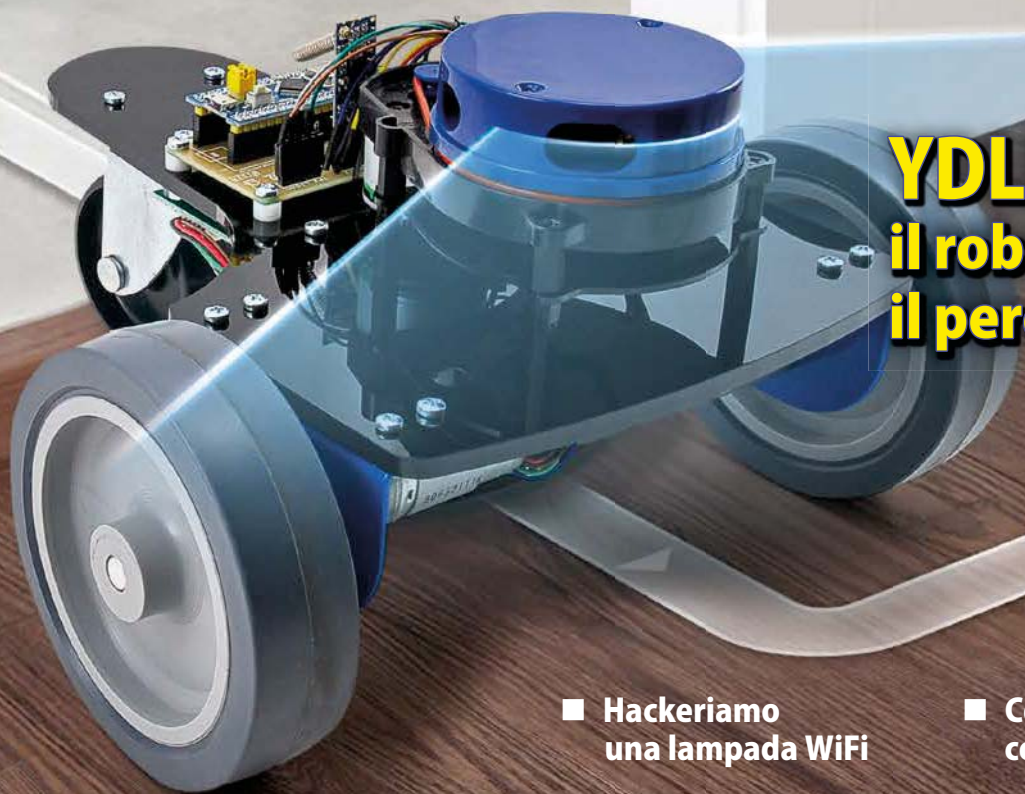


Elettronica In

www.elettronica.in.it

€ 6,00 - Anno XXVII - n. 254 - MAGGIO 2021



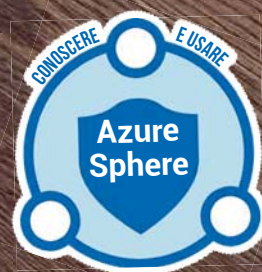
YDLidar il robot che mappa il percorso

- Hackeriamo una lampada WiFi
- Il protocollo MQTT
- Tecnologia RFID, NFC: teoria e pratica
- Controllo NeoPixel con Art-Net
- Smile Detector con Raspberry Pi

Tester servocomandi



NeoPixel WiFi



HDMI
↓
VGA



€ 11,50

Cod. **BOXHDMITOVGA**

CONVERTITORE DA HDMI A VGA 1080P PLUG AND PLAY

Permette di convertire un segnale digitale HDMI (compatibile con HDMI1.3) in un segnale VGA (SUB-D) ad alta definizione con audio in uscita. Con questo convertitore, il computer, il lettore video portatile, le console di gioco e alcuni altri dispositivi video con uscita HDMI possono connettersi a schermi HD di grandi dimensioni oppure a proiettori HD con ingresso VGA.

CONVERTITORE DA AV A HDMI PLUG AND PLAY

AV
↓
HDMI



€ 14,00

Cod. **BOXAVTOHDMI**

Permette di convertire un segnale audio/video CVBS (PAL e NTSC) in ingresso in un segnale HDMI in uscita. Con questo convertitore, il computer, il lettore video portatile, le console di gioco e alcuni altri dispositivi video con uscita CVBS possono connettersi a schermi con ingresso HDMI. L'uscita HDMI supporta una risoluzione fino a 1080P 60Hz.

Convertitori di segnale

Permettono di convertire un segnale digitale HDMI in VGA, AV in HDMI, VGA in HDMI, HDMI in AV.

VGA
↓
HDMI



€ 11,50

Cod. **BOXVGATOHDMI**

CONVERTITORE DA VGA A HDMI PLUG AND PLAY

Permette di convertire un segnale audio e video VGA (SUB-D) in un segnale HDMI (tipo A) ad alta definizione con audio. Con questo convertitore, il computer, il lettore video portatile, le console di gioco e alcuni altri dispositivi video con uscita VGA possono connettersi a schermi HD di grandi dimensioni oppure a proiettori HD con ingresso HDMI. L'ingresso VGA supporta video fino a 1920x1080 a 60 Hz, l'uscita HDMI può supportare video fino a 1080P.

CONVERTITORE DA HDMI AD AV PLUG AND PLAY

Permette di convertire un segnale HDMI in ingresso in un segnale video composito CVBS (PAL e NTSC) in uscita. Con questo convertitore, il computer, il lettore video portatile, le console di gioco e alcuni altri dispositivi video con uscita HDMI possono connettersi a schermi con ingresso CVBS. L'ingresso HDMI supporta una risoluzione fino a 1920x1080 a 60 Hz.

HDMI
↓
AV

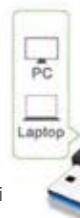


€ 11,00

Cod. **BOXHDMITOAV**

CAVO SATA-USB 3.0 PER RASPBERRY PI E PC

Cavo adattatore Plug & Play da USB 3.0 (compatibile USB 1.0, 1.1 e 2.0) a SATA 22 pin. Permette di collegare alla porta USB di Raspberry Pi o di un PC, un hard disk SATA o SSD da 2,5". N.B. non adatto per hard disk SATA da 3,5". Non compatibile con Windows 10.



€ 7,90

Cod. **CAVOSATAUSB3**





Non fermate la ricerca

Ogni mese, nelle pagine di questa rivista, trovate articoli che sono il frutto del lavoro dei nostri collaboratori, che ci permettono di proporre progetti interessanti ed innovativi.

Il team di Ricerca e Sviluppo è fondamentale sia per noi, che per aziende che vivono di tecnologia. Tuttavia la crisi dovuta al COVID-19 ha messo in difficoltà il reparto R&S di molte aziende, ponendo queste ultime di fronte a sfide improvvise, come improvvisa è stata questa stessa pandemia.

La crisi dovuta al COVID-19 ha messo in difficoltà il reparto R&S di molte aziende

In una crisi finanziaria (ed il COVID oltre ad aver richiesto un immenso tributo umanitario, ha causato difficoltà economiche inimmaginabili) la soluzione più

rapida per recuperare liquidità è sicuramente quella di tagliare le spese superflue. La storia (anche italiana) insegna che a subire tagli di budget a volte è proprio la ricerca, considerata, erroneamente, non necessaria. Infatti, investire in questo momento su ricerca e sviluppo potrebbe sembrare inutile, tutto è fermo, poche le vendite, pochi i clienti, ma è anche l'unico modo per permettere ad un'azienda di creare quella serie di prodotti e innovazioni che la porterà un passo avanti alla concorrenza, e quindi in una posizione migliore, per i tempi in cui l'economia ripartirà.

Non sottovalutiamo quindi la ricerca e lo sviluppo di nuovi prodotti e tecnologie, perchè una cosa è certa, prima o poi ne usciremo: non facciamoci trovare impreparati.

ON LINE

elettronica.in

SI RINGRAZIA

Futura Elettronica
Mouser
Texas Instruments



In copertina:
La tecnologia LiDAR è ormai alla portata di tutti. La utilizziamo in questo robot in grado di mappare la stanza in cui si trova.

Boris Landoni

boris.landoni@elettronica.in

Elettronica In

Rivista mensile, anno XXVII n. 254

MAGGIO 2021

EDITORIALE

Direttore Responsabile

Boris Landoni

boris.landoni@elettronica.in

Redazione

Stefano Garavaglia, Paolo Gaspari,
Davide Scullino, Gabriele Daghetta

redazione@elettronica.in

Grafica

Alessia Sfulcini

alessia.sfulcini@elettronica.in

Ufficio Pubblicità

Monica Premoli (0331-752668)

monica.premoli@elettronica.in

Ufficio Abbonamenti

Tel. 0331-752668

abbonati@elettronica.in

Direzione, Redazione, Pubblicità

FUTURA GROUP srl - Divisione Editoriale
via Adige 11 - 21013 Gallarate (VA)

Tel. 0331-752668

Abbonamenti

Annuo 10 numeri Euro 45,00

Annuo 10 numeri digitale Euro 39,00

Le richieste di abbonamento vanno inviate a:

FUTURA GROUP srl
via Adige 11, 21013 Gallarate (VA)

Tel. 0331-752668

oppure tramite il sito

<https://www.elettronica.in/abbonamenti/>

Distribuzione per l'Italia

SO.DI.P. Angelo Patuzzi S.p.A.

via Bettola 18 - 20092 Cinisello Balsamo (MI)

Tel. 02-660301 Fax 02-66030320

Stampa

ROTO3 Spa - Via Turbigo, 11/b
20022 CASTANO PRIMO (MI)

Tiratura
18.000 copie



Mensile
associato all'USPI,
Unione Stampa
Periodica Italiana

Rivista mensile registrata presso il Tribunale di Milano
con il n. 245 il 03/05/1995.

Prezzo di copertina Euro 6,00.

Gli arretrati nei formati cartaceo e digitale (pdf) sono acquistabili
sul sito della rivista al prezzo di Euro 6,00.

Poste Italiane Spa - Spedizione in abbonamento Postale - D.L.
353/2003 (conv. in L. 27/02/2004) art. 1 comma 1 - DCB Milano.

Futura Group srl è iscritta al Registro Operatori della
Comunicazione n. 23650 del 02/07/2013.

Impaginazione ed immagini sono realizzati in DeskTop Publishing
con programmi Adobe InDesign e Adobe Photoshop per Windows.
Tutti i contenuti della Rivista sono protetti da Copyright.

Ne è vietata la riproduzione, anche parziale, la traduzione e più in
generale la diffusione con qualsiasi mezzo senza l'autorizzazione
scritta da parte dell'Editore. I circuiti, il firmware ed il software
descritti sulla Rivista possono essere realizzati solo per uso
personale, ne è proibito lo sfruttamento a carattere commerciale
e industriale. Tutti possono collaborare con Elettronica In. L'invio
di articoli, materiale redazionale, programmi, traduzioni, ecc.
implica da parte del Collaboratore l'accettazione dei compensi e
delle condizioni stabilite dall'Editore (www.elettronica.in/ase.pdf).
Manoscritti, disegni e foto non richiesti non verranno in alcun caso
restituiti. L'utilizzo dei progetti e dei programmi pubblicati non
comporta alcuna responsabilità da parte della Società Editrice.

© 2021 Futura Group srl

Futura
Group
Edizioni

Contenuti

01

RUBRICHE

- 01 Editoriale
- 05 T&T Tips & Tricks
- 09 Embedded IoT
Digital Transformation of Things
- 13 Componenti & Sistemi
- 126 Scienza & Tecnologia

Progetto di Copertina

YDLidar ROBOT



50

Networking

NeoPixel

43



19

ARTICOLI

LABORATORIO

19 Tester servocomandi

Consente di verificare il buon funzionamento dei classici servo da modellismo testandoli con due larghezze di impulso in corrispondenza delle quali l'alberino deve assumere posizioni specifiche.

DIDATTICA

27 Il protocollo MQTT

Conosciamo il protocollo di comunicazione dati ideale per il monitoraggio a distanza tramite sensori, prediletto dalle applicazioni IoT in virtù della capacità di funzionare con bassi data-rate.

Didattica

Il protocollo MQTT



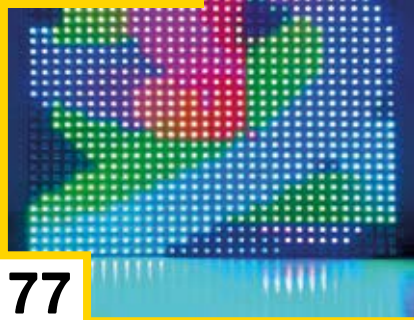
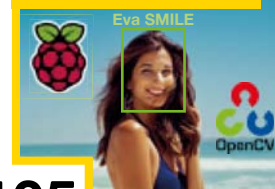
27

Laboratorio

Tester servocomandi



19

Fishino**Controllo NeoPixel con Art-Net****77****Laboratorio
Hacking WiFi****65****Didattica****Corso
Azure Sphere****117****Applicazioni****Tecnologia RFID, NFC:
teoria e pratica****89****Raspberry Pi
Smile Detector****105****NETWORKING****43****NeoPixar**

Uniamo le qualità della stampa 3D, l'efficienza e versatilità dei LED NeoPixel e la facilità di connessione a una rete WiFi tipica del ESP8266 per realizzare una lampada d'arredamento adatta ad ogni luogo.

ROBOTICA**50****YDLidar robot**

Robot su tre ruote a navigazione autonoma, in grado di muoversi e orientarsi all'interno di spazi chiusi con la capacità di ricostruire una mappa dell'ambiente circostante tramite sensore LIDAR.

LABORATORIO**65****Hacking WiFi**

Analizziamo i messaggi WiFi che vengono scambiati tra un app del cellulare ed una lampadina smart WiFi per riprodurli dal nostro pc e successivamente da un ESP8266.

FISHINO**77****Controllo NeoPixel con Art-Net**

Realizziamo una matrice di LED RGB pilotata da Fishino Mega programmata come interfaccia verso il protocollo Art-Net.

APPLICAZIONI**89****Tecnologia RFID, NFC:
teoria e pratica**

Conosciamo e sperimentiamo la tecnologia di identificazione tramite radiofrequenza, usando il modulo PN532.

RASPBERRY PI**105****Smile Detector**

Analizziamo i volti all'interno di un video o di una ripresa live e identifichiamo chi sta sorridendo mediante un algoritmo di Intelligenza Artificiale che identifica le emozioni.

DIDATTICA**117****Conoscere e usare Azure Sphere**

Collegiamo in modo sicuro la nostra board Azure Sphere al dispositivo creato in Azure IoT Central. Terza puntata.

Ottimizza la tua certificazione di sicurezza funzionale con Texas Instruments

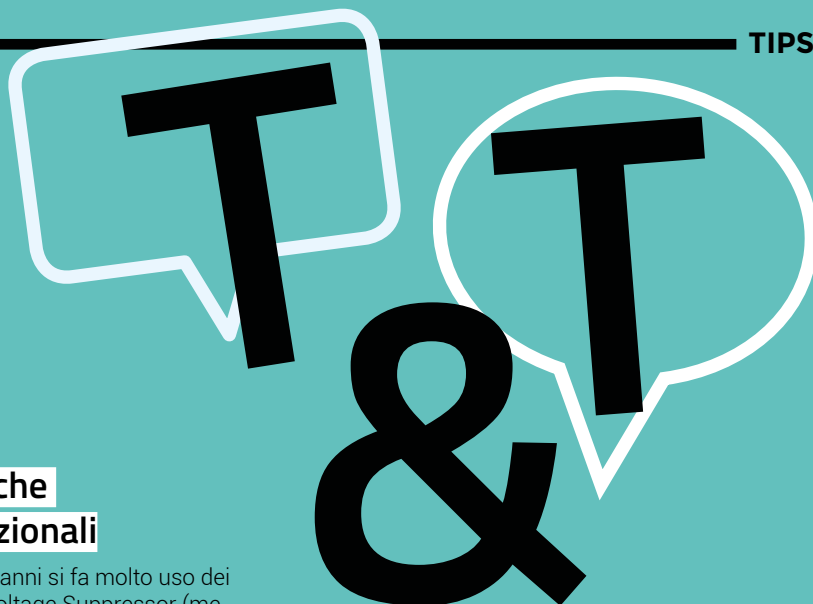
Ottieni in modo efficiente le certificazioni ISO 26262 e IEC 61508 e altre ancora, utilizzando i nostri prodotti, la documentazione disponibile e esperti di sicurezza ben informati.



Per ulteriori informazioni, tra cui analisi su richiesta e rapporti diagnostici, percentuale di FIT di sicurezza funzionale e il nostro processo di sviluppo certificato, visitare

ti.com/functionalsafety





TVS anche unidirezionali

Negli ultimi anni si fa molto uso dei Transient Voltage Suppressor (meglio noti come TVS) per proteggere anche linee dati e circuiti in bassa tensione continua; tipicamente i TVS sono diodi Zener bidirezionali composti da due Zener identici posti in antiserie e uniti per il catodo, in modo da cortocircuitare sovratensioni di ogni polarità che superino la somma:

$V_z + V_s$

dove V_z è la tensione di Zener e V_s la caduta diretta fra anodo e catodo del diodo che, per polarità della tensione, viene a trovarsi polarizzato direttamente. Tale somma corrisponde quindi alla cimatura di eventuali sovratensioni ed è la tensione d'intervento del TVS.

Dunque il TVS è un diodo Zener bidirezionale dimensionato in modo da sopportare intense correnti quando interviene e di dissipare potenze impulsive anche particolarmente elevate. Un esempio di TVS bidirezionale è il Littelfuse SP4337, che consiste in un array di diodi pensato per la protezione di bus.

Non sempre, tuttavia, il TVS è

Tips & Tricks sono i nostri suggerimenti in pillole e i piccoli trucchi riguardanti sia l'hardware che il firmware, che sperimentatori, tecnici e progettisti elettronici vorrebbero conoscere per sviluppare meglio la propria passione: quella per l'Elettronica applicata. Ecco quelli che abbiamo preparato questo mese per voi.

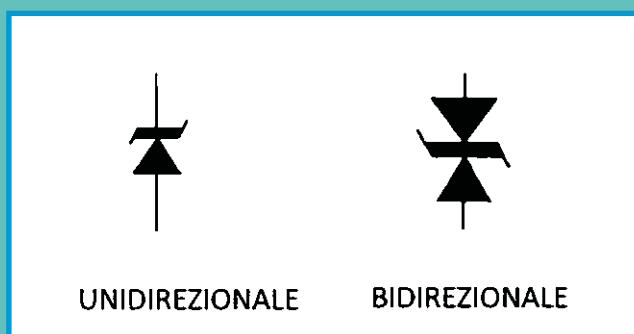
bidirezionale, perché in commercio si trovano anche diodi TVS unidirezionali: in questo caso si tratta di Zener specifici per la soppressione di sovratensioni, costruiti quindi per lavorare in regime transitorio gestendo correnti di picco e breve durata, molto elevate. Un esempio di diodo TVS unidirezionale è l'STMicroelectronics ESDA9P25-1T2.

Digitale Terrestre: canali e programmi

Nella televisione tradizionale (analogica) quando un canale non si riceveva, alla base potevano esserci problemi come un guasto in memoria o

nella sintonia, ma anche la mancata trasmissione da parte dell'emittente. Oggi invece con il Digitale Terrestre se un canale non si vede e il televisore fornisce l'avviso "servizio non disponibile" o "nessun servizio" il problema può essere dovuto all'emittente, ma molto più facilmente alla mancata ricezione del segnale del relativo transponder. Per spiegare ciò bisogna sapere che mentre nell'analogico ogni canale TV occupa un canale RF largo 7-8 MHz, nel DTT (Digitale Terrestre) a modulare la portante del singolo canale RF non è il videocomposito né l'audio analogico, bensì una stringa di dati che esprime in forma numerica i segnali televisivi di un certo numero di emittenti. Ne deriva che ogni singola "frequenza" delle bande TV veicola molti canali.

Il contenuto del segnale DTT, che occupa la banda di un normale programma TV analogico, si chiama Transport Stream (TS) o Multi Program Transport Stream (MPTS) quando contiene i dati di più servizi (i servizi sono i canali televisivi nel DTT, ossia i contenuti dell'emittente) anche se spesso viene chiamato impropriamente Multiplex o Transponder (con riferimento al ripetitore). Un MPTS contiene più programmi video, audio, informazioni relative ai programmi trasmessi come l'EPG e altri dati, ad



↑ Simboli del disegno tecnico per i due tipi di TVS.



↑ Tipica schermata delle informazioni di un decoder per Digitale Terrestre.

esempio per servizi interattivi. I segnali audio e video vengono digitalizzati e codificati, quindi inviati ad un blocco chiamato TS MUX (Transport Stream MultipleXer), che forma il flusso digitale, ossia il Transport Stream, da inviare al sistema che prepara i dati per la trasmissione. Il Transport Stream (o anche MPEG-2 TS) è a sua volta composto da pacchetti dati, chiamati TS packet o semplicemente TS, che si ripetono ciclicamente a frequenza molto elevata, così da garantire un aggiornamento della visione e dell'audio adeguati alle esigenze dell'HDTV.

Con questa tecnica, nello spazio di un

canale RF corrispondente ad un canale della TV tradizionale si possono inserire fino a 6 canali TV e due radiofonici, ma anche fino ad 8 TV più 2 radiofonici: tutto dipende da quanto è possibile comprimere i dati, non a caso l'imminente DVB-T2 supporterà fino a 10 servizi per canale RF.

Il canale radio su cui trasmette un'emittente è visualizzato dal televisore insieme ad altre informazioni premendo il tasto Info (o anche "i"); quindi se non si vede un'emittente (servizio) bisogna per prima cosa aprire la pagina Info e annotare il canale UHF o VHF e la frequenza corrispondente (vedi schermata in queste pagine) che corrisponde al relativo transponder o ripetitore DTT.

Fatto ciò si va a verificare se gli altri canali trasmessi sulla stessa frequenza si ricevono: in caso affermativo il problema è nel relativo servizio, quindi l'emittente non sta trasmettendo; in caso contrario il transponder non è in onda oppure c'è un guasto nella sintonia del TV o decoder, ma molto più probabilmente si tratta di un problema d'antenna: il centralino della casa non amplifica a sufficienza il relativo canale o l'antenna sul tetto si è spostata di qualche grado. Infatti siccome le antenne TV, specie quelle per UHF, sono direttive, basta poco ad attenuare un segnale in arrivo da lontano, abbastanza da non ricevere più le emittenti ospitate da un canale RF; questo vale tanto più nelle giornate in cui la propagazione delle onde radio è scarsa.



↑ Un tipico polyswitch.

Polyswitch: i fusibili autoripristinanti

Si tratta di componenti elettronici basati su un impasto di materiali che si comporta come quello dei termistori PTC, però caratterizzato da ridottissima resistenza a freddo e da un elevatissimo coefficiente di temperatura in una zona ben precisa della curva di potenza; questo fa in modo che superando la corrente per cui sono dimensionati e scaldandosi, la loro resistenza diventa elevatissima, tanto da bloccare praticamente la corrente. Il raffreddamento e quindi il ripristino del basso valore di resistenza nominale a freddo, avviene lentamente e solo se la sovracorrente è cessata. Questi componenti, anche noti come polyswitch, simulano il comportamento del fusibile nella maggior parte delle applicazioni e presentano il vantaggio di non richiederne la sostituzione, diversamente dai veri fusibili, che dopo l'intervento vanno sostituiti perché si interrompono definitivamente.

Gradi di temperatura degli integrati

I circuiti integrati si distinguono per temperatura operativa ambiente e possono essere classificati in gradi:

- grado militare e aerospaziale, capaci di funzionare a temperature comprese tra -55 e $+125$ °C;
- grado industriale, capace di operare fra -40 e $+125$ °C;
- grado civile, operanti fra 0 e $+70$ °C.

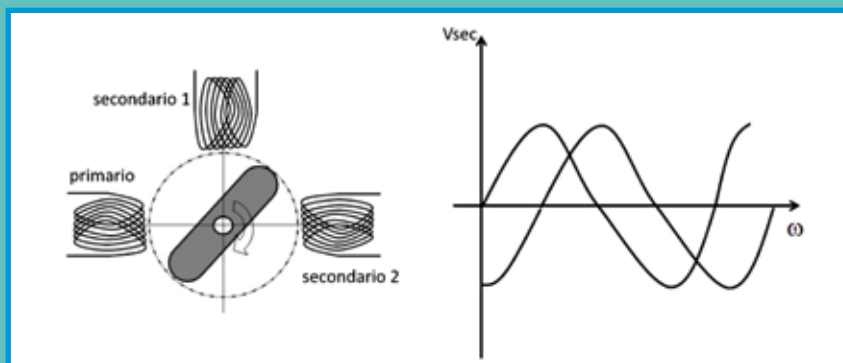
Recentemente è stato introdotto un quarto grado, che è quello automotive, con campo di temperatura operativa compreso fra -40 e 150 °C. Inoltre i componenti di grado automotive devono rispettare lo standard AEC-Q100, che prevede a sua volta cinque gradi o livelli:

- Grado 0 (A) = $-40 \div 150$ °C;
- Grado 1 (Q) = $-40 \div 125$ °C;
- Grado 2 (T) = $-40 \div 105$ °C;
- Grado 3 (I) = $-40 \div 85$ °C;
- Grado 4 (C) = $-40 \div 70$ °C.

Gli integrati che devono funzionare in apparati posti all'interno del vano motore sono di Grado 0 e 1.

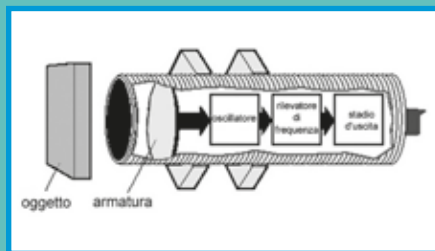
Resolver: un altro modo di dire encoder

Conosciamo gli encoder ottici, che sono dispositivi rotativi in grado di fornire



↑ **Struttura del resolver a tre avvolgimenti e andamento delle tensioni ai secondari in rapporto alla rotazione del rotore.**

↓ **Composizione del sensore capacitivo.**



impulsi corrispondenti alla posizione angolare del loro asse rotante, ma anche a velocità di rotazione e accelerazione. Per rilevare la rotazione esistono dispositivi analoghi che sfruttano le tecniche capacitiva e induttiva; in quest'ultimo caso si tratta del **resolver**, un tipo di trasduttore di spostamento induttivo, elettromeccanico, che permette la misura di spostamenti angolari attraverso il rilevamento della variazione del flusso di induzione magnetica, concatenato con un solenoide, in funzione della posizione del solenoide stesso rispetto al rotore.

Il resolver è alimentato in corrente alternata e, a differenza dell'encoder, fornisce un segnale analogico alternato, la cui ampiezza è proporzionale all'angolazione misurata. Tipicamente, per come è costruito, nella sua forma di base il resolver rileva una rotazione entro 180° ; ciò si comprende considerando che il sensore è formato da una bobina fissa (statore) e una o due rotanti solidali con l'albero. Esistono resolver più complessi, dotati di più avvolgimenti che permettono di rilevare l'angolo giro (rotazione completa).

Sostanzialmente il resolver è un trasformatore utilizzato per misurare l'angolo di rotazione e sovente si presenta come un motore elettrico, con uno o più avvolgimenti in rame sullo statore e un rotore fatto di lamierini in metallo. L'accoppiamento induttivo tra gli avvolgimenti del trasformatore varia in funzione dell'angolo di inclinazione del rotore rispetto allo statore, pertanto se si alimenta lo statore con un segnale in corrente alternata e si misura la tensione degli avvolgimenti del rotore, si ottiene un segnale elettrico in corrente alternata la cui ampiezza è proporzionale all'angolo formato dal rotore con gli avvolgimenti, giacché cambia l'intensità del flusso concatenato.

Anche se sono disponibili molte varianti

e tipi diversi, un tipico resolver ha tre avvolgimenti sullo statore: uno che fa da primario e viene alimentato a corrente alternata e due che sono i secondari. Questi avvolgimenti sono realizzati con filo di rame e di solito sono formati sull'elemento stazionario del resolver (lo statore).

Nell'immagine in questa pagina è schematizzato il resolver del genere e come si vede il suo rotore è rettilineo a due poli salienti; ne deriva che quando è allineato, ossia si trova sull'asse del primario e di un secondario, viene trasferito il massimo flusso magnetico dal primario al secondario, quindi la tensione indotta ai capi di quest'ultimo è massima; invece quando l'asse è perpendicolare il flusso è minimo, giacché dipende dal coseno dell'angolo, che a 90° vale 0. La tensione sul secondario perpendicolare è quindi nulla.

Nel caso dell'encoder in figure, essendoci due secondari ortogonali tra loro, alimentando il primario con una tensione sinusoidale si otterranno le due tensioni d'uscita (sui secondari) proposte dal grafico: una forma d'onda sinusoidale ed una cosinusoidale. Come si vede, il rapporto tra i segnali varia in base all'angolo, il che permette di dedurre da esso di quanto sta ruotando il rotore.

Il resolver è realizzato in molte varianti e viene impiegato in controlli di processo industriale, robot industriali, macchine utensili, strumenti di misura, plotter, ma anche nelle trasmissioni delle moderne automobili.

Il sensore di prossimità capacitivo

Per rilevare la vicinanza di corpi ed anche di liquidi nei serbatoi è possibile utilizzare sensori di tipo capacitivo.

Si tratta di un dispositivo capace di rilevare la prossimità di vari materiali ed anche liquidi e può avere un'uscita analogica o digitale (in questo caso si parla di interruttore di prossimità).

Il sensore capacitivo si basa sull'interferenza con la costante dielettrica di un elemento che viene avvicinato, la quale determina una variazione significativa di capacità fra le armature (opportuna-mente sagomate) del condensatore alla base del sensore; la variazione determina una mutazione di frequenza in un oscillatore nel cui circuito risonante è inserito il condensatore e tale mutazione viene rilevata attraverso un circuito discriminatore che pilota l'uscita, tipicamente a transistor, fornendo una tensione proporzionale alla distanza dell'oggetto. Il segnale generato è tipicamente espresso in millivolt/mm di distanza dall'oggetto o massa liquida. Nella versione a interruttore, un comparatore segue il discriminatore e quando viene superata la soglia attiva un transistor NPN o PNP (nel primo caso alimenta l'utilizzatore portandolo a massa, mentre nel secondo lo porta al positivo) che commuta aprendo o chiudendo il circuito.

Web Forum e supporto tecnico

Hai un problema con uno dei circuiti pubblicati? Vorresti effettuare una modifica? Accedi al nostro Web Forum dove i nostri tecnici (ma anche gli altri lettori) ti aiuteranno a chiarire qualsiasi dubbio di natura tecnica. Collegati a:

www.elettronicaain.it/webforum

INDISPENSABILE IN LABORATORIO!



L'alimentatore è lo strumento più importante di ogni laboratorio elettronico da cui si esige prima di tutto la possibilità di regolare con buona risoluzione la tensione e la corrente di uscita, requisiti soddisfatti da questi nuovi alimentatori. Mantengono i valori impostati in modo affidabile e indipendentemente dalle fluttuazioni della tensione di alimentazione o del carico collegato.



DISPONIBILE ANCHE NELLE VERSIONI:

DA 0-30 VDC / 0-10 A

COD. AL3010 € 106,00

DA 0-60 VDC / 0-5 A

COD. AL6005 € 106,00

€ 78,00

Cod. AL3005N



ALIMENTATORE LINEARE CON USCITA REGOLABILE 0-30 VDC / 0-5 A

Alimentazione: 110/220 VAC, 50/60 Hz

Uscita duale regolabile:

- Tensione: da 0 a 30 VDC

- Corrente: da 0 a 5 A

Risoluzione della tensione: 10/100 mV

Risoluzione della corrente: 1/10 mA

Precisione visualizzazione della tensione: $\pm 1\%$ +1 cifra

Precisione visualizzazione della corrente: $\pm 1\%$ +2 cifre

Dimensioni: 220x165x85 mm

Peso: 1,350 kg



Prezzi IVA inclusa.

Nasce il dispositivo di screening Covid-19 non diagnostico basato su Machine Learning: si tratta del monitor Tiger tech Covid plus, che identifica alcuni biomarcatori che possono essere indicativi di infezione da Sars-CoV-2, nonché altre condizioni ipercoagulabili (come sepsi o cancro) o stati iperinflammatori (come reazioni allergiche gravi), in individui asintomatici di età superiore ai 5 anni. Non è destinato all'uso in soggetti con sintomi della malattia ed è adatto a monitorare pazienti senza febbre. Il dispositivo è un bracciale con sensori di luce che rilevano i segnali delle pulsazioni del sangue per un periodo

› Covid-19: screening col Machine Learning



da 3 a 5 minuti, da cui il processore estrae la frequenza del battito cardiaco (BPM) e insieme ad altre informazioni le immette in un modello di apprendimento automatico probabilistico addestrato per fare previsioni sull'infezione; una segnalazione luminosa indica con un colore differente se la persona cui lo strumento è applicato evidenzia i biomarcatori o meno.

Le prestazioni cliniche del monitor sono state studiate in ambito ospedaliero su 467 individui asintomatici, inclusi 69 confermati positivi, evidenziando una concordanza positiva del 98,6% e una concordanza negativa del 94,5%.

www.covidplusmonitor.com

› IoT e AI prevengono le infestazioni di insetti

Grazie a una nuova trappola per il monitoraggio automatico dei fitofagi basata su IoT, Machine Learning e modelli previsionali, è possibile prevenire le infestazioni da parassiti, che secondo la FAO causano perdite annue tra il 20 e il 40% della produzione mondiale. Per contrastare il fenomeno xFarm, piattaforma dell'omonima azienda con focus sull'Agricoltura 4.0, ha messo a punto una rete di monitoraggio automatico della presenza di fitofagi che aiuta a prevenire le infestazioni e a ridurre il numero dei trattamenti fitosanitari, a favore del contenimento dei costi e dello sviluppo di un'agricoltura sostenibile. Si tratta di xTrap, una trappola automatica IoT dotata di una fotocamera ad alta risoluzione che permette di tenere sotto controllo di diverse specie di insetti, un algoritmo di Machine Learning per il riconoscimento e conteggio automatico degli insetti, e un modello previsionale in grado di

predire l'andamento delle infestazioni. Il tutto permette agli agricoltori di visualizzare quotidianamente da remoto l'andamento delle catture e ricevere consigli per capire in modo tempestivo quando e dove effettuare i trattamenti. Tre sono le versioni: Delta, che prevede l'impiego di ferormoni, Night, che si

serve della luce UV e Color, che utilizza fogli cromatici. I dati raccolti sono disponibili sulla piattaforma xFarm che, insieme alla relativa App, accompagna aziende agricole, enti di ricerca e soprattutto consorzi nella digitalizzazione dei processi agricoli.

xfarm.ag



› Renesas rende sicura l'IoT

I microcontrollori a 32 bit Arm Cortex-M della famiglia RA, Renesas elevano il livello di sicurezza IoT grazie all'ottenimento delle certificazioni PSA Certified Level 2 e Security Evaluation Standard for IoT Platforms (SE-SIP, che è una versione ottimizzata della Common Criteria methodology (ISO 15408-3) per la valutazione dei componenti IoT e piattaforme connesse. Alle serie di dispositivi RA4 e RA6 di Renesas, supportati dal Flexible Software Package (FSP) già certificati PSA Livello 1, si affianca il nuovo microcontrollore RA6M4 certificato PSA Livello 2. I dispositivi RA6M3, RA6M4 e RA4M2 di Renesas hanno raggiunto la certificazione SE-SIP1 contro attacchi sia a livello fisico che logico.

Inoltre i microcontrollori RA implementano tecnologie per applicazioni IoT che necessitano elevata sicurezza, combinando la presenza della Secure Crypto Engine con le certificazioni NIST CAVP, oltre alla possibilità di sfruttare la Arm TrustZone per Armv8-M. Integrano funzionalità hardware integrate per la sicurezza, a partire dall'acceleratore per algoritmi AES, sino al sottosistema isolato per la gestione delle procedure di crittazione. La Secure Crypto Engine offre la cifratura sia simmetrica che asimmetrica, funzionalità hash, generazione di numeri casuali (TRNG), gestione delle chiavi in modo avanzato, inclusa la creazione delle stesse e il "wrapping" della chiave unica del microcontrollore.

› L'AI aiuta i malati di SLA a comunicare

Brain Control BCI è un ausilio di comunicazione diretta tra cervello e computer per i malati operativo presso il reparto di neurologia e neurofisiopatologia dell'Ospedale di Pistoia. Consiste di un caschetto EEG (per elettroencefalogramma) che, collegato in wireless con un tablet o un PC, invia i segnali elettrici del cervello rilevati, opportunamente condizionati, che il computer elabora trasformandoli in comandi per interagire col sistema. Brain Control BCI è un'interfaccia cervello-computer che nei pazienti malati di SLA locked in, ad esempio, consente di tradurre i segnali del sistema nervoso

centrale in parole e azioni.

BrainControl BCI rende possibile comunicare con l'esterno grazie al pensiero, dando modo di rispondere a semplici domande e di comunicare. Tramite il caschetto intercetta l'attività elettrica sulla corteccia cerebrale, interpreta e classifica le onde generate dal movimento immaginato, attraverso un software basato sull'Intelligenza Artificiale. Grazie al training, con esso il paziente impara ad usare pensieri "decodificabili" per selezionare la risposta voluta sul dispositivo.

BrainControl fa parte di una gamma di dispositivi che comprende anche

BrainControl Sensory e Avatar: tre soluzioni, compatibili con i differenti gradi di mobilità e interazione del paziente. BrainControl Sensory è pensato per pazienti con movimenti residui volontari di qualsiasi parte del corpo (movimenti oculari, movimenti della mano, delle dita, dello zigomo, ecc.); grazie a una serie di sensori, quali puntatori oculari, sensori di movimento, emulatori mouse, è possibile sfruttare i movimenti residui del paziente per creare un'interazione con il mondo esterno. BrainControl Avatar è pensato per superare i vincoli delle persone con difficoltà motorie: permette di visitare da remoto installazioni, musei, spazi espositivi ed eventi in genere. In maniera del tutto indipendente, è possibile comandare un alter ego robotico, regolando audio, video e altezza del campo visivo. L'esperienza sarà immersiva e reale, come se si stesse visitando di persona l'ambiente prescelto. Inoltre, il monitor e gli altoparlanti di cui è dotato l'avatar consentono di rendersi visibile a distanza, se lo si desidera, dando spazio a una personificazione dell'Avatar che permette alle persone intorno di interagire con esso, stabilendo una comunicazione nuova e vivace fra le persone sul posto e la persona connessa da remoto.

www.braincontrol.eu

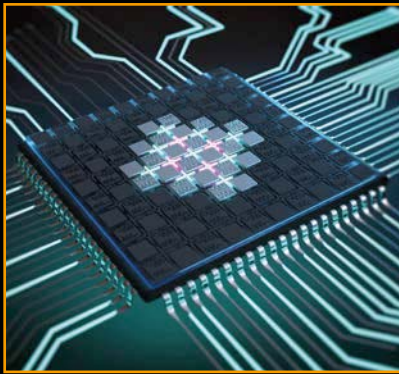


Un "access management circuit" impedisce l'accesso alla crypto engine dall'esterno se non viene eseguita la corretta sequenza di operazioni, inoltre la RAM dedicata assicura che le chiavi in chiaro non siano mai disponibili alla CPU o su bus relativi alle periferiche. La certificazione PSA rende disponibile un framework per realizzare connessioni sicure per i dispositivi, dall'analisi fino alla valutazione e certificazione. La certificazione PSA Level 2 garantisce protezione da attacchi software a diversi livelli ed è conseguita mediante supporto da parte di laboratori di terze parti che seguono la linea guida stabilite dal PSA Root of Trust (PSA-RoT): sfruttano l'analisi delle vulnerabilità e test di penetrazione di PSA-RoT per verificare che i nove requisiti di sicurezza siano stati soddisfatti.

www.renesas.com



› Infineon sviluppa tecnologia per Quantum Computer



Infineon Technologies sta lavorando allo sviluppo del "Computer quantistico tedesco basato su Qubit superconduttivi" (GeQCoS) insieme a scienziati di cinque istituti di ricerca in Germania, per promuovere lo sviluppo e l'industrializzazione del calcolo quantistico, tema promettente per via delle enormi possibilità offerte dalla tecnologia in fatto di velocità di elaborazione, che sono enormemente maggiori di quelle dei più veloci computer tradizionali odierni.

Infineon in questo progetto porta la sua esperienza nella produzione industriale di chip di semiconduttori e negli approcci al calcolo quantistico come

quello che si basa sulle cosiddette "trappole ioniche".

I computer quantistici (o Quantum Computer che dir si voglia...) finora sono stati considerati come fenomeni da laboratorio limitati a risolvere problemi accademici specifici e fondamentalmente a dimostrare come funzionano.

Un'architettura adatta per il calcolo dei problemi pratici richiede -invece- ulteriori miglioramenti a tutti i livelli, dai moduli hardware elementari e qubit (quantum bit, ossia bit quantistici), al software e all'applicazione.

L'obiettivo del progetto, finanziato dal Ministero tedesco dell'Istruzione e della ricerca con 14,5 milioni di euro, è sviluppare un processore quantistico rivoluzionario basato su qubit superconduttori e dimostrare le sue capacità speciali su un prototipo entro quattro anni. Per raggiungere questo obiettivo lavorano insieme scienziati del Walther Meißner Institute (WMI) dell'Accademia bavarese delle scienze e delle discipline umanistiche e dell'Università tecnica di Monaco, il Karlsruhe Institute of Technology (KIT), l'Università Friedrich Alexander di Erlangen-Norimberga (FAU), il

Forschungszentrum Jülich (FZJ) e il Fraunhofer Institute for Applied Solid State Physics (IAF) e Infineon.

Anche per i qubit sono necessari metodi per la produzione di massa di microstrutture, pur mantenendo una qualità costante e ripetibile; qui entra in gioco Infineon con le sue competenze tecnologiche, che allo scopo ha già sviluppato un nuovo chip per la realizzazione di un processore quantistico a trappola ionica insieme a fisici sperimentali dell'Università di Innsbruck e sta collaborando con altri partner per gettare le basi per diffondere e applicare le tecnologie quantistiche.

Il potenziale di creazione di valore dell'applicazione del calcolo quantistico è molte volte maggiore rispetto alla tecnologia stessa; il progetto congiunto pone le basi per l'attuazione dell'attuale iniziativa federale per costruire un computer quantistico "made in Germany". L'obiettivo è creare una rete a livello nazionale per i qubit superconduttori in stretta collaborazione tra il mondo della ricerca scientifica, da cui nascono le conoscenze, e quello dell'industria, che deve tramutarle in tecnologia produttiva.

www.infineon.com

Per lo smart working, per chattare con amici e parenti comodamente da casa!

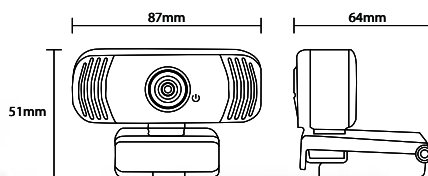
Prezzo IVA inclusa.



€24,00
Cod. FR795

WEBCAM FULL HD 1080P

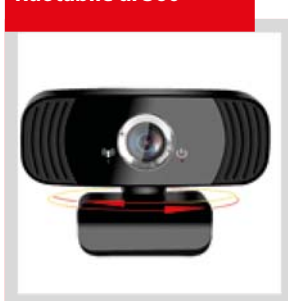
Webcam Full HD 1080P particolarmente indicata per seguire lezioni online, effettuare videoconferenze, videochat, videosorveglianza, registrazioni video, ecc. Dispone di lenti prive di distorsione, microfono omnidirezionale con riduzione del rumore, regolazione manuale della messa a fuoco, angolo di ripresa: 60° (verticale) – 120° (orizzontale), rotazione: 360°, LED di stato per alimentazione e attivazione webcam. Completamente Plug & Play (non richiede l'installazione di nessun driver), può funzionare con Windows 10/8/7/ Vista, Mac OS X, Linux, ChromeOS, Ubuntu e Android 4.0 o versioni successive. Basta collegarla alla porta USB del PC, attendere che il sistema operativo la rilevi e poi utilizzarla con Skype, Zoom, Facebook, ecc. Grazie alla sua base multifunzionale è possibile posizionarla e utilizzarla su PC fissi, PC portatili, smart TV, appoggiarla sulla scrivania oppure montarla su treppiede per macchine fotografiche.



- ✓ **Processore:** 3518EV300
- ✓ **Sensore immagine:** CMOS a colori
- ✓ **Risoluzione:** 1920 x 1080
- ✓ **Video stream:**
 - H.264: 1920 x 1080 / 1920 x 720 / 640 x 360
 - MJPEG: 1920 x 1080 / 1920 x 720 / 640 x 360
 - YUV: 640 x 360
- ✓ **Video frame rate (fps):** 30 fps (1280 x 720 / 1920 x 1080)
- ✓ **Uscita Video:** USB Video streaming
- ✓ **Controllo Immagine:**
 - messa a fuoco manuale
 - bilanciamento del bianco automatico
 - controllo automatico dell'esposizione (AEC, Automatic Exposure Control)
- ✓ **Otturatore elettronico:** automatico
- ✓ **Microfono:** omnidirezionale
- ✓ **Rapporto segnale/rumore:** > 50dB
- ✓ **Angolo di ripresa:** 60° (verticale) 120° (orizzontale)
- ✓ **Rotazione:** 360°
- ✓ **Sistemi operativi supportati:** Windows 10/8/7/Vista, Mac OS X, Linux, ChromeOS, Ubuntu e Android 4.0 o versioni successive
- ✓ **Alimentazione:** 5VDC / 160mA (tramite porta USB)
- ✓ **Temperatura di lavoro:** da -40°C a +70°C
- ✓ **Materiale:** ABS
- ✓ **Dimensioni (mm):**
 - Webcam: 87 x 64 x 51mm
 - lunghezza del cavo USB: 1,5m
- ✓ **Peso:** 180g



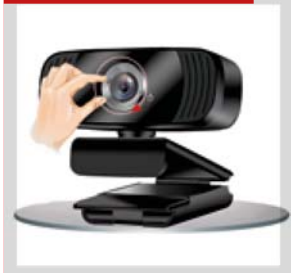
Ruotabile di 360°



Vari metodi di posizionamento



Supporta la messa a fuoco manuale



Alte potenze per i MOSFET SiC Toshiba

Toshiba Electronics Europe ha sfruttato le proprie competenze tecnologiche nei processi a semiconduttori a banda larga (WBC) per introdurre un modulo MOSFET compatto ma efficiente.

Il nuovo MG800FXF2YMS3 incorpora dispositivi MOSFET da 3300V a doppio canale al carburo di silicio (SiC) in grado di supportare correnti da 800A.

Le principali applicazioni di questi moduli ad alta densità di potenza includono gli azionamenti industriali, le apparecchiature per il controllo dei motori, gli inverter di potenza per i siti di generazione di energia rinnovabile, e i convertitori/inverter necessari per l'infrastruttura elettrica in ambito ferroviario.

Le elevate prestazioni dei nuovi moduli MOSFET SiC sono dovute anche a package iXPLV, basati su un'avanzata tecnologia di connessione interna con sinterizzazione d'argento, che consente

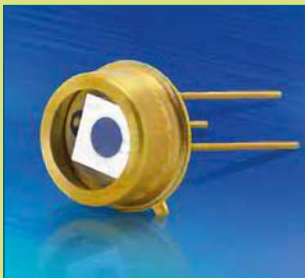


di ottenere alti livelli di efficienza in condizioni operative. È possibile supportare temperature di canale fino a 175°C, ed è inoltre garantito l'isolamento fino a 6 kV RMS. Le perdite di commutazione

all'accensione e allo spegnimento sono mantenute rispettivamente a 250mJ e 240mJ, con valori tipici di induttanza parassita previsti di appena 12 nH.

toshiba.semicon-storage.com

Fotodiodo UV ad elevate prestazioni



Opto Diode, un'azienda del gruppo ITW, introduce un nuovo fotodetector (fotorivelatore a fotodiodo) circolare ad alte prestazioni: si tratta dell'UVG5S, un fotodiodo sensibile alla gamma dell'ultravioletto con un'area attiva circolare di ben 5 mm². Il nuovo dispositivo è ideale per il rilevamento delle lunghezze d'onda tra 225 nm e 400 nm e presenta un degrado della risposta inferiore al 2% dopo l'esposizione a 7.000J/cm² a 254 nm. Con un'efficienza

quantistica interna al 100% da 200 nm a 365 nm, il fotodiodo è assemblato in un contenitore metallico un filtro UV in vetro. Le caratteristiche elettro-ottiche includono una risposta da 0,09 A/W (minimo) a 0,115 A/W (tipico) e la corrente di dark di 1nA. La tensione di breakdown varia da 25 V (minimo) a 50 V (tipico), e la capacità va da 500 pF (minimo) a 1500 pF (massimo).

Il tempo di risposta è di 1 μs (minimo) a 2 μs (massimo). La temperatura di immagazzinamento va da -20° a 80°C, mentre la temperatura di saldatura ammessa è di 240°C.

Il sensore UVG5S è ideale per il monitoraggio di sorgenti laser e altre applicazioni con sorgenti di energia UV ad elevata potenza.

optodiode.com

Fino a 50A in SMD, con Schurter

La famiglia di fusibili ceramici SMD UMT-H Schurter conta ora ben 26 valori di corrente e si arricchisce con le versioni da 40 e 50 A; si tratta di correnti d'intervento eccezionali, considerato che si tratta di dispositivi per montaggio superficiale. I fusibili SMD con la caratteristica Time-Lag T Schurter, certificati UL248-14, sono ora assortiti fra 160 mA e 50A, con tensioni di lavoro di 277 V in c.a. e 250 V in c.c. permesse dalla distanza tra i pad, che è maggiore di 10 mm e tale da evitare la scarica se il fusibile si interrompe.

A consentire la gestione di valori di corrente tanto elevati concorre l'ampia superficie dei pad in Ni-Sn, che permettono di realizzare la saldatura su PCB in grado di assicurare un'eccellente dissipazione del calore dovuto alla resistenza di contatto. I fusibili paralle-

lepidi Schurter della serie UMT-H nascono per la protezione primaria nell'elettronica industriale, nella tecnologia medica e nelle telecomunicazioni, nonché nella mobilità elettrica. Sono ideali anche per la gestione dei motori, giacché la loro caratteristica a risposta ritardata Time-Lag T permette di supportare correnti di spunto molto più elevate di quelle nominali.

www.schurter.com



Nuove DDR5 da SMART Modular



SMART Modular Technologies presenta nuove memorie DDR5, progettate per fornire maggiore larghezza di banda di memoria e maggiori prestazioni per aiutare a elaborare i dati negli HPC, nell'AI e nell'Edge Computing. La tecnologia DDR5 fornisce il doppio delle prestazioni della

DDR4, con un data-rate da 4,8 Gbps a 6,4 Gbps e una migliore efficienza energetica, grazie a un regolatore di tensione on-DIMM 12V (PMIC) e una tensione I/O di 1,1V. Altri fattori che elevano le prestazioni delle DDR5 sono la topologia dual-channel DIMM, il doppio la lunghezza del burst a BL16

e il numero di banchi. Le DDR5 dispongono di due canali secondari indipendenti da 40 bit sullo stesso modulo e hanno una densità massima del *die* di 64 Gb. La famiglia di moduli DDR5 di SMART Modular include RDIMM, UDIMM e SODIMM che vanno da 16 GB a 64 GB. www.smartm.com

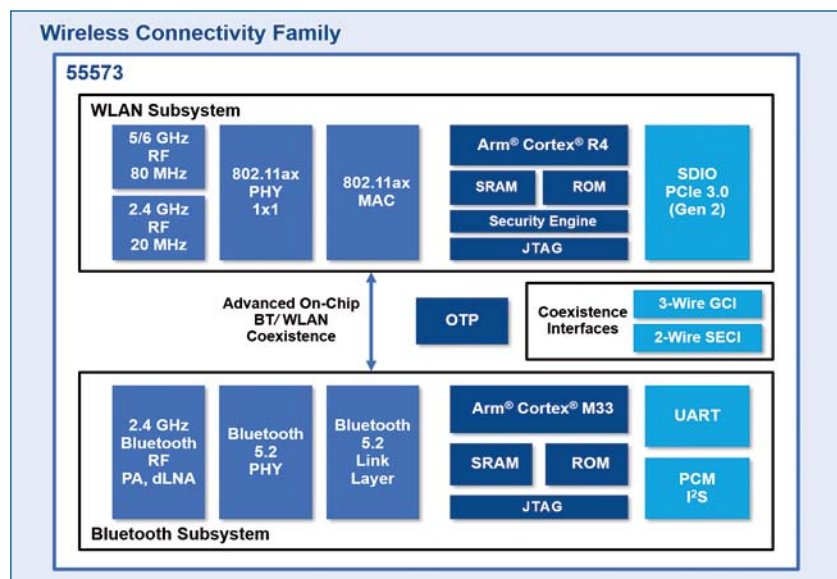
Soluzioni combo WiFi6 + Bluetooth 5.2 da Infineon

La nuova serie di prodotti AIROC WiFi 6/6E e Bluetooth 5.2 di Infineon Technologies risponde alla crescente domanda di connettività wireless sicura e a basso costo; contribuisce inoltre a ridurre la congestione delle reti wireless domestiche.

Con i nuovi dispositivi, Infineon consente ai produttori di sviluppare facilmente soluzioni intelligenti e affidabili che fanno funzionare l'IoT: dai sensori e microcontrollori all'alimentazione alla connettività e al software protetti.

Diversamente dalle precedenti generazioni di WiFi che si concentravano sulla velocità di comunicazione, il WiFi 6 implementa soluzioni per ridurre la congestione della rete e migliorarne l'efficienza, ridurre i consumi, la latenza e la portata.

La serie AIROC di componenti per la connettività wireless combinata WiFi 6/6E e Bluetooth 5.2 opera nello spettro 2,4 GHz, 5 GHz e nel più recente spettro greenfield 6 GHz per offrire prestazioni robuste e latenza minima, rendendole



ideali per applicazioni di streaming audio e video di alta qualità come le console di gioco, AR/VR, altoparlanti intelligenti, dispositivi di streaming multimediale e infotainment automobilistico. Anche le

applicazioni che richiedono una risposta immediata, come i sistemi di sicurezza e l'automazione industriale, trarranno vantaggio dai nuovi prodotti di Infineon. www.infineon.com

Unmanaged Switch PoE con isolamento

Phoenix Contact sta ampliando la sua gamma di Unmanaged Switch POE (Power-over-Ethernet) Gigabit con nuovi dispositivi a 5 e 8 porte dotati di isolamento galvanico, guide di supporto e caratterizzati da temperatura di esercizio fra -10 °C a +60 °C e da un contenitore slim. La Quality of Service (QoS) im-

plementata, la capacità di definire le priorità dei protocolli di automazione ed il supporto per i jumbo frame completano la gamma di funzionalità. Gli switch della serie 1000 consentono l'alimentazione di un massimo di otto dispositivi tramite lo stesso cavo Ethernet e soddisfano gli standard IEEE 802.3 at (PoE+) potendo

alimentare i Powered Device (PD) con un massimo di 30W a 48Vcc utilizzando cavi lunghi fino a 100 metri. Alcuni Switch della serie 1000 supportano meccanismi per il rilevamento delle interruzioni di comunicazione sulle singole porte o di problemi di alimentazione. www.phoenixcontact.com



Il time server Microchip a prova di disturbi GPS e spoofing



Le reti mission-critical possono mantenere un funzionamento affidabile solo se ricevono l'orario di rete esatto dai server di riferimento, ma i server possono essere vulnerabili ai disturbi e allo spoofing GPS. Microchip propone una soluzione a tale problema integrando la tecnologia BlueSky negli strumenti di rete SyncServer S600, prima a integrare il rilevamento e la protezione da jamming e spoofing GPS, in combinazione con la registrazione e l'analisi dei dati in radiofrequenza (RF) locali in un time server. Lo strumento SyncServer S600 Series Stratum 1 insieme ai rilevatori di jamming

e spoofing intelligenti della tecnologia BlueSky monitora continuamente lo stato della costellazione GPS locale ed esamina l'integrità del segnale RF locale e GPS per garantirne la validità. Se viene rilevata un'anomalia, la soluzione invia un allarme e, se necessario, SyncServer può attingere a sorgenti temporali alternative o ad un oscillatore interno. Ciò assicura un orario certo con degrado temporale minimo e prevedibile per le operazioni di rete vitali in applicazioni che vanno dal settore bancario e del commercio di azioni ai servizi elettrici, aerospaziale e difesa. La tecnologia SyncServer BlueSky include una suite di strumenti di registrazione, creazione di grafici e misurazione per caratterizzare i segnali satellitari GPS locali e gli eventi RF locali nel tempo.

www.microchip.com

Fotocamera LIDAR da STM e Intel

La tecnologia di scansione con micro-specchi di ST è stata scelta da Intel per la fotocamera di profondità ad alta risoluzione LiDAR Intel RealSense L515 in grado di catturare milioni di punti di profondità al secondo. Grazie a un array MEMS di micro-specchi è possibile la scansione ad alta risoluzione dell'ambiente circostante per applicazioni industriali come bracci robotizzati per il prelievo di oggetti, misure volumetriche, logistica e scansione 3D.

La fotocamera LiDAR misura 61 mm di diametro e 26 mm di altezza e contiene, oltre al micro-specchio, un sensore a fotodiodi; cattura 30 fotogrammi al secondo in un campo visivo di 70x55 gradi. La telecamera fornisce infor-



mazioni senza pixel interpolati, con un livello di sfuocatura praticamente a zero pixel grazie, al tempo di esposizione di 50 ns.

Intel RealSense mette a disposizione un'ampia gamma di soluzioni basate sulla visione, dalle più piccole fotocamere LiDAR ad alta risoluzione al mondo, alle fotocamere di profondità stereo, che offrono opportunità di miglioramento in applicazioni come robotica, scansione 3D, riconoscimento facciale, misura e logistica.

www.intelrealsense.com

Nuovi SoC UltraScale + da Xilinx a 16 nm

Xilinx ha ampliato il proprio portafoglio di processori UltraScale+ per applicazioni che richiedono soluzioni edge intelligenti. Con fattori di forma che sono il 70% più piccoli rispetto al tradizionale packaging su scala di chip, i nuovi dispositivi Artix e Zynq UltraScale+ possono ora affrontare una gamma più ampia di applicazioni nelle applicazioni industriali, di visione artificiale, in campo medicale, broadcast, consumer, automobilistico e di rete. Con l'efficiente tecnologia a 16 nanometri, i dispositivi Artix e Zynq UltraScale+ si avvalgono del package InFO (Integrated Fan-Out) di TSMC che consente di soddisfare la necessità di applicazioni edge intelligenti offrendo densità di elaborazione elevata, prestazioni per watt e scalabilità in opzioni di packaging compatte.

La famiglia Artix UltraScale+ si basa sulla sua architettura FPGA collaudata in produzione ed è ideale per una vasta gamma di applicazioni tra cui la visione artificiale con tecnologia dei sensori avanzata, rete ad alta velocità e trasmissione video ultracompatte 8K-ready. I dispositivi Ar-

tix UltraScale+ implementano transceiver da 16 gigabit/secondo per supportare protocolli emergenti e avanzati in rete, visione e video, fornendo allo stesso tempo il calcolo DSP più elevato della sua categoria. Il SoC multiprocessor ZU1 è progettato per la connettività ai margini e per i sistemi IoT industriali e sanitari, tra cui telecamere di visione integrate, streaming AV-over-IP 4K e 8K-ready, test portatile apparecchiature, nonché applicazioni mediche e di consumo.

ZU1 è progettato per applicazioni miniaturizzate ad alta intensità di calcolo e alimentato da un sottosistema di processori multicore basato su un processore Arm eterogeneo, pur essendo in grado di migrare a footprint di pacchetti comuni per un maggiore calcolo.

Le famiglie Artix e Zynq UltraScale+ supportano l'autenticazione RSA-4096, la decrittografia AES-CGM, le contromisure DPA e l'IP di monitoraggio della sicurezza proprietario Xilinx che si adatta alle minacce alla sicurezza durante il ciclo di vita del prodotto.

www.xilinx.com





ABBONAMENTO
ANNUALE



Abbonati a Elettronica In!
Realizza i **Progetti** descritti sulla rivista,
rimani aggiornato con i nostri **Corsi**,
approfondisci la conoscenza delle
tecniche e dei componenti
più avanzati con i **Tutorial**
e le nostre **News**.



Visita: www.elettronica.in/abbonamenti



ABBONAMENTO ANNUALE PER RIVISTA DIGITALE

In **OMAGGIO** avrai
il **10% DI SCONTO** per i tuoi ordini
su **Futurashop.it**

COME ABBONARSI

→ **on-line**

compilando il modulo riportato
nella pagina "**Abbonamenti**"
del nostro sito www.elettronica.in.

→ **e-mail**

scrivendo a
abbonati@elettronica.in

→ **telefono**

telefonando a
+39-0331-752668

ABBONAMENTO ANNUALE PER RIVISTA CARTACEA

In **OMAGGIO** avrai
il **10% DI SCONTO** per i tuoi ordini
su **Futurashop.it**

GRATIS 1 VOLUME A SCELTA della collana
"L'elettronica per tutti" oppure

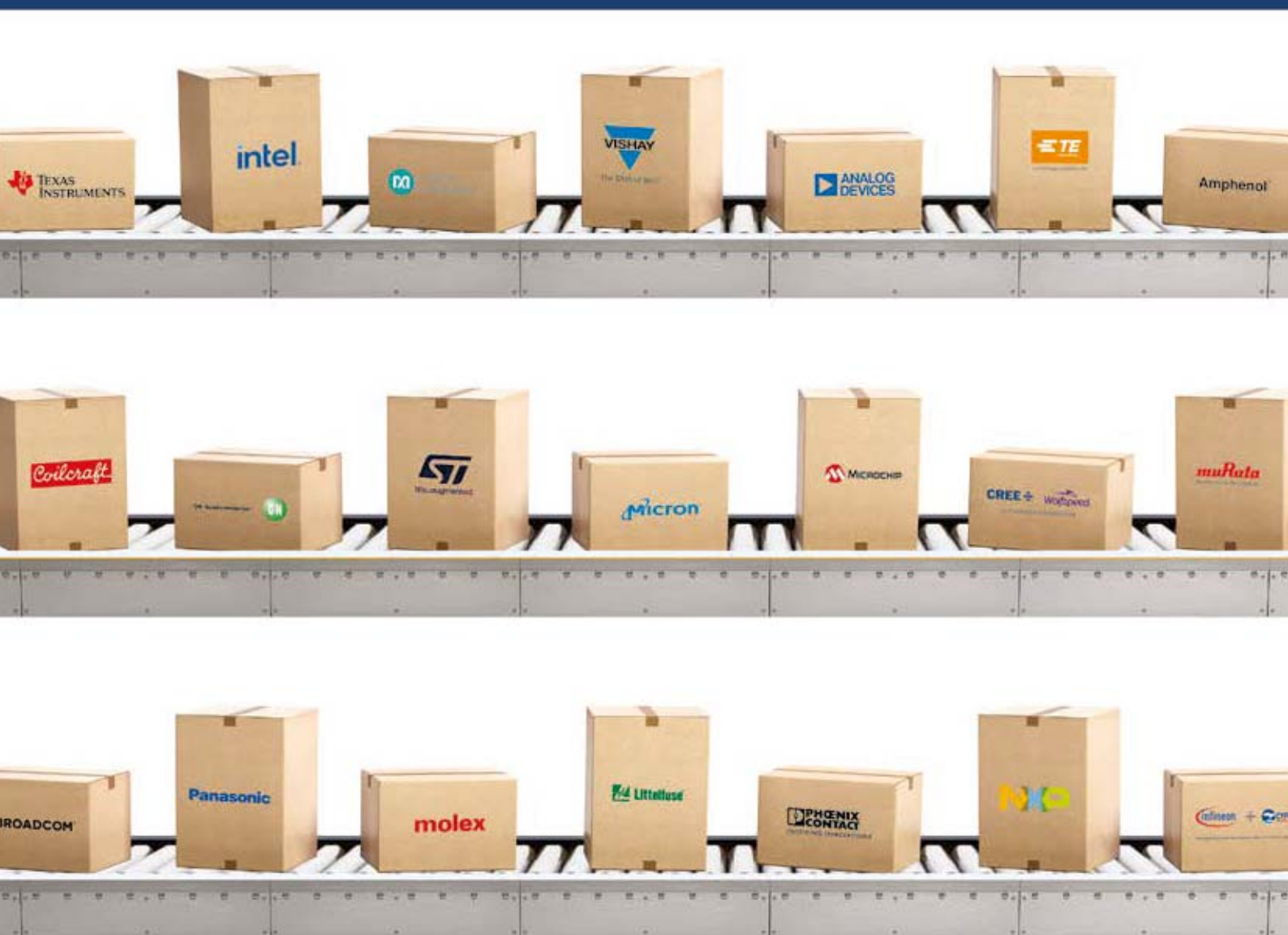
50% DI SCONTO per l'acquisto di un volume
della libreria "Elettronica In"

La possibilità di **LEGGERE GRATUITAMENTE**
la versione digitale di ELETTRONICA IN



Un'offerta sempre più grande

La più ampia selezione di semiconduttori e componenti elettronici disponibili e pronti per la spedizione



[mouser.it](https://www.mouser.it)

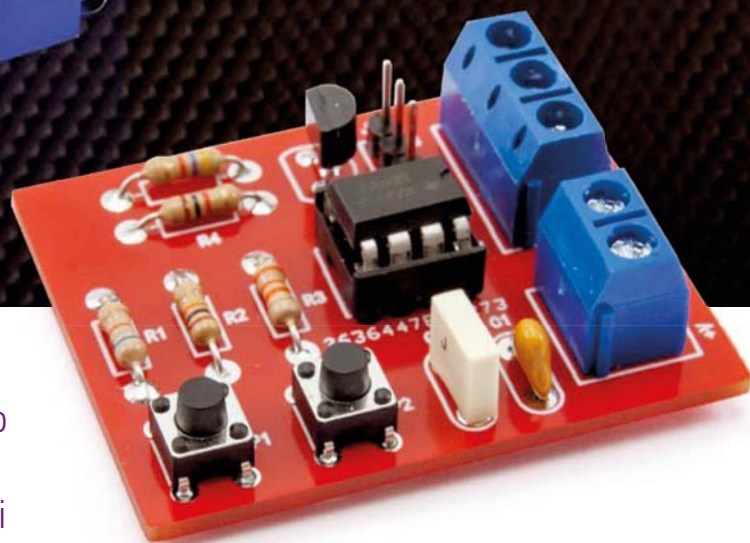


TESTER SERVOCOMANDI

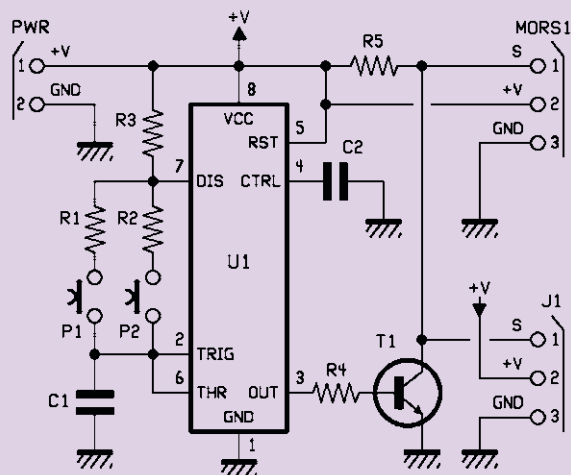


di **ROBERTO TESTI**

Consente di verificare il buon funzionamento dei classici servo da modellismo testandoli con due larghezze di impulso in corrispondenza delle quali l'alberino deve assumere posizioni specifiche.



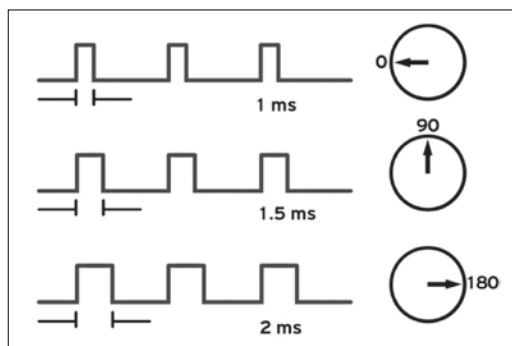
Può capitare di avere dubbi sul funzionamento di uno o più servocomandi in un macchinario o in un radiomodello (automobile o aereo radiocomandato) che non risponde come dovrebbe; in casi del genere bisogna innanzitutto identificare dove sia il problema, giacché il servo risponde a segnali di comando specifici inviati, a seconda dell'ambito di applicazione, dalla scheda di controllo o dal ricevitore del radiocomando. Diventa quindi fondamentale andare per esclusione facendo una diagnosi differenziale e



verificando se ad essere guasto è il servocomando o il circuito che gli invia gli impulsi di controllo. Si può quindi iniziare testando i servo con un circuito in grado di simulare i classici impulsi a livello TTL (0/5V) modulati in larghezza, generabili senza difficoltà con circuiti elettronici economici e di facile costruzione come quello che vi proponiamo in queste pagine. Testare il buon funzionamento del circuito di controllo richiederebbe, invece, l'utilizzo dell'oscilloscopio, con cui visualizzare la larghezza degli impulsi, per questo è meglio partire dalla verifica dei servo.

Prima di vedere com'è fatto il nostro circuito, è opportuno riassumere come funzionano i servocomandi, cui è dedicato un riquadro di approfondimento in questo articolo: tali motori integrano un circuito driver e un potenziometro (o encoder, in quelli professionali) che restituisce un feedback sulla posizione assunta dall'alberino allo scopo di stabilizzarla al valore corrispondente alla durata dell'impulso ricevuto. La corrispondenza tra durata dell'impulso e posizione angolare dell'alberino è proposta nella **Fig. 1**: 0 significa tutto ruotato a sinistra (in senso antiorario), 90 è la posizione centrale e 180 la completa rotazione in senso orario. I servo hanno una connessione standard mostrata

Fig. 1
Posizioni assunte dall'alberino del servocomando in funzione della durata dell'impulso di comando.



nella **Fig. 2**, il che rende facile collegarli e testarli, perché non c'è bisogno di andare a cercarne la piedinatura.

Ciò detto, abbiamo le informazioni che servono a capire perché il circuito che vi descriveremo è fatto come è fatto e perché funziona come spiegheremo qui di seguito.

SCHEMA ELETTRICO

Diamo dunque uno sguardo al nostro tester per servocomandi, guardandone lo schema elettrico che è davvero molto semplice, grazie all'utilizzo di un solo circuito integrato che, per sua natura, non poteva non essere l'immane timer 555. Nello specifico, il chip (U1 nello schema elettrico...) è utilizzato nella configurazione da multivibratore astabile con due reti di temporizzazione selezionabili e attivabili ciascuna mediante un pulsante dedicato, il che determina la generazione di un impulso di durata differente; per l'esattezza, P1 attiva la generazione degli impulsi più lunghi e P2 quella dell'impulso più corto. Ciascuno consente quindi di forzare il servocomando ad assumere una certa posizione.

All'uscita del 555 è stato posto un transistor NPN configurato a emettitore comune che funziona da amplificatore e che inverte il livello logico del segnale, ottenendo quello che va al pin S del connettore J1 (e corrispondente morsetto della morsettiera MORS1) e pilota il servo; ad S è collegato il collettore del transistor T1.

Per capire come funziona il circuito bisogna dare uno sguardo all'interno del 555: questo integrato contiene due comparatori aventi in comune un partitore di tensione a scala di resistenze che polarizza l'ingresso invertente dell'operazionale superiore (quello più vicino all'alimentazione positiva) e il non-invertente di quello inferiore (più vicino alla massa); dei comparatori vengono resi accessibili dall'esterno gli ingressi non-invertenti di quello superiore (i termini superiore e inferiore si riferiscono al potenziale di riferimento che ricevono dal partitore multiplo, quindi quello superiore riceve la tensione più elevata), corrispondente al piedino 6 (THR, ossia Threshold) e l'invertente di quello superiore, corrispondente al piedino 2 (TRIG, ossia Trigger).

Il nodo tra la prima e la seconda resistenza dall'alto (dal positivo di alimentazione) del partitore multiplo di riferimento dei due comparatori (alimentato, come il resto degli stadi interni al 555, dal piedino 8) viene portato all'esterno tramite il piedino 5 (CTRL, Control Voltage) e ciò permette di influen-

zare dall'esterno i potenziali di riferimento dei comparatori, così da controllare con una tensione esterna (o la connessione di resistenze verso GND o positivo di alimentazione) la frequenza di lavoro nella modalità astabile. Più esattamente, nella configurazione astabile, abbassando la tensione sul piedino 5 si accorcia il periodo della commutazione dell'uscita e quindi si incrementa la frequenza generata, viceversa lasciando salire la tensione fino al livello caratteristico (a vuoto, ossia senza alcun circuito collegato al piedino 5) o incrementandola si allunga il periodo e si riduce la frequenza prodotta.

Nel nostro caso non utilizziamo la funzione CTRL e filtriamo con il condensatore C2 la tensione al pin 5, assicurando una migliore stabilità delle tensioni di riferimento degli operazionali e la filtrazione di eventuali disturbi.

Le uscite dei comparatori entrano una nel RESET (quella del comparatore superiore) e l'altra nel SET (comparatore inferiore) di un flip-flop di tipo RS, che è un circuito logico la cui uscita diretta (Q) va a livello alto (corrispondente a circa il potenziale del piedino 8) quando il SET è a livello alto, ovvero assume lo zero logico se SET è posto a circa zero volt; l'uscita presenta un comportamento esattamente opposto quando ad essere stimolato è l'ingresso RESET, il quale, posto a livello logico alto, resetta il flip-flop, intendendo con ciò che ne azzerà lo stato di Q. Il flip-flop ha anche un'uscita complementata ($\bar{Q}$), la cui condizione logica è sempre l'inverso di quella Q; nel 555 è collegata, tramite una resistenza, alla base di un transistor NPN configurato ad open collector il cui emettitore è connesso a massa e il collettore al piedino DIS (Discharge, 7). Il transistor può sopportare una corrente di collettore di 200 mA, utilizzato in modo sink (chiuso a massa). L'uscita diretta, Q, è invece collegata al piedino OUT (3) dell'integrato attraverso un buffer interno push-pull, capace di erogare un massimo di 200 mA. Il negativo di alimentazione, vale a dire la massa di riferimento del 555 corrisponde al piedino 1, mentre al 4 è connesso il reset a logica invertita del flip-flop: questo piedino permette di resettare forzatamente il circuito dall'esterno applicando un livello logico basso.

Spiegato ciò, vediamo come il 555 funziona nel nostro circuito: è configurato come astabile che viene acceso e impostato dalla pressione di uno dei pulsanti (i due non vanno premuti insieme, altrimenti si ottiene una frequenza non utilizzabile con i servo).

Per farvi capire come funziona il circuito immagi-

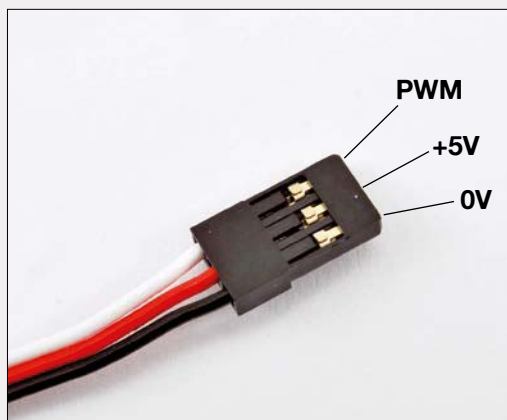


Fig. 2
Connessioni dei connettori femmina volanti dei servo per modellismo. Il PWM trasporta gli impulsi di comando (S) e lo 0 V corrisponde al negativo dell'alimentazione (GND) ovvero alla massa dell'elettronica di interna.

niamo di inserire una sola rete di temporizzazione alla volta, ovvero la sola resistenza R1 premendo P1 o la R2 premendo P2; precisiamo inoltre che fin quando non si preme nessuno dei pulsanti l'uscita del 555 è a livello alto e il collettore del T1, di conseguenza, si trova a circa 0V.

Analizziamo il funzionamento del circuito partendo dalla condizione di condensatore C1 scarico e ipotizzando di premere P1: ipotizzando C2 già carico e quindi influente sulla tensione di riferimento applicata dal partitore multiplo ai comparatori interni al 555, la condizione iniziale del circuito vede i piedini 2 e 6 a circa zero volt e quindi l'uscita del comparatore inferiore a livello alto; ciò determina il livello logico alto sull'ingresso SET del flip-flop e lo stesso sull'uscita Q (la $\bar{Q}$ si porta a zero logico lasciando interdetto il transistor collegato al DIS e quindi il condensatore libero di caricarsi). A questo punto C1 inizia a caricarsi attraverso la corrente che fluisce nella serie R1 ed R3, fino a che la tensione ai suoi capi oltrepassa la tensione di soglia del comparatore inferiore, che è pari a $1/3$ di quella applicata al piedino 8; ora l'uscita del suddetto comparatore si porta a zero logico e libera il SET del flip-flop, la cui uscita rimane comunque a livello alto fin quando la tensione ai capi su C, continuando a crescere, non si porta a un valore superiore ai $2/3$ della tensione di alimentazione del 555, ovvero alla soglia di commutazione del comparatore superiore.

Quando ciò accade, l'uscita del comparatore superiore si porta a livello logico alto e resetta il flip-flop, la cui uscita diretta passa a zero logico mentre la $\bar{Q}$ si porta a livello alto, mandando in saturazione il transistor interno collegato al piedino 7, il quale attraverso R1 scarica il condensatore C1.

Il 555 rimane in questa condizione fin quando la tensione su C1 non scende al disotto della soglia di commutazione del comparatore inferiore, allorché

CARATTERISTICHE TECNICHE

Tensione di alimentazione:
5Vcc

Corrente assorbita:
max. 35 mA

Ampiezza impulsi uscita:
4,9 V

Durata impulsi:
minimo 0,7 ms,
massimo 4,7 ms

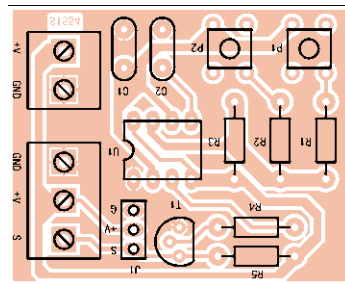
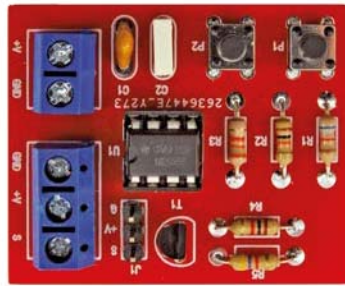
Frequenza degli impulsi:
min 85 Hz, max 334 Hz

Elenco Componenti:

R1: 68 kohm
 R2: 10 kohm
 R3: 33 kohm
 R4: 1 kohm
 R5: 4,7 kohm
 C1: 100 nF multistrato
 C2: 10 nF poliestere
 T1: BC547
 U1: ne555
 P1, P2: microswitch no

Varie:

- zoccolo 4+4
- strip maschio 3 poli
- morsetto 2 poli
- morsetto 3 poli
- stampato S1554 45x37



l'uscita di quest'ultimo torna a livello logico alto e setta il flip-flop, che riporta Q a 1 logico e /Q a zero. In questa condizione il condensatore riprende a caricarsi perché il piedino 7 torna aperto. La carica avviene nuovamente fino a 2/3 della tensione applicata tra i piedini 1 e 8 del 555, quindi il ciclo ricomincia. Per effetto di ciò abbiamo l'uscita OUT dell'U1 che si mantiene a livello alto nel periodo che serve al condensatore affinché la tensione ai suoi capi passi da 1/3 a 2/3 della tensione di alimentazione (siccome il 555 è alimentato a 9V, il passaggio è fra 3 e 6 volt, corrispondenti rispettivamente a 1/3 e 2/3 della tensione fra i pin 8 e 1) e a livello basso (zero volt circa) nel tempo che trascorre da quando i condensatori, in fase di scarica, vedono calare la tensione ai loro capi da 2/3 della tensione di alimentazione ad 1/3.

I tempi di carica e di scarica dipendono dal valore di R1 e dalla somma di R1 ed R3. Variando il rapporto fra queste due resistenze è possibile variare il duty-cycle dell'onda rettangolare fornita dal piedino 3. La durata dell'impulso d'uscita a livello alto si determina come:

$$t_h = 0,693 (R1+R3) C1$$

Invece la pausa (livello basso sul piedino 3 del 555) dura un tempo pari a:

$$t_l = 0,693 \times R1 \times C1$$

e coincide con la scarica del condensatore. La durata del periodo T è pari a $t_h + t_l$ e il duty-cycle vale:

$$dc = t_h/T$$

La frequenza prodotta dal circuito è dunque pari al rapporto:

$$f = 1/(t_h + t_l)$$

A noi comunque interessa la durata degli impulsi, perché i servomotori rispondono in base alla larghezza degli stessi e non alla frequenza; andiamo quindi a sostituire i valori nell'equazione che fornisce il t_l (durata dell'uscita OUT a livello basso) in quanto all'uscita del 555 è presente il transistor che inverte la fase del segnale e il relativo livello logico: quando OUT è a zero logico il collettore del T1 è posto a 5V dalla resistenza R5, mentre in corrispondenza del livello alto lo stesso collettore è a zero.

Per P1 abbiamo dunque una durata dell'impulso di comando sul pin S pari a:

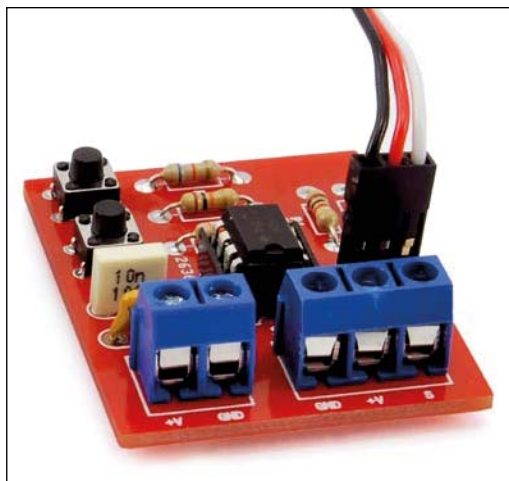
$$t_l = 0,693 \times R1 \times C1 = 0,693 \times 68 \text{ k}\Omega \times 100 \text{ nF} = 4,71 \text{ ms}$$

che corrispondono a una durata che va ben oltre ciò che serve a produrre la massima rotazione in senso orario del servocomando, mentre premendo P2 si determinano impulsi di larghezza pari a:

$$t_l = 0,693 \times R2 \times C1 = 0,693 \times 10 \text{ k}\Omega \times 100 \text{ nF} = 0,693 \text{ ms}$$

corrispondenti alla minima durata degli impulsi, quindi alla posizione di albero tutto in senso antiorario. Bene, stabilito questo vediamo come

➔ Il circuito ha due morsettiere per alimentazione e connessione servo fuori standard.



Anche detti servomotori o più semplicemente “servo”, i servocomandi sono dei motori elettrici in continua (a spazzole) che dispongono di un motoriduttore (gruppo di ingranaggi necessari a ridurre la velocità di rotazione dell'alberino) e di un alberino che aziona leve o crociere, ma anche ruote e flange, con le quali si azionano direttamente o indirettamente (tramite leve ed aste di rinvio) vari meccanismi. I servocomandi dispongono di un'elettronica interna che permette di identificare gli impulsi in arrivo dal ricevitore e posizionare di conseguenza l'alberino; la stessa elettronica provvede a rilevare la posizione raggiunta dall'alberino stesso, tramite un encoder o un potenziometro (montati coassialmente all'alberino) e a sospendere l'alimentazione del motore una volta raggiunta la posizione corrispondente alla durata degli impulsi. Questo modo di funzionamento permette di mantenere l'alberino e il leveraggio da esso azionato, in una posizione stabile.

Va ricordato che impulsi della durata di $0,7 \pm 1$ ms determinano la massima ampiezza della rotazione in senso orario dell'albero del servocomando, mentre a $1,7 \pm 2$ ms si ottiene il massimo spostamento in senso antiorario; un impulso di $1,2 \pm 1,5$ millisecondi determina la posizione centrale dell'alberino.

I servocomandi da modellismo esistono in vari tipi e possono essere, in base alla gestione del motore, distinti in analogici e digitali: sono analogici i servo “normali”, che funzionano con il segnale già descritto, ossia pilotati da impulsi di larghezza variabile tra 0,7 e circa 2ms. Invece i servocomandi digitali dialogano con il ricevitore tramite un vero e proprio protocollo di comunicazione dati (di solito proprietario per ogni casa costruttrice) che non si limitano al controllo dello spostamento,

come per gli analogici; il protocollo dati consente di inviare insieme in una sola stringa di trasmissione i comandi a più di 8 servo (limite della tecnica analogica).

Inoltre i servo digitali sono programmabili, nel senso che il loro movimento può essere programmato preventivamente per essere ripetibile. Molti servocomandi digitali sono compatibili anche con l'analogico e accettano anche un segnale a impulsi $0,7 \pm 2$ ms. Usare servo digitali con segnali analogici permette di beneficiare delle caratteristiche di precisione, potenza e stabilità di questi servo ma non le altre funzioni dei digitali.

Indipendentemente dal fatto che siano analogici o digitali, i servocomandi possono effettuare una rotazione dell'alberino minore di un giro ($0^\circ \div <360^\circ$) oppure consentire di avere la rotazione continua, come quella dei classici motori elettrici per trazione dei robot su ruote che avete visto apparire anche tra le pagine di Elettronica In.

Altre considerazioni sui servocomandi riguardano la meccanica: indipendentemente dalla categoria di appartenenza (cioè analogico o digitale) questi dispositivi si differenziano per la qualità degli ingranaggi e in generale per la costruzione della meccanica: quelli più economici hanno la meccanica in plastica più o meno resistente, mentre quelli più pregiati e costosi hanno i ruotismi e gli ingranaggi realizzati in metallo, che tipicamente è l'ottone (lega di rame e zinco) che presenta poco attrito ed è ideale per il funzionamento a lungo senza bisogno di lubrificazione con grassi vari.

I servomotori con la meccanica in metallo vengono commercialmente indicati dalla dicitura “Metal Gear” (che significa ingranaggi metallici) il che permette di riconoscerli con facilità.

➔ Vista interna

di un servocomando:

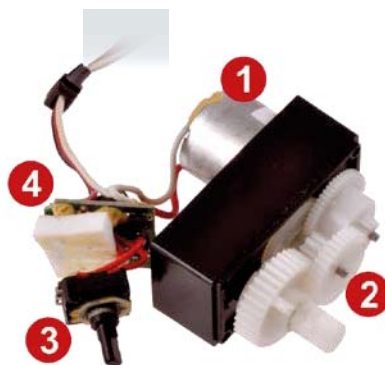
1) motore in continua;

2) motoriduttore con

alberino;

3) potenziometro;

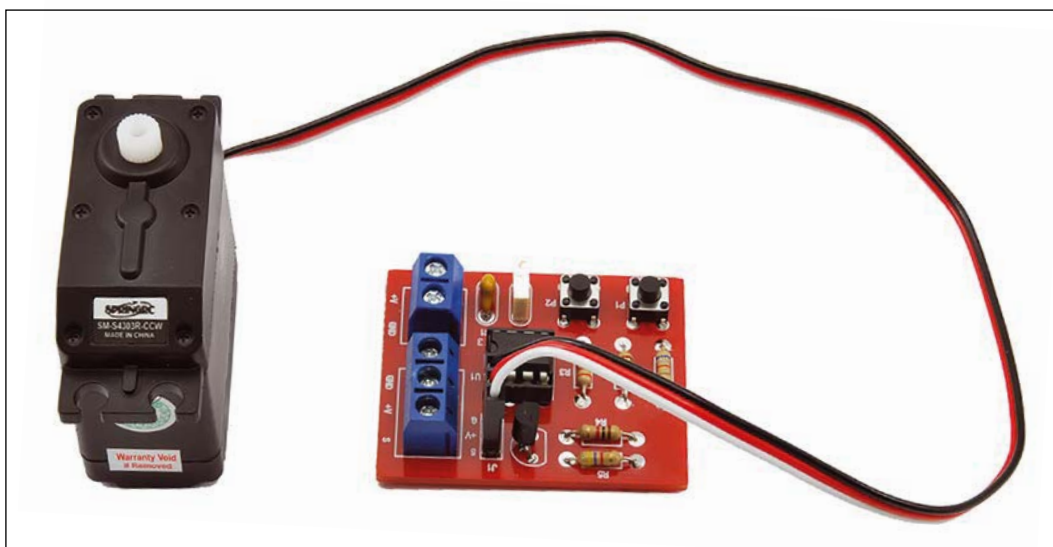
4) elettronica di controllo.



il 555 è integrato nel circuito: l'alimentazione proviene da una morsettiera e raggiunge i piedini 8 (alimentazione) e 4 (reset, disabilitato perché posto a livello alto) mentre l'uscita pilota il transistor T1, che polarizzato in base dalla resistenza R4 e sul collettore dalla R5, ribalta il segnale dell'uscita OUT portandolo dal proprio collettore alla linea S. Quest'ultima rappresenta il comando del servo ed

è ripetuta sia sul connettore (J1) strip tripolare a passo 2,54 mm (dove si può inserire il connettore volante dei servocomandi standard da modellismo dinamico) e sulla morsettiera MORS1, tripolare anch'essa, prevista per connettere servomotori di cui sono accessibili direttamente i tre fili, ovvero che abbiano un connettore con piedinatura non standard inadatto al J1.

➔ Il circuito collegato a un servo a rotazione continua.



REALIZZAZIONE PRATICA

Passiamo adesso alle note costruttive, partendo dal circuito stampato, che è un singola faccia (quindi realizzabile da tutti con facilità) e va ottenuto per fotoincisione seguendo la traccia lato rame scaricabile dal nostro sito www.elettronica.in.it oppure potete utilizzare il servizio PCBPRODUCTION dal sito futurashop.it; allegando il file Gerber sembra scaricabile dal sito della rivista.

Una volta ottenuto il circuito stampato dovete iniziare il montaggio dei componenti (tutti a montaggio tradizionale) con le resistenze e i condensatori, quindi procedendo con l'integrato 555, che potete montare su uno zoccolo dip 4+4 contatti (in tal caso inserirete l'integrato una volta saldati tutti gli altri componenti).

Passate quindi ai due pulsanti miniatura (normalmente aperti) da circuito stampato e al pin-strip per la connessione diretta ai servocomandi dotati di connettore standard; completate il montaggio con la morsettiera di alimentazione e con quella per la connessione alternativa del servo (MORS1). Per l'orientamento dei componenti polarizzati fate riferimento al disegno di montaggio visibile in queste pagine.

Per funzionare, l'intero circuito richiede un'alimentazione a tensione continua stabilizzata al valore di 5V e una corrente di almeno 200 mA (perché va considerato l'assorbimento del servocomando da testare); i 5V vanno applicati alla morsettiera PWR, con il positivo sul + e il negativo sul -.

L'alimentazione può anche essere prelevata da una batteria di pile composta da tre o quattro stilo in serie (non servono esattamente 5V perché c'è un po' di tolleranza).

CONCLUSIONI

Bene, termina qui la descrizione del tester per piccoli servomotori, che non dovrà mancare sul banco da lavoro del radiomodellista; facciamo notare che il circuito è adatto a pilotare vari tipi di servo, anche quelli più potenti della media utilizzata nel modellismo dinamico. In casi del genere, ossia se bisogna provare dei servo il cui assorbimento supera 1 ampere, suggeriamo di fornire l'alimentazione al di fuori dal circuito (quindi non attraverso il connettore J1 o la morsettiera MORS1) ma distintamente, portando dal circuito solamente il filo del segnale di controllo dal contatto S.

Un'ultima nota riguarda il test dei servocomandi a rotazione continua: il nostro circuito permette di verificare anche questi, perché premendo P1 l'alberino del servo deve mettersi a girare in senso orario, mentre con il P2 premuto si deve ottenere da esso la rotazione in senso contrario, vale a dire antiorario. □



Cosa occorre?

Il timer 555 utilizzato in questo progetto (cod. 555) è disponibile a Euro 0,40. Sul sito futurashop.it trovare molti servo per modellismo, sia classici che a rotazione continua. Il PCB può essere realizzato utilizzando il servizio di realizzazione PCB (cod. PCBPRODUCTION) disponibile sul sito futurashop.it. I prezzi si intendono IVA compresa.

Il materiale va richiesto a:

Futura Elettronica, Via Adige 11, 21013 Gallarate (VA)
Tel: 0331-799775 - <http://www.futurashop.it>

Strumenti di misura per veri professionisti!



Rilevatore da parete 3 in 1 (legno, metalli, cavi elettrici)

Rileva la presenza di cavi elettrici sotto tensione (con passaggio di corrente), strutture in legno (telai, travetti, barre, ecc), oggetti metallici (travi, telai, tubi, tondini, perni, barre in metallo). Dispone di ampio display retroilluminato, buzzer di segnalazione, pulsante di calibrazione automatica. Si spegne automaticamente dopo 3 minuti di inutilizzo. Alimentazione con batteria da 9 V (acquistabile separatamente), dimensioni 156x70x26 mm.

€29,50

Cod. FR760

Rilevatore da parete professionale (legno, metalli, cavi elettrici)

Rileva la presenza di cavi elettrici sotto tensione (con passaggio di corrente), strutture in legno (telai, travetti, barre, ecc), oggetti metallici (travi, telai, tubi, tondini, perni, barre in metallo). Dispone di ampio display LCD retroilluminato, buzzer di segnalazione, spegnimento automatico dopo 5 minuti di inutilizzo. Alimentazione tramite batteria al litio (integrata) da 3,7 volt 1500 mAh. Dimensioni 195x90x35 mm.



- ✓ Professionale
- ✓ Batteria al litio ricaricabile integrata da 3,7 V - 1500 mAh con autonomia fino a 10 ore
- ✓ Case robusto
- ✓ Prestazioni migliorate

€78,00

Cod. FR801

Misuratore di distanza fino a 80 m con LASER verde e messaggi vocali.

Metro LASER multifunzione compatto con display LCD retroilluminato, messaggi vocali, menu a icone, livella digitale, memoria interna (in grado di memorizzare fino a 100 misurazioni) e batteria ricaricabile agli ioni di litio. È in grado di effettuare le seguenti misurazioni: distanza (da 20 cm a 80 metri), area, volume, misurazione pitagorica, misurazione verticale e molte altre. Visualizza lo stato di carica delle batterie; si spegne automaticamente se inattivo (laser dopo 30 secondi, strumento dopo 2 minuti). Grado di protezione IP54. Dimensioni 120x50x26 mm.



€79,00

Cod. FR798

Misuratore di distanza LASER da 20 cm a 30 m

Compatto metro LASER con display LCD retroilluminato. Permette di misurare in modo estremamente facile e veloce la distanza (da 20 cm a 30 metri), l'area e il volume. Dispone di spegnimento automatico del LASER e del dispositivo; visualizza anche lo stato di carica delle batterie. Alimentazione con 2 batterie mini stilo da 1,5 volt tipo AAA (incluse). Dimensioni 105x34x20 mm



€27,00

Cod. FR797

Prezzi IVA inclusa

Taglia e incidi con **Cricut MAKER**



Prezzo IVA inclusa

Macchina da taglio intelligente



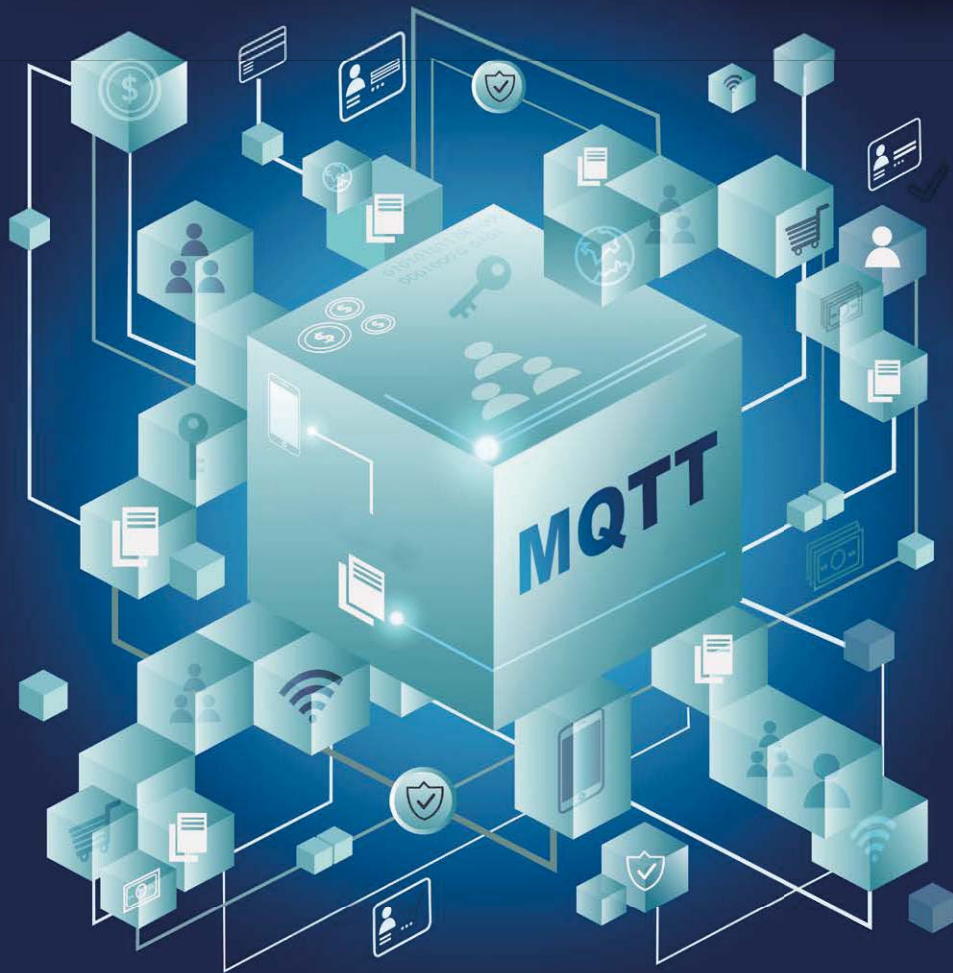
€ 439,00

cod. CRICUTMAKER

Dai vita alla fantasia e alla creatività con Cricut Maker l'innovativa macchina da taglio. Grazie a Design Space, il software di progettazione gratuito, potrai realizzare praticamente qualsiasi progetto fai-da-te, dall'arte 3D all'arredamento per la casa, gioielli, progetti di carta e molto altro ancora e inviarti direttamente a Cricut Maker dal tuo tablet, dal PC o dallo smartphone. Cricut Maker taglia in modo rapido e accurato oltre 300 materiali: tessuti, pelle, carta, cartamodelli, cuoio, fogli di balsa e tanto altro. Espandi le potenzialità di Cricut Maker utilizzando utensili dedicati, alcuni inclusi nella confezione, altri acquistabili separatamente.



IL PROTOCOLLO MQTT



dell'ing. **MATTEO DESTRO**

Conosciamo il protocollo di comunicazione dati ideale per il monitoraggio a distanza tramite sensori, prediletto dalle applicazioni IoT in virtù della capacità di funzionare con bassi data-rate.

In questi ultimi anni si è assistito a un incremento esponenziale delle applicazioni IoT (Internet of Things) le quali permettono di connettere alla rete sia oggetti di uso comune, come ad esempio il frigorifero piuttosto che una stazione meteo, oppure dispositivi utilizzati in ambienti industriali come robot, macchine utensili ecc. I dispositivi che vengono connessi alla rete hanno la necessità di inviare una serie di dati a dei server i quali hanno lo scopo di memorizzare e elaborare le informazioni ricevute. I dati generati devono essere inviati tramite un protocollo di comunicazione snello e veloce e per questo tipo di applicazioni si è pensato di sfruttare il protocollo MQTT ovvero Message Queue Telemetry Transport. Un esempio di servizio internet che accetta l'invio dei dati da elaborare tramite protocollo MQTT è il noto, e già ampiamente usato sulle pagine della rivista, Thingspeak il quale mette a disposizione una serie di servizi per l'elaborazione dei dati ri-

cevuti tramite protocollo MQTT. Il protocollo MQTT è uno standard ISO (International Organization for Standardization) e OASIS (Organization for the Advancement of Structured Information Standards) aperto ovvero un protocollo di publish e subscribe usato per ricevere e inviare messaggi tra dispositivi. Tale protocollo sfrutta la connessione TCP/IP ma tuttavia può essere implementato su qualsiasi protocollo di rete che supporti una connessione bi-direzionale senza perdita di informazioni. Data la natura del protocollo questo viene utile anche per quel tipo di applicazioni che richiedono di connettere dispositivi situati in postazioni remote con larghezza di banda limitata per la trasmissione dei dati.

LA STORIA

La prima versione del protocollo MQTT è stata scritta da Andy Stanford-Clark (IBM) e Arlen Nipper (Eurotech Inc.) nel 1999 ed è stata usata per monitorare gli oleodotti all'interno del framework SCADA (Supervisor Control And Data Acquisition). L'obiettivo degli sviluppatori era realizzare un protocollo con una buona efficienza di banda, leggero e che necessitasse di poca energia per funzionare in quanto i dispositivi che ne facevano uso erano alimentati a batteria.

Nel 2013 IBM ha rilasciato la versione 3.1 di MQTT da cui è nata una variante denominata MQTT-SN, dove SN sta per *Sensor Network*. Questa variante permette di utilizzare MQTT su dispositivi a batteria che non sfruttano il protocollo di comunicazione TCP/IP come ad esempio lo ZigBee, il Bluetooth ecc. Successivamente è stata rilasciata nell'Ottobre del 2014 la nuova revisione 3.1.1 la quale presentava lievi miglioramenti rispetto alla precedente. Infine nel Marzo del 2019 è stata rilasciata la nuova versione 5.0 la quale aggiunge diverse nuove funzionalità. Attualmente la più utilizzata è la versione 3.1.1 anche se presto o tardi si migrerà verso la nuova versione 5.0.

Storicamente le due lettere *MQ* della sigla derivano dal progetto IBM *MQSeries* sviluppato nel Dicembre del 1993 e poi rinominato nel 2002 in *WebSphere MQ* ed ancora nell'Aprile del 2014 in *IBM MQ*. Il protocollo sviluppato prevedeva dei messaggi di *publish* e *subscribe* e fu progettato esplicitamente per dispositivi con risorse limitate e con larghezza di banda ridotta.

Nelle specifiche IBM della revisione 3.1 il nome formale del protocollo doveva essere *MQ Telemetry Protocol* tuttavia si è sempre usata l'abbreviazione MQTT che tutti noi oggi conosciamo.

CONCETTI BASE DEL PROTOCOLLO MQTT

Il protocollo MQTT definisce due entità di rete differenti: una nominata Message Broker e l'altra Clients. Un broker MQTT non è altro che un Server che riceve i messaggi da dei Clients e instrada gli stessi ai destinatari. Invece un Client MQTT è un dispositivo, solitamente con a bordo un microcontrollore, che esegue una libreria MQTT appositamente scritta (in linguaggio C, python o altro) che invia dati a un broker MQTT connesso in rete. Le informazioni che devono essere inviate, ad esempio misure di temperatura, pressione, umidità ecc, vengono organizzate in topics, quindi quando si hanno dei nuovi dati a disposizione questi vengono impacchettati in un apposito pacchetto MQTT il quale viene inviato al broker a cui ci si è connessi (azione di publish). Il broker che riceve questi dati li inoltrerà a tutti quei Clients che ne hanno fatto richiesta (tramite una azione di subscribe invita dai Clients).

Chi pubblica i dati verso il broker MQTT non ha bisogno di sapere quali sono i Clients che ne stanno facendo richiesta e dove essi si trovano e viceversa chi fa la richiesta dei dati non ha nessuna necessità di sapere cosa sono e dove sono i Clients che stanno pubblicando i dati richiesti. È il broker MQTT che si occupa di instradare correttamente i dati ricevuti a chi ne ha fatto richiesta.

Se il broker MQTT riceve da un Client un messaggio per un Topic del quale al momento non ci sono richieste da parte di altri client (subscribers) allora tale messaggio e relativo Topic vengono scartati. Tuttavia chi pubblica il Topic potrebbe mandare un messaggio con settato un flag di ritenuta; in questo caso il broker MQTT è tenuto a conservare il messaggio più recente e relativo Topic. In questo modo qualora un qualsiasi client faccia richiesta di tale Topic questo sarà subito reso disponibile dal broker. Il broker MQTT può memorizzare un solo Topic per messaggio, quindi non si possono memorizzare più messaggi con lo stesso Topic.

La **Fig. 1** mette in evidenza l'interazione tra il Client A e il Client B attraverso il broker MQTT. Il Client B esegue solo operazioni di publish, in questo caso una temperatura, mentre il Client A esegue sia operazioni di subscribe che di publish. In altre parole fa richiesta al broker della temperatura messa a disposizione dal Client B e a sua volta pubblica la temperatura da lui stesso misurata.

Il Topic usato dal Client B è *temperature/roof* (temperatura tetto) mentre il Topic usato dal Client A è *temperature/floor* (temperatura pavimento). Il Client B quando ha pubblicato per la prima volta

la temperatura ha impostato anche a true il flag Retain descritto prima.

In questo modo nel momento in cui il Client A ha inoltrato richiesta tramite subscribe del Topic temperature/roof ha ricevuto anche il primo valore pubblicato dal Client B che altrimenti sarebbe andato perso. Invece la successiva pubblicazione viene girata subito al Client A perché il broker MQTT ha memorizzato il fatto che il Client A ne fece richiesta tramite un precedente subscribe.

Per non ricevere più i valori di temperatura pubblicati dal Client B il Client A dovrebbe inviare un pacchetto MQTT di unsubscribe con relativo Topic.

Un'altra peculiarità messa a disposizione dal protocollo MQTT è la seguente: un Client che si connette al broker per effettuare delle azioni di Publish al momento della connessione può impostare un messaggio di testo, il quale viene memorizzato dal Server, che verrà utilizzato in caso di disconnessione improvvisa dalla rete da parte del Client che sta eseguendo le azioni di Publish.

In conclusione si può affermare che il broker MQTT si comporta come un ufficio postale il quale non fa uso dell'indirizzo per recapitare un messaggio ma sfrutta il concetto di Topic per identificare il messaggio che contiene i dati desiderati durante le operazioni di subscribe.

Quindi qualsiasi Client che desidera ricevere i dati associati ad un determinato Topic non deve fare altro che eseguire una richiesta di subscribe al broker con riferimento al giusto Topic (se preferite in italiano possiamo tradurlo come argomento).

Inoltre esiste il concetto di uno a molti ovvero più Clients diversi possono fare richiesta al broker di ricevere i dati inerenti al medesimo Topic; viceversa più Clients possono pubblicare svariati Topic che poi sono destinati a un unico Client che ne fa richiesta, concetto di molti a uno.

Ogni Client connesso al broker è in grado sia di pubblicare un Topic sia di fare una richiesta di Topic al broker MQTT. Ad esempio un Client potrebbe pubblicare delle misure di temperatura di un processo di lavorazione e ricevere indietro, attraverso una richiesta di subscribe, dei comandi da eseguire a seconda dei dati inviati.

Per quanto riguarda la sicurezza il protocollo MQTT invia le credenziali di connessione in chiaro in formato testo e non include nessuna misura di protezione e autenticazione. Infatti con dei software come Wireshark è possibile intercettare i pacchetti MQTT inviati da un client e vederne il contenuto. Se si vuole aggiungere un grado di protezione si può farlo sfruttando TLS per criptare le informazio-

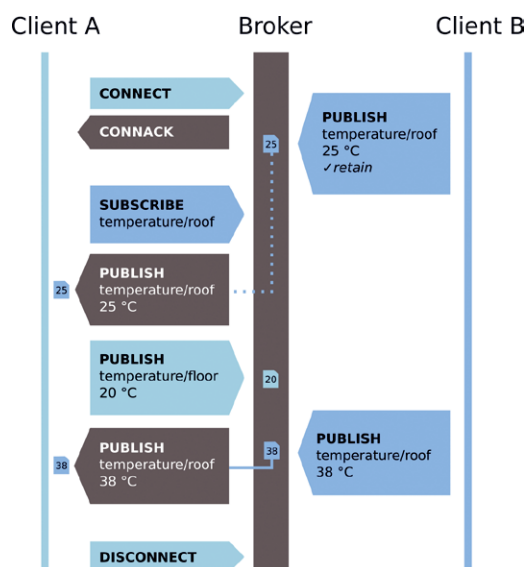


Fig. 1
Comunicazione
tra client A e
B mediata dal
broker MQTT.

ni inviate. La porta di default utilizzata da MQTT è la 1883 mentre quella da usare con TLS è la 8883. Infine possiamo riassumere alcune dei vantaggi che un broker MQTT può dare:

- elimina la vulnerabilità e l'insicurezza delle connessioni client;
- può facilmente scalare da un singolo dispositivo a centinaia di dispositivi;
- gestisce e traccia tutte le connessioni includendo credenziali di sicurezza e certificati;
- riduce lo sforzo della rete senza compromettere la sicurezza (reti cellulari o satellitari).

STRUTTURA PROTOCOLLO MQTT 3.1.1

Il protocollo MQTT lavora scambiando una serie di pacchetti di controllo e in questa sezione cercheremo di descrivere il formato di ognuno di questi pacchetti in modo da dare un'idea dei concetti che ci sono alla base del protocollo stesso.

Un pacchetto MQTT è composto da un massimo di tre parti, poste nel seguente ordine:

- Fixed Header, presente in tutti i pacchetti MQTT;
- Variable Header, presente in alcuni pacchetti MQTT;
- Payload, presente in alcuni pacchetti MQTT.

Ogni pacchetto MQTT contiene un *Fixed Header* che è composto da un minimo di due byte a un massimo di 5 byte.

Il primo byte è formato da due parti distinte:

- il nibble meno significativo contiene i flag del pacchetto;
- il nibble più significativo contiene il codice del pacchetto MQTT che si sta inviando.

Tabella 1
Questa tabella
identifica
il "Fixed Header".

Bit	7	6	5	4	3	2	1	0
Byte 1	Codice MQTT che identifica il pacchetto che si sta inviando				flag specifici per ogni codice MQTT			
Byte 2 Byte 5	Lunghezza rimanente del pacchetto che si sta inviando. Il numero di byte necessario a contenere questa informazione è variabile							

I byte successivi dal secondo al quinto esprimono la lunghezza del pacchetto MQTT che si sta inviando. Esiste un apposito algoritmo per calcolare la lunghezza dei pacchetti MQTT e quindi il numero di byte necessari a contenere tale informazione. Quanto detto può essere riassunto nella **Tabella 1**. I possibili codici dei pacchetti MQTT gestibili con la revisione 3.1.1 sono quelli proposti nella **Tabella 2**. Per quanto riguarda i flag, sono specifici per ogni codice MQTT; per alcuni codici sono riservati per usi futuri mentre per altri sono impostabili dall'utente. Se si invia un codice MQTT con i flag impostati in modo errato il server sarà costretto a chiudere la connessione. La **Tabella 3** è riassuntiva di quanto appena spiegato. Come si può osservare in essa, soltanto il codice di *Publish* permette di impostare i flag mentre tutti gli altri codici sono riservati e devono essere impostati come in **Tabella 3**. Vedremo più avanti il significato dei quattro flag configurabili. Ora vediamo come si calcola la lunghezza del pacchetto dati e quanti byte usare per questa informa-

zione. Cominciamo con il dire che se la lunghezza del pacchetto è inferiore a 128 byte allora basta un singolo byte per indicare la lunghezza del pacchetto (7 bit meno significati, il bit più significativo rimane impostato a 0). Viceversa se la lunghezza del pacchetto è superiore a 127 byte allora si ha che:

- i sette bit meno significativi contengono il dato mentre il bit più significativo viene posto a 1 per indicare che c'è almeno un altro byte ad indicare la lunghezza del pacchetto;
- il numero massimo di byte che identificano la lunghezza è quattro.

Per capire meglio conviene fare un esempio, supponiamo che il pacchetto dati da inviare abbia una lunghezza pari a 321 Byte la quale necessita di due byte per essere rappresentata. Quindi si deve così procedere:

- dividere il numero 321 per 128 ottenendo come risultato 2 e un resto pari a 65;
- quindi il primo byte da inserire sarà composto dal resto ottenuto dalla divisione più 128 (che in

Tabella 2
Codici dei pacchetti
validi per la
revisione 3.1.1
dell'MQTT.

Name	Value	Direction of flow	Description
Reserved	0	#	Riservato
CONNECT	1	Client → Server	Codice per richiesta connessione al Server da parte del Client
CONNACK	2	Server → Client	Codice di ACK al precedente pacchetto
PUBLISH	3	Client → Server or Server → Client	Codice per pubblicazione messaggio
PUBACK	4	Client → Server or Server → Client	Codice di ACK al precedente pacchetto
PUBREC	5	Client → Server or Server → Client	Codici di interscambio usati da Client e Server per consegna pacchetto assicurata. Quando si pubblica un messaggio in questa modalità si ha uno scambio di pacchetti di validazione tra Client e Server e viceversa
PUBREL	6	Client → Server or Server → Client	
PUBCOMP	7	Client → Server or Server → Client	
SUBSCRIBE	8	Client → Server	Codice per richiesta messaggio (subscribe)
SUBACK	9	Server → Client	Codice di ACK al precedente pacchetto
UNSUBSCRIBE	10	Client → Server	Codice per annullamento richiesta messaggio (unsubscribe)
UNSUBACK	11	Server → Client	Codice di ACK al precedente pacchetto
PINGREQ	12	Client → Server	Codice richiesta PING al Server
PINGRESP	13	Server → Client	Codice risposta al PING da parte del Server
DISCONNECT	14	Client → Server	Codice per richiesta disconnessione dal Server da parte del Client. In questo caso il Server non manda nessun ACK
Reserved	15	#	Riservato

Name	Fixed header flag	Bit 3	Bit 2	Bit 1	Bit 0
CONNECT	Riservato	0	0	0	0
CONNACK	Riservato	0	0	0	0
PUBLISH	Usati in MQTT 3.1.1	DUP	QoS	QoS	RETAIN
PUBACK	Riservato	0	0	0	0
PUBREC	Riservato	0	0	0	0
PUBREL	Riservato	0	0	1	0
PUBCOMP	Riservato	0	0	0	0
SUBSCRIBE	Riservato	0	0	1	0
SUBACK	Riservato	0	0	0	0
UNSUBSCRIBE	Riservato	0	0	1	0
UNSUBACK	Riservato	0	0	0	0
PINGREQ	Riservato	0	0	0	0
PINGRESP	Riservato	0	0	0	0
DISCONNECT	Riservato	0	0	0	0

Tabella 3
Flags dei
pacchetti
MQTT.

binario equivale a 0b10000000). Il risultato della somma da 193 (0b11000001);

- il secondo byte conterrà il risultato della divisione ovvero il numero 2;
- quindi in esadecimale abbiamo i seguenti due byte: 0xC1 – 0x02. Questi due byte così calcolati codificano la lunghezza del pacchetto di 321 byte;
- per decodificare il numero di byte presenti nel pacchetto, sfruttando l'esempio in corso, si può usare la seguente: $(0xC1 \& 0x7F) + 0x80 * 2$;
- il simbolo & identifica l'operazione booleana AND bit a bit mentre * indica l'operazione matematica di moltiplicazione;
- il risultato dell'operazione da 321.

Nelle applicazioni IoT difficilmente si dovranno codificare pacchetti con lunghezza superiore a 16383 byte, che codificato diventa 0xFF – 0x7F. Tuttavia per completare il discorso proviamo a codificare una lunghezza di pacchetto pari a 25589 che necessita di tre byte. Di seguito il procedimento:

- dividere il numero 25589 per 128 ottenendo come risultato 199 con resto 117;
- iterare la divisione, ovvero dividere 199 per 128 ottenendo come risultato 1 con resto 71;
- quindi il primo byte sarà dato dal resto della prima divisione più 128 ottenendo così 0xF5;
- il secondo byte sarà dato dal resto della seconda divisione più 128 ottenendo così 0xC7;
- infine l'ultimo byte conterrà il risultato dell'ultima divisione fatta ovvero 0x01;
- quindi la codifica di 25589 restituisce tre byte: 0xF5 – 0xC7 – 0x01;

- per decodificare il numero facciamo il processo inverso visto prima ovvero: $(0xF5 \& 0x7F) + 0x80 * ((0xC7 \& 0x7F) + 0x80 * 0x01)$. Il risultato dell'operazione da 25589.

Alcuni pacchetti MQTT contengono un *Variable Header* il quale è collocato tra il *Fixed Header* e il *Payload* come visto all'inizio del capitolo.

Il contenuto del *Variable Header* dipende dal tipo di pacchetto MQTT.

All'interno del *Variable Header* trova posto il *Packet Identifier* composto da due byte. Il *Packet Identifier* è presente solo per alcuni pacchetti MQTT tra cui:

- MQTT PUBLISH solo se il campo *QoS* è maggiore di zero
- MQTT PUBACK
- MQTT PUBREC
- MQTT PUBREL
- MQTT PUBCOMP
- SUBSCRIBE
- SUBACK
- UNSUBSCRIBE
- UNSUBACK

I pacchetti MQTT SUBSCRIBE, UNSUBSCRIBE e PUBLISH (con *QoS* maggiore di zero) devono contenere un *Packet Identifier* e questo valore deve essere maggiore di zero. Quindi tutte le volte che il Client deve inviare un nuovo pacchetto di questo tipo, deve assegnargli un *Packet Identifier* libero ovvero non utilizzato. Se invece il Client deve re-inviare un pacchetto MQTT già inviato deve riutilizzare lo stesso *Packet Identifier* utilizzato in precedenza.



Tabella 4
Composizione del
Fixed Header del
pacchetto MQTT
CONNECT.

Bit	7	6	5	4	3	2	1	0
Byte 1	0x01				0x00 – Reserved			
Byte 2 Byte 5	La lunghezza è data dai 10 byte del <i>Variable Header</i> più la lunghezza del <i>Payload</i> che è variabile							

Un *Packet Identifier* ritorna disponibile per il suo utilizzo nel momento in cui il Client ha processato il relativo pacchetto di ACK. Quindi PUBACK per i pacchetti PUBLISH inviati con QoS uguale a uno o PUBCOMP per i pacchetti PUBLISH inviati con QoS uguale a due. Invece per i pacchetti SUBSCRIBE e UNSUBSCRIBE si deve fare riferimento ai corrispondenti SUBACK e UNSUBACK. Tutto ciò vale anche per il Server quando è lui a inviare i pacchetti MQTT di cui sopra. Il pacchetto PUBLISH con QoS uguale a zero *NON* contiene nessun *Packet identifier*. Infine alcuni pacchetti MQTT possono contenere una sezione detta *Payload* ed è la parte finale dell'intero pacchetto. Il Payload è previsto per i pacchetti MQTT CONNECT, SUBSCRIBE, SUBACK e UNSUBSCRIBE, mentre per il PUBLISH è opzionale.

CODIFICA COMANDI MQTT

Prima di poter inviare i pacchetti del protocollo MQTT è necessario instaurare una comunicazione TCP/IP con il Server e solo dopo si può iniziare a inviare i pacchetti dati con il protocollo MQTT. Il primo pacchetto MQTT che analizzeremo è *CONNECT*; un Client può inviare questo pacchetto una sola volta a meno che la comunicazione non sia stata chiusa o interrotta. Il pacchetto *CONNECT* è composto da un *Fixed Header*,

da un *Variable Header* e da un *Payload*. Il *Fixed Header* è composto come mostrato nella **Tabella 4**. Al Fixed Header segue il Variable Header che è formato da un Protocol Name, da un Protocol Level, dai Connect flag e infine dal Keep Alive. La **Tabella 5** riassume quanto spiegato. Per quanto riguarda il byte 8, ossia i Flag di connessione, vale quanto segue:

- UN è lo User Name Flag. Questo bit viene usato per indicare se nel Payload è presente la stringa user name. Se settato a 1 stringa user name presente nel Payload se settato a 0 non presente;
- P è il Password Flag. Questo bit viene usato per indicare se nel Payload è presente la stringa password. Dipende dallo User Name Flag, se questo è settato a 0 allora anche questo bit deve essere settato a 0 viceversa se il bit User Name Flag è settato a 1 allora questo bit può essere settato sia a 1 (password presente nel Payload) che a 0 (password non presente nel Payload);
- WR è il Will Retain Flag. Questo bit viene usato per indicare se il messaggio Will Message deve essere conservato dal Server nel momento in cui viene pubblicato. Quindi se il bit Will Flag è settato a 0 allora anche il bit Will Retain deve essere settato a 0 e quindi il messaggio non viene

Tabella 5
Composizione del
Variable Header del
pacchetto MQTT
CONNECT.

Description		7	6	5	4	3	2	1	0
Protocol Name									
Byte 1	Lunghezza (MSB – LSB)	0	0	0	0	0	0	0	0
Byte 2		0	0	0	0	0	1	0	0
Byte 3		0	1	0	0	1	1	0	1
Byte 4		0	1	0	1	0	0	0	1
Byte 5		0	1	0	1	0	1	0	0
Byte 6		0	1	0	1	0	1	0	0
Protocol Level									
Byte 7	Livello 4	0	0	0	0	0	1	0	0
Connect flag									
		UN	P	WR	WQoS		W	CS	RSV
Byte 8	flag di connessione	x	x	x	x	x	x	x	0
Keep Alive									
Byte 9	Intervallo di tempo espresso in secondi ⁸					Keep Alive MSB			
Byte 10						Keep Alive LSB			

conservato. Viceversa se il bit Will Flag è settato a 1 e anche il bit Will Retain è settato a 1 allora il Server deve conservare il messaggio dopo la sua pubblicazione;

- WQoS è il Will QoS Flag. Questi due bit specificano il livello QoS che deve essere usato quando si pubblica il messaggio Will Message. Se il bit Will Flag è settato a 0 allora anche questi due bit devono essere settati a 0. Se invece il bit Will Flag è settato a 1 allora questi due bit possono essere settati a 0, 1 o 2 ma non si può impostare il valore 3;
- W è il Will Flag. Se questo bit è settato a 1 indica che, se la connessione viene accettata dal Server, il messaggio Will Message deve essere memorizzato dal Server e associato alla connessione. Il messaggio memorizzato viene pubblicato quando la connessione viene persa a meno che tale messaggio non sia stato cancellato dal Server o se la connessione è stata chiusa con una procedura pulita;
- CS è il Clean Session Flag. Questo bit viene usato per controllare lo stato della sessione. Se il bit è settato a 0 il server deve riprendere la comunicazione con il Client in base allo stato della corrente sessione. Se non c'è nessuna sessione associata all'identificativo del Client allora il Server deve creare una nuova sessione. Se invece il bit è settato a 1 sia il Client che il Server devono scartare la precedente sessione e iniziarne una nuova. Per i dettagli consultare la guida OASIS MQTT Version 3.1.1;
- RSV è un flag Reserved.

Andiamo oltre e vediamo il parametro Keep Alive: indica un intervallo di tempo espresso in secondi ed è il massimo intervallo di tempo che può intercorrere tra un pacchetto e il successivo. Quindi è responsabilità del Client assicurarsi che l'intervallo di tempo tra l'invio dei pacchetti non ecceda quello impostato. In assenza di pacchetti MQTT da inviare il Client deve inviare il pacchetto MQTT PINGREQ. Tale pacchetto può essere inviato in qualsiasi momento e può anche essere usato per capire se la connessione è attiva e il Server sta lavorando. Se il Server non riceve nessun pacchetto MQTT da parte del Client dopo che è trascorso un tempo pari a una volta e mezzo il tempo di *Keep Alive* allora deve procedere alla disconnessione dello stesso. Il *Payload* del pacchetto in esame è composto da una serie di campi distinti. Per alcuni di essi la presenza nel *Payload* è determinata dai flag impostati nel *Variable Header*; di seguito elenchiamo i campi e

il loro significato:

- *Client Identifier*: non è altro che l'ID di identificazione del Client verso il Server ed è univoco per ogni Client. Questo campo è obbligatorio ed è quindi sempre presente nel *Payload* del pacchetto MQTT CONNECT. Questo campo è codificato tramite stringa UTF-8. La lunghezza dell'ID può variare da 1 a un massimo di 23 caratteri. Sono accettati numeri e lettere minuscole e maiuscole. Tutti gli altri caratteri non sono ammessi. In testa alla stringa ci sono sempre due byte che indicano la lunghezza della stringa;
- *Will Topic*: se il flag *Will Flag* è settato a 1 allora il successivo campo sarà la stringa *Will Topic*. Anch'essa è codificata tramite UTF-8 e in testa alla stringa ci sono due byte che ne indicano la lunghezza;
- *Will Message*: se il flag *Will Flag* è settato a 1 allora il successivo campo sarà *Will Message* ma in questo caso il messaggio è codificato come una sequenza di byte. In testa ci sono due byte che ne indicano la lunghezza;
- *User Name*: se il flag *User Name* è settato a 1 allora il successivo campo sarà la stringa *User Name* codificata in formato UTF-8. In testa alla stringa due byte indicante la sua lunghezza;
- *Password*: se il flag *Password* è settato a 1 allora il successivo campo sarà la stringa *Password* codificata in formato UTF-8. In testa alla stringa due byte indicante la sua lunghezza.

Con questo abbiamo terminato la descrizione del pacchetto MQTT CONNECT. Il Server che riceve tale pacchetto deve validarlo e in caso di anomalia chiudere la connessione senza inviare il pacchetto MQTT *CONNACK*.

Se la validazione va a buon fine il Server deve:

1. se il Client ha un ID che è già connesso al Server, quest'ultimo disconnetterà il client attualmente connesso;
2. dovrà eseguire il processo di *Clean Session*;
3. dovrà riconoscere il pacchetto dati ricevuto e per fare ciò dovrà preparare il pacchetto MQTT *CONNACK* contenente un codice di ritorno pari a zero (se diverso da zero vuol dire che il Client non è stato riconosciuto dal Server in modo corretto);
4. invierà il pacchetto MQTT *CONNACK* al Client e inizierà a monitorare il tempo di *Keep Alive*.

Il secondo pacchetto MQTT che analizziamo è *CONNACK*, ossia il pacchetto che invia il Server in risposta al pacchetto *CONNECT* inviato dal Client. Quindi il primo pacchetto MQTT inviato dal Server



→ **Tabella 6**
Fixed Header del
pacchetto MQTT
CONNACK

Bit	7	6	5	4	3	2	1	0
Byte 1	0x02				0x00 – Reserved			
Byte 2	La lunghezza rimanente del pacchetto è di soli 2 byte							

al Client deve essere *CONNACK*. Se il Client non riceve dal Server questo pacchetto in un tempo ragionevole dovrebbe chiudere la connessione. La definizione di tempo ragionevole dipende dall'applicazione e dall'infrastruttura di comunicazione. Il pacchetto MQTT *CONNACK* è formato da un *Fixed Header*, da un *Variable Header* ma non da un *Payload* infatti ne è privo.

Il *Fixed Header* è composto come in **Tabella 6**.

Il *Variable Header* è invece composto come mostrato nella **Tabella 7**. Il byte 1 contiene i flag del pacchetto e solo il flag al bit 0 è utilizzato ed è chiamato *Session Present Flag*. Se il Server accetta una connessione con impostato il flag *Clean Session* a 0 allora il valore da impostare nel flag *Session Present Flag* dipende dal fatto che il Server abbia già memorizzato una sessione con l'ID del Client. Se il Server ha già una sessione memorizzata allora deve settare il flag a 1 altrimenti deve settare il flag a 0. Il byte 2 contiene i codici di risposta da parte del Server alla richiesta di connessione da parte del Client. Di seguito i possibili codici di risposta:

- 0x00 = connessione accettata;
- 0x01 = connessione rifiutata per versione protocollo non corretto (il Server non supporta il livello del protocollo MQTT richiesto dal Client);
- 0x02 = connessione rifiutata per identificativo rigettato; l'ID del Client è nel formato UTF-8 ma non permesso dal Server;
- 0x03 = connessione rifiutata per Server non disponibile (la connessione di rete è stata stabilita ma il servizio MQTT non è disponibile);
- 0x04 = connessione rifiutata per User Name o Password errata;

- 0x05 = connessione rifiutata - non autorizzato; il Client non è autorizzato a connettersi;
- 0x06 ÷ 0xFF = riservati per sviluppi futuri.

Come accennato prima, questo pacchetto MQTT non ha il *Payload*.

Il terzo pacchetto MQTT da analizzare è *PUBLISH*, il quale viene utilizzato dal Client per inviare un messaggio al Server (ad esempio i dati di misura di temperatura, umidità e pressione da pubblicare) oppure viceversa è il Server che invia dei dati al Client che ne ha fatto richiesta attraverso una operazione di *SUBSCRIBE*.

Il pacchetto MQTT *PUBLISH* è formato da un *Fixed Header*, da un *Variable Header* e dal *Payload*. Il *Fixed Header* è composto come in **Tabella 8**.

Riguardo al Byte 1 vale quanto segue:

- DUP è un flag che serve per indicare se è il primo tentativo di invio del pacchetto dati. Se il flag è settato a 0 indica che è il primo invio del pacchetto altrimenti è un re-invio dello stesso;
- QoS è un campo composto da due bit, indicante il grado di garanzia di consegna del pacchetto (ci sono tre livelli di garanzia: QoS = 0 → la consegna del pacchetto avviene al massimo una volta; QoS = 1 → la consegna del pacchetto avviene almeno una volta; QoS = 2 → la consegna del pacchetto avviene esattamente una volta);
- RETAIN; se questo flag è settato a 1 durante l'invio del pacchetto *PUBLISH* dal Client al Server, indica al Server che deve memorizzare il messaggio ricevuto e il valore del campo QoS dello stesso cosicché questo messaggio possa essere consegnato a chi ne farà richiesta tramite

→ **Tabella 7**
Struttura del
Variable Header del
pacchetto MQTT
CONNACK

Bit	Description	7	6	5	4	3	2	1	0
Connect Acknowledge flag		Reserved							SP
Byte 1		0	0	0	0	0	0	0	x
Connect Return Code									
Byte 2		x	x	x	x	x	x	x	x

→ **Tabella 8**
Struttura del Fixed
Header del pacchetto
PUBLISH.

Bit	7	6	5	4	3	2	1	0
Byte 1	0x03			DUP		QoS level		RETAIN
Byte 2	La lunghezza rimanente del pacchetto è di soli 2 byte							

Bit	7	6	5	4	3	2	1	0
Byte 1	0x04				0x00 – Reserved			
Byte 2	La lunghezza rimanente del pacchetto è di soli 2 byte							

Tabella 9
Struttura del
Fixed Header
del Pacchetto
MQTT
PUBACK

una operazione di SUBSCRIBE; se il flag è settato a 0 indica al Server che non deve memorizzare il messaggio ricevuto e non deve rimuovere o sovrascrivere qualsiasi messaggio attualmente memorizzato.

La lunghezza del pacchetto è data dalla lunghezza del campo *Variable Header* più la lunghezza del campo *Payload*.

Il *Variable Header* del pacchetto PUBLISH è così composto:

- 1 *Topic Name* → Identifica il canale sul quale si vuole pubblicare i dati presenti nel *Payload*. Il *Topic Name* è codificato in formato stringa UTF-8;
- 2 *Packet Identifier* → Questo campo è presente solo se *QoS* è settato a 1 o 2. Il *Packet Identifier* è un codice univoco a 16 bit.

Infine il campo *Payload* il quale contiene il messaggio che sta per essere pubblicato. Il contenuto e il formato dei dati dipendono dall'applicazione. Il Server che riceve il pacchetto MQTT PUBLISH deve rispondere come segue:

- nessuna risposta se il campo *QoS* è settato a 0;
- rispondere con il pacchetto MQTT PUBACK se il campo *QoS* è impostato a 1;
- rispondere con il pacchetto MQTT PUBREC se il campo *QoS* è impostato a 2.

A questo punto analizziamo i due possibili pacchetti di risposta all'invio del pacchetto MQTT PUBLISH. Il primo dei quali è PUBACK (Publish acknowledgment). Tale pacchetto viene inviato come risposta al pacchetto MQTT PUBLISH con campo *QoS* impostato a 1. Il suo *Fixed Header* è composto come in **Tabella 9**.

Il campo *Variable Header* contiene soltanto il codice *Packet Identifier* a 16 bit ed è composto da due byte; il Byte 1 contiene i bit di peso maggiore (MSB) del *Packet Identifier* e il Byte 2 contiene invece gli LSB. Questo pacchetto non prevede il campo *Payload*. Il secondo possibile pacchetto di risposta al pacchetto MQTT PUBLISH inviato è PUBREC (Publish Received). Tale pacchetto viene inviato come risposta al pacchetto MQTT PUBLISH con campo *QoS* impostato a 2. Questa risposta identifica la fase 1 del processo di risposta e validazione pacchetto pubblicato. Il suo *Fixed Header* è composto come

in **Tabella 9**, quindi da 2 byte, ma con la differenza che i bit 4÷7 del Byte 1 valgono 0x05.

Come per il precedente pacchetto il campo *Variable Header* contiene soltanto il codice *Packet Identifier* a 16 bit. Anche in questo caso non è previsto il campo *Payload*.

Il Client, quando riceve il pacchetto MQTT PUBREC deve rispondere con il pacchetto MQTT PUBREL (Publish Release). Siamo quindi nella fase del processo di risposta e validazione del pacchetto pubblicato. Il suo *Fixed Header* è composto come quello di PUBACK (**Tabella 9**) e di PUBREC, quindi dai soliti 2 byte, però la seconda parte del Byte 1 (bit 4÷7) è 0x06.

Come per il precedente pacchetto il campo *Variable Header* contiene soltanto il codice *Packet Identifier* a 16 bit. Anche in questo caso non è previsto il campo *Payload*.

Infine il Server quando riceve il pacchetto MQTT PUBREL deve rispondere con il pacchetto MQTT PUBCOMP (Publish Complete). Siamo quindi nella fase 3 del processo di risposta e validazione del pacchetto pubblicato. Il suo *Fixed Header* è composto come i precedenti (sempre 2 byte) ma la prima parte del Byte 1 vale 0x07.

Come per il precedente pacchetto il campo *Variable Header* contiene soltanto il codice *Packet Identifier* a 16 bit. Anche in questo caso non è previsto il campo *Payload*. Con la fase 3 si conclude l'invio del pacchetto MQTT PUBLISH con campo *QoS* settato a 2 (**Fig. 2**).

Passiamo ora a descrivere il pacchetto MQTT SUBSCRIBE il quale viene inviato dal Client al Server per creare una o più richieste di Subscription. In pratica il Client con questo pacchetto MQTT dice al Server quali sono i Topics di suo interesse e di conseguenza il Server dovrà inviarglieli ogni qual volta ci saranno degli aggiornamenti riguardanti tali Topics (la **Fig. 3** mostra una situazione in cui un Client pubblica un Topic: in questo caso una temperatura, e tre Client ne fanno richiesta tramite il pacchetto MQTT SUBSCRIBE). Il Server ottempererà a queste richieste inviando un pacchetto MQTT PUBLISH con campo *QoS* opportunamente configurato. Il pacchetto MQTT SUBSCRIBE è composto da un *Fixed Header*, da un *Variable Header* e da un *Payload*. Il suo *Fixed Header* è composto come propone la **Tabella 10**.



Tabella 10
Struttura del
Fixed Header
del pacchetto
SUBSCRIBE.

Bit	7	6	5	4	3	2	1	0
Byte 1	0x08				0x02 – Reserved			
Byte 2	La lunghezza è data dai 2 byte del <i>Variable Header</i> più la lunghezza del <i>Payload</i> che è variabile							

Come per il precedente pacchetto, il campo *Variable Header* contiene soltanto il codice *Packet Identifier* a 16 bit.

Infine il *Payload* del pacchetto in esame il quale serve al Client per fare una lista dei Topics di interesse, questa lista prende il nome di *Topic Filters*. La suddetta lista è codificata come stringa in formato UTF-8. Quindi un *Topic Filters* presente nel *Payload* sarà codificato come proposto nella **Tabella 11**.

Come si può osservare, in testa abbiamo due byte per definire la lunghezza della stringa che identifica il *Topic*, segue la stringa stessa, e successivamente un byte con i flag. Di questi flag si può impostare il solo campo *QoS*. Tutti gli altri bit risultano riservati

per scopi futuri. Questa struttura può essere ripetuta più volte nel *Payload*.

Nel momento in cui il Server riceve un pacchetto MQTT *SUBSCRIBE* da un Client questo deve rispondere con un pacchetto MQTT *SUBACK* e tale pacchetto deve avere lo stesso codice *Packet Identifier* usato dal Client.

Il pacchetto MQTT *SUBACK* inviato dal Server al Client deve contenere un codice di ritorno per ogni coppia *Topic Filters* – *QoS*. Questo codice di ritorno deve anche indicare il massimo *QoS* che è garantito per tale Subscription oppure ritornare un codice di failure.

Quindi il pacchetto MQTT *SUBACK* sarà formato da un *Fixed Header*, da un *Variable Header* e da un *Payload* che contiene tutti i codici di ritorno per ogni *Topic Filters* inviato con il pacchetto di Subscribe.

Il suo *Fixed Header* è composto come in **Tabella 12**. Come per il precedente pacchetto, il campo *Variable Header* contiene soltanto il codice *Packet Identifier* a 16 bit.

Infine il *Payload* che contiene la lista dei codici di ritorno. Ogni codice di ritorno corrisponde a un *Topic Filter* ricevuto dal Server con il pacchetto MQTT *SUBSCRIBE*; ne deriva che l'ordine dei codici di ritorno inseriti nel *Payload* deve essere lo stesso dei *Topic Filter* ricevuti dal Server. Quindi il formato del *Payload* prevederà un Return Code i cui bit saranno,

Fig. 2
Schematizzazione
del QoS dell'MQTT.

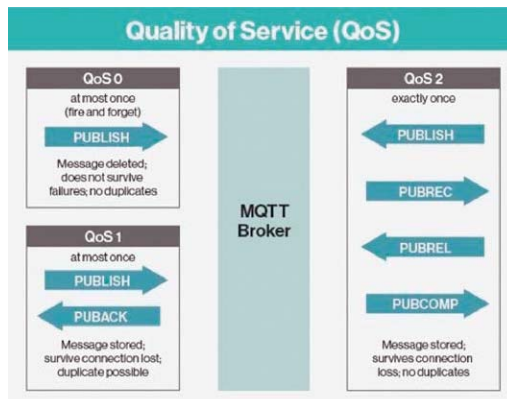
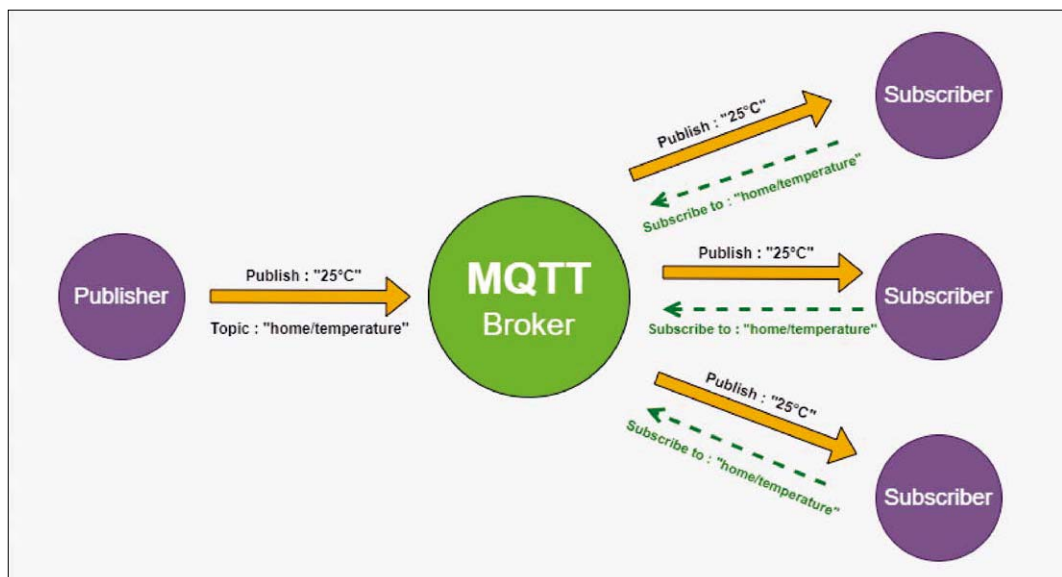


Fig. 3
Funzionamento del
protocollo MQTT.



Description	7	6	5	4	3	2	1	0
Topic Filter								
Byte 1	Length MSB							
Byte 2	Length LSB							
Byte 3...N	Topic Filter (String UTF-8)							
Request QoS								
	Reserved						QoS	
Byte N+1	0	0	0	0	0	0	x	x

Tabella 11
Topic Filters
per i pacchetti
MQTT
SUBSCRIBE.

Bit	7	6	5	4	3	2	1	0
Byte 1	0x09				0x00 – Reserved			
Byte 2	La lunghezza è data dai 2 byte del <i>Variable Header</i> più la lunghezza del <i>Payload</i> che è variabile							

Tabella 12
Fixed Header
del pacchetto
MQTT
SUBACK.

nell'ordine dallo 0 al 7: xx00000x (le "x" identificano i bit da settare per formare i codici di ritorno); in pratica avrà sempre i bit 2+6 a 0 logico. Quindi i codici di ritorno ammessi sono:

- 0x00 → Success – Massimo QoS acconsentito 0
- 0x01 → Success – Massimo QoS acconsentito 1
- 0x02 → Success – Massimo QoS acconsentito 2
- 0x80 → Failure

Se con il pacchetto MQTT *SUBSCRIBE* si indicano al Server i *Topic* di interesse con il pacchetto MQTT *UNSUBSCRIBE* si dice al Server quali *Topic* non si vuole più ricevere. Il pacchetto MQTT *UNSUBSCRIBE* è formato da un *Fixed Header*, da un *Variable Header* e da un *Payload* che contiene i *Filter Topics* di cui non si ha più interesse. Il suo *Fixed Header* è composto come in **Tabella 13**.

Come per il precedente pacchetto il campo *Variable Header* contiene soltanto il codice *Packet Identifier* a 16 bit. Infine il *Payload* che contiene la lista dei *Topic Filters* che il Client non desidera più ricevere dal Server. Anche in questo caso i *Topic Filters* sono codificati come stringa in formato UTF-8. Affinché il Server possa rimuovere i *Topic* non più di interesse per un dato Client, ci deve essere una corrispondenza uno a uno tra il *Topic* memorizzato dal Server e il *Topic* che si vuole eliminare altrimenti questo non sarà possibile. Se invece questa associazione va a buon fine allora il *Topic* corrispondente verrà eliminato e il Client non ne riceverà più i dati. Al pacchetto MQTT *UNSUBSCRIBE* il Server risponde

sempre con il pacchetto MQTT *UNSUBACK* il quale deve avere lo stesso codice *Packet Identifier*. Questo pacchetto MQTT verrà usato come risposta anche se nessun *Topic* è stato rimosso dal Server.

Parliamo quindi del pacchetto MQTT *UNSUBACK* il quale viene inviato dal Server al Client ed è composto da un *Fixed Header* e da un *Variable Header*. Non è presente il *Payload*. Il suo *Fixed Header* è composto come in **Tabella 14**. Come per il precedente pacchetto il campo *Variable Header* contiene soltanto il codice *Packet Identifier* a 16 bit. Nessun *Payload* è richiesto.

Rimangono da descrivere ancora tre pacchetti MQTT che vanno a completare tutta la rassegna. Tra questi abbiamo il pacchetto MQTT *PINGREQ* e la relativa risposta da parte del server *PINGRESP* e il pacchetto MQTT *DISCONNECT* per sconnettere il Client dal Server.

Il pacchetto MQTT *PINGREQ* viene inviato dal Client al Server ed è usato per:

- indicare al Server che il Client è connesso anche se non sta inviando altri pacchetti MQTT;
- verificare che il Server sia attivo;
- verificare che la connessione di rete sia attiva e funzionante.

Questo pacchetto MQTT ha solo il *Fixed Header* ed è composto come proposto nella **Tabella 15**.

Il pacchetto MQTT *PINGRESP* viene inviato dal Server al Client in risposta al pacchetto MQTT descritto sopra. Anch'esso presenta solo il *Fixed Header* ed è

Bit	7	6	5	4	3	2	1	0
Byte 1	0x0A				0x02 – Reserved			
Byte 2	La lunghezza è data dai 2 byte del <i>Variable Header</i> più la lunghezza del <i>Payload</i> che è variabile							

Tabella 13
Fixed Header
del pacchetto
UNSUBSCRIBE.



Tabella 14
Fixed Header del
pacchetto UNSUBACK

Bit	7	6	5	4	3	2	1	0
Byte 1	0x0B				0x00 – Reserved			
Byte 2	La lunghezza rimanente del pacchetto è di soli 2 byte							

Tabella 15
Struttura del
pacchetto PINGREQ

Bit	7	6	5	4	3	2	1	0
Byte 1	0x0C				0x00 – Reserved			
Byte 2	La lunghezza rimanente del pacchetto è 0 byte							

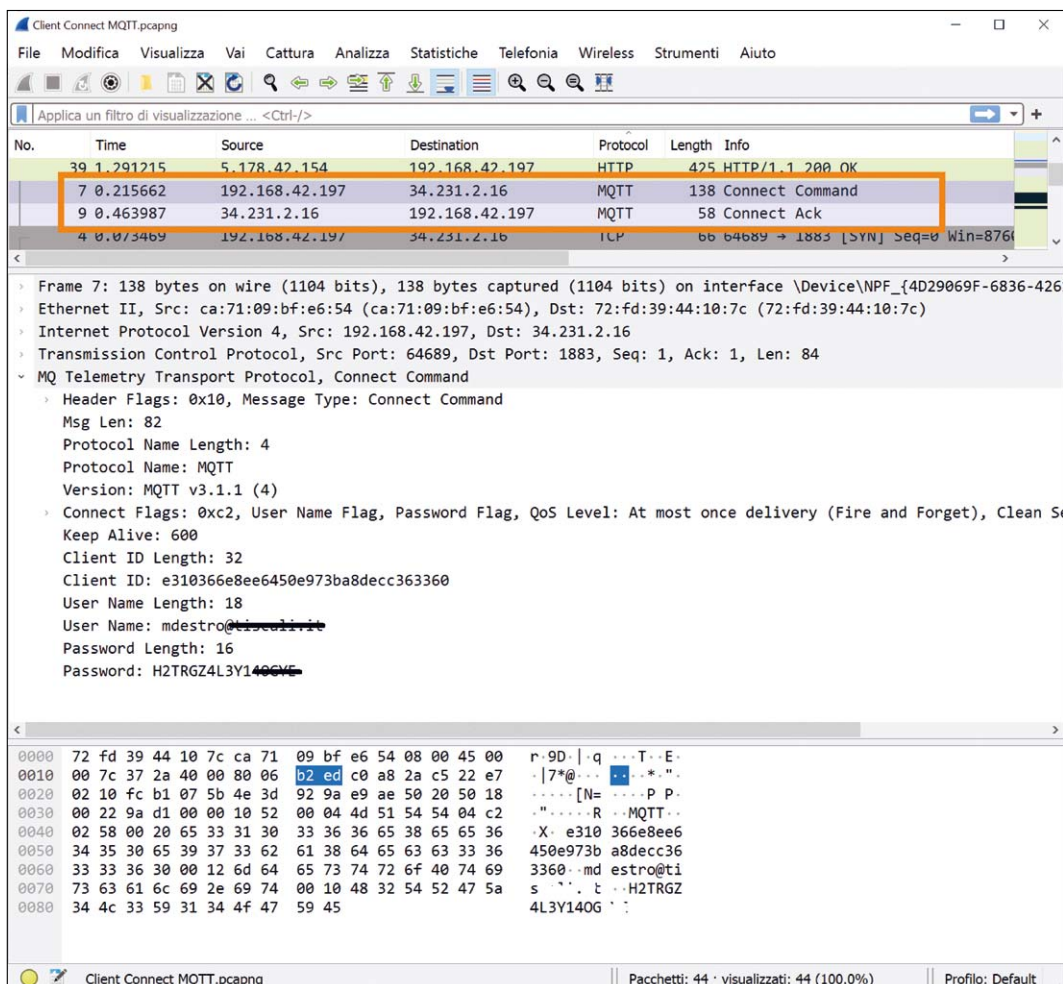
composto come PINGREQ, con la differenza che nel Byte 1 l'insieme dei bit dal 4 al 7 vale 0x0D. Infine vediamo il pacchetto MQTT *DISCONNECT*, il quale viene inviato dal Client al Server al fine di disconnettersi da quest'ultimo. Con questo comando viene effettuata una disconnessione pulita. Anch'esso presenta solo il *Fixed Header* ed è strutturalmente come i due precedenti, con la differenza che i bit 4÷7 del Byte 1 contengono il valore 0x0E. Bene, con la descrizione di quest'ultimo pacchetto abbiamo terminato la trattazione dei pacchetti

MQTT resi disponibili nella revisione 3.1.1.

ALCUNI ESEMPI DI PACCHETTI MQTT

Per fissare gli argomenti trattati facciamo qualche esempio sfruttando il noto software *Wireshark* per intercettare i pacchetti MQTT inviati dal Client. La **Fig. 4** mostra l'invio del pacchetto MQTT *CONNECT*, dal Client al Server, seguita dal pacchetto MQTT *CONNACK* inviata dal Server al Client (sezione cerchiata di arancione nella parte alta della figura). Nella parte bassa si vedono in dettaglio i dati pre-

Fig. 4
Invio dei pacchetti
CONNECT e
CONNACK.



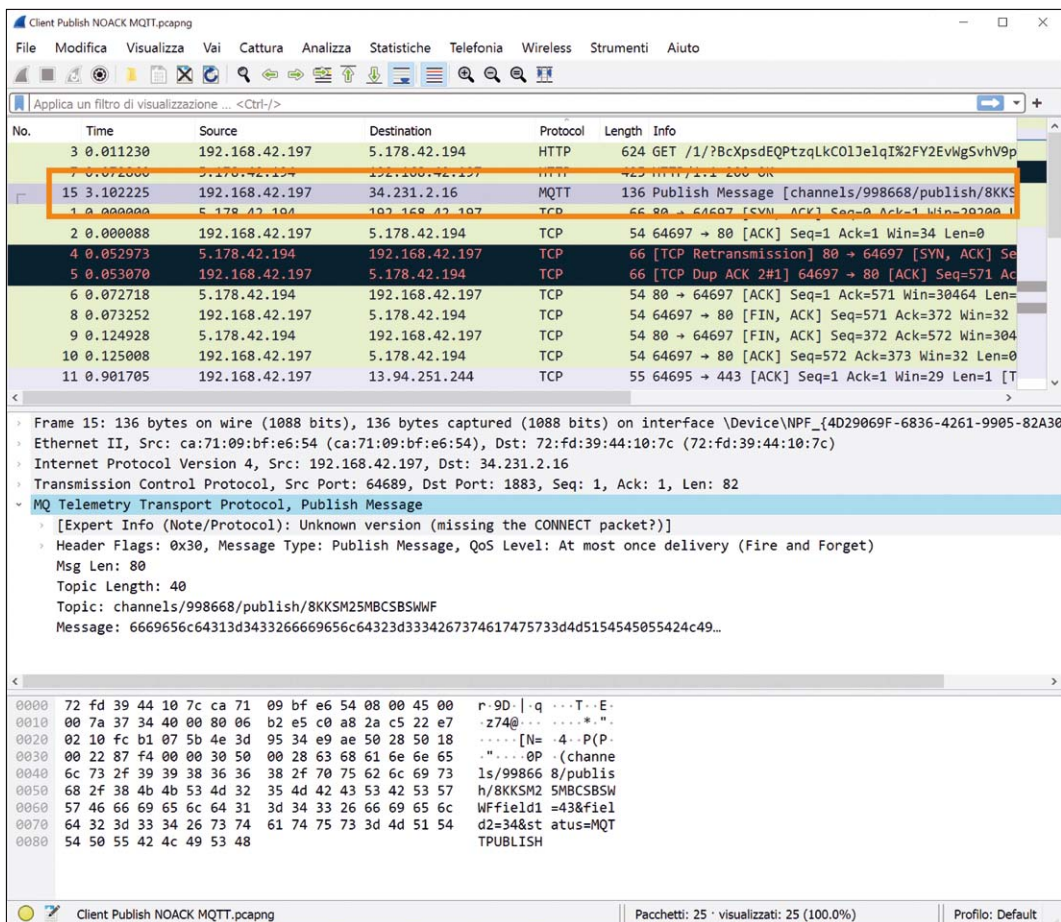


Fig. 5
Invio del
comando
PUBLISH.

senti nel pacchetto MQTT *CONNECT*, in particolare si può osservare quanto segue:

- il pacchetto ha lunghezza totale 82 byte;
- versione del protocollo MQTT settata a 4 che equivale alla revisione 3.1.1;
- Keep Alive impostato a 600 secondi;
- Client ID utilizzato *e310366e8ee6450e973ba-8decc363360*;
- User Name e Password;

Si noti che prima di ogni stringa ci sono due byte che ne indicano la lunghezza.

La Fig. 5 invece mostra l'invio del comando MQTT *PUBLISH* con campo *QoS* impostato a 0 (fire and forget). Infatti nella parte alta della figura si vede che nell'elenco c'è un solo pacchetto MQTT, inviato dal Client al Server, che corrisponde al pacchetto *PUBLISH* senza nessun pacchetto di risposta da parte del Server. In dettaglio abbiamo:

- il pacchetto ha lunghezza totale 80 byte;
- segue l'informazione della lunghezza del Topic ovvero 40 byte;
- segue il nome del Topic in formato stringa UTF-8.

Il Topic dipende dall'applicazione;

- infine c'è il messaggio, sempre in formato stringa, contenente i dati da pubblicare.

La Fig. 6 mostra sempre un pacchetto MQTT *PUBLISH* ma con campo *QoS* impostato a 1 (at least once delivery). Come si può osservare in alto nella figura al pacchetto MQTT *PUBLISH* inviato dal Client al Server segue un pacchetto MQTT *PUBACK* inviato dal Server al Client. Nel pacchetto inviato dal Client al Server ora compare anche la voce *Message Identifier*. La Fig. 7 mostra ancora un pacchetto MQTT *PUBLISH* ma con campo *QoS* impostato a 2 (exactly once delivery). Come si può osservare in alto nella figura al pacchetto MQTT *PUBLISH*, inviato dal Client al Server, segue un pacchetto MQTT *PUBREC* inviato dal Server al Client. Il Client risponde con un pacchetto MQTT *PUBREL* e il Server conclude con un *PUBCOMP*. Infine osserviamo la Fig. 8, la quale mostra un pacchetto MQTT *SUBSCRIBE*. Come si può osservare in alto nella figura al pacchetto MQTT *SUBSCRIBE*, inviato dal Client al Server, segue un pacchetto MQTT *SUBACK* inviato dal Server al

Fig. 6
Invio pacchetto
PUBLISH con
campo QoS
impostato a 1.

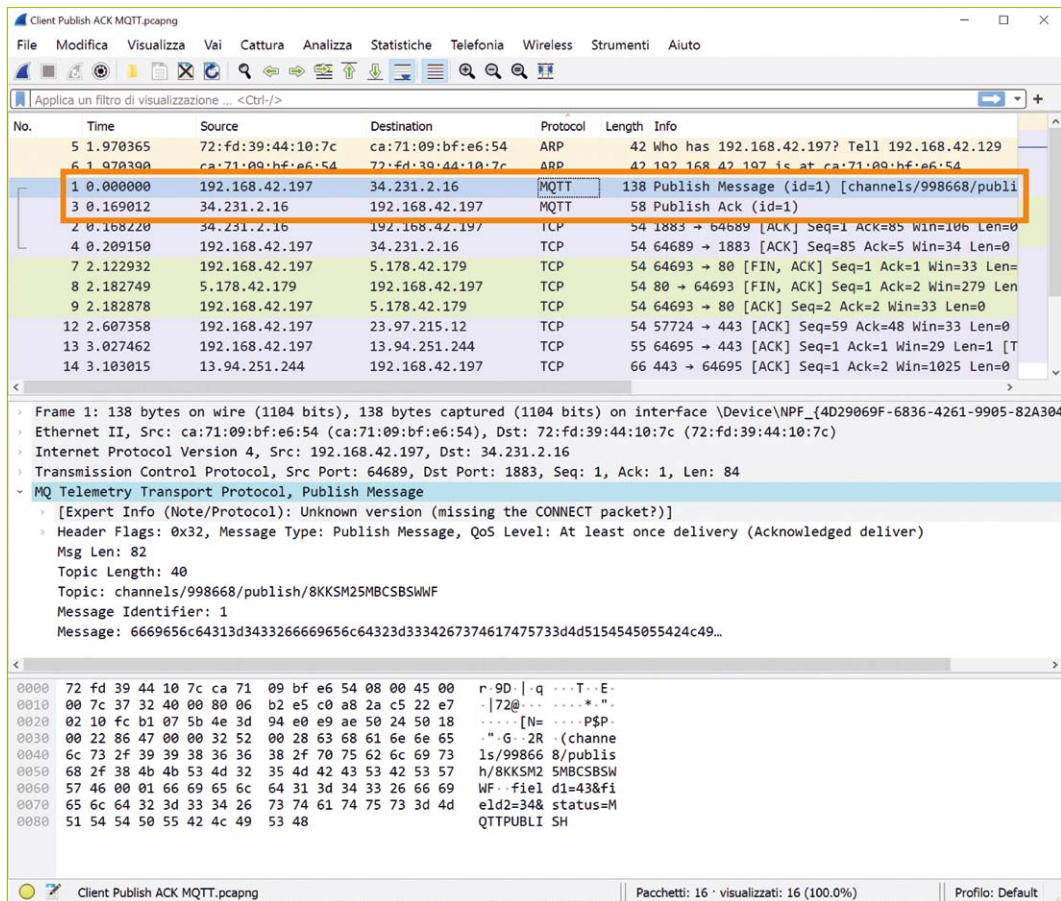
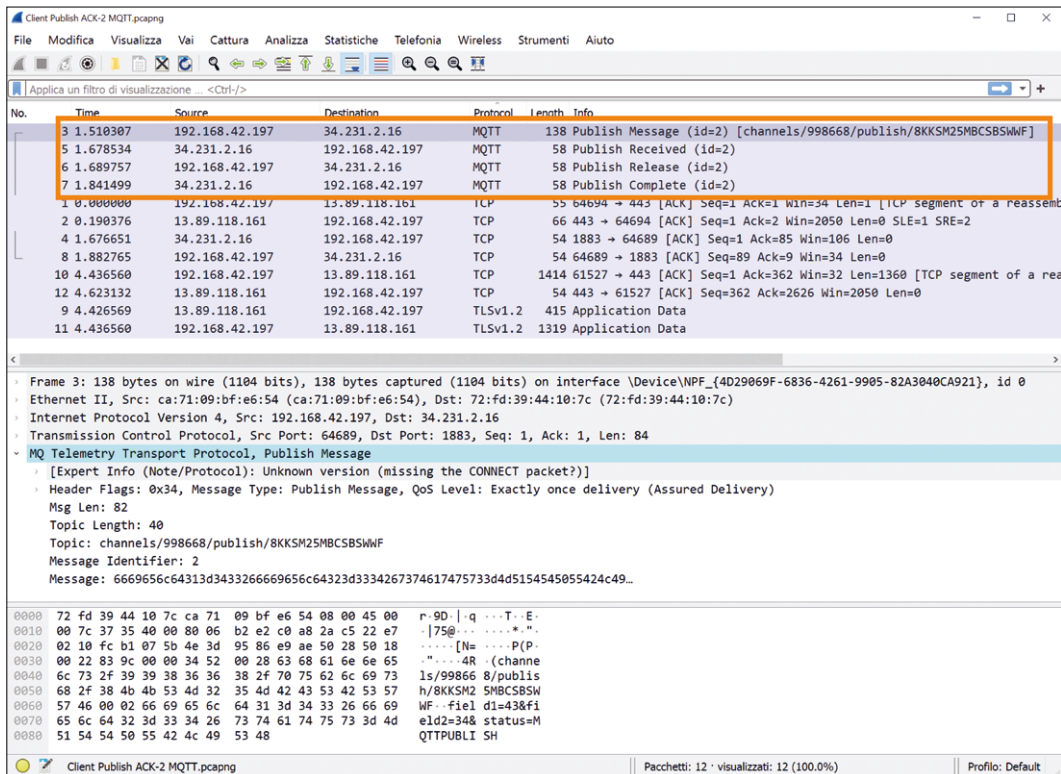


Fig. 7
Invio pacchetto
MQTT PUBLISH
con campo QoS
impostato a 2.



Client. Come si può osservare dalla figura il Client fa richiesta al Server tramite il *Topic* di lunghezza 56 byte. Ovviamente il nome del *Topic* dipende dall'applicazione. Si osservi che subito dopo il Server invia al Client un pacchetto MQTT *PUBLISH* con il *Topic* richiesto.

UNO SGUARDO VELOCE A MQTT 5.0

La nuova revisione del protocollo MQTT presenta della variazioni rispetto alla precedente 3.1.1.

Di seguito alcune considerazione in merito:

1. è stato aggiunto un nuovo pacchetto MQTT nominato *AUTH* ovvero *Authentication Exchange*;
2. è stato aggiunto un nuovo campo nominato *Properties*, che trova posto prima del *Payload*, il quale contiene un set di proprietà. Quindi il nuovo campo richiede un byte che ne identifica la lunghezza seguita dalle proprietà. Questo campo è presente neli pacchetti MQTT *CONNECT*, *CONNACK*, *PUBLISH*, *PUBACK*, *PUBREC*, *PUBREL*, *PUBCOMP*, *SUBSCRIBE*, *SUBACK*, *UNSUBACK*, *DISCONNECT* e *AUTH*;
3. è stato aggiunto un nuovo campo nominato

Reason Code il quale è un byte senza segno che indica il risultato di una operazione. Se il valore è inferiore a 0x80 indica una operazione conclusa con successo viceversa indica una operazione non andata a buon fine. Il *Reason Code* trova posto nei pacchetti MQTT *CONNACK*, *PUBACK*, *PUBREC*, *PUBREL*, *PUBCOMP*, *DISCONNECT* e *AUTH*. Invece i pacchetti MQTT *SUBACK* e *UNSUBACK* contengono una lista di uno o più *Reason Code*;

4. la versione del protocollo presente nel pacchetto MQTT *CONNECT* passa dal valore 0x04 a 0x05;
5. il flag bit *Clean Session* diventa *Clean Start*.

Queste sono a grandi linee le modifiche apportate alla nuova revisione del protocollo MQTT.

La nostra libreria di gestione del protocollo MQTT, di cui discuteremo in un altro articolo, è basata sulla revisione 3.1.1 che è la più utilizzata al momento.

Ovviamente non escludiamo di estendere la nostra libreria anche alla revisione 5.0 nel futuro. La libreria è stata pensata per essere utilizzata con i motori GSM e connessioni TCP/IP.

The image shows a Wireshark network traffic capture titled "Client Subscribe MQTT - Field.pcapng". The main packet list pane displays several packets. Three packets are highlighted with an orange box:

- Packet 3: Time 0.031176, Source 192.168.43.15, Destination 34.237.200.136, Protocol MQTT, Length 117, Info "Subscribe Request (id=5) [channels/998668/su]".
- Packet 21: Time 0.179110, Source 34.237.200.136, Destination 192.168.43.15, Protocol MQTT, Length 59, Info "Subscribe Ack (id=5)".
- Packet 53: Time 0.274110, Source 34.237.200.136, Destination 192.168.43.15, Protocol MQTT, Length 118, Info "Publish Message [channels/998668/subscribe/f]".

The packet details pane for the selected packet (Frame 3) shows:

- MQ Telemetry Transport Protocol, Subscribe Request
- [Expert Info (Note/Protocol): Unknown version (missing the CONNECT packet?)]
- Header Flags: 0x82, Message Type: Subscribe Request
- Msg Len: 61
- Message Identifier: 5
- Topic Length: 56
- Topic: channels/998668/subscribe/fields/field1/PODDI37ZU2FS1W5D
- Requested QoS: At most once delivery (Fire and Forget) (0)

The packet bytes pane at the bottom shows the raw data in hexadecimal and ASCII.

Fig.8
Invio del
pacchetto
SUBSCRIBE.

Stop ai fili con i nuovi interruttori wireless a 433 MHz!

Interruttori wireless da utilizzare in abbinamento al ricevitore wireless a 433 MHz a 1 canale (FR790). Non richiedono nessuna alimentazione in quanto si auto-alimentano tramite un generatore elettrodinamico posto al loro interno. Ideali per controllare facilmente senza fili le luci in abitazioni, garage, negozi o altri dispositivi funzionanti a 220 VAC con carico inferiore a 1000 W e in tutte le situazioni in cui è necessario installare un interruttore ad un impianto già esistente senza aggiungere

cavi di collegamento. La procedura di abbinamento tra ricevitore e pulsante è molto semplice e offre la possibilità di memorizzare fino a 5 dispositivi.



cod. FR790
€ 16,00

**Ricevitore wireless 433 MHz
1 Canale - 220 VAC**

**Interruttori wireless
a 433 MHz
autoalimentati**



cod. FR791
€ 9,00



2 CANALI

cod. FR792
€ 16,00



RICEVITORE

2 CANALI 220VAC
cod. EMY220RX2 **€ 16,00**

2 CANALI 12VDC
cod. EMY12RX2 **€ 16,00**

4 CANALI 12VDC
cod. EMY12RX4 **€ 19,00**

**Istruzioni
in italiano!**

RICEVITORE

1 CANALE 220VAC
cod. EMY220RX1 **€ 13,00**

1 CANALE 12VDC
cod. EMY12RX1 **€ 13,00**

TRASMETTITORE

1 CANALE
cod. EMYTX1
€ 11,00

**TRASMETTITORE
2 CANALI**
cod. EMYTX2 **€ 12,00**

**TRASMETTITORE
4 CANALI**
cod. EMYTX4 **€ 13,00**

SISTEMA DI RADIOCONTROLLO A 433 MHz a 1-2-4 canali 220VAC o 12VDC

Trasmettitori e ricevitori a 1-2-4 canali operanti sulla banda dei 433 MHz. Forniscono un segnale stabile a lunga distanza e un bassissimo consumo di corrente in modalità stand-by. I ricevitori possono essere alimentati a 220VAC o a 12VDC in base al modello. Ai ricevitori va abbinato esclusivamente il trasmettitore con lo stesso numero di canali.



Aprire il cancello



Attivare l'irrigazione



Accendere le luci di casa



Controllare le tende

NEOPIXAR

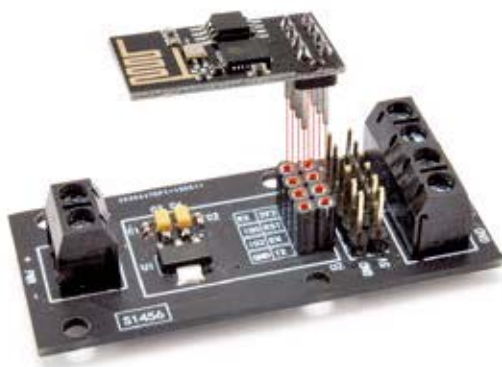


di **SAMUELE BRUNINI**

Uniamo le qualità della stampa 3D, l'efficienza e versatilità dei LED NeoPixel e la facilità di connessione a una rete WiFi tipica del ESP8266 per realizzare una lampada d'arredamento adatta ad ogni luogo.

Sappiamo ormai da tempo quante cose si possano realizzare facilmente e a buon prezzo quando si ha in casa una stampante 3D; tra queste, sicuramente molti elementi di arredo e d'utilità, sia monoblocco che composti o contenenti meccanismi; un esempio di ciò ve lo diamo in questo articolo, dove vi spieghiamo come autocostruire una lampada da tavolo a LED con braccio snodato che consente di alzarla e abbassarla, ma anche di inclinarla in avanti e indietro, come si farebbe con le tradizionali lampade in metallo. Stampa 3D è sinonimo di tecnologia e modernità, quindi non potevamo proporvi di realizzare con essa una lampada comune, ma abbiamo pensato di fare qualcosa che fosse al passo con i tempi e perfettamente integrato in un mondo connesso: una lampada WiFi comandabile a distanza

Modulo ESP
con la motherboard
FT1456K.



tramite un'interfaccia utente in esecuzione su uno smartphone o tablet, ottenuta appoggiandosi al servizio Cloud offerto da Blynk, che avete imparato a conoscere nel fascicolo n° 252 con il progetto "smart sensor connesso con Blynk." L'interfaccia (GUI) ci consentirà di controllare non solo la luminosità, ma anche la tinta cromatica, dato che in linea perfetta con lo stile tecnologico del progetto non abbiamo utilizzato LED qualsiasi, bensì i popolarissimi NeoPixel, che sono degli RGB controllabili serialmente in maniera semplice da Arduino e schede compatibili, grazie a una specifica libreria resa disponibile da Adafruit.

IL PROGETTO

La nostra lampada da tavolo è dotata di un controllo smart, implementato grazie al modulo basato su ESP8266 che ne è il cuore. L'interfaccia di controllo è basata su Blynk per permettere il controllo dell'apparecchio con il proprio smartphone o tablet in tutta comodità, con dei controlli semplici e intuitivi. È possibile anche controllare la lampada con Alexa con dei semplici comandi vocali,

come ad esempio "alexa accendi/spegni lampada" oppure regolare l'intensità della luce pronunciando "alexa accendi lampada al 50%".

L'illuminazione è costituita da un anello di 16 LED RGB Neopixel ciascuno contenente il chip WS2812, con la peculiarità di non aver necessità di driver esterni, infatti per controllare i LED è necessario un solo segnale digitale che proverrà dal pin 2 del modulo ESP8266. Nell'anello NeoPixel ciascuno dei 16 diodi LED RGB può essere gestito individualmente; ognuno può produrre 256 tonalità del proprio colore, determinando un totale di 16.777.216 colori. È possibile impostare a piacimento la frequenza di refresh, in modo da rendere impercettibili determinati giochi di luce. Il canale dati per la comunicazione è seriale di tipo One-Wire e può essere controllato tranquillamente con microcontrollori come l'ATmega di Arduino, i PIC ed anche schede Arduino, Fishino e moduli ESP, come abbiamo fatto nel nostro caso.

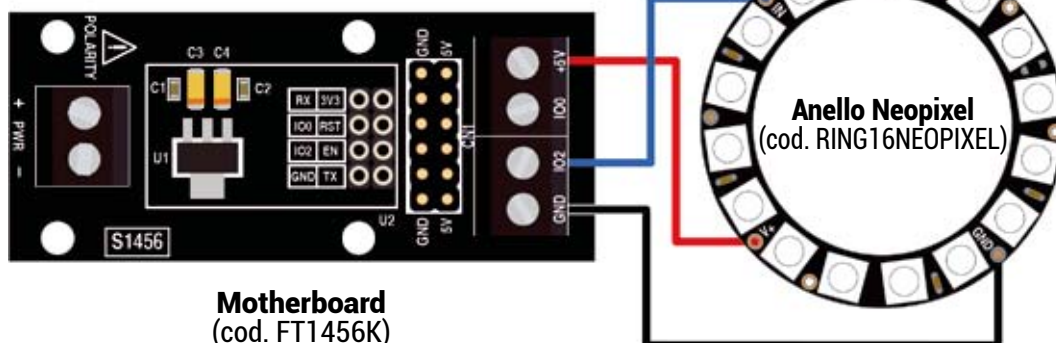
Per semplificare i collegamenti abbiamo utilizzato la scheda di espansione in kit per moduli ESP8266; il sistema è alimentato da una tensione di 5V e per standardizzare la connessione abbiamo optato per una porta microUSB.

Per quel che riguarda il circuito, nel dettaglio il progetto prevede i seguenti elementi:

- ESP8266 Modulo WiFi (cod.ESP8266);
- Scheda elettronica per ESP8266 (cod.FT1456K);
- Anello con 16 LED RGB WS2812 (cod. RING16NEOPIXEL);
- BreakoutBoard con microUSB (cod. BBMICROUSB-FEMM).

Per quanto riguarda la parte meccanica della lampada, i file .STL necessari a stampare in 3D le singole parti sono i seguenti:

schema di CABLAGGIO



- Lampada = www.thingiverse.com/thing:27062
- Base = www.thingiverse.com/thing:4129338

PREPARARE L'HARDWARE

La connessione tra i componenti è molto semplice e va eseguita seguendo lo schema di cablaggio visibile in queste pagine.

La scheda sulla quale si monta il modulo WiFi è la motherboard per ESP8266 codice FT1456K della Futura Elettronica (www.futurashop.it) la quale è sostanzialmente un piccolo PCB in grado di ospitare il modulo WiFi ESP8266 che dispone di un doppio strip femmina (2x2,54 mm) per innestare il modulo stesso, di una morsettiera a 2 vie per l'alimentazione (che è a 5Vcc) e di una morsettiera a 4 vie per portare all'esterno +5V, massa e le linee IO0 e IO2 (due GPIO del modulo WiFi) che trasportano i dati. La scheda riporta anche un connettore maschio (CN1) con gli stessi segnali del modulo ESP8266 (saranno quindi disponibili TX ed RX dell'UART, il reset ed altro ancora) insieme a GND e +5V.

CREAZIONE DEL PROGETTO SU BLYNK

Andiamo ora a vedere la parte riguardante il controllo da smartphone (o tablet). Per prima cosa è necessario scaricare l'applicazione Blynk da AppStore o PlayStore a seconda del dispositivo che si vuole utilizzare. Fatto ciò occorre aprire l'applicazione: nel caso si abbia già un account su blynk sarà semplicemente necessario eseguire il login con mail e password utilizzata per la registrazione; se invece è la prima volta che si utilizza Blynk, bisogna pigiare il pulsante "Create New Account" come in **Fig. 1**.

A questo punto ci si trova davanti alla schermata di registrazione: in questa è necessario inserire la propria e-mail e password; una volta confermato con il pulsante "Next" si riceverà una e-mail all'indirizzo inserito poco prima, con un link da aprire per terminare la registrazione. Svolti questi semplici passaggi la registrazione è completata e vi ritroverete nella schermata proposta nella **Fig. 2**.

Adesso bisogna premere il pulsante del QR code in alto a destra nella schermata (evidenziato dal quadratino rosso nella **Fig. 2**) e consentire l'utilizzo della fotocamera per poter inquadrare il QR code; questo passaggio sarà necessario per poter clonare il progetto esistente direttamente sul vostro account.

Una volta scansionato il QR code si aprirà la pagina visibile nella **Fig. 4**, relativa al progetto che stiamo creando, nella quale troviamo le principali infor-

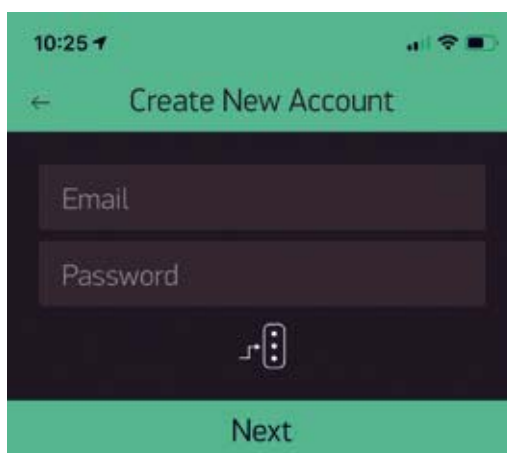


Fig. 1
Creazione dell'account in Blynk.

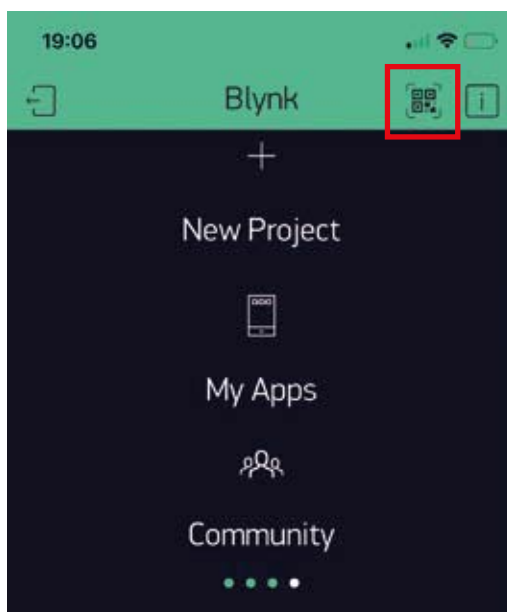


Fig. 2
Schermata di creazione di un nuovo progetto e, sotto, il QR code da scansionare.



mazioni compreso il token di autenticazione che ci servirà tra breve. L'ultima cosa che rimane da fare è copiare l'AUTH TOKEN dalla pagina delle impo-

Fig. 3
Schermata di creazione del progetto con evidenziato il token per l'autenticazione.

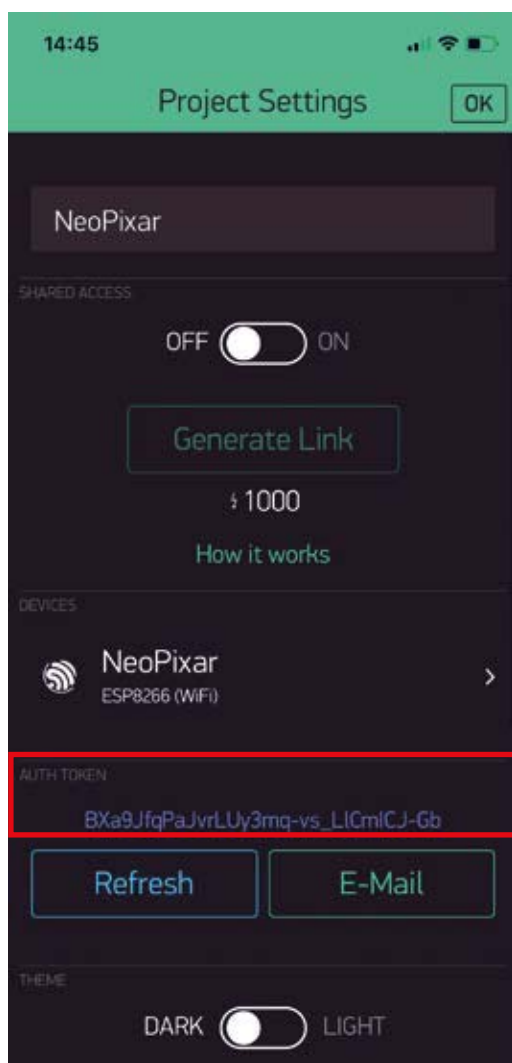
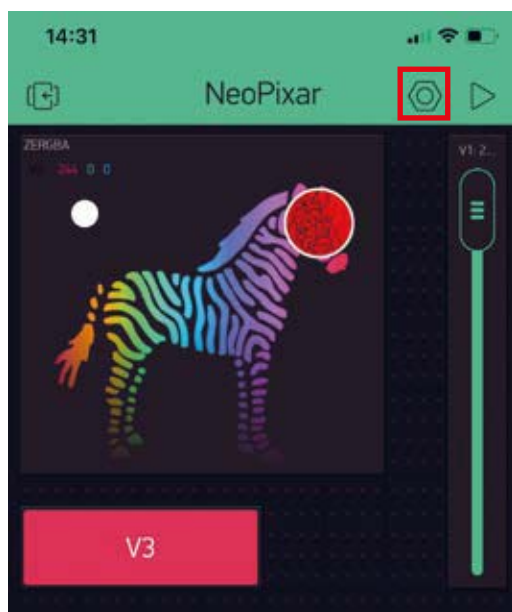


Fig. 4
Il pulsante delle impostazioni nella schermata.



stazioni perché sarà il codice che andrà inserito nel firmware per creare il collegamento tra l'ESP8266 e l'applicazione; alla pagina delle impostazioni si accede premendo il pulsante a forma di dado presente in alto a destra dello schermo, come proposto nella **Fig. 4**.

È possibile copiare manualmente il codice oppure inviarselo per e-mail (consigliato) così da poter fare direttamente copia e incolla dal messaggio e-mail all'IDE di Arduino.

La schermata dell'IDE Arduino con lo sketch dell'applicazione una volta inserito tale parametro si presenta come proposto nella **Fig. 5**, dove AUTH TOKEN è quello evidenziato dal bordo rosso; alla stessa maniera nella figura trovate evidenziati i parametri della rete WiFi (ssid e password). Ricordiamo che per poter programmare e configurare l'ESP8266 nell'IDE Arduino è necessario installare il driver di terze parti Espressif (www.espressif.com).

IL FIRMWARE

Affinché il dispositivo possa funzionare è necessario modificare alcuni parametri personali del firmware ad esempio i dati del WiFi (esclusivamente i 2,4 GHz) o l'Auth Token prelevato prima dall'applicazione, questi valori vanno modificati inserendoli in mezzo agli apici, rispettando le maiuscole e minuscole

Se si vuole far apparire il dispositivo sull'app di alexa con un nome dispositivo diverso da Lampada, modificare il valore con quello desiderato. Quando si dirà ad Alexa di accendere il dispositivo bisognerà pronunciare le frasi in questo modo:

Tabella 1
Connessioni tra scheda ESP8266 e converter TTL/USB.

SCHEDA ESP8266	CONVERTITORE SERIALE
TX	RX
RX	TX
CHPD	3.3V
GPIO 0	GND
GND	GND
PWR +	5V
PWR-	GND


```

ver1|Arduino 1.8.10
File Modifica Sketch Spuntini: Arduino
ver1
1 //
2 NeoPixel NeoPixel ESP8266
3 created by
4 Daniele Brunialti
5 //
6
7 #define SERIAL_PRINT Serial
8 #include <BlynkSimpleEsp8266.h>
9 #include <ESP8266WiFi.h>
10 #include <ESP8266DNS.h>
11 #include <WiFiUdp.h>
12 #include <ArduinoOTA.h>
13 #include <Adafruit_NeoPixel.h>
14 #include "fauxmoESP.h"
15
16 const char* ssid = "ESP8266"; //ssid WIFI
17 const char* password = "password"; //password WIFI
18
19
20 #define PIN 2 //pin
21 #define Imp "Lampada" //dispositivo riconosciuto da alexa
22 char auth[] = "AUTH_TOKEN";
23
24 int E, G, B;
25
26 fauxmoESP fauxmo;
27

```

Fig. 5
Impostazioni
dello sketch
nell'IDE
Arduino.

"Alexa accendi NOME DISPOSITIVO"
"Alexa Spegni NOME DISPOSITIVO"
"Alexa accendi NOME DISPOSITIVO al 40%"

Prima di avviare l'upload del firmware è necessario installare tutte le librerie che troverete nel file zip, è sufficiente copiarle e inserirle nella cartella *documenti > arduino > libraries* e avviare l'ide di arduino, in caso le si avesse già installate in precedenza, controllare semplicemente se sono aggiornate all'ultima versione. Come di consueto per flashare il firmware sull'ESP8266 è necessario mettere a massa il pin GPIO 0 per abilitare la modalità flash. È possibile utilizzare qualsiasi convertitore seriale TTL/USB che si voglia; per le nostre prove abbiamo utilizzato l'FT782 (www.futurashop.it). I collegamenti tra la motherboard ESP8266 e il convertitore sono quelli riportati nella **Tabella 1**. Collegare i jumper direttamente sulla scheda di espansione dell'ESP8266 fare bene attenzione al

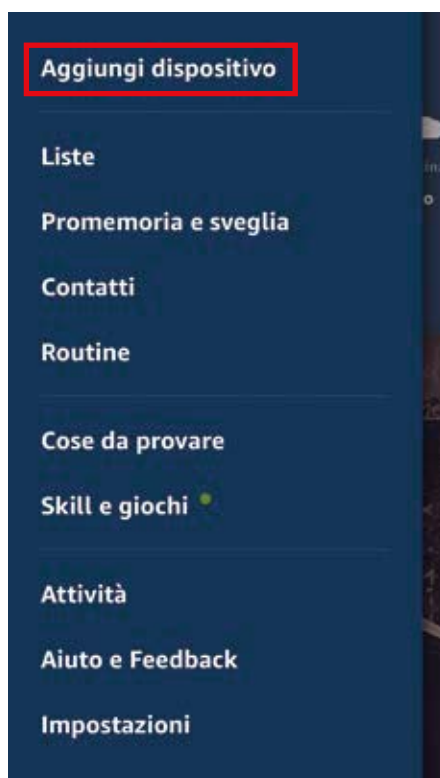
collegamento del GND tra convertitore seriale e scheda ESP.

Una volta terminato il caricamento è possibile scollegare il convertitore e il GPIO 0 in questo modo l'ESP potrà avviarsi caricando lo sketch dalla sua memoria interna, a questo punto il lavoro è



Il nostro
prototipo con
l'elettronica
inserita nella
base.

→ **Fig. 6**
Creazione nuovo
dispositivo nell'app.



→ **Fig. 7**
Scelta dispositivi.



finito e sarà già possibile controllare la lampada attraverso l'applicazione Blynk.

CONTROLLO CON ALEXA

Vediamo adesso come fare per gestire la nostra lampada NeoPixel attraverso l'assistente vocale Alexa; prima di cominciare bisogna assicurarsi che il telefono/tablet e il dispositivo Alexa-compatibile siano collegati alla stessa rete WiFi della lampada, perché diversamente il controllo sarà impossibile. Per configurare la lampada con Alexa, è sufficiente aprire l'applicazione "Amazon Alexa" e premere "aggiungi dispositivo" come mostrato nella **Fig. 6**; in seguito, nella prossima schermata scorrete l'elenco fino all'ultima voce e premete il pulsante "altro" (**Fig. 7**) allorché si aprirà un'altra schermata con alcune avvertenze sarà necessario premere su "trova dispositivi". Adesso l'app incomincerà a ricercare il dispositivo; la ricerca durerà mediamente 45 secondi. Il termine della ricerca l'applicazione vi avviserà di aver trovato un dispositivo con il nome scelto in precedenza. Ora dovete premere su Configura dispositivo e in seguito su "fatto". Adesso il dispositivo è correttamente configurato e pronto per l'utilizzo.

CONCLUSIONI

Anche l'applicazione qui descritta mostra come, sfruttando il servizio Cloud Blynk, è possibile trasformare una manciata di elettronica e un dispositivo di uso comune come lo smartphone in un'idea innovativa. □

Cosa occorre?

I componenti utilizzati in questo progetto sono disponibili presso Futura Elettronica. La breakout board in grado di ospitare il modulo WiFi (cod. FT1456K) è in vendita a Euro 5,00, l'ESP8266 (cod. ESP8266) è disponibile a Euro 5,00, l'anello neopixel con 16 led (cod. RING16NEOPIXEL) costa Euro 12,90, la breakout board con microusb (cod. BBMICROUSBFEMM) è in vendita a Euro 0,50. Per la stampa delle parti plastiche è disponibile il servizio di stampa 3D direttamente dal sito bit.ly/3DSericeFE. I prezzi si intendono IVA compresa.

Il materiale va richiesto a:

Futura Elettronica, Via Adige 11, 21013 Gallarate (VA)
Tel: 0331-799775 - <http://www.futurashop.it>

SCEGLI LA TUA STAMPANTE 3D

Affidati a chi le progetta e le sviluppa

STAMPANTE 3D 40x40x40 cm in KIT

stampante 3D FDM (Fused deposition modeling)
capace di produrre stampe di **40x40x40 cm** (64.000 cm³)

€ **999,00**

Cod. **3D4040**

- ✓ Struttura interamente in alluminio
- ✓ Area di stampa: X 40 cm, Y 40 cm, Z 40 cm
- ✓ Diametro filamento: 1,75 mm
- ✓ Tipo di filamento: ABS, PLA, NYLON ed altri ancora
- ✓ Diametro ugello fornito: 0,4 mm
- ✓ Diametro ugelli opzionali: da 0,3 mm a 0,8 mm
- ✓ Velocità di stampa massima: 300 mm/s (in funzione dell'oggetto da stampare)
- ✓ Piatto di stampa fisso: vetro temperato da 6 mm
- ✓ Riscaldatore per piatto di stampa: 40 x 40 cm – 12V/240 W con adesivo 3M (opzionale)
- ✓ Controllabile da PC o da modulo LCD (opzionale)
- ✓ Alimentazione tramite modulo switching 220 VAC/12 VDC 350 W
- ✓ Istruzioni di montaggio in italiano con illustrazioni



Versione montata
cod. 3D4040/M
€ 1.299,00 IVA inclusa.



STAMPANTE 3DRAG IN KIT

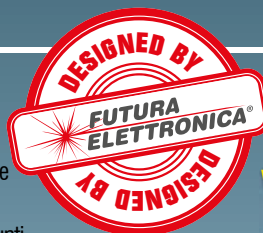


- ✓ Struttura in alluminio per combinare rigidità e leggerezza
- ✓ Assemblaggio semplificato con giunti metallici e parti in ABS stampate a iniezione
- ✓ Piano di stampa con movimentazione X/Y
- ✓ Estrusore movimentato sul solo asse Z
- ✓ Volume di stampa massimo: 20 x 20 x 20 cm
- ✓ Risoluzione: X e Y - 0,015 mm; Z - 0,39 micron
- ✓ Estrusore con ugello da 0,5 mm per filo da 3 mm in PLA e ABS
- ✓ Velocità di stampa tipica: 120 mm/s
- ✓ Piatto riscaldato

€ **499,00**

Cod. **3DRAG/K**

Versione montata
cod. 3DRAG/M
€ 699,00 IVA inclusa.



VASTA GAMMA DI ACCESSORI!
SCOPRILI TUTTI SU
www.futurashop.it

**E CON LA STAMPANTE 3DRAG
PUOI ANCHE...
STAMPARE
CON IL CIOCCOLATO**

**Set estrusore Choco
3DRAG - in kit**
Cod. 3DCHOCO - € 178,00



I robot stanno progressivamente penetrando nella nostra vita, non soltanto nell'industria, nella logistica, nella chirurgia e nell'assistenza sanitaria, quindi in ambito professionale, ma ce li troviamo in casa ad esempio sotto forma di aspirapolvere capaci di muoversi nell'ambiente orientandosi e scansando gli ostacoli, ma anche raggiungendo autonomamente la stazione di ricarica delle batterie. La tecnologia di questi elettrodomestici indipendenti è molto interessante, tant'è che in altre occasioni ci siamo cimentati nella realizzazione di robot su ruote capaci di circolare autonomamente. Peraltro oggi è implementabile con prodotti facilmente reperibili e di basso costo. In questo articolo vogliamo tornare sull'argomento per "snocciolare" il problema della navigazione autonoma di un robot all'interno di un ambiente chiuso sconosciuto. Approfondiremo, inoltre, il problema della localizzazione all'interno di ambienti chiusi e la costruzione di una mappa dell'ambiente circostante mediante una tecnologia recente come il **LIDAR** (Light Detection and Ranging o Laser Imaging Detection and Ranging), impiegato, ad esempio, nell'ADAS (autoveicoli a guida autonoma). Per loro natura gli argomenti trattati non sono dei più semplici, ma proveremo a spiegarli nella maniera più comprensibile possibile.

Il problema generale della navigazione autonoma dei robot è oggi uno dei più studiati, sotto l'impulso della necessità del volo automatico dei droni, ma soprattutto della guida autonoma delle automobili.

Noi ci concentreremo però sulla navigazione in ambienti interni, problema che si incontra nei robot aspirapolvere, che devono percorrere tutto lo spazio a disposizione in modo da poter raccogliere la polvere in ogni punto raggiungibile; il nostro scopo finale è, invece, la mappatura dell'ambiente circostante

(map building): è quindi essenziale che la navigazione avvenga negli spazi più ampi piuttosto che in quelli più stretti dove potrebbe raccogliersi la polvere. Se volessimo realizzare un robot per la navigazione autonoma, alcuni sensori ad ultrasuoni sarebbero più che sufficienti per fornire al robot le informazioni necessarie per riconoscere gli ostacoli nelle vicinanze, anche se la navigazione non sarebbe molto efficiente in quanto non sarebbe possibile valutare con precisione il percorso migliore (path planning) a causa delle poche informazioni disponibili. Per raggiungere il nostro obiettivo dobbiamo necessariamente utilizzare un sensore molto avanzato in grado di fornire al robot molte più informazioni sull'ambiente circostante; stiamo parlando di sensori laser scanner di tipo LIDAR.

Robot su tre ruote a navigazione autonoma, in grado di muoversi e orientarsi all'interno di spazi chiusi con la capacità di ricostruire una mappa dell'ambiente circostante tramite sensore LIDAR.

LIDAR rappresenta una tecnica grazie alla quale è possibile misurare la distanza degli oggetti tutto attorno, tramite impulsi laser; la tecnica usata è simile a quella usata nei radar, che però utilizzano segnali radio invece che impulsi luminosi.

Essendo il fascio laser di tipo puntiforme, la misura sarà estremamente accurata a differenza di quella ottenibile con i sensori ad ultrasuoni, che eseguono una valutazione in un arco di cerchio ampio non meno di 30 gradi.

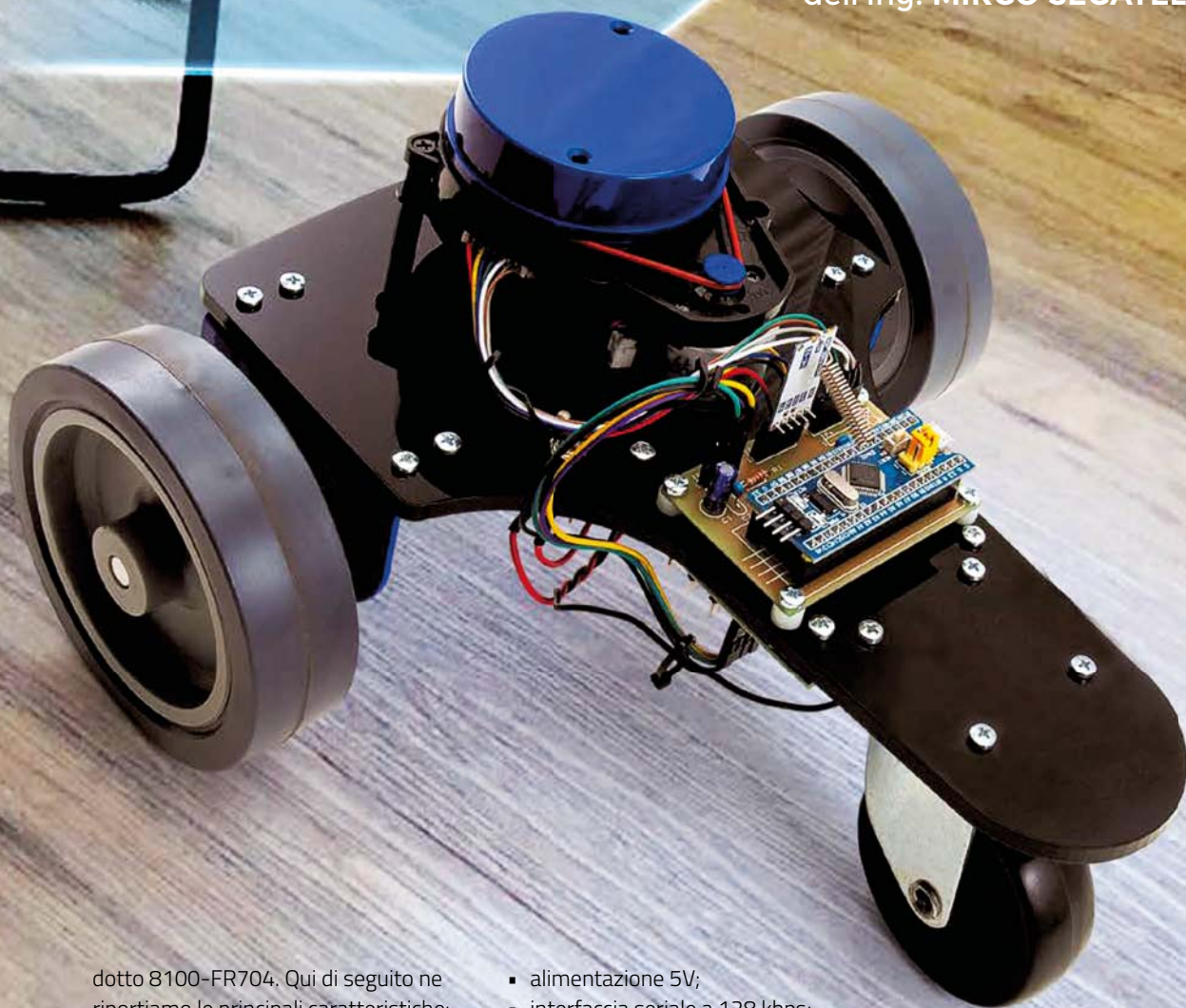
IL SENSORE LASER SCANNER 2D

Per il nostro progetto ci avvarremo di un sensore YDLidar X4 (**Fig. 1**) prodotto dall'azienda cinese EAI e distribuito in Italia dalla Futura Elettronica, quindi reperibile sul relativo sito Internet (www.elettronica.in.it) con il codice pro-



YDLidar ROBOT

dell'Ing. MIRCO SEGATELLO



dotto 8100-FR704. Qui di seguito ne riportiamo le principali caratteristiche:

- scansione su 360°;
- rilevamento da 12 cm a 10 metri;
- laser all'infrarosso in classe I (sicuro per l'occhio nel caso si intercetti il raggio);
- campionamento fino a 5.000 letture al secondo;
- risoluzione angolare 0,5°;
- risoluzione nella distanza 1 mm;

- alimentazione 5V;
- interfaccia seriale a 128 kbps;
- assorbimento 350 mA (in scansione).

I sensori LIDAR sono composti da due parti: una fissa ed una mobile posta in rotazione da un motore. La parte mobile invia costantemente impulsi laser nell'ambiente circostante e valuta l'angolo di incidenza del segnale riflesso tramite un piccolo sensore CCD.

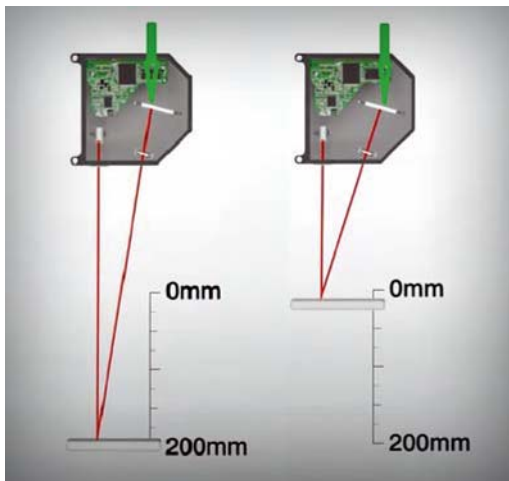
A differenza della tecnica TOF (acronimo inglese di Time of Flight) la tecnica a triangolazione permette di ottenere più misure in minor tempo e con maggiore precisione.

Facendo riferimento alla **Fig. 2**, un oggetto posto lontano farà in modo che il fascio laser riflesso cada nella

Fig. 1
Sensore
YDLidar X4.



Fig. 2
Tecnica della
triangolazione
in un sensore
laser.



zona sinistra del sensore CCD mentre un oggetto posto nelle vicinanze farà cadere il fascio laser più a destra, un'opportuna logica a microcontrollore ricaverà la distanza dell'oggetto. La distanza misurata avviene in una ben precisa direzione che viene rilevata durante la rotazione, pertanto ogni singola misurazione consta di un angolo e di una distanza a cui può essere

associato anche un valore che indica la qualità (e quindi l'affidabilità) della misura in quanto alcune misurazioni potrebbero fallire per diversi motivi, spesso dipendenti dalla forma e tipo di superficie colpito dal raggio laser. Ruotando rapidamente, la parte mobile fornirà centinaia, ma anche migliaia di letture al giro, tracciando una mappa accurata dell'ambiente circostante.

Ancora più complicati sono i laser scanner 3D che permettono una ricostruzione spaziale dell'ambiente circostante nelle tre dimensioni.

Nella confezione del sensore troverete anche due cavetti di collegamento ed un convertitore seriale USB per poter interfacciare facilmente il sensore con un PC. Dal sito del produttore potete scaricare il software *PointCloudViewer* (per PC) che permette di visualizzare in una mappa 2D i dati forniti dal sensore, come visibile in Fig. 3.

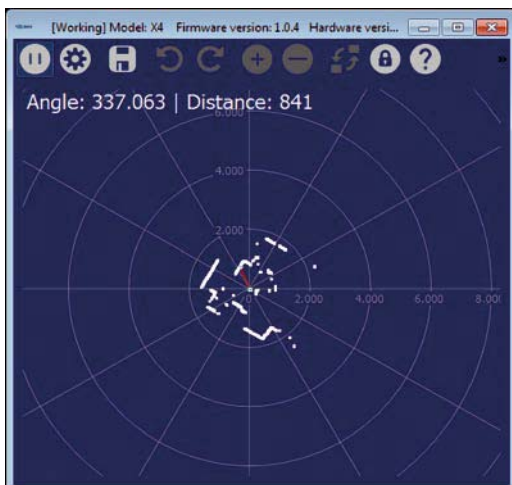
REALIZZAZIONE PRATICA

Il sensore LIDAR fornisce un flusso di dati notevole e l'interfaccia seriale a 128 kbps mal si sposa con le logiche Atmel tipo ATmega328 e ATmega2560 di Arduino; sarebbe meglio optare per una scheda Raspberry Pi oppure, come nel caso del robot proposto in queste pagine, per una scheda basata su STM32 programmabile in ambiente Arduino. Allo scopo, la stessa azienda produttrice del

Tabella 1
Materiale occorrente.

COMPONENTE	CODICE FUTURA ELETTRONICA
1x YDLidar X4	8100-FR704
1x Driver MD25	7300-MD25
2x Motore MG30 con ruota	7300-EMG30
2x Ruota diametro 100mm	7300-WHEEL100
2x staffa 90° per motore MG30	7300-EMG30SUPP
2x Modulo radio HC-12	2846-HC12
1x Board con Microcontrollore STM32F103C8T6	2846-STM32ARMCORE
1x Ruota pivotante diametro 600mm	-
1x Batteria ricaricabile LiPO 2.200 mAh - 11,1 V	7893-25C2200MAH11V1

Fig. 3
Mappa creata
con il software
PointCloudViewer.



sensore fornisce l'apposita libreria in grado di interpretare il flusso di dati in uscita dal sensore, semplificando enormemente lo sketch da scrivere per la gestione del LIDAR. Nel nostro caso, dunque, abbiamo utilizzato una board basata su STM32F103 con core a 32 bit funzionante a 72 MHz. Questa MCU dispone di una porta USB nativa e 3 porte seriali hardware che useremo per interfacciare il sensore LIDAR e la telemetria. In **Fig. 4** lo schema a blocchi del sistema con l'interconnessione dei vari componenti. Il materiale necessario a realizzare il nostro robot è riportato nella **Tabella 1**. Per poter creare un robot per la navigazione autonoma abbiamo bisogno di una base robotica con ben precise caratteristiche, prima fra tutte la necessità di sgravare la CPU principale dal compito di gestire i motori e gli encoder. Abbiamo anche la necessità che il robot si muova con precisione ed in modo fluido con un controllo diretto di accelerazione e velocità, come vedremo questo è un punto fondamentale per la successiva implementazione dell'odometria.

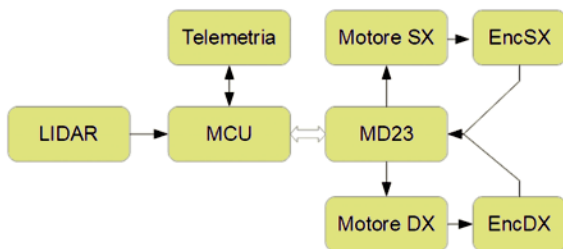


Fig. 4
 Schema a blocchi del sistema.

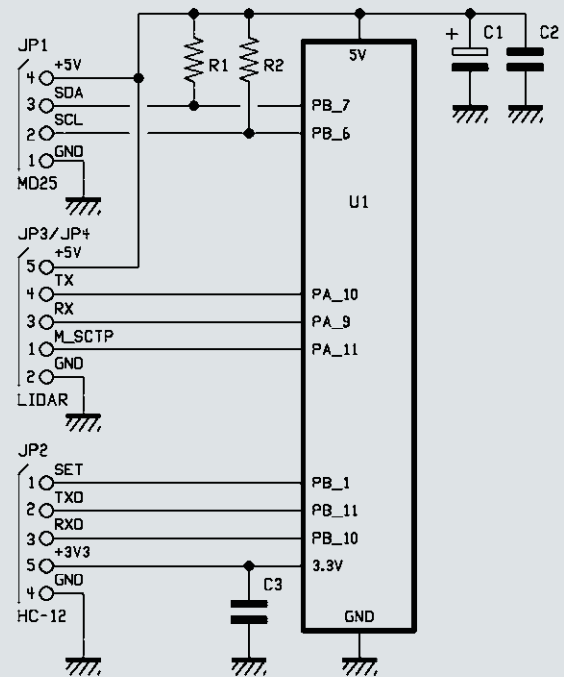
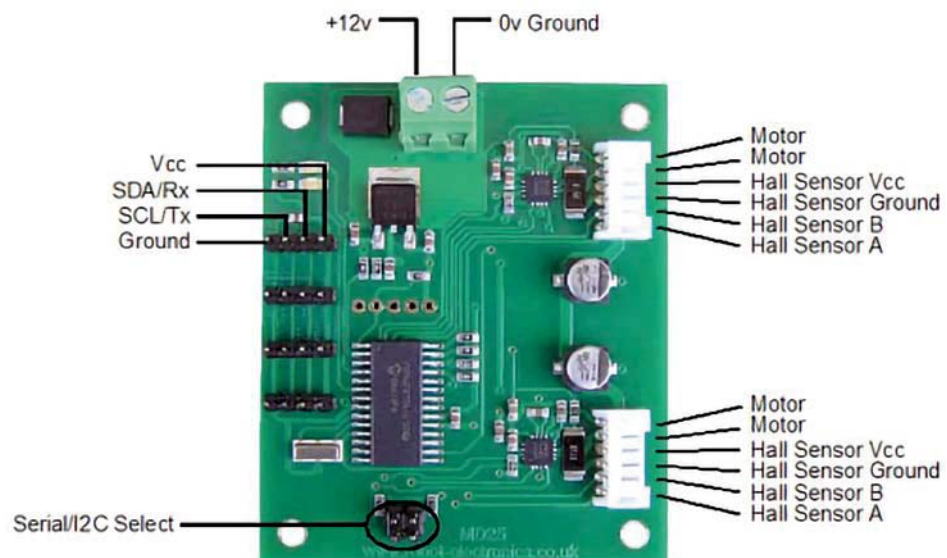
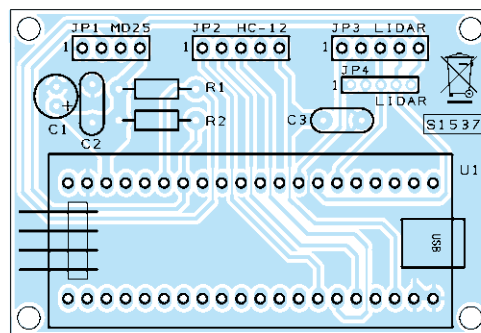
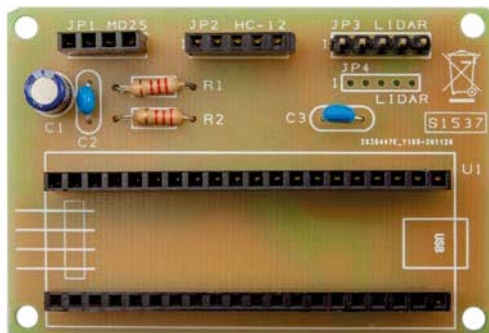


Fig. 5
 Il driver per i motori MD25.





Elenco Componenti:

R1, R2: 2,2 kohm
 C1: 10 μ F 25 VL elettrolitico
 C2: 100 nF 63V poliestere
 C3: 470 nF 63V poliestere
 U1: STM32ARMCORE
 JP1: strip femmina 4 poli
 JP2: strip femmina 5 poli
 JP3: strip maschio 5 poli

Varie:

- LIDAR X4, sensore LIDAR cod. FR704
- modulo HC12
- driver per 2 motori Dc con ingresso encoder cod. MD25

- motoriduttore 12 Vdc con encoder cod. EMG30
- staffa di supporto per motoriduttori cod. EMG30SUPP
- connettore DEANS maschio
- interruttore a levetta
- strip femmina 5 poli
- strip femmina 20 poli
- strip maschio 5 poli 90°
- cavetto 100 mm con connettore 4P maschio - 4P femmina P,2,54 mm
- ruota pivotante $\varnothing$ 60 mm con piastra di fissaggio 47x47 mm
- ruota diametro 100 mm

- cod. WHEEL100
- vite M3x6 TB a croce
- vite M3x12 TB a croce
- vite M3x14 TB a croce
- vite M3x16 TB a croce
- dado M3
- distanziale in ABS 3x6 mm cod. DISTABSM3X6
- distanziale in ABS 3x4 mm cod. DISTABSM3X4
- piastra in plexiglass sagomata per LIDAR ROBOT
- circuito stampato S1537 (64x43,5 cm)

- consumo: 530 mA;
- consumo a vuoto: 150 mA;
- consumo in stallo: 2,5 A;
- risoluzione encoder: 360 impulsi/giro.

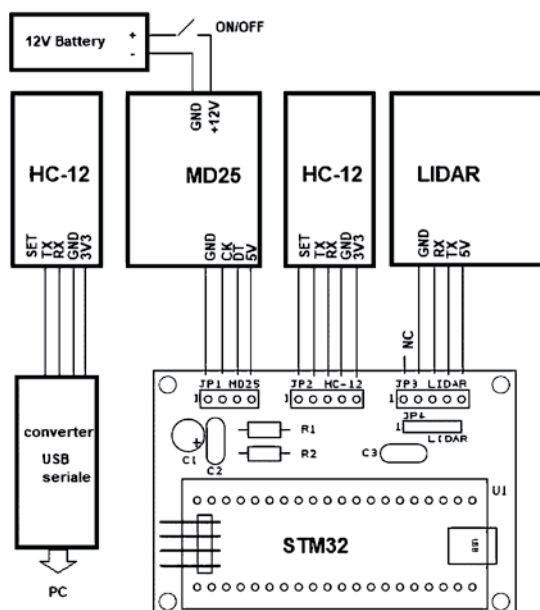


Fig. 6
Interconnessione degli elementi.

Per il controllo dei motori abbiamo optato per un driver intelligente siglato MD25 (**Fig. 5**). Questo driver è stato specificatamente progettato per funzionare in congiunzione con i motori EMG30 con un'alimentazione principale a 12V e dispone di uno stabilizzatore interno a 5V per fornire energia ai dispositivi esterni, nel nostro caso la board con STM32. Il driver utilizza la lettura degli encoder per mantenere la velocità specificata ed impostare l'accelerazione/decelerazione programmata. Inoltre è prevista la lettura della tensione di alimentazione e per garantire la massima sicurezza implementa un blocco nel caso si interrompa la comunicazione con la logica di comando oppure la tensione scenda sotto i 9 volt. Per il controllo utilizzeremo l'interfaccia I²C-Bus, pertanto i ponticelli denominati "Serial/I2C Select" andranno lasciati liberi. La corrente massima gestita è di 2,8A per motore.

Per assemblare tutte le parti meccaniche abbiamo previsto una base da stampare in PLA con una stampante 3D. Per chi non disponesse di una stampante 3D potrà optare per del comune plexiglass con spessore 4-5mm sagomato e forato seguendo le indicazioni dei file allegati.

▼ Listato 1

```
int searchPath()
{
    int searchAngle, optiAngle;
    int minDist, maxDist=0;
    for (searchAngle=90+arcAngle/2; searchAngle<270-arcAngle/2; searchAngle++) {
        minDist=10000;
        for (int i=searchAngle-arcAngle/2; i<searchAngle+arcAngle/2; i++) {
            if ( my_dist[i] > 0 && my_dist[i] < minDist) {
                minDist=my_dist[i];
            }
        }
        if (minDist>maxDist) {
            maxDist=minDist;
            optiAngle=searchAngle;
        }
    }
    if (maxDist>minGap)
        return(optiAngle);
    else
        return 0;
}
```

Per completare il robot le due ruote di trazione ed una ruota posteriore pivotante con diametro di 60mm.

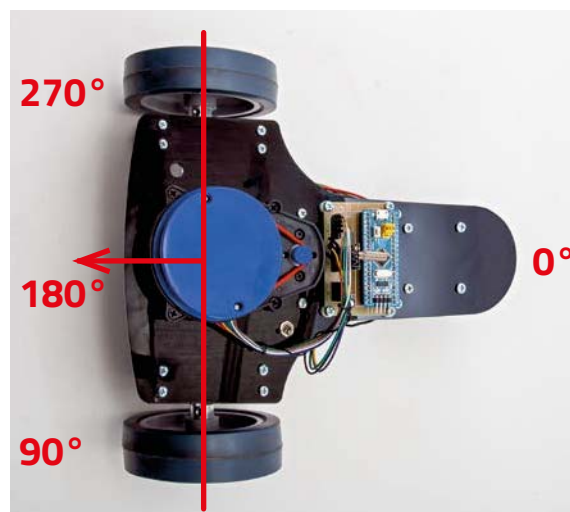
Abbiamo previsto un piccolo stampato che raccoglie i pochi componenti necessari, essendo a singola faccia è facilmente realizzabile a partire dai file forniti assieme a tutti i file di questo numero della rivista. L'alimentazione è affidata ad una batteria a 3 celle LiPo da 2.200 mAh che è connessa tramite un interruttore (per accensione e spegnimento) al modulo MD25, sarà lo stabilizzatore interno a fornire i 5V per alimentare il resto dei componenti. Sulla board con STM32 vanno saldati i due strip maschio forniti in dotazione, sulla scheda invece i relativi strip femmina, aiutatevi con le immagini in queste pagine, che ritraggono il prototipo da noi realizzato e collaudato, l'interconnessione dei vari elementi è raffigurata in **Fig. 6**.

NAVIGAZIONE AUTONOMA

Passiamo ora a descrivere la logica di comando che è stata implementata per realizzare la navigazione autonoma. Pur avendo usato una logica basata su STM32 abbiamo scritto il codice nel classico ambiente di programmazione Arduino IDE, pertanto le righe di codice vi risulteranno famigliari e maggiormente comprensibili. Ci sono tre principali sotto programmi che vengono eseguiti in continuazione: YDLidarACQ, odometry, navigation. Il primo sotto programma, come dice il nome stesso, viene usato per leggere i dati dal LIDAR ed archivarli in un vettore di 360 valori, ciascuno per

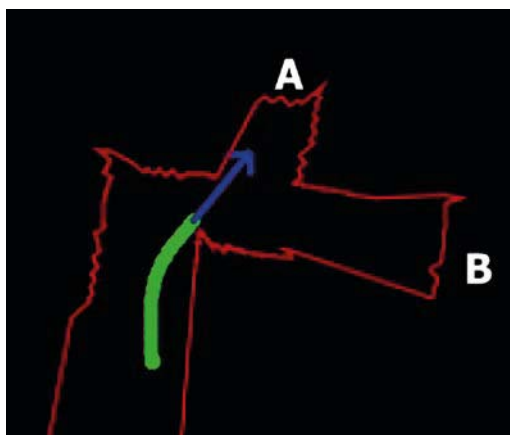
ciascun angolo in gradi. Quindi la casella di indice 90 conterrà la distanza dell'oggetto posto a 90° rispetto il sensore Lidar che però è stato installato ruotato di 180° per evitare che il motore sporgesse nella parte frontale; fate riferimento alla **Fig. 7** per il corretto orientamento.

Il LIDAR fornisce un flusso di dati considerevoli e alcuni di essi non sono validi (la misura è zero) oppure affetti da errore, come detto prima a causa della variabilità della superficie degli oggetti colpiti dal laser. Pertanto procederemo ad inserire i valori nel vettore facendo una media dei valori rilevati, scartando i valori errati e riducendo gli errori di lettura. Il vettore così determinato viene passato

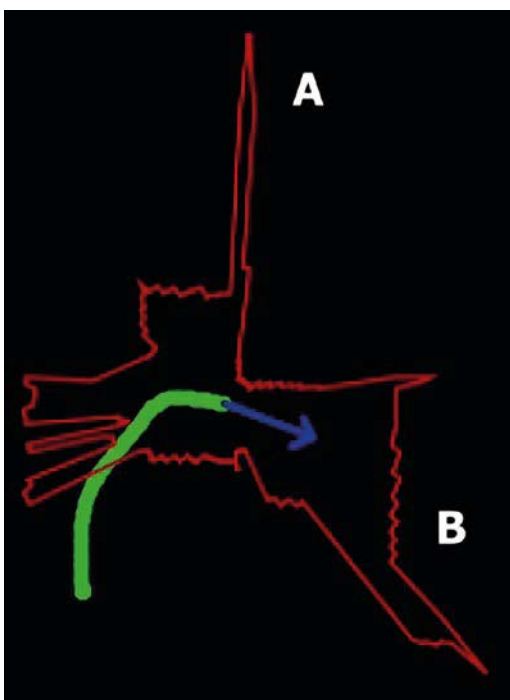


◀ **Fig. 7**
Orientamento
del robot.

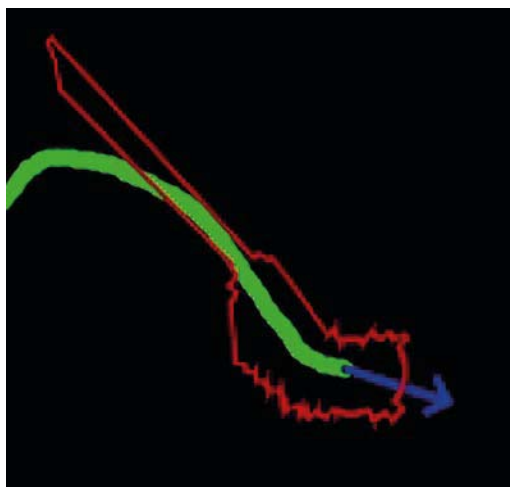
➔ **Fig. 8**
Spiegazione
della funzione
searchPath.



➔ **Fig. 9**
Funzione
searchPath in un
varco stretto.



➔ **Fig. 10**
Funzione
searchPath nel
punto cieco del
percorso.



alla funzione navigation che la usa per determinare il percorso del robot.

Relativamente a questa funzione di path planning si potrebbero scrivere libri e internet è pieno di articoli accademici. È necessario implementare un algoritmo di esplorazione, secondo un ragionevole criterio di selezione delle zone da esplorare.

Il nostro robot non deve raggiungere un preciso obiettivo ma dovrà circolare negli ambienti circostanti per mapparne gli spazi pertanto il nostro algoritmo dovrà semplicemente determinare la migliore traiettoria di marcia valutando in quale direzione c'è più spazio da percorrere.

Già questo non è un problema banale perché se anche il LIDAR dovesse fornire l'indicazione che di fronte al robot l'ostacolo si trova a 3 metri non è detto che ci sia un varco sufficientemente ampio perché il robot ci passi. Pertanto l'algoritmo dovrà identificare in quale direzione vi sia il varco più profondo sufficientemente ampio per il transito del robot tenendo conto dei parametri arcAngle e minGap definiti nel software (si veda il **Listato 1**). Il primo rappresenta l'arco di circonferenza ritenuto sufficiente per il transito del robot ed è impostato al valore 40°, Il secondo parametro è la minima profondità del varco per la quale il robot proseguirà la marcia. Se non vi è un varco sufficientemente ampio e sufficientemente profondo la funzione ritorna il valore nullo ed il robot si trova di fatto in un "vicolo cieco" altrimenti ritorna l'angolo verso il quale il robot dovrà dirigersi. Aiutiamoci con alcune figure per spiegare questo algoritmo evidenziando in verde il percorso del robot, in blu la direzione ed in rosso la traccia fornita dal LIDAR. In **Fig. 8** sia il percorso A che B sono sufficientemente ampi però il percorso B è più profondo, questa sarà la direzione da seguire. In **Fig. 9** il percorso A è più profondo però il varco è troppo stretto, viene optato per il percorso B. In **Fig. 10** il robot ha raggiunto un punto cieco perché non vi è sufficiente spazio per proseguire, l'unica alternativa è una rotazione di 180° e riprendere il cammino da dove è arrivato.

Alla funzione descritta precedentemente se ne affianca un'altra di nome obstacle, riportata nel **Listato 2**, che ha il compito di analizzare l'ambiente nell'intorno del robot per capire se ci sono ostacoli che possano intralciare la marcia, soprattutto nei fianchi del robot.

La funzione obstacle ritorna il valore dell'angolo verso il quale è presente l'ostacolo più vicino al robot (se quest'ultimo è più vicino del parametro minObstacle), altrimenti ritorna zero ad indicare nessun impedimento alla marcia.

▼ Listato 2

```
int obstacle() {
  int searchAngle, obstAngle;
  int minDist=10000;
  for (searchAngle=90+30; searchAngle<270-30; searchAngle++) {
    if ( my_dist[searchAngle] > 0 && my_dist[searchAngle] < minDist )      {
      minDist=my_dist[searchAngle];
      obstAngle=searchAngle;
    }
  }
  if (minDist<minOstacle) {
    Serial.println("obstAngle=" + String(obstAngle) + "\tmaxDist=" + String(minDist));
    return(obstAngle);
  } else
    return 0;
}
```

▼ Listato 3

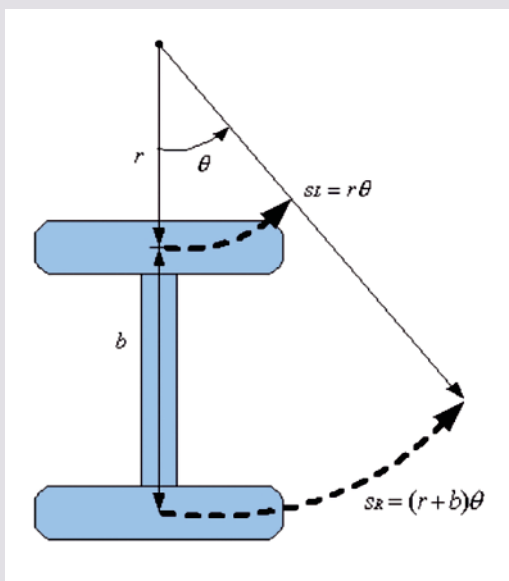
```
void navigation()
{
  int searchAngle=180;
  if (!isStart) stopMotor();
  if (isData) //ci sono dati validi sui quali poter decidere la navigazione
  {
    int steer;

    //verifico se ci sono ostacoli nelle vicinanze
    int obstAngle = obstacle();
    Serial.println("obstAngle=" + String(obstAngle));

    if (obstAngle>90 && obstAngle<=180)
    {
      steer=float(robotSpeed)*(obstAngle-90)/300.0;
    }
    else if (obstAngle>180 && obstAngle < 270)
    {
      steer=float(robotSpeed)*(90-obstAngle)/300.0;
    }
    else
    {
      //non ci sono ostacoli cerco il percorso migliore
      int optiDir = searchPath(); //cerco l'angolo nel quale c'è il varco più profondo
      Serial.println("optiDir=" + String(optiDir));
      if (optiDir<1) //non c'è un varco abbastanza profondo
      {
        //vicolo cieco torno indietro
        // la ruota deve percorrere pi*roadWay/2
        float dist=PI*roadWay/2.0;
        long numtick= 360.0*dist/(PI*DR2);
        Serial.print("at a dead end! ");
        Serial.println(numtick);
        stopMotor();
        delay(500);
        enc1=encoder1();
        enc2=encoder2();
        move(128-robotSpeed, 128+robotSpeed);
        while(encoder2()<enc2+numtick) ;
        stopMotor();
        delay(500);
        isStart=true;
        steer=0;
      }
      else
      {
        steer=float(robotSpeed)*float(optiDir-180)/100.0; //la sterzata è proporzionale a dove devo andare
      }
    }
  }

  if (isStart) move(128+robotSpeed+steer, 128+robotSpeed-steer);
}
```

➔ Fig. 11
Odometria
del robot.



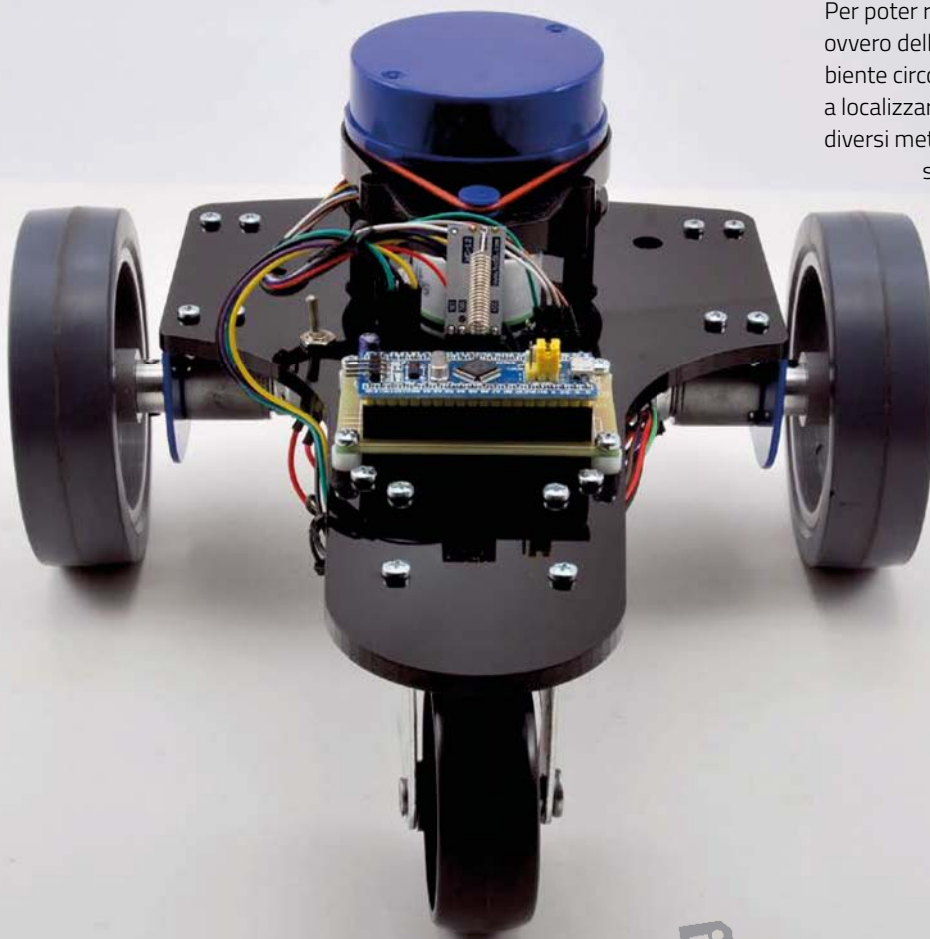
La funzione obstacle ha precedenza rispetto la funzione searchPath perché è prioritario che il robot non vada a sbattere con niente, comprese le gambe di qualcuno che sta passando casualmente nei paraggi. Utilizzeremo le informazioni precedentemente ricavate per comandare opportunamente

la marcia del robot impostando separatamente la velocità di rotazione delle due ruote applicando un controllo di nome PID (funzione navigation riportata nel **Listato 3**). Vogliamo che il robot si muova in modo fluido e costante senza improvvisi cambi di direzione che potrebbero portare ad errori nella stima della posizione. In sintesi il controllo PID prevede di impostare una velocità di marcia (robotSpeed nel programma) che viene mantenuta finché non vi sia la necessità di cambiare direzione. Il cambio di direzione avviene aumentando la velocità di una ruota e conseguentemente la diminuzione dell'altra lasciando inalterata la velocità del robot. La virata avviene però proporzionalmente alla nuova direzione da seguire, pertanto se la via da seguire si trova a 220 gradi la sterzata sarà maggiore del caso di una virata verso i 200 gradi (riferitevi alla **Fig. 7**); se il percorso ottimale fosse a 180° non vi sarebbe alcuna sterzata ed il robot proseguirebbe dritto.

Consigliamo di visionare i filmati pubblicati sul nostro canale YouTube (trovate l'indirizzo nell'apposito box) per comprendere come questi algoritmi abbiano effetto sul robot.

ODOMETRIA

Per poter risolvere il problema del map-building ovvero della costruzione di una mappa dell'ambiente circostante è necessario che il robot riesca a localizzarsi all'interno di questa mappa. Ci sono diversi metodi per raggiungere questo scopo a seconda di dove il robot si muove, ad esempio in un drone in volo un modulo IMU potrebbe già essere una buona base di partenza, nel nostro caso il robot muovendosi su ruote sfrutterà la lettura degli encoder per determinare il percorso svolto, in questo caso si parla esplicitamente di odometria (Dead Reckoning in inglese). Fortunatamente



l'odometria per robot su ruote che utilizza gli encoder come sensori è già stata ampiamente studiata e riportata nella documentazione del settore. In parole semplici possiamo dire che conteggiando gli impulsi degli encoder della ruota sinistra e destra possiamo ricavare il numero di rotazioni di ciascuna ruota e noto il loro diametro valutare il percorso svolto. Questa metodologia soffre di alcune limitazioni intrinseche nel sistema utilizzato. Per prima cosa il sistema di posizionamento è relativo, quando il robot entra in funzione esso si trova necessariamente nelle coordinate 0,0. Questo di per sé non è un problema però alcuni errori sistematici possono influire negativamente nell'accuratezza della stima della posizione tra i quali ricordiamo le imprecisioni meccaniche nella costruzione, diverso diametro delle ruote la risoluzione finita degli encoder. A questo poi si sommano gli errori casuali dovuti allo scivolamento delle ruote, movimento su pavimenti imprevisti o non solidali (tappeti) forze esterne che portano a deviazioni impreviste. Se necessario il riallineamento del robot avviene tramite il posizionamento di boe attive che forniscono al robot una posizione certa su cui basarsi per una taratura. Vediamo ora in dettaglio quali calcoli è necessario fare, tutto si basa su alcuni passaggi di trigonometria. Noto il percorso di ciascuna ruota possiamo ricavare il percorso medio del robot e l'angolo di rotazione conseguente alla marcia, si faccia riferimento alla **Fig. 11** per la successiva spiegazione. Ad intervalli regolari viene letto il valore dei due encoder e calcolata la distanza percorsa da ciascuna

ruota con le formule:

$$SL = \text{countL} \cdot DR \cdot \pi / 360$$

$$SR = \text{countR} \cdot DR \cdot \pi / 360$$

Dove countL e countR sono rispettivamente i conteggi di ciascun encoder nell'arco temporale considerato e DR è il diametro della ruota e b è la carreggiata del robot. Noti questi due valori possiamo determinare lo spazio totale percorso dal robot ST, l'angolo di rotazione del robot θ e le coordinate raggiunte. I valori X_0 e Y_0 e θ_0 sono i valori precedentemente ricavati.

$$ST = (SL + SR) / 2$$

$$\theta = \theta_0 + (SR - SL) / b$$

$$X = X_0 + ST \cdot \cos(\theta)$$

$$Y = Y_0 + ST \cdot \sin(\theta)$$

Notate come le nuove coordinate sono determinate a partire da quelle precedenti, da cui il problema che un eventuale errore sistematico andrebbe a sommarsi in continuazione causando un errore nel lungo periodo a volte inaccettabile. Noto lo spostamento del robot ed il tempo tra una lettura e l'altra è anche possibile determinare la velocità del robot. Nel **Listato 4** potete vedere le righe di codice che implementano l'equazione appena descritte. Dalle prove da noi condotte abbiamo comunque riscontrato che il sistema funziona molto bene tanto

Listato 4

```
void odometry() {
    timeNow=millis();
    enc1=encoder1();
    enc2=encoder2();
    timeDt=timeNow-timePre;
    timePre=timeNow;
    deltaEnc1 = enc1-enc1_old;
    deltaEnc2 = enc2-enc2_old;
    distW1=PI*DR1*deltaEnc1/encTick; //distanza percorsa dalla ruota 1 in mm
    distW2=PI*DR2*deltaEnc2/encTick; //distanza percorsa dalla ruota 2 in mm
    distW1W2=(distW1+distW2)/2.0; //distanza percorsa dal robot
    Speed=1000*(float(distW1+distW2)/2.0)/float(timeDt); // velocità del robot in mm/sec
    Angle=Angle+float(distW1-distW2)/roadWay; //angolazione del robot in radianti
    posX = posX_old + distW1W2*sin(Angle);
    posY = posY_old + distW1W2*cos(Angle);
    posX_old=posX;
    posY_old=posY;
    enc1_old = enc1;
    enc2_old = enc2;
}
```

Fig. 12
Software STM32
Flash loader
demonstrator
riconoscimento
MCU..

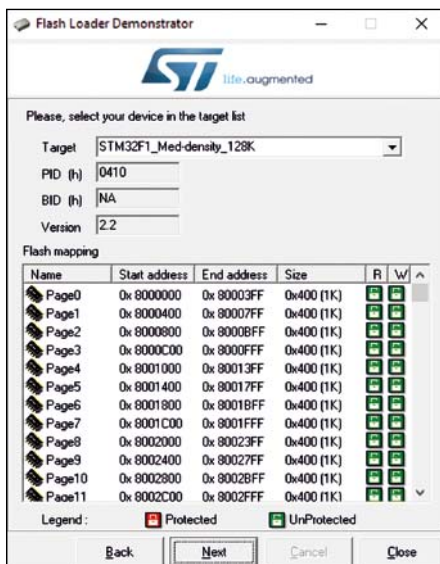


Fig. 13
Software STM32
Flash loader
demonstrator
caricamento
firmware.



Fig. 14
Software STM32
Flash loader
demonstrator
caricamento
completato.



che con un percorso di 13 metri l'errore è rimasto limitato a pochi centimetri. La cosa importante è curare attentamente la meccanica e l'allineamento di tutte le parti e misurare con la massima attenzione il diametro delle ruote, ritoccando eventualmente il valore inserito nel software.

TELEMETRIA

Il robot così programmato è in grado di operare in completa autonomia, infatti l'acquisizione e l'elaborazione di tutti i segnali avviene all'interno della CPU del robot, è però interessante poter vedere ed analizzare la mappa ricreata dal robot in tempo reale, in una postazione remota che raccolga in una dashboard i dati raccolti dal robot. Vi è quindi la necessità di poter trasmettere verso

Tabella 2
Collegamenti elettrici per la programmazione.

STM32	CONVERTITORE USB/SERIALE
GND	GND
PA9 (TX)	RX
PA10 (RX)	TX
3V3	VCC (3,3V)

una postazione remota i dati acquisiti tramite un sistema di telemetria in tempo reale. Una prima soluzione potrebbe essere l'utilizzo di un economico modulo Bluetooth tipo HC-05 programmato per il trasferimento seriale alla velocità di 57.600 bps, però il segnale si smorzerebbe rapidamente in presenza di muri o lunghe distanze. Una seconda soluzione è stata l'utilizzo di moduli XBee che però presentano un costo elevato, la versione definitiva vede l'utilizzo dei moduli ricetrasmittitori HC-12 (codice HC12). Si tratta di moduli molto compatti a basso costo con una portata che può arrivare sino al chilometro, perfetti per la nostra applicazione. Questi moduli sono abbastanza esigenti in fatto di alimentazione quindi è necessario porre nelle immediate vicinanze dei pin di alimentazione un condensatore del valore di 470 nF. Appena usciti dalla confezione sono pronti al funzionamento ma la velocità di trasmissione è impostata a 9.600 baud, mentre per la nostra applicazione la velocità di trasmissione deve essere impostata a 57.600 baud. Per modificare le impostazioni è necessario interfacciare i moduli al PC tramite un convertitore USB-Seriale impostato su 3,3V (che ci

servirà anche per programmare il microcontrollore STM32, il convertitore USBTOUART è perfetto allo scopo) e portare la linea SET a livello logico basso, in questo modo il modulo si pone in modalità di comando AT. Dal lato PC dovremmo avviare un qualsiasi software per comunicazioni seriali come Putty o CoolTermWin, aprire la comunicazione seriale a 9600 baud ed inviare i seguenti comandi: AT+P4 e successivamente AT+57600.

Il comando AT+P4 permette di ridurre la potenza di trasmissione riducendo la corrente assorbita, mentre AT+57600 imposterà la corretta velocità di comunicazione; ad ogni comando il modulo risponderà con la conferma di avvenuta modifica. Dovremmo programmare due moduli HC-12: uno sarà connesso alla scheda di controllo del robot mentre il secondo sarà connesso al PC tramite l'adattatore USB-Seriale. Nelle foto in queste pagine potete vedere il robot completamente assemblato, la batteria è fissata sotto per mantenere basso il baricentro e migliorare la stabilità.

PROGRAMMAZIONE DELLA SCHEDA

Terminato l'assemblaggio dell'hardware proviamo alla programmazione dell'STM32, per fare questo dovete procurarvi un convertitore USB-Seriale funzionante a 3,3 volt e connetterlo come descritto in **Tabella 2**. Assieme ai file della rivista troverete i sorgenti del progetto ma anche la versione compilata molto più immediata da utilizzare per la programmazione del microcontrollore. Su questa scheda sono presenti due jumper usati per selezionare il modo di funzionamento; per abilitare la funzione di boot da seriale dovrete posizionare il jumper di nome BOOT0 su 1 e BOOT1 su 0 e successivamente premere il pulsante di reset. Dovete scaricare e installare il software STM32 Flash loader demonstrator disponibile sul sito ufficiale della ST e seguire la procedura seguente: avviate il software e selezionate la COM del convertitore, quindi cliccate su NEXT e verificate che il software riconosca il microcontrollore (**Fig. 12**), proseguite e aprite il firmware compilato *YDLidarRobot_V102.bin* ed avviate la programmazione (**Fig. 13**), attendete il completamento delle operazioni, **Fig. 14**. Terminata la programmazione staccate il programmatore e riposizionate entrambi i jumper in posizione 0.

BASE STATION

La postazione remota che si occupa di ricevere i dati dal robot (Base Station) consiste in un PC con un software da noi elaborato. Come software di

Guarda il robot in azione su YouTube!



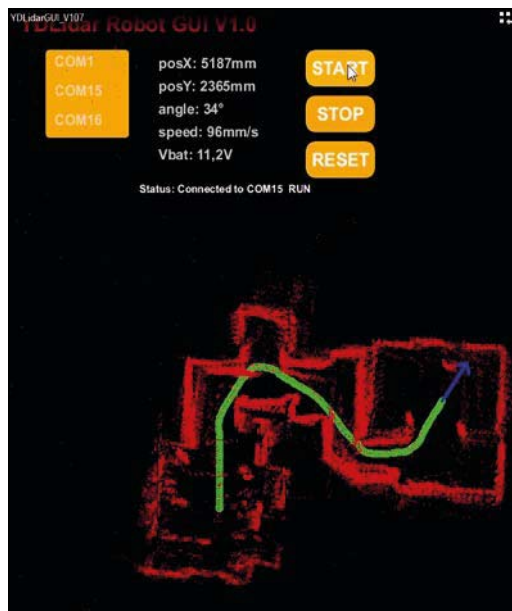
Vuoi vedere in azione il nostro robot in azione? Nulla di più semplice: collegati alla pagina <https://youtu.be/6wm6O00cPrs> Potrai anche vedere i video di moltissimi altri nostri progetti proposti in passato.

sviluppo abbiamo scelto Processing che è il cugino di Wire il linguaggio di programmazione usato per Arduino IDE, un linguaggio approntato alla grafica che mette a disposizione alcuni strumenti avanzati per la grafica 2D e 3D. Inoltre abbiamo il vantaggio di usare due ambienti di programmazione simili semplificando il lavoro di programmazione; assieme ai file di questo numero trovate tutti i sorgenti. L'interfaccia è alquanto spartana e volutamente semplificata per poter essere fruibile agevolmente anche su dispositivi a schermo piccolo come note-



Fig. 15
Il software
YDLidar GUI.

Fig. 16
Funzione SLAM.



book. Quello che vedremo a video sarà il tracciato del robot (traccia verde) con il contorno delle ultime misure effettuate dal modulo LIDAR (traccia rossa) ovvero gli ostacoli nei dintorni del robot, come visibile in **Fig. 15**.

Il software riporta anche la posizione calcolata dal robot, la sua velocità attuale, l'angolazione e la tensione della batteria. Con i tre pulsanti di colore arancione potremmo comandare il robot ed eventualmente resettare i dati di misura e del grafico. Il software non necessita di installazione ed è multiplatforma, pertanto potrà essere usato tanto su SO Windows che Linux. Appena avviato, in alto a sinistra, potrete selezionare la COM sulla quale è presente il convertitore USB-Seriale con il modulo radio HC-12; questa operazione consente anche di avviare la comunicazione. Il software attende risposta dal robot e se per qualche motivo la connessione venisse interrotta il software lo avviserà con apposito messaggio.

FUNZIONE SLAM

Il lavoro del software sulla base station in congiunzione con i dati rilevati dal robot ci serviranno allo SLAM (dall'inglese Simultaneous Localization and Mapping) processo per cui un robot che si muove in un ambiente sconosciuto, costruisce la mappa di tale ambiente e si localizza all'interno di essa. La funzione SLAM si attiva cliccando sulla casella di spunta in alto a destra. Dovremmo fare alcuni calcoli che sfoceranno in un grafico bidimensionale in cui vengono plottati in continuazione con persistenza i dati forniti dal modulo LIDAR op-

portunamente posizionati ed orientati secondo le informazioni sul posizionamento ed orientamento del robot come, visibile in **Fig. 16**.

Si tratta di un processo abbastanza complesso che necessita di parecchio calcolo computazionale che nelle forme più avanzate considera e corregge l'imprecisione dei dati rilevati affetti da rumore. Sempre nuovi algoritmi si stanno sviluppando attorno a questa tecnologia spinti dall'esigenza della guida autonoma sia indoor che outdoor anche in ambienti 3D. Spesso questi algoritmi necessitano però di piattaforme hardware più performanti e grandi capacità di memoria. La mappa così ottenuta potrà essere usata dal robot per migliorare l'efficienza nei successivi spostamenti e per identificare punti di interesse nel percorso come ingressi e uscite piuttosto che basi di ricarica. Immaginate un robot aspirapolvere con la batteria quasi scarica, se avesse in memoria la mappa dell'area di lavoro e conoscesse la sua posizione, potrebbe pianificare il percorso più breve per raggiungere la base.

CONCLUSIONI

Il lavoro per arrivare ad un robot funzionante è stato notevole, soprattutto nello sviluppo del software ma il risultato è decisamente affascinante. Meglio di tante parole, i video realizzati durante i test finali possono chiarire il funzionamento. Questo articolo potrà fornire importanti informazioni per capire meglio come vengono progettati e realizzati i robot aspirapolveri che sempre più spesso vediamo in funzione nelle nostre abitazioni e che hanno dato notevole contributo al miglioramento di questa tecnologia, abbassando notevolmente i costi dei laser scanner. Algoritmi sempre più ottimizzati permetteranno di ottenere migliori prestazioni in sistemi embedded sempre più economici. □



Cosa occorre?

I componenti utilizzati in questo progetto sono disponibili presso Futura Elettronica. Il sensore LIDAR (cod. FR704) è in vendita a Euro 124,00, il driver motori (cod. MD25) a Euro 70,00, il motore con encoder (cod. EMG30) a Euro 48,00, la ruota (cod. WHEEL100) a Euro 17,00, il modulo radio (cod. HC12) a Euro 7,80, la board con STM32 (cod. STM32ARMCORE) a Euro 7,50. I prezzi si intendono IVA compresa.

Il materiale va richiesto a:

Futura Elettronica, Via Adige 11, 21013 Gallarate (VA)
Tel: 0331-799775 - <http://www.futurashop.it>

Nuovo allarme evoluto WiFi-GSM/GPRS



€119,00

Cod. FR783



Controllabile con:



Smart Life



Tuya Smart



Amazon Alexa



Google Home

Centralina antifurto wireless programmabile, controllabile da remoto tramite Smartphone sfruttando la rete GSM/GPRS o Wi-Fi e direttamente con Amazon Alexa o Google Home. Le App "Smart Life" o "Tuya Smart" permettono di configurare e gestire il funzionamento dell'allarme in ogni suo dettaglio. È possibile aggiungere sensori ed accessori opzionali per coprire fino a 99 zone. La confezione comprende: la centralina, due radiocomandi, un sensore wireless per porte/finestre, un sensore PIR wireless, due chiavi TAG RFID, la batteria di backup e l'alimentatore di rete. Manuale di utilizzo in italiano.

Prezzi IVA inclusa



Accessori
opzionali:

Sensore PIR
wireless



€14,00

Cod. FR783PIR

Sensore
porte/finestre
wireless



€11,00

Cod. FR783MAGN

Radiocomando
supplementare



€9,00

Cod. FR783RAD

Transponder
a portachiavi



€5,00

Cod. FR783TAG

Sirena wireless
da interno



€29,00

Cod. FR783SIR

Sirena wireless
da esterno con
pannello solare



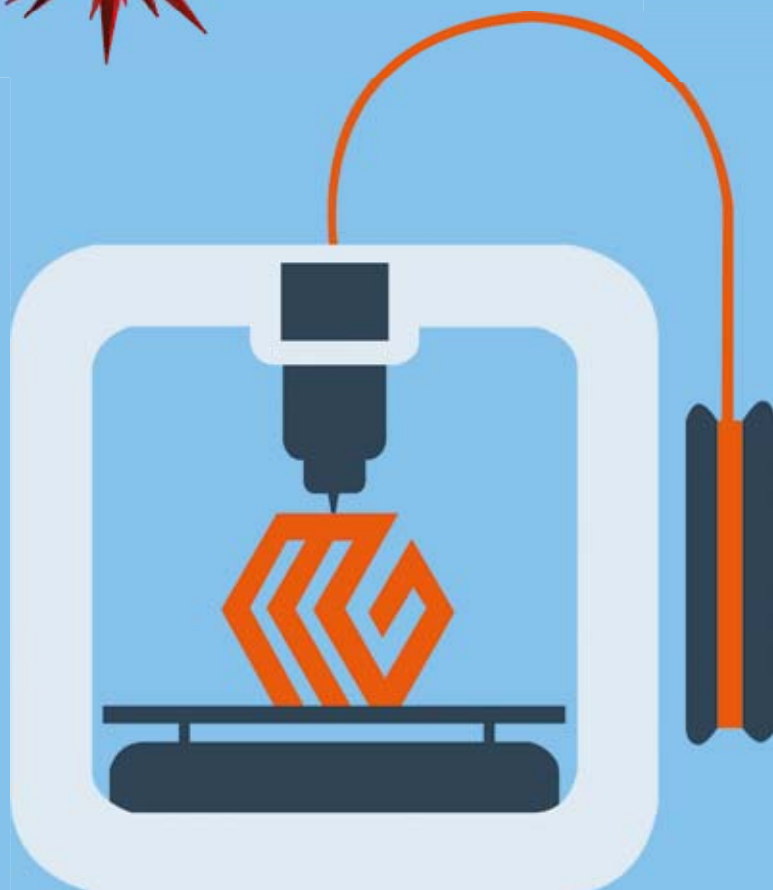
€65,00

Cod. FR783SIREXT

Trasformiamo le tue idee in realtà!

Servizio taglio laser e incisione

Futura Elettronica ti offre il servizio di taglio laser ed incisione per realizzare pannelli, parti meccaniche, mascherine, scritte, decori ed altro ancora in plexiglass (o metacrilato).



Servizio stampa 3D

Futura Elettronica mette inoltre a disposizione le proprie stampanti 3D FDM, per chi vuole realizzare un oggetto tridimensionale con dimensioni fino a 40x40x40 cm.

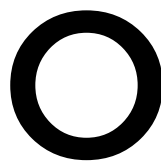
Informazioni dettagliate su
www.futurashop.it

HACKING WiFi

di ANDREA CHIAPPORI



Analizziamo i messaggi WiFi che vengono scambiati tra un'app del cellulare ed una lampadina smart WiFi per riprodurli dal nostro pc e successivamente da un ESP8266.

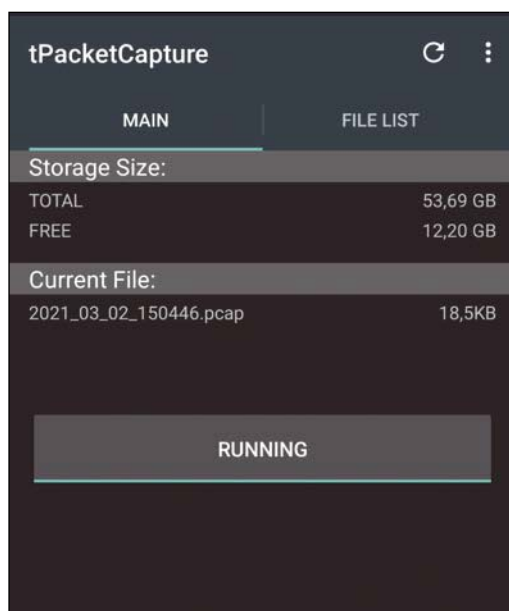


Ormai la domotica appartiene al nostro vivere quotidiano e facilmente avremo in casa una lampadina WiFi controllabile da qualche app del nostro cellulare o da un gateway hardware come Alexa, Google Assistant e tanti altri.

Ne esistono tantissime, ognuna di marche e modelli diversi, alcune controllabile solo in tonalità di bianco, altre sono a colori RGB, altre ancora in entrambe le modalità (RGBW). In questo articolo analizzeremo a fondo i messaggi che vengono scambiati tra l'app del nostro cellulare e una tra le lampade WiFi più usate, cercando di capire il protocollo usato e provando a riprodurre gli stessi messaggi per controllare la lampadina da PC e successivamente dall'ormai famoso chip ESP8266, con il quale saremo in grado di accendere la lampadina e cambiarne il colore a nostro piacimento, senza dover necessariamente usare l'app del cellulare o sfruttare server esterni centralizzati.

Il procedimento che andremo ad illustrare ci porterà all'interno dei pacchetti TCP che contengono i comandi di pilotaggio della lampadina, navigando tra i vari byte per cercare di

Fig. 1
App per catturare
traffico di rete su
cellulare.



capirne il significato ed intervenire per modificarli opportunamente e prendere il controllo della lampadina stessa senza dover dipendere dalla specifica app fornita in dotazione. Ovviamente ad un certo punto dovremo scendere in dettagli che sono specifici solo per quella particolare lampadina che abbiamo utilizzato, ma con questo articolo vogliamo illustrare un processo che potrà essere applicato anche su altri prodotti WiFi.

Stiamo parlando di lampadine a LED che possono essere regolate nelle varie tonalità di colore (RGB), o solo nel bianco (W) o in entrambe le modalità (RGBW), nel nostro caso abbiamo usato un tipo di lampadina non brandizzata (quindi esternamente non ha presenta alcun marchio visibile) disponibile da Futura Elettronica con il codice LEDWIFIBULB. Si tratta di una lampadina prodotta da "Tuya Smart" una grossa compagnia cinese specializzata nella produzione di hardware IoT, che quindi si occupa, non solo di illuminazione, ma di più svariati dispositivi per la casa come condizionatori, interruttori, prese intelligenti, ecc..

Generalmente, un po' in tutte le lampadine, si può intervenire nella luminosità e nella saturazione per quanto riguarda il colore mentre nella modalità luce bianca si potrà intervenire nella luminosità e nella temperatura.

Ogni lampadina presenta un'app di riferimento (nel nostro caso SmartLife) con la quale viene, in una prima fase associata al nostro router, e successivamente controllata in ogni suo aspetto.

Con questa app possiamo anche creare scene ed effetti particolari ed impostare timer di accensione

e spegnimento. In realtà possono esistere molte altre app per controllare queste lampade, come TuyaSmart, TuyaApp, FamilyCloud che, anche se apparentemente sembrano pressoché identiche, differiscono per la modalità con cui inviano comandi alla lampadina; alcune ad esempio usano server cinesi centrali con cui scambiano comandi tramite protocollo MQTT o Web API. Esistono poi anche diversi plugin in rete (quasi per ogni tipo di lampadina) per far in modo che esse possano essere associate a numerose piattaforme di assistenza domotica come Home Assistant, OpenHab, Domoticz e tante altre che sono sviluppate per Raspberry Pi, e che possono fornire spunto di analisi.

PRIMI PASSI

Il nostro obiettivo è quello di prendere il controllo della nostra lampadina inizialmente da PC (per comodità di intervento) e successivamente da un microcontrollore senza usare nessuna app, anche se la useremo per la nostra analisi. Prima di tutto installiamo la nostra lampadina, associandola al nostro router, secondo le istruzioni del produttore, come da "normali" utilizzatori.

Solitamente si tratta di una particolare sequenza di accensione e spegnimento della lampadina in modo che quest'ultima entri in modalità configurazione. In questa fase il modulo WiFi interno alla lampadina entra in modalità Access Point, venendo a comportarsi come un router, con un proprio nome e password; il cellulare cercherà in automatico questa rete WiFi per connettersi ad essa. Ovviamente questa connessione è solo temporanea perché il nostro cellulare perderà la rete correntemente in uso e non potrebbe più connettersi in Internet, ma serve esclusivamente per trasferire alcuni dati alla nostra lampadina, come ad esempio il nome della rete del nostro router e la password. Una volta configurato, il modulo WiFi, torna in modalità Station, ma con le nuove informazioni acquisite, potrà connettersi al nostro router e quindi anche alla rete internet. Fin qua nulla di particolare, ma siamo già in grado di ottenere informazioni sul tipo di lampadina, perché nella fase di configurazioni, se eseguiamo una scansione delle reti WiFi, potremmo trovare il nome del modulo della lampadina; se ad esempio, dovesse iniziare con la stringa ESP seguito da qualche numero, capiremmo (come nel nostro caso) che all'interno della nostra lampadina si trova proprio un modulo ESP8266 o un ESP32 ed è un primo passo per capire con quale modulo abbiamo a che fare.

APP TPACKETCAPTURE

A questo punto, dopo aver verificato che con l'app standard sul cellulare, riusciamo ad accendere e cambiare colore (o luminosità di bianco) della nostra lampadina, possiamo indagare sui pacchetti di rete che questa app invia alla lampadina.

Per fare questo ci vengono in aiuto alcune app per cellulare, che memorizzano su file, tutto il traffico di rete tra cellulare e mondo esterno, sia in entrata che in uscita. Hanno solitamente un pulsante di start, uno di stop e la possibilità di salvare il file nella memoria interna. Nel nostro caso abbiamo usato TpacketCapture, come mostrato in **Fig. 1**, ma ne esistono diverse.

Quindi come prima cosa, dopo aver fatto partire l'acquisizione, apriamo l'app specifica della lampadina e premiamo il comando per accendere la lampadina; stoppiamo acquisizione, recuperiamo dal cellulare il file appena generato (solitamente un file .pcap) e lo analizziamo su PC tramite il programma Wireshark uno dei più conosciuti analizzatore del traffico di rete, scaricabile gratuitamente a questo indirizzo www.wireshark.org/download.html.

WIRESHARK

Ovviamente nel nostro file sarà presente tutto il traffico di rete e non solo quello generato dalla

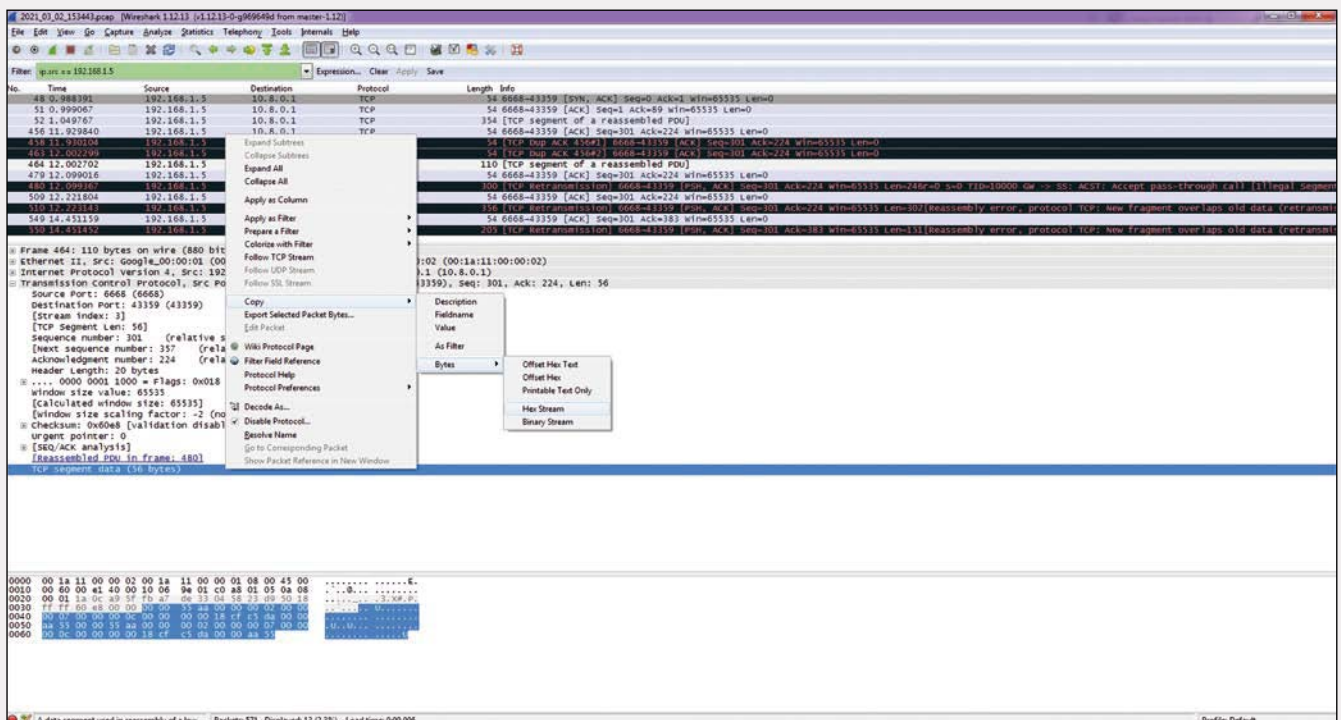
nostra app, ma con Wireshark possiamo applicare alcuni filtri per visualizzare solo quello interessato. Innanzitutto dovremmo conoscere l'indirizzo IP che il nostro router ha assegnato alla lampadina ed al nostro cellulare e questo si può facilmente ottenere dalle impostazioni del nostro router; quindi imposteremo un filtro del tipo: `ip.dest == 192.168.1.5` (dove l'indirizzo IP sarà quello della lampadina) e verranno visualizzati i pacchetti TCP, solitamente più di uno, inviati dal cellulare alla lampadina (**Fig. 2**) che possiamo analizzare singolarmente.

Se siamo fortunati, questi dati potrebbero essere già in chiaro e quindi potremmo leggere alcune stringhe di testo, magari già esplicative, che rappresentano il comando inviato alla lampadina; in questo caso il lavoro è praticamente già finito anche perché alcune lampade WiFi in commercio forniscono la possibilità di controllare le proprie lampade e rendono pubblici i vari comandi nella sezione developer del proprio sito. Basterà incapsulare i vari comandi in un pacchetto TCP ed inviarlo alla lampadina nelle modalità che vedremo più avanti.

Se invece non siamo così fortunati ed i dati inviati sono in formato binario, occorrerà investigare ancora per cercare di capire il formato.

Potrebbe esserci anche il caso in cui non riusciamo

Fig. 2
Interfaccia utente di Wireshark durante una cattura.



↓ Listato 1 - Codice C# per invio di un pacchetto TCP

```
public int SendMessageTCP(string[] stream)
{
    Thread t = new Thread(new ThreadStart(ListenData));
    t.Start();

    try
    {
        IPAddress ipAddress = IPAddress.Parse(_ipLamp);
        IPEndPoint remoteEP = new IPEndPoint(ipAddress, _portLamp);

        // Create a TCP/IP socket
        if (senderTcp == null)
        {
            senderTcp = new Socket(ipAddress.AddressFamily, SocketType.Stream, ProtocolType.Tcp);
            senderTcp.Connect(remoteEP);
            Console.WriteLine("Socket connected to {0}", senderTcp.RemoteEndPoint.ToString());
        }

        for (int i = 0; i < stream.Length; i++)
        {
            string msg = stream[i];
            byte[] msgbytes = convertStreamHex(msg);

            senderTcp.BeginSend(msgbytes, 0, msgbytes.Length, SocketFlags.None, new AsyncCallback(SendCallBack), null);
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.ToString());
        return -1;
    }
    return 0;
}
```

mo a vedere nessun pacchetto TCP inviato alla lampadina, questo potrebbe essere dovuto al fatto che l'app si appoggia direttamente a server esterni ed invii messaggi alla lampadina attraverso essi usando alcuni protocolli come ad esempio MQTT. In questo caso si consiglia di scaricare altre app compatibili o versioni precedenti della stessa app in cui la comunicazione avviene ancora tramite traffico TCP.

DUPLICAZIONE COMANDI

Torniamo ora al caso in cui i pacchetti TCP che stiamo esaminando su Wireshark siano in formato binario; essi sono quelli che hanno fatto accendere la nostra lampadina e anche se non sappiamo ancora cosa vogliano significare i singoli byte che li compongono, possiamo comunque replicarlo dal nostro PC ed, in seguito, da un ESP8266. Ogni linguaggio di programmazione (come C++, C#, python, java) ha le proprie funzioni primitive per invio e ricezione di pacchetti TCP e UDP che possiamo utilizzare per il nostro scopo. Basterà

conoscere l'IP e la porta (anch'essa visibile tramite Wireshark) della lampadina ed inviare lo stesso messaggio, o, per sicurezza, la stessa sequenza di messaggi contenuti nei pacchetti TCP che abbiamo ritrovato su Wireshark.

Per farlo ci viene in aiuto proprio l'interfaccia grafica di Wireshark che permette di copiare negli appunti, tramite previa selezione dei dati e poi "Copy Byte HexStream" (come mostrato in **Fig. 2**) solo la parte dati del pacchetto tcp verso la lampadina, quindi quello che chiameremo payload e che è privo di tutte gli header del protocollo TCP. Tale copia è in formato testo e rappresenta una stringa di byte consecutivi in formato esadecimale, quindi dal momento che solitamente queste primitive operano su array di byte, occorrerà effettuare una opportuna conversione.

Nel nostro caso useremo c# con l'IDE di Visual Studio, ed il caso specifico di invio di un messaggio TCP è visualizzato nel **Listato 1**.

In particolare il parametro che passiamo alla funzione è un array di stringhe perché gli passeremo tutti i



In seguito vedremo che in realtà ne può bastare solamente uno.

Associamo questo codice ad un pulsante che chiameremo ad esempio, "Turn On" e verifichiamo che premendolo si accenda la lampadina.

Ora si possono ripetere questi passaggi per lo spegnimento della lampadina, e successivamente per un determinato colore e per i vari comandi a cui siamo interessati, uno alla volta.

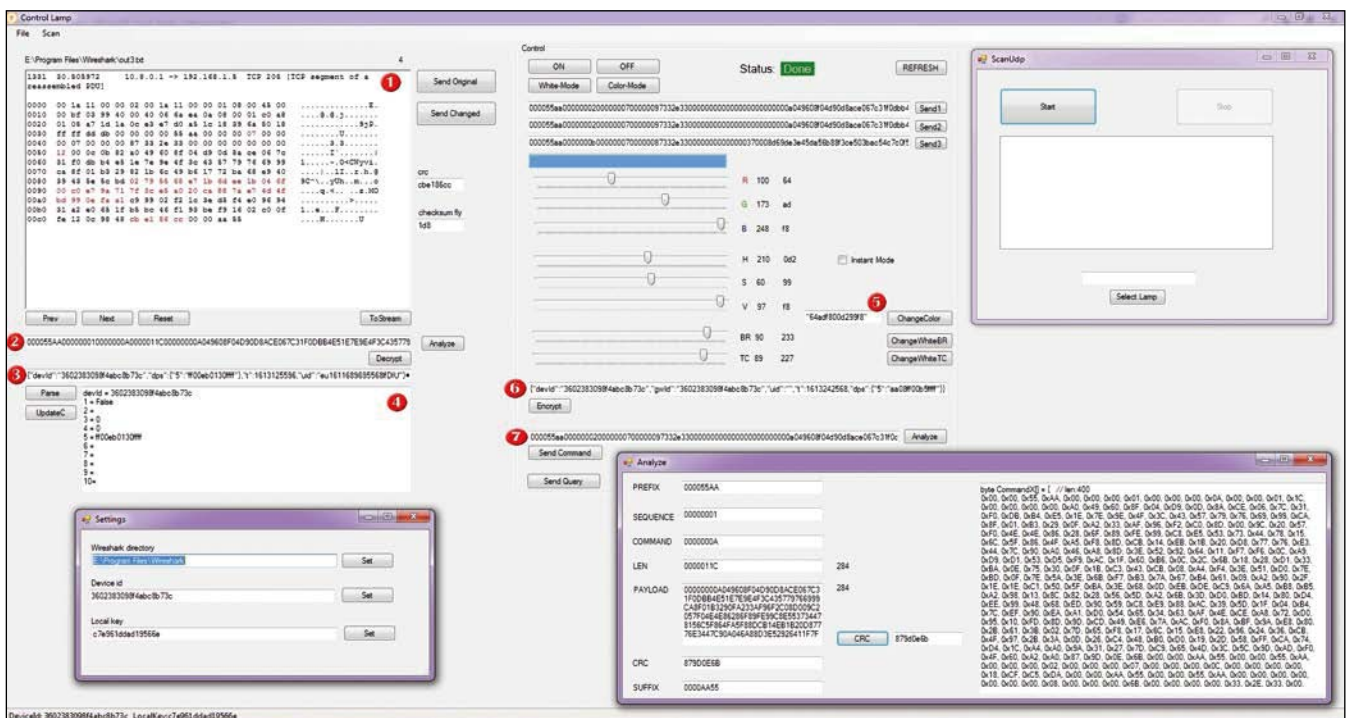
Questa non è di sicuro la soluzione ottimale e versatile per una serie di motivi; intanto si tratta di comandi cablati nel codice, potrebbe andare ancora bene se volessimo solo accendere e spegnere la lampadina con un colore fisso, ma se volessimo scegliere anche solo tra una decina di colori dovremmo inviare pacchetti TCP preconfezionati che comunque occuperebbero molto spazio in memoria, specialmente quando andremo a fare il porting su un microcontrollore in cui lo spazio diventa una variabile critica.

Inoltre non dimentichiamo che stiamo parlando di pacchetti binari di cui non conosciamo ancora nulla e che potrebbero non avere più lo stesso significato dal momento in cui svitiamo la nostra lampadina dalla sede e la riavvitiamo per esempio in un'altra stanza; e dover rifare ogni volta questi procedimenti porta comunque via molto tempo.

Prima abbiamo usato un programma in C# per inviare un pacchetto TCP alla lampadina, ma avremmo potuto usare alcuni programmi per PC che permettono invio (e anche ricezione) di messaggi personalizzati TCP/UDP verso qualsiasi altro dispositivo nella rete, come ad esempio PacketSender (scaricabile gratuitamente dal sito <https://packetsender.com>) che può risultare sempre utile per prove veloci, tuttavia se optiamo per una soluzione su misura per la nostra lampadina avremo più vantaggi creando un nostro programma personalizzabile che in futuro potremmo usare e perfezionare anche per altri tipi di lampadine a seconda delle esigenze particolari.

Riprendiamo allora il nostro progetto in C# che ci è servito per i comandi base ed aggiungiamo una serie di controlli per facilitare la comprensione del formato dei byte nei pacchetti TCP sulla base di osservazione, esperimenti, tentativi ed un po' di fortuna. Sappiamo già come accenderla e spegnerla, vogliamo capire come scegliere un colore a nostro piacimento modificando i byte giusti. Un metodo potrebbe essere quello di sniffare il traffico tra cellulare e lampadina (sempre usando l'app TPacketCapture) mentre cambiamo colore ripetutamente e poi analizzare su Wireshark le differenze tra i vari pacchetti generati, osservare

Fig. 3
Programma C#
per analizzare
comunicazione
con la lampadina
WiFi.



le differenze e provare qualche cambiamento, rinviando il pacchetto modificato. Così, però, saremmo costretti a passare troppe volte tra il nostro programma e Wireshark e allora integreremo tutto nel programma, dandogli direttamente in pasto il file *.pcap* che verrà convertito internamente in un file di testo con la successione di tutti i pacchetti interessati. Per fare ciò useremo il programma *tshark.exe* che è contenuto all'interno della cartella di Wireshark e che praticamente altro non è che la versione da linea di comando di Wireshark. Il parametri che passeremo a *tshark.exe* sono:

```
-Y "ip.dst == 192.168.1.5 && tcp.len==151 &&
ip.proto== TCP && not tcp.analysis.retransmission"
-r color.pcap -w out.pcap -T text -x -P -S *** >
color.txt
```

Dopo aver scritto un po' di codice saremo in grado di scorrere tra i pacchetti inviati alla lampadina e responsabili del cambio di colore, segnare di colore rosso i byte che cambiano di volta in volta ed osservare quelli che rimangono uguali, cercando di capire se i primi byte possono rappresentare un header fisso e contenere alcune informazioni quali il tipo di comando, la lunghezza dell'intero pacchetto e qualche checksum di controllo. L'interfaccia del programma, presente in **Fig. 3** può sembrare complessa, ma è il frutto di test e delle varie difficoltà incontrate e ha preso forma via via sviluppandosi da sinistra a destra. O meglio, inizialmente era presente solo la parte sinistra che è tutto ciò che ci serve per studiare il cuore del payload inviato alla lampadina ed i vari meccanismi per decodificarlo, mentre sulla parte destra ci sono i controlli che invieremo alla lampadina una volta che capiremo come codificare i comandi.

UTILIZZO DEL PROGRAMMA C#

Segue una breve descrizione di alcuni controlli del programma C# utilizzato per effettuare vari test sulla lampadina WiFi, in riferimento ai pallini rossi nella **Fig. 3**.

- 1) Schermata di testo in cui vengono visualizzati i pacchetti TCP letti dal file pcap (precedentemente acquisiti da Wireshark o Tcpdump su cellulare) dopo averli caricati dalla voce di menù "File". Il pacchetto visualizzato può essere rinviato alla lampadina con pulsante "Send Original" o modificato testualmente per poi inviarlo con pulsante "Send Changed"
- 2) Casella di testo in cui viene trasferito solo il payload del pacchetto in seguito alla pressione

del tasto "ToStream". Questa casella di testo verrà inoltre popolata in automatico con la risposta della lampadina ai vari comandi.

- 3) Casella di testo in cui viene caricato il messaggio decodificato dal payload in seguito alla pressione del tasto "Decrypt". Ovviamente se si volesse decodificare un altro payload è possibile copiarlo direttamente nella casella di testo 2.
- 4) Schermata di testo in cui viene mostrata la stringa di testo in formato JSON in modalità più facilmente leggibile, in seguito alla pressione del tasto "Parse". Premendo il tasto "UpdateC" i valori contenuti verranno letti e gli slide del colore (RGB e HSV) sulla parte destra verranno aggiornati di conseguenza
- 5) Casella di testo che contiene la stringa di comando per regolare il colore della lampadina. Può essere modificato a mano, ma si aggiorna in automatico appena vengono modificati gli slide del colore (RGB e HSV). Premendo il tasto "Change color" a fianco, verrà aggiornato di conseguenza il comando nella casella 6, pronto per essere inviato.
- 6) Casella di testo che contiene il comando da inviare alla lampadina per cambiare colore. Può essere modificato manualmente o aggiornato in automatico con il testo fornito dalla casella 5, se vogliamo modificare il colore, oppure dai pulsanti "ChangeWhiteBR" e "ChangeWhiteTC", se vogliamo variare i parametri del bianco (in base alla posizione degli slide BR e TC).
- 7) Casella di testo in cui viene caricato il payload del pacchetto finale in seguito alla pressione del tasto "Encrypt". Può essere modificato manualmente inserendo un qualsiasi pacchetto valido.

Premendo il pulsante "Send Command" verrà confezionato un pacchetto TCP con questo pacchetto come payload ed inviato direttamente alla lampadina.

Premendo invece il pulsante "Send Query" verrà inviato un payload con il comando 0x0A che forzerà la lampadina a rispondere con un pacchetto contenente il suo stato, che sarà automaticamente mostrato nella casella 2 pronto per essere decodificato.

Altri pulsanti di controllo sono:

Pulsante "ON": invia comando per accendere la lampadina.

Pulsante "OFF": invia comando per spegnere la lampadina.

Pulsante "White-Mode": invia comando per passare a modalità bianco.



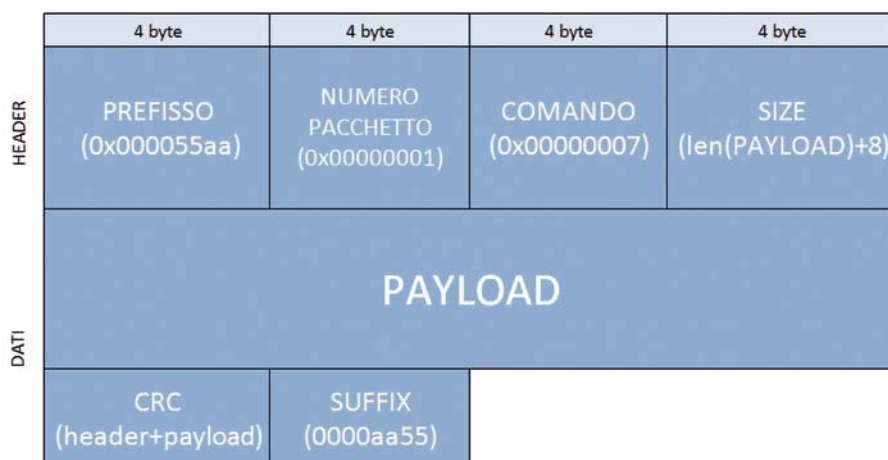


Fig. 4
Struttura
del payload
contenuto
nel pacchetto
TCP.

Pulsante "Color-Mode": invia comando per passare a modalità colore

Pulsante "REFRESH": nel caso in cui lo stato fosse error, resetta la creazione del socket di rete

CheckBox Instant Mode: se spuntato invia direttamente un comando alla lampadina sul rilascio della slide H forzando il cambio colore.

Premendo poi i pulsanti "Analyze", vicino alle caselle di testo 2 e 7, si aprirà una finestra che conterrà nella parte sinistra una suddivisione del pacchetto nei vari campi e nella parte destra una rappresentazione del pacchetto in forma di array da poter usare in un programma in C che useremo per i porting successivi. Premendo la voce di menù "Scan", si aprirà una finestra con cui il programma si mette in ricezione dei pacchetti UDP inviati da una o più lampade ogni tre secondi e ci permetterà di selezionare quella che vorremo controllare. Sempre dalla voce di menù "File", abbiamo la possibilità di usare la finestra "Settings" per impostare alcuni parametri essenziali come la directory in cui risiede la nostra installazione di Wireshark (per utilizzare il comando `xall` interno del programma), il DeviceID della lampadina e la Localkey entrambi usati all'interno dei comandi. Da notare, a tal proposito, come mentre il DeviceID è univoco della lampadina e non cambia mai, la LocalKey cambia ogni volta che la lampadina viene associata al router.

Detto questo, possiamo modificare qualche byte nella textbox dell'interfaccia indicata con 1 ed inviare il nuovo pacchetto così formato alla lampadina tramite pulsante "Send changed", tuttavia ancora non funziona perché probabilmente cambiando il valore di un byte cambierà anche la checksum o qualche CRC.

A questo punto siamo su PC quindi se avviamo

Wireshark facendolo acquisire in tempo reale, vedremo anche la risposta della lampadina con un flag di RST indicante un pacchetto non valido. Dopo ulteriore analisi e tentativi si trova il gruppo di byte sul quale viene calcolato il CRC e confrontato con i quattro byte in fondo al pacchetto. Si tratta di un CRC a 32 bit ed è facilmente riscontrabile sugli altri pacchetti utilizzando il tool online per il calcolo del CRC (<https://www.scadacore.com/tools/programming-calculators/online-checksum-calculator>) che fornisce tutti i tipi di CRC da uno stream di dati. Ora possiamo cambiare qualche byte, ricalcolare il CRC ed impostare quei quattro byte del CRC a quel valore, noteremo che non viene generato RST nella risposta della lampadina, quindi il comando viene accettato, ma ancora nessun cambiamento di colore. L'ulteriore complicazione è dovuta al fatto che i dati sono criptati e come ogni codice criptato occorre una chiave, ma sappiamo che si tratta di una lampadina Tuya e possiamo ottenere qualche informazione in più. Nel frattempo, però, dopo una serie di confronti tra i vari pacchetti, possiamo stabilire in linea di massima la struttura di un pacchetto base (per lampadine Tuya) che risulta essere come in **Fig. 4** e come appare anche in **Fig. 3** nella finestra "Analyze".

RECUPERO LOCALKEY

Stiamo entrando in una analisi sempre più approfondita e di conseguenza queste procedure diventano sempre più dipendenti dal tipo di lampadina. Altri modelli avranno metodi più o meno differenti per il recupero di queste informazioni, ma una volta compresi questi passaggi, anche se solo per una determinata lampadina, potremmo essere facilitati qualora si presentassero difficoltà con

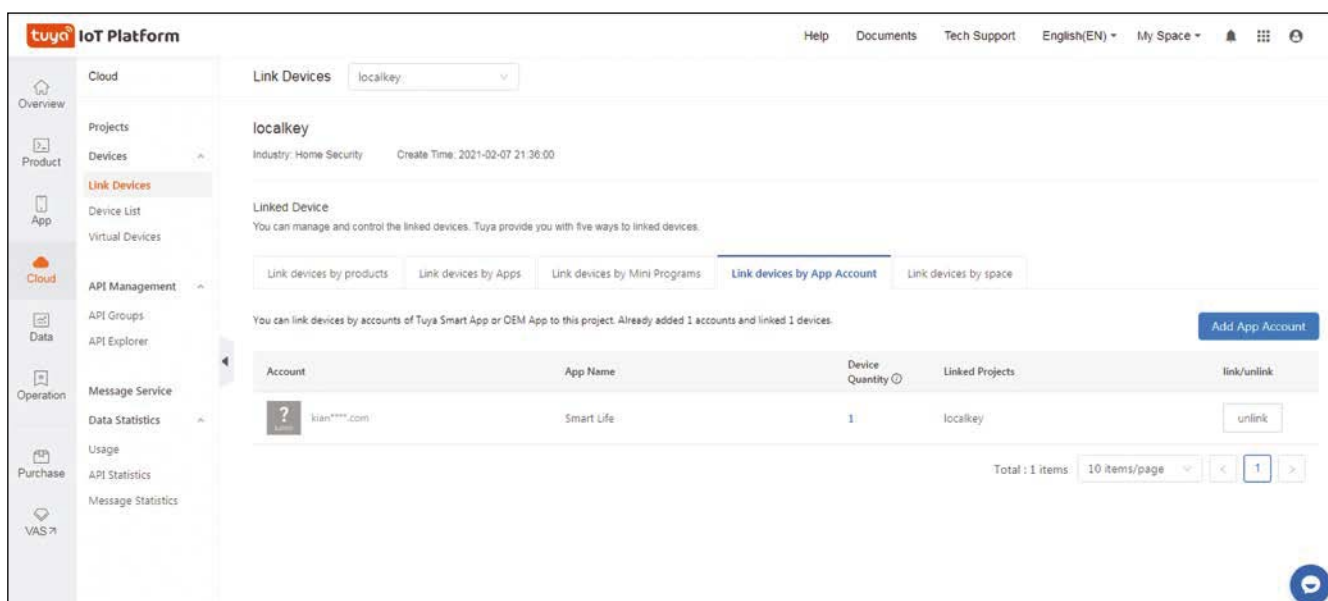


Fig. 5
Sito web
per poter
associare la
lampadina
al cloud e
recuperare la
localkey.

altri modelli. Il modo più convenzionale per il recupero della chiave Tuya (chiamata Localkey) è quello consigliato dalla stessa casa e prevede l'accesso alla sezione developer del sito ufficiale TuyaSmart (<https://iot.tuya.com>). Dopo essersi registrati, occorre accedere al menù "Cloud- Link Devices", creare un progetto nuovo e nella tab "API Groups" abilitare tutte le voci per quel progetto, selezionare il tab "Link devices" by "App Account" come visualizzato in **Fig. 5**; dopodichè premere sul pulsante "AddAppAccount" e scansionare con l'app della lampadina (SmartLife o similari) il codice QR che appare a schermo. A questo punto la lampadina viene associata al cloud Tuya, ma per ottenere la localkey occorre eseguire una Web Api verso i server Tuya passandogli alcuni dati che abbiamo a disposizione, come la coppia ClientId - ClientSecret del progetto creato precedentemente sul sito ed il DeviceId della lampadina che la identifica univocamente e che troviamo navigando all'interno dell'app sotto la voce "Info Dispositivo". La Web API che ci interessa è una GET come la seguente:
<https://openapi.tuya.eu.com/v1.0/devices/3602383098f4abc8b73c>
In cui è 3602383098f4abc8b73c è il DeviceId della lampadina e gli altri parametri (ClientId e ClientSecret) sono passati nell'header della richiesta. Per fare questo useremo Postman (<https://www.postman.com>), un comodo tool gratuito nato proprio per lo sviluppo di api-web e che useremo per questo scopo. Tra i parametri che passeremo nell'header, c'è anche il "token-access" che è obbligatorio e che ci viene fornito da una

Web API di questo tipo:

https://openapi.tuya.eu.com/v1.0/token?grant_type=1

Con tutti questi parametri otterremo una risposta in formato JSON che conterrà anche la nostra localkey. Insieme al progetto scaricabile dal nostro sito troverete comunque il file da importare in Postman ed avviare questa procedura, cambiando opportunamente i parametri secondo le vostre esigenze; anche perché l'access-token è temporaneo e dopo il periodo di scadenza occorre rieseguire l'API per ottenerne un altro valido.

CLOUD

Abbiamo appena associato la lampadina al cloud Tuya ed abbiamo la localkey che ci servirà per decriptare i pacchetti TCP ma con la stessa localkey e Postman potremmo usare anche le altre WEB API (sia GET che POST) sugli stessi server per prendere il controllo della lampadina esattamente come fa l'app SmartLife. Questo perché tramite il sito Tuya visto in precedenza c'è anche una pagina di test (API Explorer) che permette di inviare comandi e leggere lo stato dei dispositivi collegati ed ottenere le risposte online. Questi stessi comandi possono poi essere inviati tramite Postman ed in futuro anche da un nostro sito o applicazione web. Noi però, come primo obiettivo, vogliamo decriptare i pacchetti WiFi per controllare la lampadina ed essere svincolati dall'uso di server esterni che in futuro potrebbero smettere di funzionare, quindi riprenderemo i nostri tentativi per decriptare i nostri pacchetti TCP.

DECODIFICA

Riprendiamo allora il programma C# ed aggiungiamo codice e controlli per provare a decriptare i pacchetti acquisiti in precedenza e cercare di capire quale algoritmo di crittazione viene usato dalla nostra lampadina. Ne esistono davvero tanti e con varianti diverse, quello più diffuso è AES (soprattutto da quando il DES si è rilevato debole rispetto a certi attacchi di forza bruta) ed utilizza un sistema di cifratura a blocchi. Ne esistono di diversi tipi, a seconda della dimensione di questi blocchi, ma noi ora abbiamo la localkey (nel nostro caso c7e961ddad19566f) e siccome è lunga 16 byte, quindi 128 bit, dovremmo orientarci tra gli algoritmi a 128 bit. Non sappiamo ancora quale parte del nostro pacchetto corrisponde al messaggio da decriptare, ma possiamo ipotizzare che i primi byte che abbiamo individuato come header (perché contengono alcune informazioni dei dati veri e propri) non ne facciano parte e nemmeno gli ultimi byte che (come illustrato in **Fig. 4**) sembrano essere il CRC e footer. Per effettuare tutta una serie di tentativi su diversi algoritmi di cifratura e su diversi sottoinsieme del pacchetto ci vengono in aiuto alcuni siti web che rilasciano l'algoritmo di crittazione e decriptazione online in cui si possono modificare al volo diverse variabili. Uno di questi è ad esempio www.devglan.com/online-tools/. Dopo un po' di prove otteniamo come output della

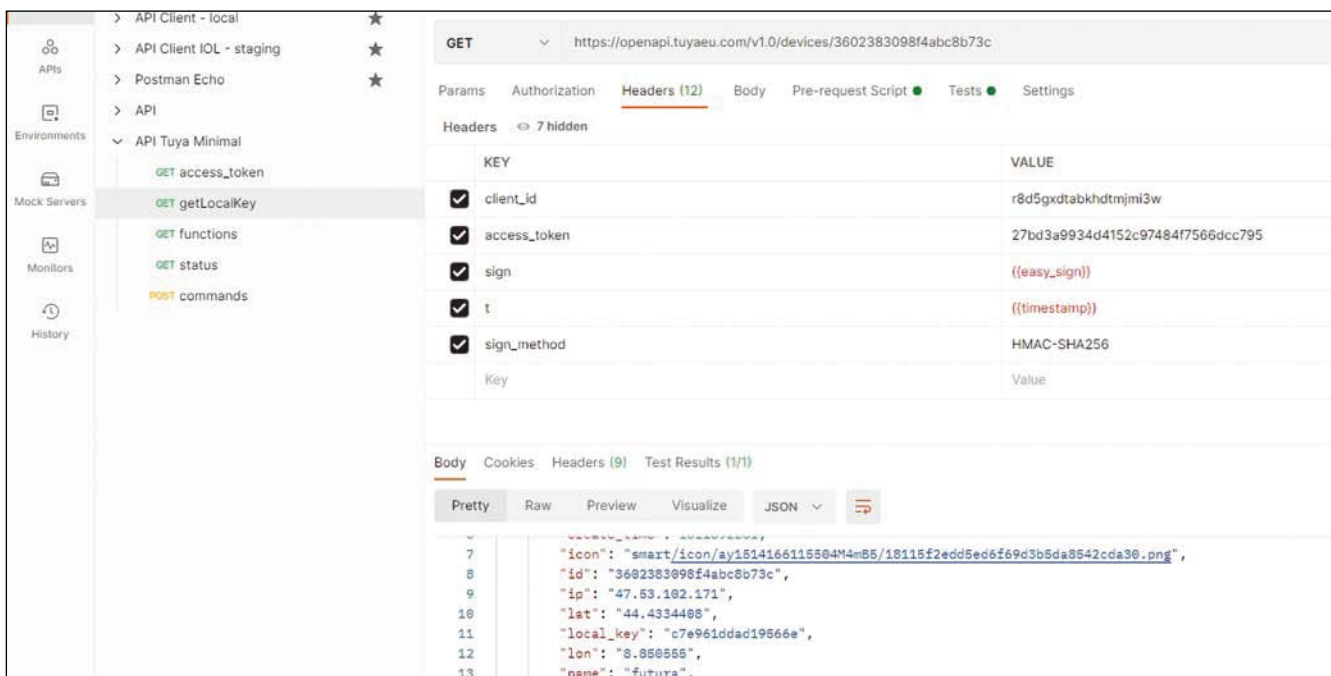
decodifica una stringa di testo in formato JSON che ha tutta l'aria di un comando, scoprendo che l'algoritmo usato è AES-128 in modalità ECB. La modalità ECB opera considerando ogni blocco come un'unità indipendente e risulta meno sicura di altre (come la CBC) in cui esiste anche una correlazione tra i vari blocchi, ma questo per fortuna, ci evita l'utilizzo di un vettore di inizializzazione che avrebbe portato altre difficoltà.

Alla fine otteniamo una stringa di questo tipo:

```
{ "devId": "3602383098f4abc8b73c", "t": "1613427708", "dps": { "1": false, "2": "colour", "3": 79, "4": 0, "5": "aa08ff00b5ffff", "6": "bd76000168ffff", "7": "ffff500100ff00", "8": "ffff8003ff000000ff000000ff0000000000000000", "9": "ffff5001ff0000", "10": "ffff0505ff000000ff00ffff00ff00ff0000ff000000" } }
```

In prima istanza potrebbe sembrare indecifrabile, ma ora possiamo fare diversi cicli di acquisizione con l'app TpacketCapture mentre da cellulare inviamo alla lampadina configurazioni di colore prestabilite e poi, osservando i vari messaggi decriptati, sarà tutto più chiaro. Scopriremo che il parametro "devId" è appunto il deviceId della lampadina, "t" è il timestamp e "dps" è la collezione dei comandi (nominati da 1 a 10) e separati per funzionalità come viene meglio dettagliato nell'apposito riquadro in queste pagine.

Fig. 6
Utilizzo di Postman per lo sviluppo.



Per la nostra esercitazione abbiamo impartito alcuni comandi esemplificativi alla lampada WiFi, che riportiamo nella tabella in questo riquadro.

I restanti comandi sono utilizzati per gestire le scene, gli allarmi e i timer e non vengono ancora trattati in questa sede.

Da notare che questi comandi per il colore o per il bianco, possono essere inviati anche a lampadina spenta; verranno normalmente acquisiti da essa e alla successiva accensione assumerà la tonalità di colore o di bianco impostata in precedenza. Allo stesso modo se cambiamo un colore mentre

siamo in modalità bianco, non succederà nulla, ma se poi passiamo alla modalità colore troveremo la lampadina accesa col colore impostato precedentemente.

Particolare attenzione per il comando "5" che rappresenta il colore sia in modalità RGB che HSV ed espresso in una stringa in cui i valori vengono rappresentati in esadecimale.

Da notare anche il valore minimo della luminosità del bianco e quello del valore V sia 25 (0x19) anziché 0; questo non ha una particolare spiegazione, semplicemente è il risultato di test empirici.

COMANDO	DESCRIZIONE	PARAMETRI	ESEMPIO
1	Accensione	true	"1":true
	Spegnimento	false	"1":false
2	Modalità	white	"2": "white"
		colour	"2": "colour"
		scene	"2": "scene"
		music	"2": "music"
3	Luminosità del bianco	25 -255	"3": 243
4	Temperatura del bianco	0 - 255	"4": 23
5	Settaggio colore	"RRGGBB0HHSSVV" RR: 0 - 255 GG: 0 - 255 BB: 0 - 255 HH: 0 - 360 SS: 0 - 100 VV: 25 - 100	"5": "004e000078ff5e"

CODIFICA

A questo punto, dopo aver appreso il formato dei comandi ed il loro significato, possiamo rieseguire questi passi in maniera inversa e codificare una stringa personalizzata per poi inviare il pacchetto TCP così formato alla lampadina che si comporterà di conseguenza. Riprendiamo nuovamente il programma C#, che completeremo aggiungendo i controlli (visibili nella parte destra di **Fig. 3**) con i quali potremo inserire manualmente una stringa ed eseguire tutti i passaggi, dalla creazione del pacchetto al controllo CRC e infine alla codifica (utilizzeremo a tal proposito la libreria CryptoStream fornita da Microsoft e configurabile per AES-128 ECB). Per rendere più pratici i test aggiungeremo anche alcune slide con cui selezionare il colore desiderato (sia RGB che HSV) e che andranno a modificare la stringa nella casella 5. Aggiungeremo altre slide per cambiare la tonalità del bianco, due pulsanti per cambiare modalità (bianco o colore) ed infine i pulsanti per mostrare il pacchetto codifi-

cato prima di inviarlo sulla rete. In particolare sul rilascio della slide Hue che controlla la tonalità del colore il pacchetto TCP verrà inviato direttamente alla lampadina per provarla velocemente senza dover fare ogni volta tutti i passaggi.

Da notare inoltre come cambiando gli slide per controllare il colore tramite RGB devono cambiare di conseguenza anche gli slide HSV, in modo che la stringa per il comando "5" sia coerente; cambiando solo HSV o solo RGB potrebbero esserci problemi e la lampadina non cambierà colore.

RICEZIONE MESSAGGI

Finora abbiamo parlato di invio di messaggi verso la lampadina e tutto funziona, ma avendo ormai la possibilità di inviare messaggi dal programma in C# (quindi dal nostro PC) e sniffare i pacchetti scambiati tramite Wireshark con più comodità, possiamo analizzare anche le possibili risposte della lampadina ai nostri comandi; basterà impostare i filtri su Wireshark ed osservare la comunicazione nelle due direzioni. Noteremo che ad ogni pacchetto inviato, la lampadina risponde con un pacchetto TCP analogo, che potremo copiare e incollare nella casella di testo del nostro programma, eseguirne la decodifica e leggerne il risultato nella casella di testo sottostante. Vedremo che questa altro non è che una stringa JSON con lo stato della lampadina per gli stessi comandi che abbiamo impostato. Per rendere tutto integrato nel programma C# abbiamo usato un thread di ricezione che viene attivato dopo l' invio di un comando e che riceve i pacchetti trasmessi dalla lampadina copiandoli nella stessa casella di testo, pronti per essere decodificati. Esiste anche un pacchetto particolare, che non ha bisogno di comandi (stringa vuota), ma nell'header, più precisamente nel campo Comando della **Fig. 4**, deve contenere il byte 0x0A (al posto del solito 0x07 usato per i controlli), tramite il quale la lampadina non esegue nessuna azione, ma risponde inviando un pacchetto in cui la stringa JSON decodificata contiene lo stato di tutti i dieci comandi. In realtà, come si può verificare con Wireshark, la lampadina, mentre è spenta, invia sempre, ogni tre secondi un pacchetto UDP in broadcast, ovvero con indirizzo IP di destinazione settato a 255.255.255.255, pronto per essere ricevuto da chiunque, per segnalare la sua presenza nella rete. A quel punto i dispositivi vicini sanno l' IP verso cui possono mandare i comandi e si potrebbe anche il codificare il pacchetto UDP che, a seconda dei modelli di lampadina, può contenere informazioni più generali come il tipo di prodotto, il

modello del dispositivo o la versione. Questo meccanismo è usato da diversi tipi di lampadine, anche se alcune rimangono silenti fino a che non ricevano esse stesse un particolare pacchetto broadcast che le attivi. È comunque pratica comune usare i pacchetti UDP, solitamente meno affidabili, per mandare informazioni di poco contenuto informativo e ad intervalli frequenti, mentre per i comandi in cui vogliamo essere più sicuri della loro trasmissione si ricorre a protocollo TCP che prevede meccanismi di ritrasmissione più affidabili.

CONCLUSIONI

Abbiamo analizzato questo processo per apprendere qualche meccanismo in più che si cela dietro ai più comuni dispositivi WiFi; questo non vuole essere sicuramente una soluzione universale per tutti i casi perché, ovviamente lampadine di marche diverse avranno un loro protocollo proprietario e, a seconda dei casi, potrebbe essere più o meno complicato, ma imparare a districarsi anche solo con una, può essere un buon motivo per apprendere il funzionamento ed una ottima pratica per quando ci dovremo scontrare con un altro dispositivo.

Allo stesso modo, il programma in C# non ha la pretesa di sostituirsi ad un'app su cellulare sia a livello di utilizzo sia di estetica, ma è stato creato appositamente per analizzare la comunicazione con la lampadina e capire i vari passaggi di codifica per poi eseguire il tutto su un microprocessore dotato di interfaccia di rete in cui la memoria è limitata e le attese per il caricamento del firmware per le numerose prove che sono state richieste durante questo processo, avrebbero allungato di gran lunga i tempi di sviluppo. Prossimamente eseguiremo il porting su microprocessore e nel prossimo articolo controlleremo una di queste lampadine tramite un ESP8266. ■



Cosa occorre?

La lampadina smart utilizzata in questo articolo è disponibile presso Futura Elettronica (cod. LEDWIFIBULB) a Euro 19,90. I prezzi si intendono IVA compresa.

Il materiale va richiesto a:

Futura Elettronica, Via Adige 11, 21013 Gallarate (VA)
Tel: 0331-799775 - <http://www.futurashop.it>

FORMATO COLORE

In elettronica ed informatica siamo più abituati a rappresentare un colore come composto dalle tre componenti di rosso, verde e blu.

Esistono però anche altri formati, come ad esempio HSV e HSL che non si basano su componenti distinte, ma sulla tonalità (Hue, da cui la lettera H) che varia da 0 a 360 e viene rappresentato proprio come un cerchio.

Partendo dall'angolo zero (colore rosso) e muovendosi lungo la circonferenza attraverseremo linearmente tutti i colori fino

a tornare al rosso. Se poi per ogni tonalità mi allontano dalla circonferenza avvicinandomi al centro cambia il valore di saturazione (da 0 a 100) indicato con S) fino a diventare bianco nel centro del cerchio quando vale 0. Se poi a parità di posizione cambio il valore di intensità (varia anch'esso da 0 a 100 e viene indicato con V o con L) posso arrivare fino al colore 0. La differenza tra HSV e HSL sta solamente nel terzo valore che nel primo caso è chiamato appunto Value e quando vale 100 indica il colore pieno per

quella tonalità, mentre nel caso di HSL è chiamato Luminosità ed al valore 100 assume colore nero, quindi al valore 50 diventa assimilabile al formato HSV. Nel nostro caso la lampadina può essere controllata in entrambe le modalità, ma essendo il comando del colore una combinazione di entrambi i formati (RGB e HSV) è preferibile inviare un comando coerente in cui, quindi, i primi sei caratteri per rappresentare componenti RGB siano equivalenti agli ultimi sette caratteri che rappresentano lo stesso colore in formato HSV.

LED COB

LIGHTS

**100 LUMEN
WATT**



Realizza lettere, numeri per pannelli pubblicitari, illumina al meglio i prodotti nella tua vetrina e tanto altro con le barrette a LED COB. Ricoperte da gomma siliconica sono realizzate su supporto in alluminio per una migliore dissipazione di calore. Utilissime per svariati utilizzi!



Cod. LEDCOB200B



Cod. LEDCOB200W



Cod. LEDCOB200V



Cod. LEDCOB200R

BARRETTE DI LED COB 12V-15W LUNGHEZZA 20 CM

Tipo di LED: COB chip
Angolo del fascio: 120°
Flusso luminoso: 100 lumen/watt
Vita media: 50.000 ore
Alimentazione: 12 VDC
Potenza: 15W
Consumo: 900 mA
Temperatura di funzionamento: da -20°C a +60°C
Dimensioni: 200x10x3 mm
Peso: 11 grammi

€5,90

A BARRETTA

**Collega
insieme più
barrette di
colori
differenti!**

Prezzi IVA inclusa.

Alimentale con:

**MAX 1
barretta di
LED COB**

ALIMENTATORE SWITCHING
12V - 15W - 1,25A

€5,00

Cod. POS15W12V



**MAX 2
barrette di
LED COB**

ALIMENTATORE
SWITCHING
12V - 36W - 3A

€8,50

Cod. ADL-40-12



**MAX 6
barrette di
LED COB**

ALIMENTATORE
SWITCHING
12V - 100W -
8,5A

€20,00

Cod. MW10012



FISHINO

CONTROLLO NEOPIXEL CON ART-NET

ART
NET
ETHERNET



di EMANUELE SIGNORETTA

Realizziamo una matrice di LED RGB pilotata da Fishino Mega programmata come interfaccia per il protocollo Art-Net.

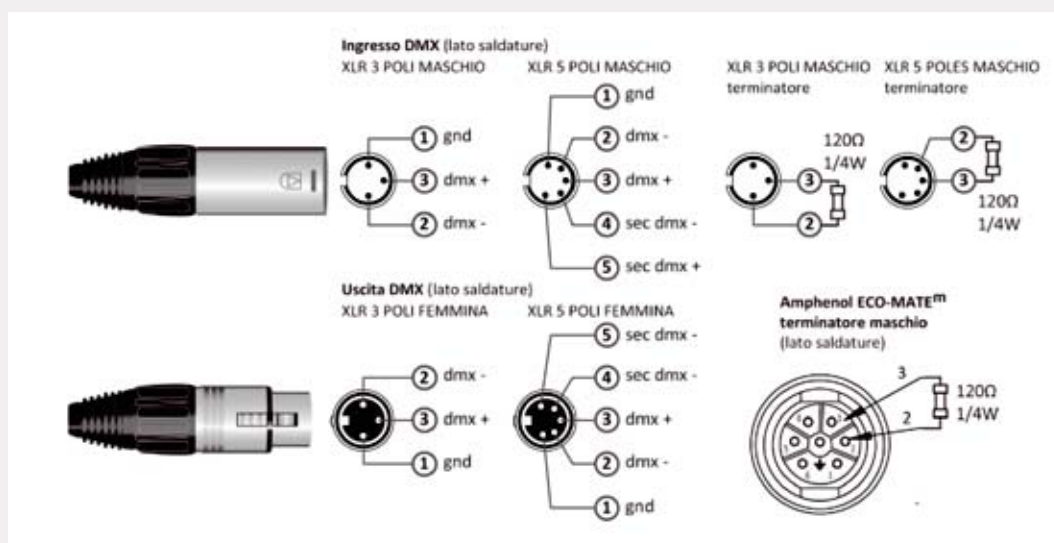
P

er la buona riuscita degli eventi di intrattenimento, è necessario progettare e installare un sistema di controllo delle luci e degli apparati scenografici presenti sul palcoscenico o in studio. Prima degli anni '80 del secolo scorso i tecnici dovevano stendere e collegare centinaia di metri di cavi a enormi

apparati di controllo.

Dal 1986 ha iniziato a prendere piede il DMX 512, con l'intento di standardizzare il controllo degli strumenti di scena in modo da far capo ad un unico dispositivo "master", soppiantando di fatto le controparti proprietarie. Ma non solo: il DMX 512 permette di decentrare la sezione di potenza in corrispondenza delle lampade e degli utilizzatori, perché dalla console non partono i cavi di alimentazione ma semplicemen-

Fig. 1
Pin-out dei
connettori
XLR adibiti alla
comunicazione
DMX.



te il cavo dei dati. Gestendo la potenza sul luogo dell'utilizzatore è possibile sia limitare la quantità di cavi, sia ridurre le perdite.

Nel ventunesimo secolo ha iniziato a diffondersi un nuovo protocollo per il controllo luci chiamato Art-Net, evoluzione del DMX.

Trasmette di fatto 32768 universi DMX, ciascuno di questi composto da 512 canali.

Ogni canale assume valori tra 0 e 255. La sostanziale differenza tra DMX e Art-Net è che il primo necessita di un cavo per ciascun universo, il secondo necessita di un solo cavo di rete o di WiFi.

In questo articolo ci occuperemo di creare una matrice di 1.024 LED, comandati da PC utilizzando il software Jinx e due schede Fishino Mega. Ma prima un po' di storia.

IL DMX

Il DMX o Digital MultipleX o DMX 512 è un protocollo di comunicazione che cominciò a diffondersi nel 1986 e che a partire dal 1990 subì numerose

revisioni. Come già detto, è stato ideato principalmente per standardizzare il controllo dei vari effetti scenografici di diversi modelli e marche.

Inizialmente è stato utilizzato per il controllo unidirezionale di dispositivi come dimmer, teste mobili e PAR. Successivamente venne implementato negli effetti scenografici come le macchine del fumo ma, proprio a causa della sua unidirezionalità, ne è proibito l'utilizzo per i dispositivi per la movimentazione delle strutture scenografiche o effetti pirotecnici. Per questi ultimi è infatti previsto l'utilizzo del protocollo MIDI.

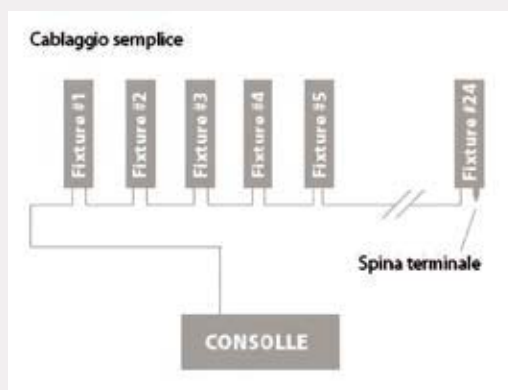
Lo standard prevede l'utilizzo di connettori XLR a 5 poli, (di cui tuttavia se ne utilizzano 3) per evitare di confondere i cavi con quelli dei dispositivi audio, o peggio ancora con le linee phantom a +48V. Tuttavia numerosi produttori utilizzano connettori XLR a 3 poli (**Fig. 1**).

Il collegamento avviene utilizzando cavi schermati bipolari di impedenza nominale di 110 Ω. All'inizio della catena DMX vi è un controller o MASTER che genera il segnale il cui prezzo può variare dalle poche decine di Euro alle diverse migliaia. I vari dispositivi SLAVE sono collegati in cascata o a stella, mediante l'utilizzo di opportuni splitter.

Al termine della catena è previsto l'utilizzo di un terminatore da 120 Ω che deve smorzare eventuali disturbi nelle zone più lontane dal controller. Ogni cavo ha il limite di trasporto di 512 canali DMX, che formano un universo. In base ai controller usati è possibile controllare più universi indipendentemente l'uno dall'altro (**Fig. 2**).

La trasmissione dei dati avviene mediante il protocollo RS-485 con livelli logici alti a 5V e livelli logici bassi a 0V. Poiché la durata della trasmissione

Fig. 2
Schema di
collegamento a
cascata di una
catena DMX.



di un singolo bit è di 4µs, in un secondo vengono trasmessi un totale di $\frac{1}{4} * 10^6$ bit corrispondenti 250 kb/s. Nella **Fig. 3** vedete lo schema di ciascuna delle sei parti componenti di un gruppo di dati. Il primo segnale ad essere trasmesso è il **BREAK**: segna l'inizio della trasmissione di dati. Trasmette 22 bit (88 µs) a livello logico 0.

Successivamente viene trasmesso il **MARK AFTER BREAK (MAB)**: imposta tutti i canali a livello logico basso. Segue lo **START CODE (SC)**: indica che sta per iniziare la trasmissione del livello dei vari canali, la sua durata è sempre la stessa ed è pari a 11 impulsi (44 µs) ed il livello logico è alto.

Con durata variabile da 0 ad 1 secondo segue il **MARK TIME BETWEEN FRAMES (MTBF)**: indica l'avvio della trasmissione del valore di un canale. Subito dopo troviamo il **CHANNEL DATA (CS)**: utilizzando un totale di 21 bit avvia e chiude la trasmissione del valore del canale, con valori che vanno da 0 a 255. Viene ripetuto il **MARK TIME BETWEEN FRAMES (MTBF)** per indicare il termine della trasmissione di un canale e l'inizio di quello successivo. In totale la trasmissione di un blocco dati avviene in 22.754 µs.

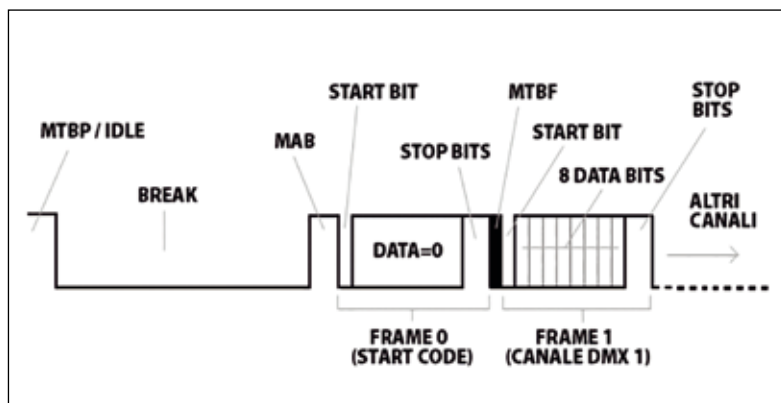


Fig. 3
Struttura di un
singolo pacchetto
DMX.

ART-NET

L' Art-Net è un protocollo di comunicazione royalty-free che implementa il DMX-512 per il controllo delle fixture e anche il protocollo per la gestione remota dei dispositivi (RDM). Viene trasmesso in UDP sulla porta 6454 ed è di tipo full-duplex. La struttura hardware di una rete Art-Net è molto semplice (**Fig. 4**): si tratta infatti di una rete LAN con un hub/switch a cui convergono eventualmente altri switch. Ai capi della rete LAN si trovano



Fig. 4
Schema di concetto di
una rete di controllo
Art-Net.

Fig. 5
Struttura di un
pacchetto Art-Net.

offset (bytes)	0	1	2	3
0	'A'	'r'	't'	'.'
4	'N'	'e'	't'	0
8	Opcode ArtDMX (0x5000) little endian		Protocol Version Hi (0)	Protocol Version Lo (14)
12	Sequence	Physical	Universe little endian	
16	Length Hi	Length Lo (2 to 512, even)	Data	Data
20	Data ...			

i "media server" e i nodi. I media server possono essere dei semplici Personal Computer dotati di un software di illuminazione, o delle vere e proprie consolle per uso specifico. Il loro scopo è generare il segnale in Art-Net, coordinare i dispositivi, l'eventuale gestione errori (per esempio una lampada bruciata). I nodi invece sono i dispositivi che ricevono i dati e li rielaborano. Tra questi ci sono i convertitori Art-Net/DMX, teste mobili, effetti scenografici e altre schede come ad esempio una Fishino cui collegare dei LED.

Sin dalla prima release l'Art-Net ha dimostrato di essere un valido protocollo, sebbene con alcune limitazioni che sono state risolte col passare del tempo. La prima versione permetteva infatti il controllo di soli 40 universi, con la seconda il numero è stato aumentato a 256. Nonostante tutto, il salto di qualità è stato molto più evidente con la terza versione, la quale permette il controllo di ben 32.768 universi. La quarta e ultima versione è stata rilasciata nel 2016 e comprende diverse migliorie. Basti pensare allo scorso festival di Sanremo l'intera scenografia è stata gestita con 3829 universi Art-Net, generati utilizzando il software Madrix

LESSICO DELL'ART-NET

L'Art-Net ha una terminologia molto variegata poiché prende alcuni termini dal DMX e altri dai normali protocolli di rete.

Analizziamone i principali in dettaglio:

- **Universo:** singolo gruppo di 512 canali DMX consecutivi;
- **Kiloverso:** gruppo di 1.024 universi;
- **Sub-Net:** gruppo di 16 universi consecutivi;
- **Net:** gruppo di 16 consecutive Sub-Nets (o 256 universi consecutivi). Il numero Massimo di Nets è 128;
- **Port-Address:** uno dei 32.768 possibili indirizzi a cui può essere trasmesso un frame DMX. È un numero di 15 bit composto dalla somma:

Net+Sub-Net+Universo;

- **Nodo:** dispositivo principalmente realizzato per trasformare l'Art-Net in DMX. Un nodo può gestire al massimo 4 porte in entrata e 4 porte in uscita;
- **ArtPoll:** un pacchetto Art-Net utilizzato per scoprire l'esistenza di nodi;
- **ArtPollReply:** un pacchetto Art-Net inviato da un nodo in risposta all'ArtPoll;
- **ArtDmx:** pacchetto Art-Net usato per trasmettere un bit di start su tutta la rete pacchetti DMX 512;
- **Subscription:** indica un nodo che riceve un particolare universo;
- **Discovery:** è un processo attraverso il quale un controller rileva quale nodo vuole ricevere uno o più universi specifici.

TIPI DI TRASMISSIONE

Come altri protocolli, anche l'Art-Net prevede varie modalità in cui possono essere trasmessi i dati; le descriviamo qui di seguito.

- **Unicast:** con questa tipologia di trasmissione si intende una singola sorgente che invia i dati ad un singolo ricevitore. È molto efficiente nell'uso della banda di rete poiché gli apparati di rete possono far sì che i pacchetti vengano trasmessi nella parte di network a cui è connesso il ricevitore. Diventa sempre meno efficiente quando più ricevitori devono ricevere gli stessi dati.
- **Multicast:** identifica uno o più trasmettitori che inviano dati ad un gruppo di ricevitori. I ricevitori si "iscrivono" per iniziare a ricevere gli stessi dati. È probabile che alcuni switch di rete abbiano una capacità limitata di supportare il multicast, convertendolo così in broadcast.
- **Broadcast:** quando una sorgente invia dati a tutti i ricevitori. È molto comodo per quanto riguarda la configurazione dei software e per la gestione dei pacchetti. Si rivela però inefficiente

se la maggior parte dei dispositivi non necessita di tutti i dati. Per quanto riguarda la gestione degli apparati di rete può capitare che la rete venga congestionata perché i dati vengono inviati a tutte le porte del gateway e degli switch.

Per maggiori dettagli è possibile consultare la documentazione ufficiale al link <https://artisticlicence.com/WebSiteMaster/User%20Guides/art-net.pdf>

STRUTTURA DI UN PACCHETTO ART-NET

Come esposto nella Fig. 5, ogni pacchetto Art-Net è composto da 24 byte, così suddivisi:

1. I primi bytes 0-9 sono uguali per tutti i pacchetti. Contengono infatti i caratteri "Art-Net", seguiti da uno 0 e dall'OpCode. L'OpCode è un numero composto da 16 bit che definisce il tipo del pacchetto Art-Net.
2. I byte 10 e 11 sono riservati alla versione del protocollo.
3. I byte 12 e 13 sono occupati rispettivamente da Sequence e Physical. Sequence viene usato per tracciare l'ordine in cui i pacchetti vengono inviati. Physical indica la porta fisica che ha inviato i pacchetti.
4. Il byte 14 indica la SubUni e sono i primi 8 bit della combinazione porta-indirizzo cui sono indirizzati i pacchetti.
5. Il byte 15 indica la Net e sono i secondi 8 bit della combinazione porta-indirizzo a cui sono indirizzati i pacchetti.
6. Il bit 16 è LengthHi ed è il MSB (Most Significant Byte) della lunghezza dell'array dei dati.

schema di CABLAGGIO

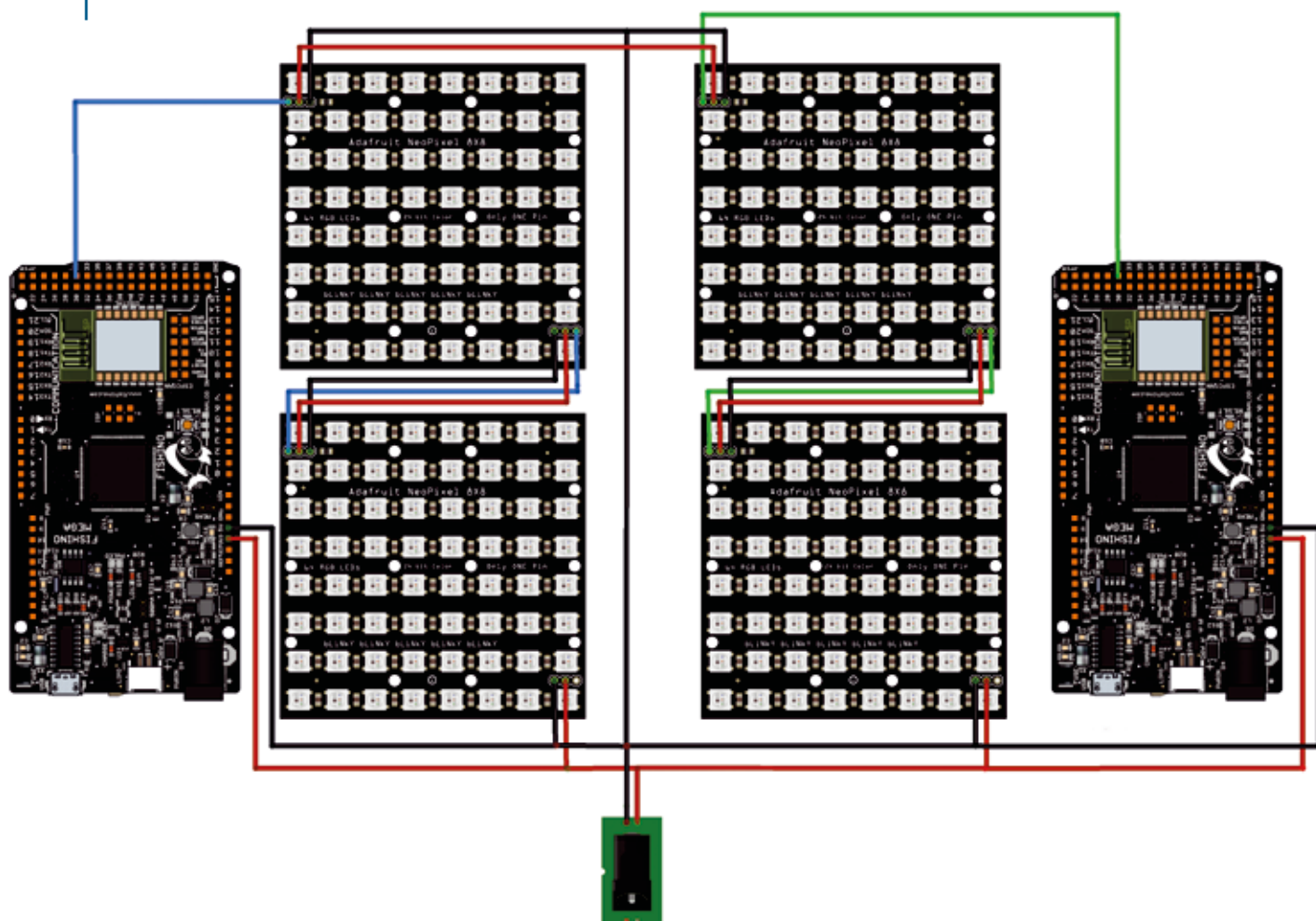




Tabella 1

Numero massimo di LED controllabili dalle schede Fishino a 8 bit.

SCHEDA	NUMERO DI LED CONTROLLABILI
Fishino Uno	300
Fishino Mega	600
Fishino Guppy	300

7. Il bit 17 è LengthLo ed è il LSB (Least Significant Byte) della lunghezza dell'array dei dati.

8. A partire dal bit 18 c'è un array di dimensione variabile (da 2 a 512 byte) che indica i valori per ciascuno dei canali (fino ad un massimo di 512).

HARDWARE E INSTALLAZIONE

Il progetto che vi proponiamo è illustrato nello schema di cablaggio, dove vediamo che il tutto è diviso in due sottosistemi composti ciascuno da una Fishino Mega che pilota due matrici; ogni matrice corrisponde a 2 universi ArtNet, quindi con la nostra configurazione andremo ad operare su quattro universi.

È possibile installare i LED in diversi modi: strip lineare, matrice o svariate altre forme geometriche. Noi ci occuperemo di realizzare una matrice di 32 righe e 32 colonne, utilizzando 4 matrici 16x16 di NeoPixel, vendute da FuturaShop (codice FLEXLEDMX1616). Ognuna è un pannello flessibile dotato di 256 LED NeoPixel (appunto disposti a matrice 16x16) con chip WS2812B, permette di creare animazioni, scritte, disegni, ecc. L'interesse tra i LED, in verticale e orizzontale, è

di 10 mm. Poiché ciascun LED necessita di una tripletta di variabili che ne identificano lo stato (Red, Green, Blue), in ciascun universo vengono controllati 170 LED. Per questo motivo le prime 2 matrici utilizzeranno gli universi numerati da 0 a 3 e le seconde due gli universi da 4 a 7.

Ogni pannello è strutturato per consentire la connessione in cascata (allo scopo di realizzare display di grandi dimensioni) e pertanto dispone di due connettori, uno di input e l'altro di output.

Per alimentare le matrici avremo bisogno di un alimentatore switching da 5V; in questo caso abbiamo utilizzato uno switching da 5V - 50 A (codice MW35005). Consigliamo di collocare e fissare le matrici ad un pannello rigido e resistente.

Sono preferibili una lastra di plexiglass o di legno, dallo spessore di almeno 1 cm in quanto isolanti elettrici (le matrici sono isolate, ma non possiamo escludere che qualche filo si stacchi e vada in cortocircuito se sollecitato).

Le matrici sono state disposte seguendo un ordine a "due colonne" che parte dal vertice superiore sinistro. Successivamente abbiamo eseguito i collegamenti tra le varie matrici, saldando opportunamente i cavi di alimentazione e del segnale. Consigliamo di eseguire le saldature con un saldatore a stilo con punta fine, ad una temperatura di circa 350°C, tenendone la punta giusto il necessario a sciogliere la lega saldante, perché esporre le matrici ad elevate temperature per tempi prolungati può danneggiare i LED.

Considerato l'elevato assorbimento dei LED e la delicatezza del PCB su cui poggiano i LED, è preferibile collegare sia il capo iniziale sia il capo finale della catena all'alimentatore mediante un cavo



Listato 1

```
// Neopixel settings
const int numLeds = 512; // change for your setup
const int numberOfChannels = numLeds * 3; // Total number of channels you want to receive (1 led = 3 channels)
const byte dataPin = 6;
Adafruit_NeoPixel leds = Adafruit_NeoPixel(numLeds, dataPin, NEO_GRB + NEO_KHZ800);

// Artnet settings
FishinoArtnet artnet;
const int startUniverse = 4; // CHANGE FOR YOUR SETUP most software this is 1, some software send out artnet first universe as 0.

// Check if we got all universes
const int maxUniverses = numberOfChannels / 512 + ((numberOfChannels % 512) ? 1 : 0);
bool universesReceived[maxUniverses];
bool sendFrame = 1;
int previousDataLength = 0;
```



bipolare di sezione adeguata.

Per quanto riguarda gli apparati di rete è da preferire creare una rete ad-hoc, in modo da evitare conflitti con altri dispositivi connessi alla rete.

Bisogna, infine, collegare il PC al gateway con un cavo di rete.

LA LIBRERIA FISHINOARTNET

Per poter sfruttare questo protocollo è necessario che ciascuna Fishino si connetta ad una rete WiFi, riceva e poi elabori i dati in arrivo sulla porta 6454. Per le schede Arduino ed ESP sono già state scritte in passato numerose librerie distinte per modulo WiFi o ethernet. Su GitHub abbiamo trovato una libreria particolarmente interessante, ossia la **"ArtnetWifi"** (<https://github.com/rstephan/ArtnetWifi/>). È lo scheletro della nuova libreria FishinoArtnet, un porting con gli adattamenti necessari per funzionare con il modulo ESP delle schede Fishino. Quest'ultima è già inclusa nel package ufficiale Fishino, ma prossimamente effettueremo una pull request alla repository originale: così facendo sarà possibile ottenere eventuali aggiornamenti con una maggior facilità. Durante la fase di test della libreria abbiamo osservato che le diverse board Fishino non riescono a controllare lo stesso numero di LED, a causa dei limiti del microcontrollore di cui ciascuna di esse è dotata. Nella **Tabella 1** viene riportato il numero massimo di LED controllabili in Art-Net senza correre il rischio che il firmware vada in crash. Al momento non è possibile compilare lo sketch per le schede Fishino a 32 bit, in quanto allo stato attuale la libreria AdafruitNeoPixel (che abbiamo utilizzato per la gestione dei led) non è compatibile con i microcontrollori PIC32.

POSSIBILI UTILIZZI

Le configurazioni che questa libreria permette di realizzare in base ai dispositivi utilizzati sono innumerevoli; infatti è possibile utilizzare Fishino per inviare pacchetti Art-Net sulla rete per controllare fixtures di qualunque tipo. Un'alternativa è quella di far ricevere comandi e gestire gli attuatori quali i motori stepper, dimmer pack, ecc.

Di seguito analizzeremo il codice della libreria contenuto nello sketch d'esempio **FishinoArtnetNeo-pixel**, utilizzato per realizzare un display a matrice da 1.024 LED RGB basati sul chip WS2812B. Il codice è quello che verrà eseguito da ogni singola Fishino Mega, la quale opererà con un proprio indirizzo di rete, quindi va da sé che per ciascuna dovrete fare la relativa "personalizzazione" dello

↓ Listato 2

```
void initTest() {
    for (int i = 0 ; i < numLeds ; i++)
        leds.setPixelColor(i, 127, 0, 0);
    leds.show();
    delay(500);
    for (int i = 0 ; i < numLeds ; i++)
        leds.setPixelColor(i, 0, 127, 0);
    leds.show();
    delay(500);
    for (int i = 0 ; i < numLeds ; i++)
        leds.setPixelColor(i, 0, 0, 127);
    leds.show();
    delay(500);
    for (int i = 0 ; i < numLeds ; i++)
        leds.setPixelColor(i, 0, 0, 0);
    leds.show();
}
```

sketch. Nel **Listato 1** sono presenti parametri di configurazione, anche se abbiamo ommesso i parametri di configurazione del WiFi. Va comunque precisato che l'indirizzo IP della Fishino Mega deve essere statico, in modo da non dover sempre cambiare configurazione dei programmi che trasmettono il segnale. La costante intera *numLeds* viene utilizzata per specificare il numero di LED utilizzati: poiché si tratta di due matrici 16 x16 il totale è 512. La costante *numberOfChannels* indica quanti sono necessari per ciascun led. Se si utilizzassero dei LED RGBW il valore sarebbe 4, ma siccome si tratta di LED RGB è necessario impostarla a 3. *dataPin* è il pin cui verrà collegata la nostra matrice, potete inserire un numero qualsiasi.

Dopo aver creato gli oggetti delle librerie Adafruit_NeoPixel e FishinoArtnet, segue la costante *startUniverse*. È necessaria una breve digressione riguardo questa costante: alcuni programmi trasmettono i dati degli universi (e anche dei canali) a partire dall'indice 0. Altri programmi seguono invece la numerazione che parte dall'indice 1. In questo caso abbiamo impostato la costante a 0. Sulla seconda scheda cambiano i dati di configurazione dell'Art-Net e di connessione al Wi-Fi.

Oltre all'indirizzo IP varia la costante *startUniverse*: infatti la prima scheda utilizza gli universi numerati da 0 a 3, mentre la seconda utilizza gli universi numerati da 4 a 7. Il **Listato 2** contiene delle istruzioni di semplice interpretazione: si tratta di una sola funzione che viene richiamata una sola volta all'interno del setup e serve a testare il funzionamento dei led in modo appropriato. Accende tutti i LED della strip ordinatamente con i colori rosso, verde e blu ad intervalli di 500 millisecondi tra l'uno e l'altro. Infine i LED vengono spenti, in attesa



↓ Listato 3

```
void onDmxFrame(uint16_t universe, uint16_t length, uint8_t sequence, uint8_t* data) {
    sendFrame = 1;
    // set brightness of the whole strip
    if (universe == 15) {
        leds.setBrightness(data[0]);
        leds.show();
    }
    // Store which universe has got in
    if ((universe - startUniverse) < maxUniverses)
        universesReceived[universe - startUniverse] = 1;
    for (int i = 0 ; i < maxUniverses ; i++) {
        if (universesReceived[i] == 0) {
            //Serial.println("Broke");
            sendFrame = 0;
            break;
        }
    }
    // read universe and put into the right part of the display buffer
    for (int i = 0; i < length / 3; i++) {
        int led = i + (universe - startUniverse) * (previousDataLength / 3);
        if (led < numLeds)
            leds.setPixelColor(led, data[i * 3], data[i * 3 + 1], data[i * 3 + 2]);
    }
    previousDataLength = length;
    if (sendFrame) {
        leds.show();
        // Reset universeReceived to 0
        memset(universesReceived, 0, maxUniverses);
    }
}
```

di ricevere altre istruzioni. Il **Listato 3** è sostanzialmente il "core" del nostro sketch. In questo snippet vengono letti tutti i dati in arrivo e vengono a loro volta scomposti in modo da essere posizionati nella giusta configurazione RGB della matrice. Il **Listato 4** contiene le funzioni *setup()* e *loop()*. Nella funzione *setup()* viene inizializzata la porta seriale, e vengono richiamate ordinatamente le funzioni che fanno connettere il Fishino al WiFi, inizializzano i LED, la libreria Art-Net e viene eseguito il test di cui al **Listato 2**.

Nella funzione *loop()* è contenuta solamente la funzione *artnet.read()*; utilizzata per ricevere ed elaborare i dati.

CONFIGURAZIONE DEL SOFTWARE

Esistono numerosi software in grado di generare pacchetti di dati secondo il protocollo Art-Net: i più completi sono senza alcun dubbio Pandora's Box della Coolux e Madrix, sebbene siano a pagamento. Sono i più consigliati a coloro i quali desiderano utilizzarli in spettacoli o eventi particolari. Tuttavia

↓ Listato 4

```
Serial.begin(115200);
ConnectWifi();
artnet.begin();
leds.begin(); // INITIALIZE NeoPixel strip object (REQUIRED)
initTest();

// this will be called for each packet received
artnet.setArtDmxCallback(onDmxFrame);
void loop() {
    // we call the read function inside the loop
    artnet.read();
}
```



esistono numerosi programmi freemium e addirittura alcuni di questi sono open source. Non possiamo non citare Open DMX e Jinx: di quest'ultimo studieremo passo passo la configurazione. Per conoscere tutti i programmi più popolari in questo settore, è opportuno visitare il link <https://dmxking.com/control-software>.

CONFIGURAZIONE JINX

Jinx è un programma freemium che consente di gestire i LED attraverso svariati protocolli (Art-Net, DMX, SaCN ecc.) e di gestirne l'output. È infatti possibile impostare l'output in modo da formare una semplice striscia o anche una matrice. Il numero massimo di LED controllabili è 48.000, per un totale di 144.000 canali, circa 282 universi Art-Net. Si può scaricare gratuitamente dal sito ufficiale <http://www.live-leds.de/>. Attualmente è disponibile in versione no installer per i soli PC Windows, ma abbiamo avuto modo di testarlo senza alcuna difficoltà su Ubuntu 20.04 utilizzando Wine. Una volta scaricato ed estratto il file .zip in una cartella qualsiasi del nostro computer, avviamo il file Jinx.exe, mentre su Ubuntu è sufficiente lanciare da terminale il comando `wine Jinx.exe`. All'avvio apparirà una schermata come quella proposta in **Fig. 6**, dove si può immediatamente notare la gestione della GUI: due riquadri laterali suddivisi a loro volta in due canali da cui è possibile

Guarda il controllo NeoPixel con Art-Net e Fishino in azione su YouTube!



Vuoi vedere in azione il nostro controllo NeoPixel con Art-Net e Fishino? Nulla di più semplice: collegati alla pagina <https://youtu.be/nxydIV0qXxM>. Potrai anche vedere i video di moltissimi altri nostri progetti proposti in passato.

selezionare le animazioni e i loro settaggi. Il riquadro centrale è dedicato al master ossia il punto a cui convergono tutti i canali e vengono "misceati" per formare l'output finale.

Fig. 6
Schermata di avvio di Jinx.

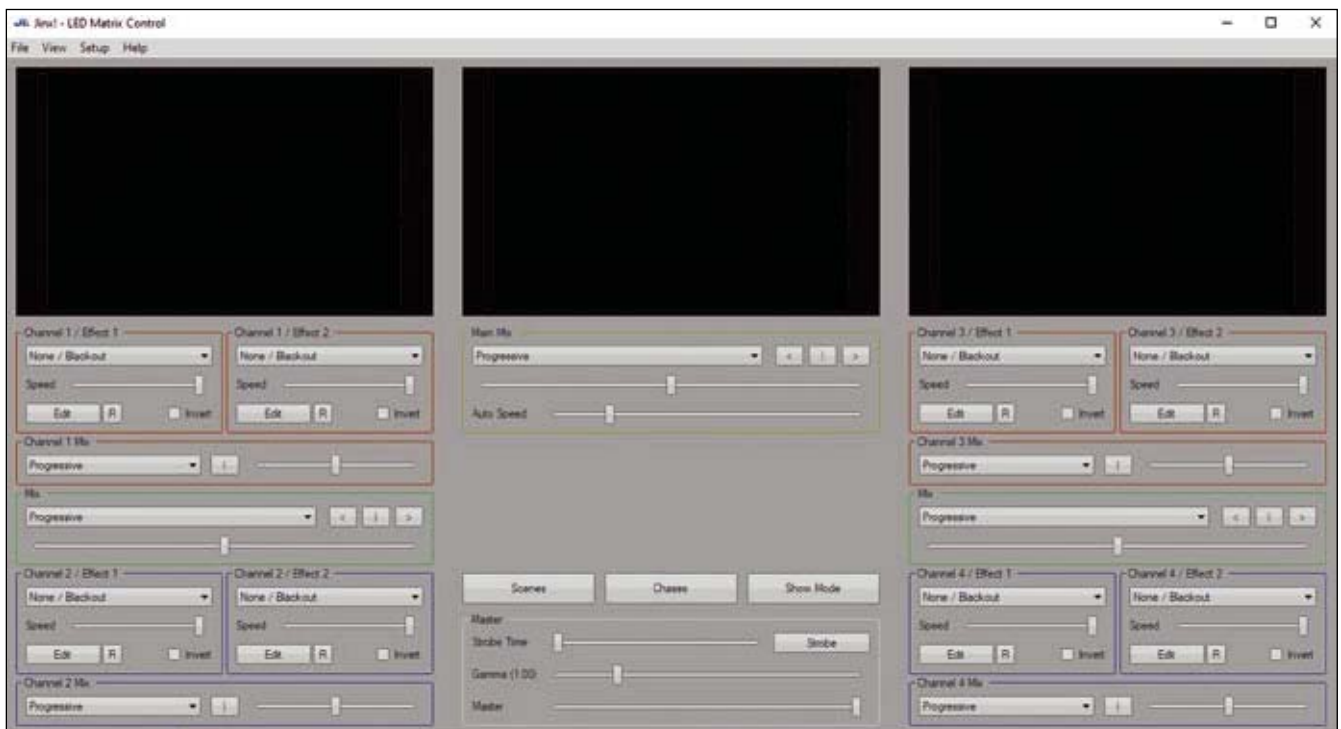
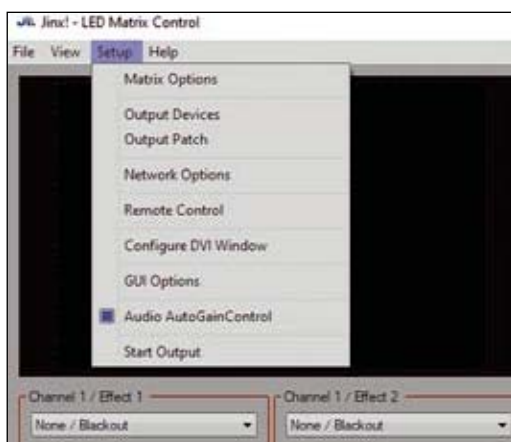


Fig. 7
Particolare del
menù "Setup".



È dedicato anche ad alcune funzioni come ad esempio lo strobe, la gestione dell'intensità e la preview finale. Passiamo ora al settaggio del software. Dalla barra degli strumenti selezioniamo la voce Setup e verifichiamo che la voce Start Output non sia spuntata (**Fig. 7**); in caso contrario rimuoviamo il segno di spunta cliccandoci sopra.

1. Rechiamoci di nuovo in **Setup → Matrix option**.

Si aprirà una finestra come quella in **Fig. 8** nella quale è necessario modificare i parametri in modo che coincidano con la nostra configurazione hardware. Pertanto nella casella *width* (numero di colonne) inseriamo il numero 32 nella casella *height* (numero di righe) il numero 32. Non è necessario modificare i restanti parametri, perciò premiamo **OK** e verremo riportati nella schermata principale.

2. Per trasmettere i dati è necessario settare i parametri per il protocollo. Spostiamoci dunque su **Setup → Output Devices**: si aprirà una nuova

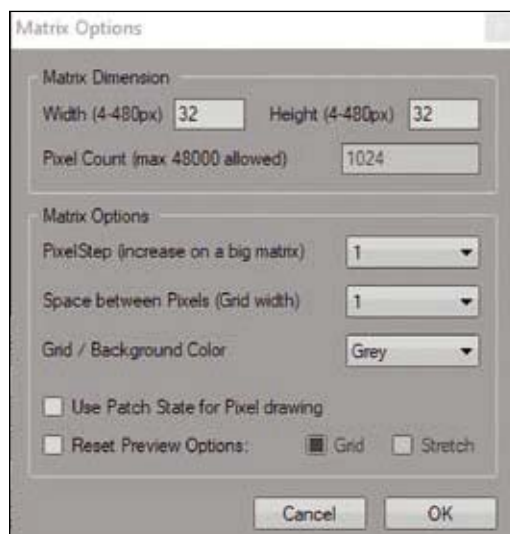


Fig. 8
Configurazione
della dimensione
della matrice.

finestra contenente un elenco (inizialmente vuoto) dei dispositivi inizializzati. Per aggiungerne uno clicchiamo sul pulsante Add e si aprirà una finestra come nella **Fig. 9**. In Device Type selezioniamo Art-Net. Togliamo il segno di spunta dalla voce Broadcast / Multicast (E 1.21) e nel campo IP Address inseriamo l'indirizzo IP del Fishino. Lasciamo invariati i campi Net e Subnet, e impostiamo il campo Universe a 0. Clicchiamo su OK per ritornare alla schermata precedente. Ripetiamo i passaggi per i restanti 6 universi incrementando ogni volta il campo Universe. Una volta completata la configurazione dei 7 universi possiamo premere su Close per tornare alla schermata principale.

3. Per trasmettere esattamente i dati al Fishino è necessario indicare a quale universo e a quale canale corrisponde ogni colore dei led. Perciò è necessario patchare la matrice. Rechiamoci su Setup e poi Output Patch. Si aprirà una finestra come in **Fig. 10**. È possibile patchare ciascun LED manualmente oppure eseguire in

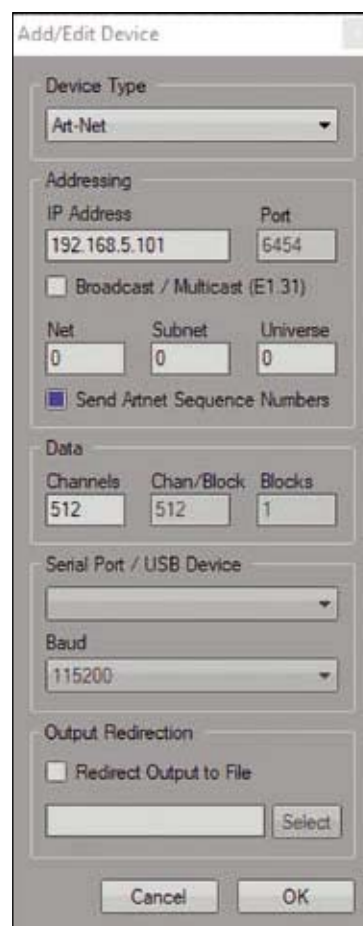


Fig. 9
Settaggio di
un singolo
dispositivo
(Universo).

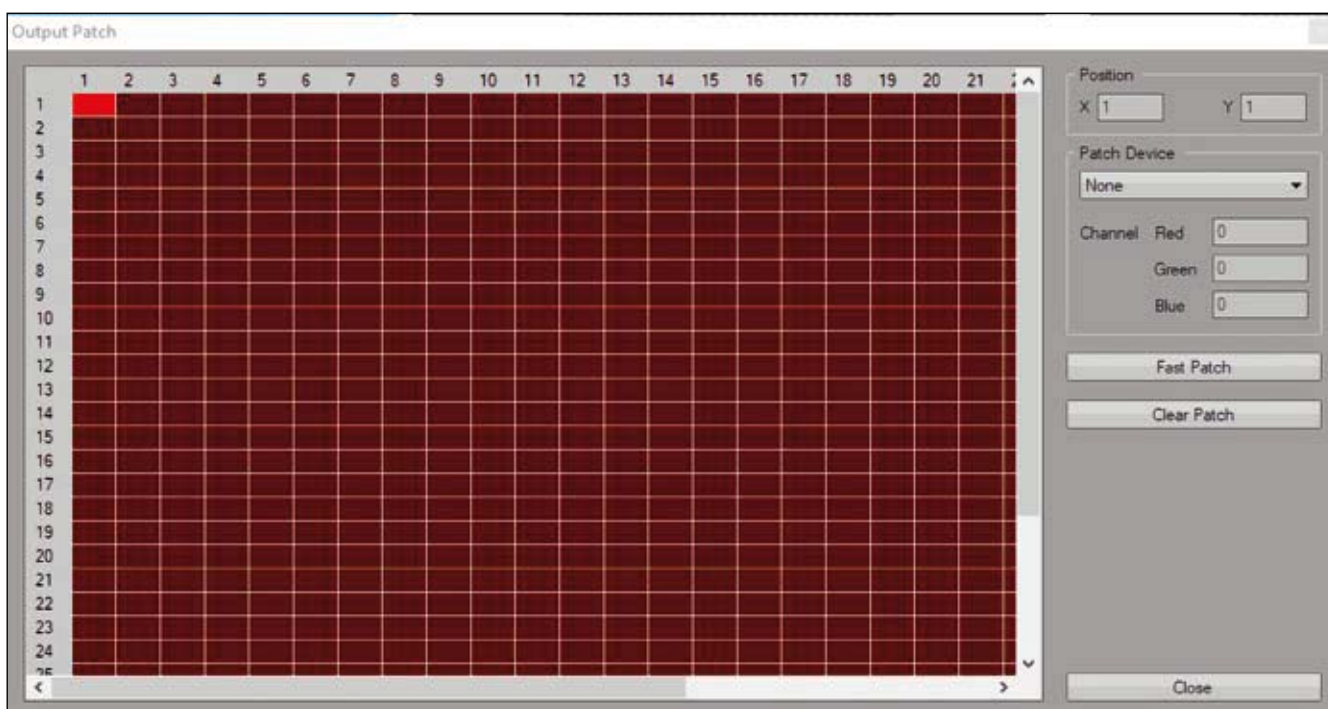


Fig. 10
Pixel mapping dei
led sulla matrice
secondo il pattern
di montaggio).

automatico la configurazione. Noi ci avvarremo di entrambe le procedure. Posizioniamoci con il cursore del mouse sul primo LED in alto a sinistra, clicchiamo su Fast Patch e si aprirà una finestra di dialogo come quella in **Fig. 11**. Nei campi X e Y inseriamo la dimensione della matrice che vogliamo patchare. Inseriamo come parametri rispettivamente 32 e 32 sebbene il numero massimo di LED che verranno patchati sia di 170 per volta. Nella finestra Patch Mode selezioniamo SnakeLines starting Top-Left. I parametri RGB Order e First Channel lascia-

moli impostati rispettivamente a RGB e 0. Patch Device deve essere impostato a [0 | 0 | 0] (rispettivamente [Net | Subnet | Universo]). Una volta eseguita questa prima parte della configurazione è necessario portarla a termine manualmente la configurazione dell'intera riga fino a patchare l'ultimo LED della matrice. Per farlo selezioniamo un LED alla volta e modifichiamone i parametri manualmente. Una volta patchati i primi 170 LED rimangono liberi 2 canali del primo universo che purtroppo non possono essere utilizzati. Infatti dobbiamo far



Fig. 11
Finestra di
pixel
mapping.

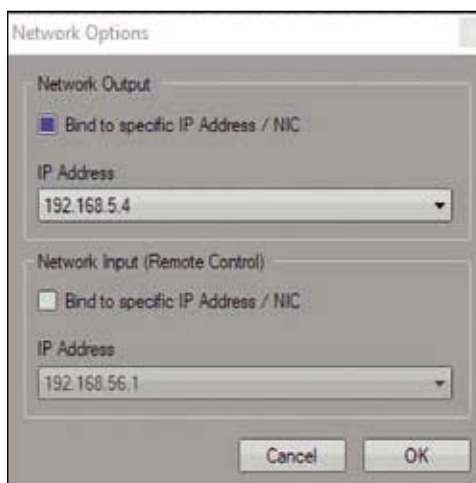


Fig. 12
Scelta della
scheda
di rete da
utilizzare.

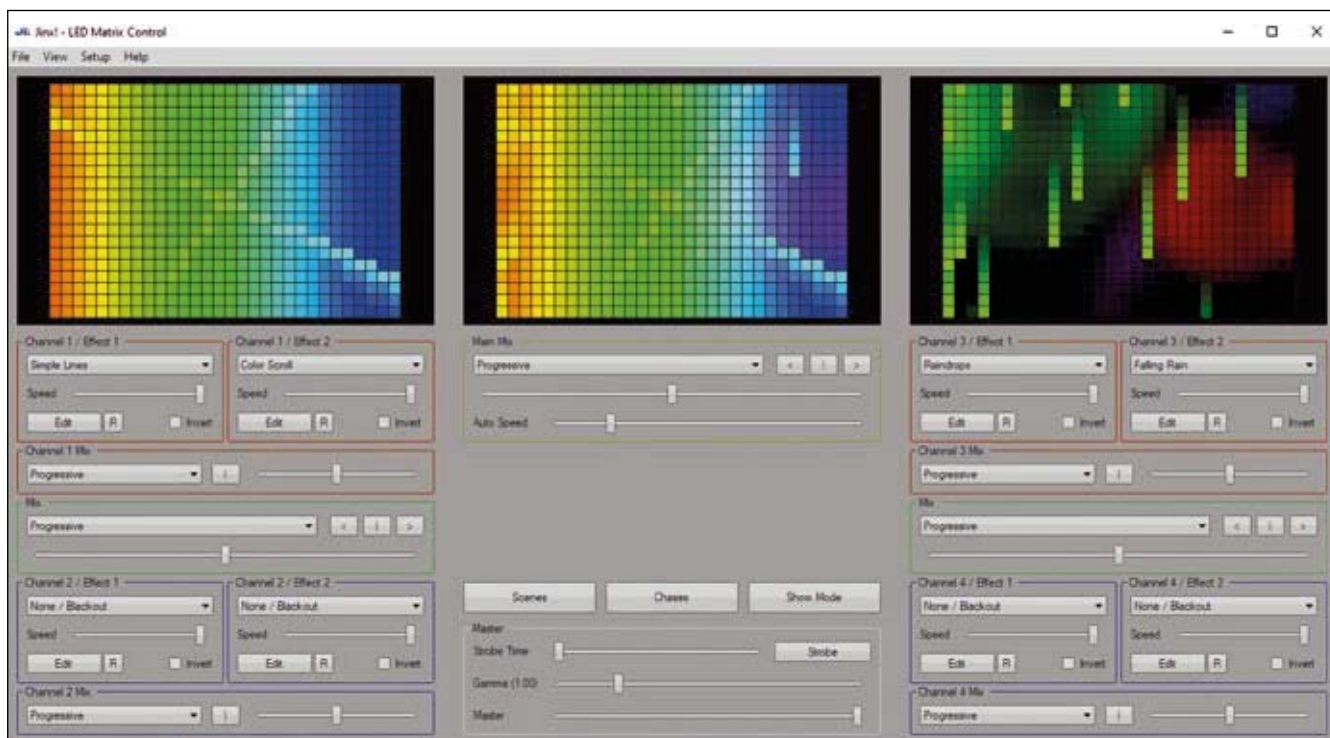


Fig. 13
Utilizzo di
Jinx una volta
terminata la
configura-
zione.

partire la configurazione iniziando dal 171esimo LED con il canale 0 dell'universo 2 ([0 | 0 | 0]). Ripetete questi procedimenti fino a quando non avete completato la configurazione di tutta la matrice. Cliccate dunque su Close.

4. L'ultimo passo consiste nello specificare la scheda di rete utilizzata per la trasmissione dei dati. Reclamoci dunque su Setup e poi su Network options. Si aprirà una finestra di dialogo come in **Fig. 12**. Nella sezione Network Output spuntiamo la voce Bind to specific IP Address / Nic. Nel campo IP Address selezioniamo l'indirizzo IP del nostro PC associato alla scheda di rete che stiamo utilizzando. Facciamo clic su OK e torniamo alla schermata iniziale.

La configurazione del programma è terminata, possiamo ora recarci in Setup poi Start Output e spuntiamo la casella per abilitare l'output su Fishino (**Fig. 13**). Possiamo dunque recarci nel primo riquadro e alla voce Channel 1/Effect 1 selezioniamo un effetto qualunque tra quelli a disposizione. Una volta selezionato si può cliccare su Edit per modificarne i parametri. Se i LED dovessero rimanere spenti è probabile che Jinx non sia stato abilitato attraverso il firewall.

CONFIGURAZIONE LIGHTJAMS

Lightjams è un software a pagamento, tra i più

completi e permette di gestire altri dispositivi e apparati scenografici. È possibile utilizzarlo gratuitamente, sebbene ogni 15 minuti il programma entri in modalità "cool down" per 30 secondi durante i quali non è possibile operare.

Vi invitiamo a visionare il video dimostrativo che si trova sul nostro sito insieme ai file del progetto (<https://youtu.be/owZ0p28XQ9Q>).

CONCLUSIONI

Si conclude qui la presentazione del protocollo Art-Net. Con i concetti fondamentali e gli strumenti forniti in questo articolo sarete in grado di realizzare numerosi progetti: l'unico limite è l'immaginazione. □

Cosa occorre?

I componenti utilizzati in questo progetto sono disponibili presso Futura Elettronica. La board Fishino Mega (cod. FISHINOMEGA) è in vendita a Euro 55,00, la matrice a LED (cod. FLEXLEDMX1616) è disponibile a Euro 49,00, mentre l'alimentatore switching 50W 5V (cod. MW05005) costa Euro 15,00. I prezzi si intendono IVA compresa.

Il materiale va richiesto a:

Futura Elettronica, Via Adige 11, 21013 Gallarate (VA)
Tel: 0331-799775 - <http://www.futurashop.it>

del Prof.
MICHELE MENNITI

TECNOLOGIA RFID, NFC: TEORIA E PRATICA

Conosciamo e sperimentiamo la tecnologia di identificazione tramite radiofrequenza, usando il modulo PN532.

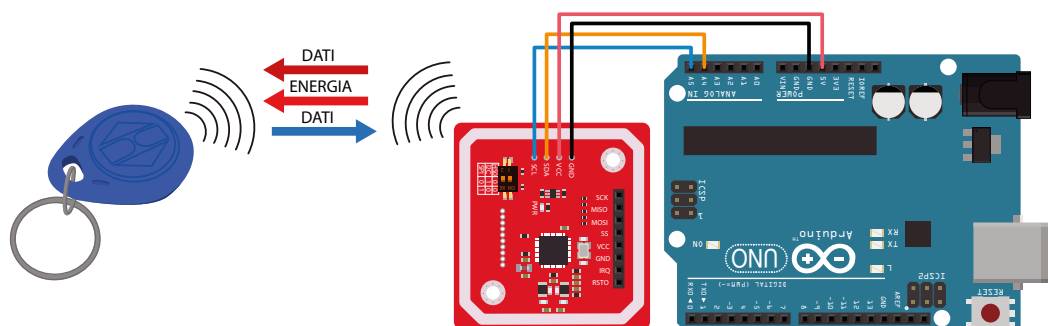
R

FID è l'acronimo di "Radio Frequency Identification" e indica una tecnologia che si basa sul riconoscimento di un oggetto identificato univocamente mediante l'UID (Unique Identifier) da parte di un lettore specifico, mediante una comunicazione in radiofrequenza.

I campi di applicazione di questa tecnologia sono numerosi e svariati: carte di credito, passaporti, sistemi antitaccheggio, controllo accessi e presenze, bigliettazione, gestione volumi biblioteche; questi sono solo alcuni esempi.

Il vantaggio principale della tecnologia rispetto ad altre (banda magnetica, chip, barcode) consiste nel fatto che oggetto e lettore non devono venire a contatto tra loro e non devono nemmeno essere in condizioni di visibilità unilaterale o reciproca. Una delle ultime implementazioni dell'RFID è costituita dalle carte senza contatto (contactless) basate sull'NFC, che

Fig. 1
Tecnologia NFC.



consentono di effettuare transazioni finanziarie in assoluta sicurezza.

Gli attori della tecnologia RFID sono i seguenti:

- **Tag o transponder:** è l'oggetto da leggere, in genere costituito da chip dotato di memoria e connesso a una bobina in funzione di antenna; può avere forma di portachiavi, Card ISO, chiave-vetta. In base alla tipologia un Tag può contenere il solo UID o anche dei messaggi di testo o dei comandi che il Reader legge o esegue nel momento in cui avviene la comunicazione.
- **Controller RFID (Reader o Lettore):** un sistema di ricetrasmittente a radiofrequenza controllato da un microcontrollore, il cui scopo è quello di leggere le informazioni (UID e altro) memorizzate nei Tag. Lo stesso lettore può, in determinate situazioni, trasformarsi in programmatore, per trasferire informazioni nei Tag.
- **Management system (Sistema di gestione):** non è una componente fondamentale, ma se esiste significa che ci si trova di fronte ad una rete di

Reader interconnessi tra loro e con il sistema informativo. Esso si occupa di rilevare la lettura dei Tag, ricavarne le informazioni in essi contenute e gestire le applicazioni che gli sono demandate.

I tag sono classificabili, in base alla tecnologia di funzionamento, in:

- **Passivi:** non hanno una propria fonte di energia, ma sfruttano quella del segnale di eccitazione proveniente dal Reader, il quale investendo l'antenna genera ai suoi capi una tensione indotta che viene raddrizzata e filtrata, così da alimentare il chip; questo si attiva e pulsa il proprio codice chiudendo in analogia i capi della bobina e determinando una reazione d'indotto nell'antenna del Reader, dove un apposito circuito rileva le variazioni di corrente trasformandole in impulsi uguali ai dati pulsati dal tag. In base alla frequenza di lavoro possono operare a distanza di pochi cm ma anche di alcuni metri.

IL MODULO RFID PN532

- Modalità lettore/scrittore RFID.
- Supporto schede Mifare 1k, 4k, Ultralight e DesFire, schede ISO/IEC 14443-4 come CD97BX, CD light, DesFire, P5CN072 (SMX), carte Innovision Jewel come la IRT5001, schede FeliCa come RCS_860 e RCS_854.
- Funzionamento come card 14443-A virtuale.
- Supporto NFC con telefoni Android.
- Antenna PCB incorporata, con distanza di comunicazione $5 \div 7$ cm.
- Cambio di livello on board, Standard 5V TTL per I²C e UART, 3,3V TTL SPI.
- Dimensioni: 42,7 x 40,4 x 4 mm.
- Tensione di alimentazione: 3,3V ÷ 5V.
- I²C/UART: 3,3V ÷ 24V TTL.
- Supporto I²C, SPI e HSU.
- SPI: 3.3V TTL con resistore da 100 ohm in serie. Può essere collegato direttamente all'interfaccia 5V di una scheda a microcontrollore come Arduino.



Fig. 2
Applicazioni
della
tecnologia
NFC.

- **Attivi:** alimentati da batterie, in genere sono costituiti da una coppia RTX come i Reader, poi sono caratterizzati da memoria di grande capacità ed addirittura possono essere dotati di sensori o altri elementi accessori. Le distanze in questi casi possono raggiungere 150+250 m.
- **Passivi assistiti da batteria:** sistemi ibridi rispetto ai primi due, in quanto la batteria è usata per alimentare solo alcuni componenti e non tutto il Tag.

Vediamo qualche informazione aggiuntiva riguardo ai Tag passivi, gli unici di nostro interesse in questo articolo; le applicazioni alle quali si prestano in genere sono correlate alle dimensioni dei Tag che,

essendo di fatto costituiti da un chip miniaturizzato e da una bobina, stampata su un supporto, con funzione di antenna, possono avere essere inseriti in carte di credito, etichette adesive, ma anche capsule, rendendoli di fatto univoci al riconoscimento. La memoria del chip può essere principalmente di "sola-lettura" o "riscrivibile"; in quest'ultimo caso è possibile memorizzare messaggi o istruzioni, modificabili a piacimento nel tempo; mediante opportuni programmi è possibile formattarli (azzeramento della memoria con possibile cambio di tipologia di Tag), cancellarli (azzeramento della memoria e inserimento di un messaggio vuoto) o resettarli (ripristino allo stato di fabbrica). I tipi di memoria possono coesistere nel chip (la parte di

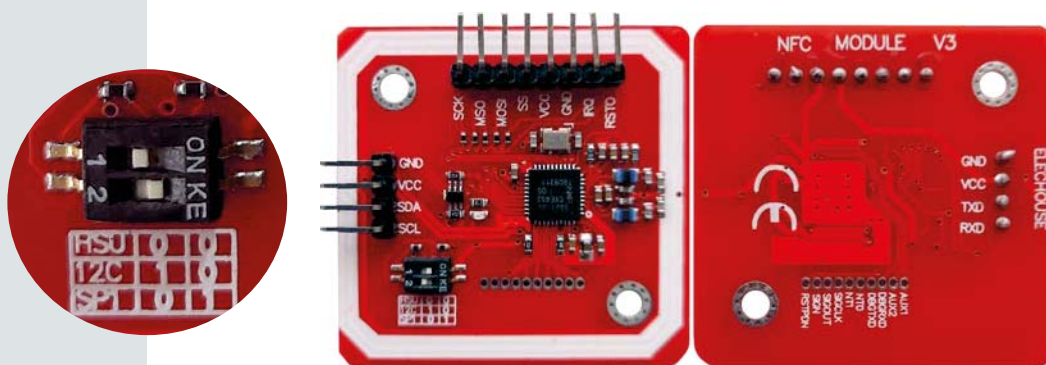


Fig. 3
I supporti
RFID-NFC.



sola lettura contiene l'UID, quella riscrivibile (e informazioni modificabili). Tipicamente un chip RFID può memorizzare da qualche byte ad alcuni kB. Le caratteristiche dell'antenna dipendono, oltre che dalla frequenza di lavoro, anche dalle necessità operative del Tag. La forma dell'antenna varia a seconda delle frequenze: per le frequenze basse si ricorre ad un certo numero di spire avvolte in aria (modelli HF) o su un cilindro di ferrite (modelli LF). Nel caso delle frequenze UHF vengono usati sistemi complessi come antenne a dipolo di varie forme. Il funzionamento di un Tag passivo è schematizzato nella **Fig.1**.

Per poter comunicare tra loro il Reader ed il Tag devono utilizzare lo stesso protocollo di comunicazione. Le tipologie di protocolli, e quindi di Tag, più diffuse sono: ISO 14443, Mifare Classic, Mifare Ultralight, FeliCa, NTag2xxx.

Le frequenze di comunicazione dell'RFID sono standardizzate e le due più usate universalmente sono:

Tabella 1
Impostazione dip-switch del modulo PN532
per il settaggio dell'interfaccia.

TIPO INTERFACCIA	DIP 1	DIP 2
HSU	OFF	OFF
I ² C	ON	OFF
SPI	OFF	ON

- banda LF (120÷145 kHz) è quella che è stata usata alle origini e si usa ancora oggi;
- banda HF; in essa rientra la frequenza fissa di 13,56 MHz, che oggi è quella universalmente usata come standard di riferimento per l'RFID;
- banda UHF, variabile da un continente all'altro e per tale ragione utilizzata solo in quelle applicazioni che richiedono distanze operative decisamente maggiori di quelle che raggiungono LF e HF.

NFC è l'acronimo di "Near Field Communication" (comunicazione a distanza ravvicinata) ed è una famiglia di sistemi di nuova generazione, strettamente correlata agli RFID a 13,56 MHz, utilizzata esclusivamente per applicazioni operative a brevissime distanze (max 10 cm). Questa nuova tecnologia include l'operatività bidirezionale P2P (acronimo di Peer-to-Peer), un particolare protocollo utilizzato per mettere in comunicazione tra loro due Reader; poiché anche questi posseggono un proprio UID e quindi sono univocamente identificabili, possono scambiare informazioni tra loro senza ricorrere ai Tag; in pratica sono costituiti da

Tabella 2
I collegamenti tra il modulo PN532 ed il mondo Arduino.

Mode	PN532 Module	Arduino UNO		Arduino Leonardo	Arduino Mega (2560)	Arduino Due	
Power	VCC	5V		5V	5V	5V	
IIC/I2C Mode	GND	GND		GND	GND	GND	
	SDA	A4/SDA		Pin 2 /SDA	Pin 20 /SDA	Pin 20 /SDA	
	SCL	A5/SCL		Pin 3/SCL	Pin 21/SCL	Pin 21/SCL	
HSU Mode	TXD	Pin 0	Could not present message in Serial Monitor on PC	Pin 0	Pin 19	Pin 19	
	RXD	Pin 1		Pin 1	Pin 18	Pin 18	
SPI Mode	SCK	Pin 13 or ICSP-3		ICSP-3	Pin 52 or ICSP-3	ICSP-3	Note: SPI on Due is still in developing
	MISO	Pin 12 or ICSP-1		ICSP-1	Pin 50 or ICSP-1	ICSP-1	
	MOSI	Pin 11 or ICSP-4		ICSP-4	Pin 51 or ICSP-4	ICSP-4	
	SS	Pin 10		Pin 10	Pin 10	Pin 10	

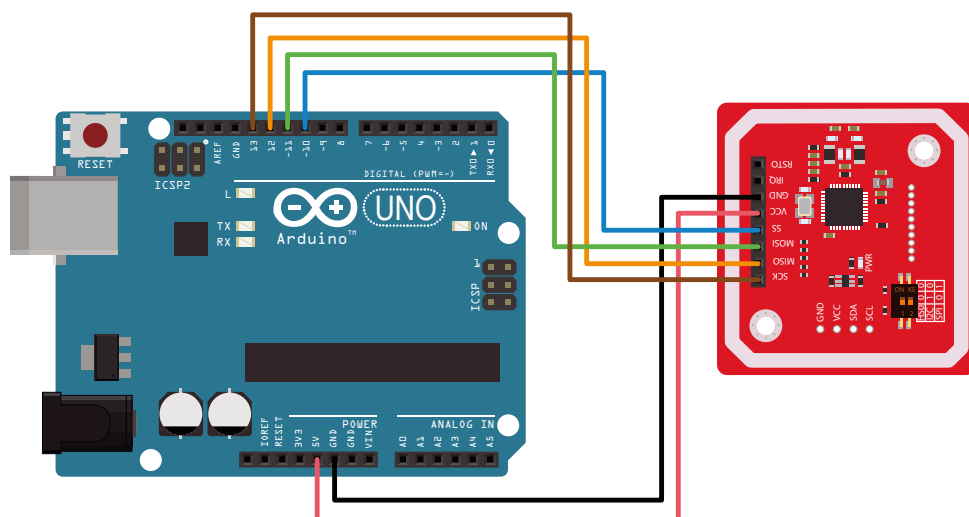


Fig. 4
Collegamento
SPI sull'header
laterale.

un unico circuito contenente sia un Reader attivo che un Tag passivo con caratteristiche operative compatibili con lo standard ISO 14443, adatte quindi ad applicazioni che richiedono grande sicurezza, come i pagamenti elettronici.

Quando entrano in contatto due apparecchi NFC, il Reader di quello che deve effettuare l'operazione si mette in contatto col Tag dell'altro, che gestisce l'operazione. Una volta che i due apparecchi sono in collegamento si attiva la comunicazione classica attiva-passiva che permette al primo apparecchio di assumere informazioni dall'altro, oppure quella innovativa attiva-attiva, che porta ad uno scambio di dati, in assoluta sicurezza. I circuiti NFC vengono inseriti, lato utente, in un apparato elettronico dotato di tutte le caratteristiche necessarie alla sua funzione (interfaccia utente, collegamento alla rete) come uno smartphone, mentre lato service (erogatore del servizio) in uno strumento specializzato per quella determinata operazione (POS, biglietteria). Esistono anche applicazioni P2P che sono praticabili mediante due sistemi utente. Il ruolo dell'NFC comunque si limita a creare un canale di collegamento sicuro tra due sistemi, non gli spetta la parte esecutiva dell'eventuale operazione finanziaria. La tecnologia NFC, specialmente se associata ad uno smartphone, permette di compiere svariate operazioni: aprire un link web, attivare o disattivare il Bluetooth o il WiFi, visualizzare (e salvare nella rubrica) un biglietto da visita in formato V-Card, aprire un'e-mail con campi precompilati, aprire o salvare un numero di telefono, avviare un'APP integrata nello smartphone, visualizzare un testo semplice, aprire un SMS precompilato (**Fig. 2**). Praticamente tutti i Tag riscrivibili di nuova generazione possono essere formattati secondo lo standard NFC, diventando interoperabili con

le numerose APP disponibili per il mondo degli smartphone. I dispositivi NFC hanno 3 diverse modalità operative che vengono utilizzate in diversi scenari di comunicazione:

- Modalità Read/Write: il dispositivo NFC legge o scrive Tag NFC;
- Modalità emulazione card: il dispositivo NFC emula un Tag o una Smart Card;
- Modalità peer-to-peer (P2P): due dispositivi NFC scambiano informazioni tra loro.

NDEF è l'acronimo di NFC Data Exchange Format ed è un formato standardizzato per i dati, che garantisce che le informazioni possano essere programmate su Tag NFC ed essere lette correttamente da dispositivi NFC diversi, con programmi e sistemi operativi differenti.

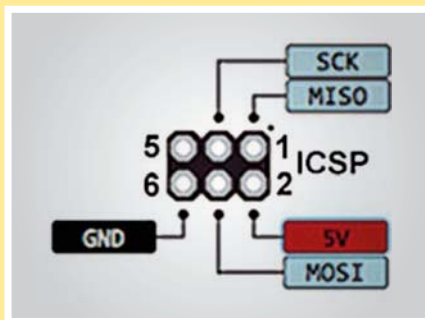
Per le nostre esercitazioni con l'RFID abbiamo acquistato una serie di supporti RFID/NFC, visibili in **Fig. 3**, si tratta di tre Tag portachiavi colorati, una CARD e due tipologie di adesivi NFC, uno neutro e l'altro con un logo colorato.

IL MODULO ELECHOUSE PN532

Ora che abbiamo le idee chiare sulla tecnologia RFID-NFC, possiamo presentare questo modulo realizzato dalla Elechouse e basato sul chip PN532, prodotto dalla NXP, il cui datasheet è reperibile al link: <https://www.nxp.com/docs/en/user-guide/141520.pdf>. Il modulo, acquistabile su futurashop.it (codice 2846-PN532) è dotato di ben tre interfacce con il mondo dei microcontrollori: I²C, SPI e HSU (High Speed UART). Il bordo bianco disegnato attorno al modulo copre e protegge l'antenna integrata del modulo che, come abbiamo visto nell'introduzione, è estremamente importante ai fini del funzionamento del circuito. A tal pro-

Fig. 5
Collegamento SPI
sul connettore ICSP
e piedinatura del
connettore.

ARDUINO UNO	MODULO PN532
pin 1 ICSP	MISO
pin 2 ICSP	VCC (uno qualsiasi dei due disponibili)
pin 3 ICSP	SCK
pin 4 ICSP	MOSI
pin 6 ICSP	GND (uno qualsiasi dei due disponibili)
pin 10 (header)	SS



posito una nota riportata nel manuale che abbiamo tradotto in Italiano e verrà fornito con il modulo, spiega che i collegamenti tra il modulo e l'Arduino vanno effettuati in modo che i fili incrocino l'antenna a 90° altrimenti ne potrebbero compromettere il buon funzionamento. Prima di collegare il modulo ad una board Arduino, occorre settarlo in base all'interfaccia che si desidera utilizzare. A tale scopo sul modulo c'è un mini dip-switch a due poli, le cui combinazioni stabiliscono il tipo di interfaccia da adottare (Tabella 1).

Per i collegamenti consigliamo di saldare sul modulo dei pin a 90°, così da soddisfare le indicazioni tecniche di Elechouse; una serie di cavetti jumper F-M e F-F permetteranno di completare il collegamento. Nel caso di un sistema stand-alone si potrebbero saldare i fili direttamente nei fori, ma sempre garantendo l'attraversamento dell'antenna a 90°. Per quanto riguarda i collegamenti tra il modulo ed Arduino occorre tenere presente che le due modalità I²C e HSU condividono gli stessi pin quindi accanto al connettore a 4 pin sono presenti due serigrafie, la prima sulla faccia superiore per l'I²C e la seconda sulla faccia inferiore per l'HSU; l'altro connettore, a 8 pin, riporta invece tutte le connessioni del collegamento SPI, oltre ad un pin aggiuntivo per l'IRQ ed un altro per il RESET. La **Tabella 2** riporta tutti i collegamenti da fare per le tre modalità in base alla board Arduino che si possiede; nelle nostre sperimentazioni comunque useremo Arduino UNO. Descriviamo ora nello specifico i tre tipi di collegamenti in base all'interfaccia selezionata sul modulo; alcuni ci torneranno utili nella sperimentazione della libreria che abbiamo testato in questo articolo.

COLLEGAMENTI PER L'INTERFACCIA SPI

Questi collegamenti si possono effettuare in due modi: ricorrendo agli header di Arduino oppure al connettore ICSP (più un pin dell'header).

La **Fig. 4** illustra il cablaggio per la prima modalità. In **Fig. 5** sono proposte le connessioni per la seconda modalità; per facilitare il collegamento ICSP ne riportiamo sotto la piedinatura. Nei dip del modulo bisogna impostare il pin 1 su OFF ed il pin 2 su ON.

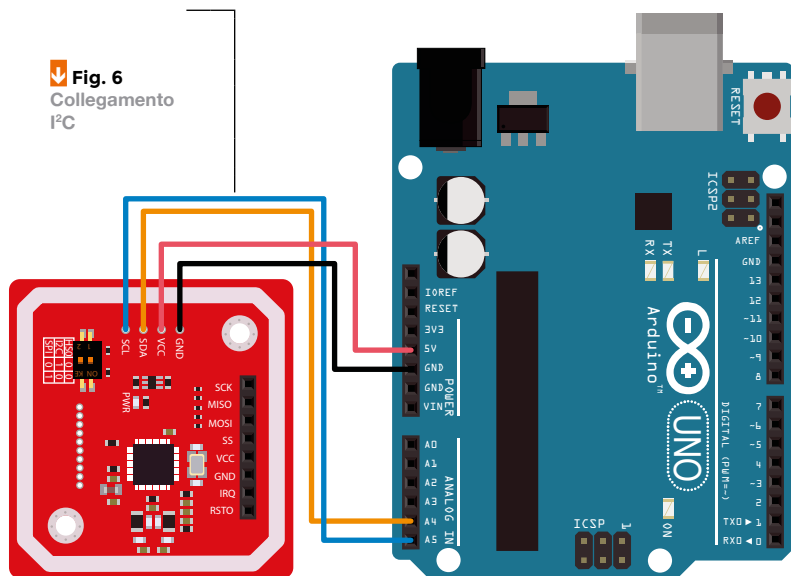
COLLEGAMENTI PER L'INTERFACCIA I²C

Alcuni sketch richiedono espressamente il collegamento I²C, che è più semplice in quanto necessita solo di quattro fili: due per l'alimentazione e due per i segnali, sono riportati in **Fig. 6**. Una volta effettuati i collegamenti occorre settare il doppio dip switch, pin 1 su ON e pin 2 su OFF. Il collegamento I²C richiede l'utilizzo di 2 resistori da 10k con funzione di pull-up, cioè collegati tra i due segnali SDA e SCL ed i 5V, in mancanza il modulo facilmente non sarà riconosciuto o lavorerà in modo anomalo. La cosa si risolve facilmente con l'aggiunta di una breadboard, di due resistori e alcuni cavetti jumper (schema proposto in **Fig. 7**).

COLLEGAMENTI PER L'INTERFACCIA HSU

L'ultima tipologia di collegamento prevista è l'HSU che, lato PN532, usa gli stessi pin per l'interfaccia-

Fig. 6
Collegamento
I²C



mento con Arduino. Infatti la serigrafia evidenzia i collegamenti I²C sul lato superiore del PCB e quelli HSU sul lato inferiore. Quindi anche in questo caso servono solo quattro fili: due per l'alimentazione e due per i segnali, come riportato nello schema di cablaggio in **Fig. 8** (pin 0 Arduino su TXD modulo e pin 1 Arduino su RXD modulo).

Una volta effettuati i collegamenti occorre settare il doppio dip switch: pin 1 su OFF e pin 2 su OFF. Considerate che, nel caso in cui questa modalità di collegamento venga adottata per un firmware che preveda l'utilizzo del Monitor Seriale, si creerebbe un conflitto, in quanto l'unica seriale di Arduino UNO sarebbe già occupata per la comunicazione con il microcontrollore di Arduino e non potrebbe inviare dati anche al Monitor Seriale. In questo caso è possibile utilizzare una board Arduino MEGA 2560, che possiede ben quattro seriali. I collegamenti da effettuare sono come quelli in **Fig. 9**. Naturalmente anche in questo caso occorre settare il doppio dip switch: pin 1 su OFF e pin 2 su OFF. Esiste comunque un'alternativa software, per chi non possiede una board Arduino MEGA e vuole usare Arduino UNO; è sufficiente includere nel firmware la libreria "SoftwareSerial.h" ed usare il relativo comando di simulazione seriale "SoftwareSerial SWSerial(2, 3);" con il quale in pratica si settano i pin 2 e 3 al posto dei pin 0 e 1, quindi i collegamenti da fare diventano quelli in **Fig. 10**:

LIBRERIA E HARDWARE PER I NOSTRI TEST

Le librerie realizzate per il chip PN532 sono numerose, ognuna ha le sue peculiarità, ma fondamentalmente sono simili tra loro; nelle numerose prove che abbiamo effettuato, la più consona ai nostri scopi si è rivelata la "SEED_ARDUINO_NFC"; per utilizzarla occorre andare al link https://github.com/Seeed-Studio/Seeed_Arduino_NFC, eseguire il comando "Download ZIP" dal pulsante CODE (**Fig. 11**), scegliere il percorso per scaricare il file "Seeed_Arduino_NFC-master.zip" ed infine rinominare il file in "Seeed_Arduino_NFC.zip".

Aprire quindi un nuovo sketch con una versione aggiornata dell'IDE (nel nostro caso 1.8.12) eseguire il comando Sketch - #include libreria - Aggiungi libreria da file .ZIP... e selezionare il file .zip, quindi chiudere l'IDE. La libreria è disponibile anche nella sezione download del sito www.elettronica.in col nome "Seeed_Arduino_NFC.zip".

Quanto all'hardware, abbiamo deciso di collegare il modulo PN532 alla board Arduino UNO mediante I²C-Bus, visto che questo metodo ci tornerà utile per il progetto che vi presenteremo a fine articolo.

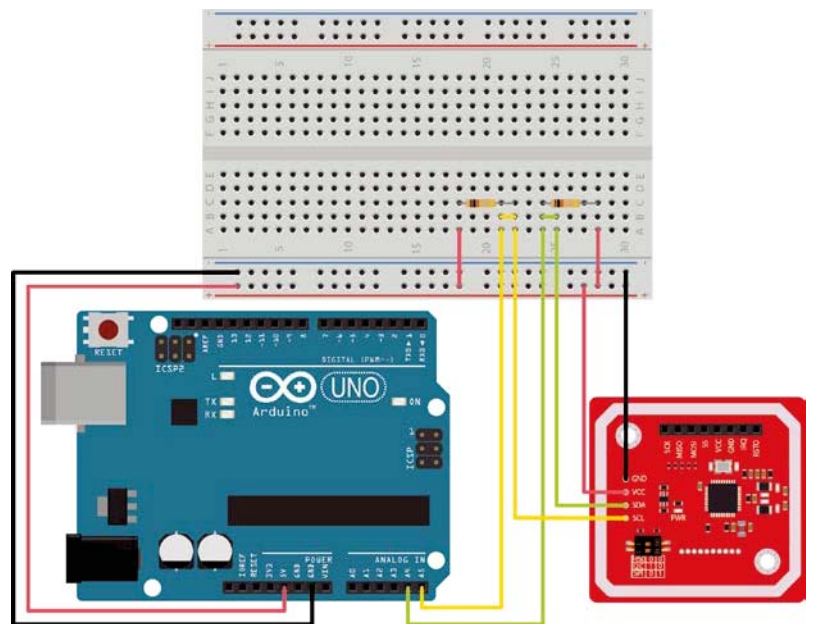


Fig. 7
Collegamento I²C con i due resistori di pull-up.

GLI SKETCH DI TEST

Useremo i supporti visti nell'introduzione e gli sketch che mette a disposizione la libreria. Apriamo l'IDE chiuso in precedenza, dopo aver incluso la libreria, ed impostiamo correttamente i parametri relativi alla board Arduino UNO ed alla COM ad essa associata.

Read Tag: lettura dei dati contenuti nei supporti

Mediante il comando "Apri" troviamo lo sketch Read Tag all'interno della nostra libreria; una volta aperto chiudere eventuali altri sketch aperti nell'IDE. Prima di poter essere caricato su Arduino lo sketch deve essere correttamente impostato, in quanto prevede la possibilità di operare con tutte le modalità del modulo: SPI, HSU, Software serial e I²C. Il principio per l'impostazione è semplice in quanto bisogna agire su un ciclo "if", impostando

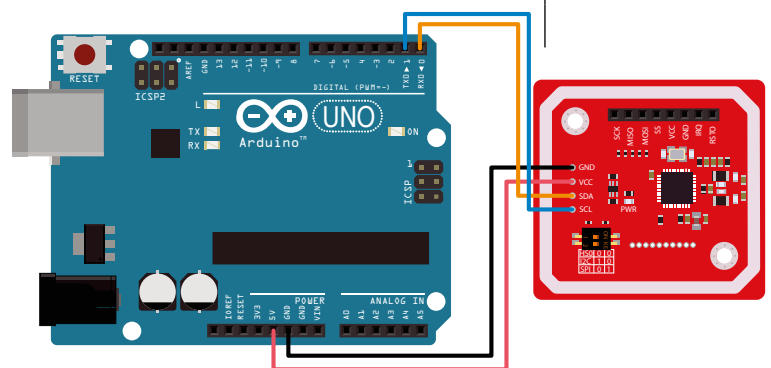
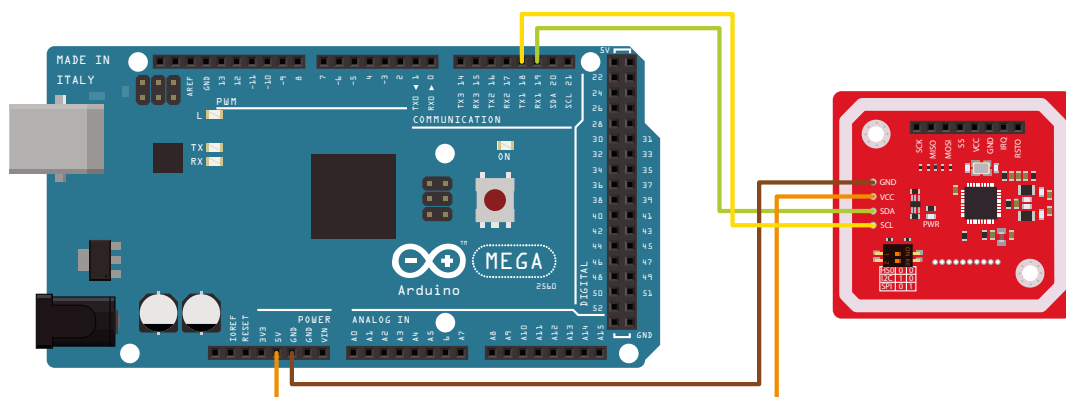


Fig. 8
Collegamento HSU con Arduino UNO.

Fig. 9
Collegamento
HSU con
Arduino MEGA.



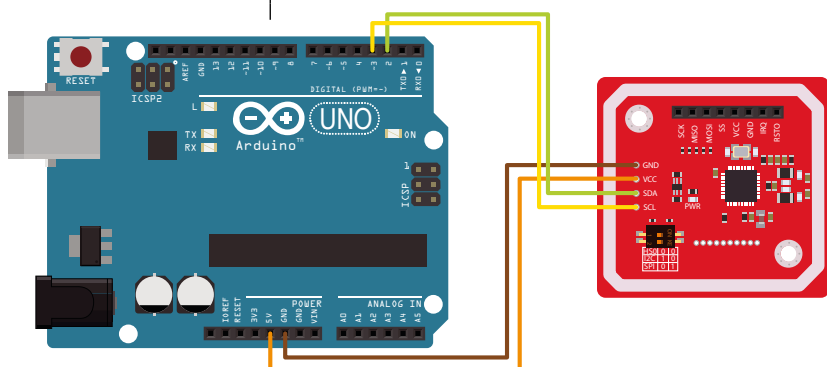
a 1 il tipo di collegamento adottato; per facilitare il compito del lettore abbiamo preparato la tabella riepilogativa **Tabella 3**. Nel nostro caso (modalità I²C) in pratica dobbiamo impostare a 0 sia l'if della riga 4 ("#if 0 // use SPI") che le "elif" delle righe 9 e 14. Se invece decidessimo di usare una delle due modalità SPI, la riga 4 andrà impostata come "#if 1 // use SPI" e la riga 7 andrà modificata in "PN532_SPI pn532spi(SPI, 10);", in quanto noi usiamo il pin 10 per pilotare la linea SS e non il pin 9 previsto dallo sketch originale.

Tabella 3
Le modifiche delle
righe degli sketch.

Modalità	riga 4	riga 9	riga 14
SPI	1	0	0
HSU	0	1	0
Soft serial	0	0	1
I ² C	0	0	0

Attenzione che i numeri di riga potrebbero cambiare, ma sarà facile localizzarli. Infine, a chi dispone di una vecchia versione di Arduino UNO, ma in generale non si sbaglia nemmeno con quelle più recenti, conviene aggiungere, come prima riga della "void setup" l'istruzione "delay(500);", lo scopo è quello di evitare blocchi indesiderati della seriale dell'Arduino, che potrebbero impedire successivi caricamenti di sketch

Fig. 10
Collegamento
HSU con
Arduino UNO
e libreria
SoftwareSerial.



sulla board. Prima del caricamento su Arduino occorre salvare lo sketch con le modifiche, il programma impedisce la sovrascrittura dei file originali, poco male, scegliamo come percorso la cartella Sketch predisposta per il nostro IDE, ed al suo interno ne creiamo una per i nostri scopi "SeedArduinoNFC_test", e vi salviamo il nostro sketch personalizzato. A questo punto possiamo usare il comando "Carica" per programmare Arduino con lo sketch di prova. Nel momento in cui finisce il caricamento dobbiamo semplicemente aprire il Monitor Seriale, nel caso succeda nulla o appaiano caratteri strani sicuramente occorre settare la velocità del collegamento seriale, portandola allo stesso valore dello sketch, ossia 9.600 baud, immediatamente apparirà il testo:

```
NDEF Reader
Found chip PN532
Firmware ver. 1.6
Scan a NFC tag
```

Il programma, una volta lanciato, ogni 5 secondi cerca di trovare un supporto RFID/NFC, appena ne avviciniamo uno viene immediatamente intercettato e letto. In questa prova alcuni supporti sono stati precedentemente programmati, per poter mostrare cosa viene visualizzato al momento della lettura. Se si vuole leggere più volte lo stesso supporto, tra una prova e l'altra deve essere allontanato dal modulo. In **Fig. 12** vedete i risultati di questo primo test.

- Tag ROSSO e ARANCIONE: non sono formattati secondo il protocollo NDEF, sono riconosciuti come NFC Tag – Mifare Classic, viene mostrato il loro UID (identificativo univoco) e segnalato che non ci sono dati memorizzati.
- Tag BLU, ADESIVO NEUTRO e CARD: sono riconosciuti come NFC Tag – NFC Forum Type 2, viene mostrato il loro UID, contengono 1 record di 3 byte vuoti.

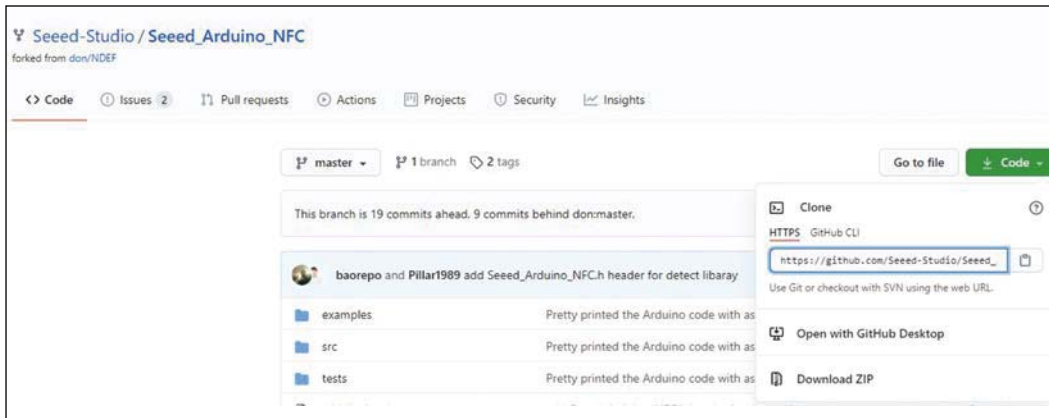


Fig. 11
Download
della libreria.

Fig. 12
I risultati della
lettura dei tag.

- **ADESIVO COLORATO:** riconosciuto come NFC Tag – NFC Forum Type 2, viene mostrato il suo UID, contiene 2 record (25 e 37 byte), il primo è di tipo "0x2 Mime Media" e contiene l'istruzione "com.android.chrome" per aprire il browser, mentre il secondo è di tipo "0x4 External" e contiene due istruzioni: il prefisso android.com:pkg e l'istruzione "com.wakdev.nfctasks".

Write Tag: scrittura di dati nei supporti

Con le stesse modalità viste per il precedente aprire lo sketch Write Tag, all'interno della libreria, e chiudere eventuali altri sketch aperti nell'IDE. Questo sketch può operare solo con le modalità SPI e I²C: nel primo caso, il nostro, occorre impostare a "1" la riga 3, nel secondo caso va lasciata a 0; occorre poi aggiungere la solita istruzione "delay(500);" come prima riga della "void setup", ed infine tradurre in modo adeguato i testi "messaggio" che vedremo poi sul Monitor Seriale:

Riga 28: "\n avvicina un supporto compatibile al lettore"

Riga 31: `message.addUriRecord("https://www.arduino.cc/");`

Riga 35: `SERIAL.println("Fatto. Prova a leggere il supporto col tuo smartphone.");`

Riga 32: `SERIAL.println("Scrittura fallita.");`

Questo programma, una volta rilevata la presenza di un supporto compatibile vi scrive un indirizzo internet in un record (riga 31); avvicinando successivamente uno smartphone al Tag/CARD, il browser predefinito aprirà il sito internet pre-memorizzato. Se invece di un URL volessimo memorizzare nel Tag un messaggio di testo (in questo caso "Ciao, Arduino!") sarà sufficiente modificare la sola riga 31, con la sintassi:

Riga 31: `message.addTextRecord("Ciao, Arduino!");`

COM7	COM7	COM7
TAG ROSSO	TAG ARANCIONE	TAG BLU
Scan a NFC tag	Scan a NFC tag	Scan a NFC tag
Tag is not NDEF formatted.	Tag is not NDEF formatted.	NFC Tag - Mifare Classic
NFC Tag - Mifare Classic	NFC Tag - Mifare Classic	UID A9 1D 1A A3
UID 76 CF B5 AC	UID 66 6B D5 A1	
No NDEF Message	No NDEF Message	NDEF Message 1 record, 3 bytes
		NDEF Record
		TNF 0x0 Empty
		Type Length 0x0 0
		Payload Length 0x0 0
		Type
		Payload
		Record is 3 bytes
COM7	COM7	
ADESIVO NEUTRO	CARD	
Scan a NFC tag	Scan a NFC tag	
NFC Tag - NFC Forum Type 2	NFC Tag - Mifare Classic	
UID 04 93 AA 6A 79 4D 80	UID CD D8 20 C1	
NDEF Message 1 record, 3 bytes	NDEF Message 1 record, 3 bytes	
NDEF Record	NDEF Record	
TNF 0x0 Empty	TNF 0x0 Empty	
Type Length 0x0 0	Type Length 0x0 0	
Payload Length 0x0 0	Payload Length 0x0 0	
Type	Type	
Payload	Payload	
Record is 3 bytes	Record is 3 bytes	
COM7		
ADESIVO COLORATO		
Scan a NFC tag		
NFC Tag - NFC Forum Type 2		
UID 04 82 9E 01 A6 39 03		
NDEF Message 2 records, 62 bytes		
NDEF Record		
TNF 0x2 Mime Media		
Type Length 0x4 4		
Payload Length 0x12 18		
Type 77 38 2F 31 w8/1		
Payload 63 6F 6D 2E 61 6E 64 72 6F 69 64 2E 63 68 72 6F 6D 65 com.android.chrome		
Record is 25 bytes		
NDEF Record		
TNF 0x4 External		
Type Length 0xF 15		
Payload Length 0x13 19		
Type 61 6E 64 72 6F 69 64 2E 63 6F 6D 3A 70 6B 67 android.com:pkg		
Payload 63 6F 6D 2E 77 61 6B 64 65 76 2E 6E 66 63 74 61 73 6B 73 com.wakdev.nfctasks		
Record is 37 bytes		



Fig. 13
I risultati della
scrittura dei
supporti test.

COM7	COM7
SUPPORTI NON FORMATTATI	SUPPORTI FORMATTATI
NDEF Writer Found chip PN532 Firmware ver. 1.6	NDEF Writer Found chip PN532 Firmware ver. 1.6
avvicina un supporto compatibile al lettore Error. Block Authentication failed for 4 Scrittura fallita.	avvicina un supporto compatibile al lettore Fatto. Prova a leggere il supporto col tuo smartphone.

Questa opzione ci tornerà utile per l'ultimo progetto pratico. Notate che i numeri di riga potrebbero cambiare, ma sarà facile localizzarli. Dopo aver salvato lo sketch modificato nella nostra cartella sketch, possiamo caricarlo sull'Arduino UNO e, al termine, aprire il Monitor Seriale. Il programma, una volta lanciato, ogni 5 secondi cerca di trovare un supporto RFID/NFC formattato, se lo trova effettua l'operazione di scrittura cancellando il contenuto precedente, in caso contrario (supporto non formattato) restituisce un errore specifico. Infatti utilizzando i supporti letti in precedenza ci siamo trovati di fronte a due esiti: i due Tag non formattati secondo lo standard NDEF

hanno restituito l'errore visibile nell'immagine di sinistra in **Fig. 13**; tutti gli altri hanno originato l'immagine di destra. Per verificare cosa è successo nei contenuti dei supporti abbiamo fatto ricorso nuovamente al programma "Read Tag": i due Tag non formattati (rosso e arancione) hanno ridato la stessa immagine della **Fig. 13**, mentre tutti gli altri hanno dato il risultato visibile nella **Fig. 14**: riconosciuto come NFC Tag – Mifare Classic, viene mostrato il suo UID, contiene 1 record (28 byte) di tipo "0x1 Well Known" e contiene l'istruzione "<https://www.arduino.cc>" per aprire il browser. A questo punto possiamo verificare il funzionamento dei supporti formattati programmati, disponendo di un qualsiasi smartphone Android dotato della funzione NFC.

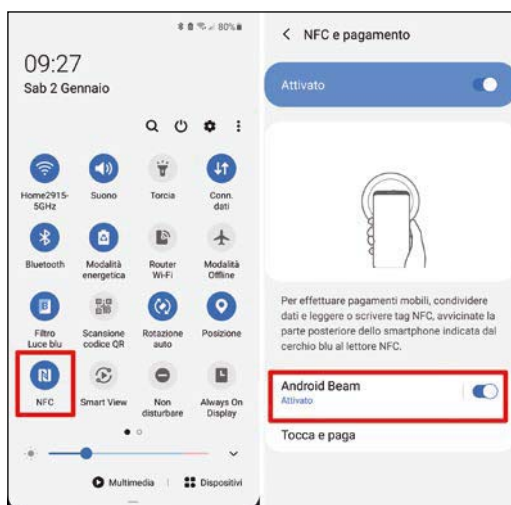
La prima operazione da fare è quella di entrare nelle Impostazioni, localizzare l'icona "NFC" ed attivarla, quindi entrare nelle sue proprietà ed attivare l'opzione "Beam" (**Fig. 15**).

A questo punto è sufficiente avvicinare il sensore dello smartphone ad uno qualsiasi dei supporti e si otterrà un segnale acustico di conferma lettura e l'immediata apertura della home page del sito ufficiale di Arduino. Nei casi in cui, come nel caso dello smartphone usato per le prove, non vi sia un browser predefinito c'è il passaggio intermedio della richiesta tipica di queste situazioni (**Fig. 16**).

Fig. 14
Lettura dei
supporti
formattati
dopo la
scrittura.

COM7
SUPPORTI FORMATTATI
NDEF Reader Found chip PN532 Firmware ver. 1.6
Scan a NFC tag
NFC Tag - Mifare Classic UID A9 1D 1A A3
NDEF Message 1 record, 28 bytes NDEF Record TNF 0x1 Well Known Type Length 0x1 1 Payload Length 0x18 24 Type 55 U Payload 00 68 74 74 70 73 3A 2F 2F 77 77 2E 61 72 64 75 69 6E 6F 2E 63 63 2F . https://www.arduino.cc Record is 28 bytes

Fig. 15
Impostazione
dello Smartphone
Android.



Write Tag Multiple Records: scrittura multipla di dati nei supporti

Aprire lo sketch WriteTagMultipleRecords, all'interno della nostra libreria e chiudere eventuali altri sketch aperti nell'IDE. Anche questo sketch può operare solo con le modalità SPI e I²C: nel primo caso, il nostro, occorre impostare a "1" la riga 3, nel secondo caso va lasciata a 0; occorre poi aggiungere la solita istruzione "delay(500);" come prima riga della "void setup", ed infine tradurre in modo adeguato i testi "messaggio" che vedremo poi sul Monitor seriale:

Riga 28: "\n avvicina un supporto compatibile al lettore"
Riga 31: message.addTextRecord("Ciao, Arduino!");

```

Riga 32: message.addUriRecord("https://www.arduino.cc/");
Riga 33: message.addTextRecord("Arrivederci, Arduino!");
Riga 36: SERIAL.println("Fatto. Prova a leggere il
supporto col tuo smartphone.");
Riga 38: SERIAL.println("Scrittura fallita.");

```

Questo programma opera in modo simile al precedente, con la differenza che prima e dopo il comando url invia dei messaggi di testo che saranno tutti visualizzati nello smartphone. I messaggi visibili nel Monitor seriale sono gli stessi del Write Tag. Anche in questo caso ricorriamo al programma "Read Tag" per verificare il Tag appena usato per il test, con il risultato visibile nella **Fig. 17**: riconosciuto come NFC Tag – Mifare Classic, viene mostrato il suo UID, contiene 3 record (21, 23, 28 byte) di tipo "Ox1 Well Known" e contiene il messaggio iniziale "Ciao, Arduino" ("en" è il prefisso usato automaticamente per lo sketch di testo), l'istruzione "https://www.arduino.cc" per aprire il browser ed il messaggio finale "Arrivederci, Arduino!". Naturalmente anche in questo caso facciamo un test di lettura con il nostro smartphone, il risultato è visibile in **Fig. 18**: ci sono i due messaggi e tra di essi l'esecuzione dell'URL.

PROGRAMMI UTILITY

Diamo ora uno sguardo ad alcune utility per pulire, cancellare o formattare i supporti. iniziamo da **CleanTag**, un programma che ripristina il Tag allo stato di fabbrica; nel caso di un Mifare Classic, ad

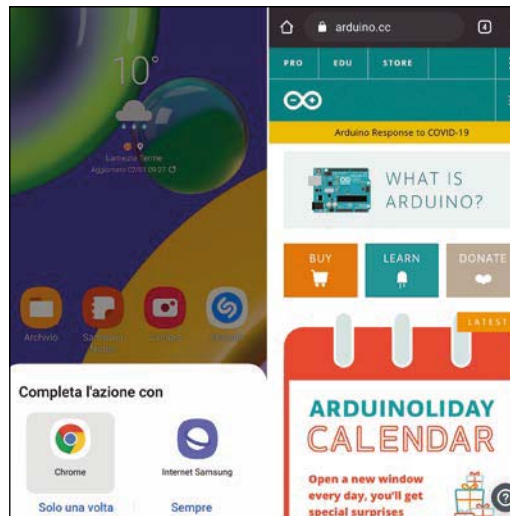


Fig. 16
Esecuzione
del comando
contenuto nel
supporto

esempio formattato come Tag NFC, viene azzerato e riportato al tipo Mifare Classic originale. Nel caso di Mifare Ultralight, i Tag vengono azzerati e lasciati vuoti. Per usarlo aprire lo sketch dalla libreria ed eseguire le due solite modifiche (riga 6 e delay in Setup) e le traduzioni:

```

Riga 31: SERIAL.println("\nposizione un Tag Mifare
Classic non formattato sul modulo.");
Riga 37: SERIAL.println("\nSuccesso, il Tag è stato
ripristinato alle condizioni di fabbrica.");
Riga 39: SERIAL.println("\nTag non ripristinabile.");

```

Fig. 18
Esecuzione
delle istruzioni
contenute nel
supporto.

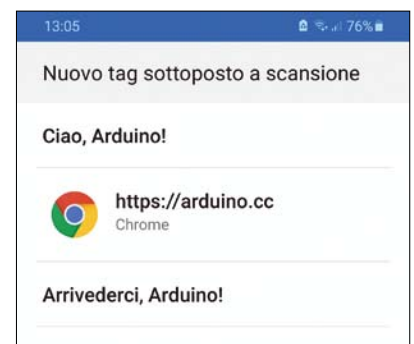
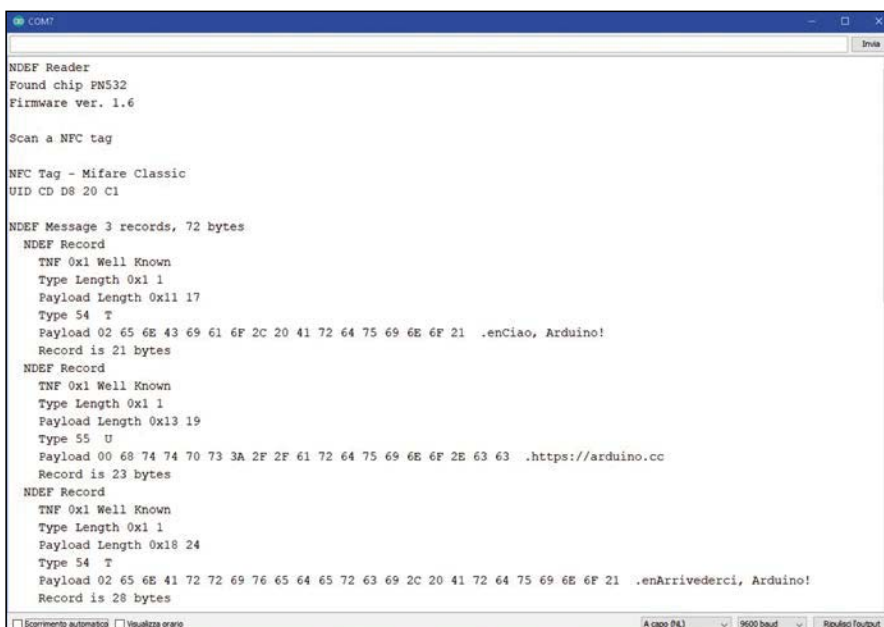


Fig. 17
Lettura del
supporto usato
per la scrittura
multipla.

Fig. 19
Ripristino di un
supporto Mifare
Classic.

```
COM7
NFC Tag Cleaner
Found chip PN532
Firmware ver. 1.6

Posiziona il Tag da ripristinare sul modulo.

Successo, il Tag è stato ripristinato alle condizioni di fabbrica.
```

Fig. 20
Formattazione di
supporto Mifare
Classic come TAG
NDEF.

```
COM7
NDEF Formatter
Found chip PN532
Firmware ver. 1.6

posiziona un tag Mifare Classic non formattato sul modulo.

Successo, il tag è stato formattato come NDEF.
```

Caricare il programma su Arduino UNO; l'operazione si effettua avvicinando il Tag al modulo, e andrà a buon fine solo se il Tag originario è di tipo Mifare Classic, altrimenti il programma darà errore. In

Fig. 19 la conferma del buon fine dell'operazione fatta su un Tag portachiavi Mifare Classic (blu). Leggendo ora il Tag con il "Read Tag" verrà restituito quanto di seguito:

```
NDEF Reader
Found chip PN532
Firmware ver. 1.6
Scan a NFC tag
Tag is not NDEF formatted
NFC Tag - Mifare classic
UID A9 1D 1A A3
No NDEF Message
```

indicante che il tag non è formattato NDEF, è di tipo Mifare classic e appare anche l'UID.

Fig. 21
Lettura del
supporto appena
formattato.

```
COM7
TAG ARANCIONE
NDEF Reader
Found chip PN532
Firmware ver. 1.6

Scan a NFC tag

NFC Tag - Mifare Classic
UID 66 6B D5 A1

NDEF Message 1 record, 3 bytes
NDEF Record
TNF 0x0 Empty
Type Length 0x0 0
Payload Length 0x0 0
Type
Payload
Record is 3 bytes
```

FormatTag

Questo programma serve per trasformare un Tag Mifare Classic in un Tag NDEF. Aprire lo sketch FormatTag all'interno della nostra libreria, e chiudere eventuali altri programmi aperti nell'IDE. Questo sketch può operare solo con le modalità SPI e I²C: nel primo caso, il nostro, occorre impostare a "1" la riga 6, nel secondo caso va lasciata a 0; occorre poi aggiungere la solita istruzione "delay(500);" come prima riga della "void setup", ed infine tradurre in modo adeguato i testi "messaggio" che vedremo poi sul Monitor Seriale:

Riga 32: `SERIAL.println("\nposiziona un Tag Mifare Classic non formattato sul modulo.");`

Riga 37: `SERIAL.println("\nSuccesso, il Tag è stato formattato come NDEF.");`

Riga 39: `SERIAL.println("\nFormattazione fallita.");`

Carichiamo ora il programma su Arduino UNO; l'operazione si effettua avvicinando il Tag al modulo, e andrà a buon fine solo se il Tag originario è di tipo Mifare Classic, altrimenti il programma darà errore. In **Fig. 20** la conferma del buon fine dell'operazione fatta su un Tag portachiavi Mifare Classic (arancione).

Leggiamo ora il Tag appena formattato con il solito "Read Tag": come mostra la **Fig. 21**, il contenuto del Tag ARANCIONE, che è diventato identico a quello blu, che era stato formattato prima dell'inizio di queste prove.

EraseTag

Questo programma Cancella un tag NFC scrivendo un messaggio NDEF vuoto. Per provarlo caricare lo

Fig. 22
Cancellazione
di un supporto
Tag NFC.

```
COM7
CARD
NFC Tag Eraser
Found chip PN532
Firmware ver. 1.6

Posiziona il Tag da cancellare sul modulo.

Successo, il Tag ora contiene un record vuoto.
```


sketch dalla libreria ed eseguire le due solite modifiche (riga 4 e delay in Setup) e le traduzioni:

```
Riga 29: SERIAL.println("\nPosiziona il Tag  
da cancellare sul modulo.");  
Riga 35: SERIAL.println("\nSuccesso, il Tag ora  
contiene un record vuoto.");  
Riga 37: SERIAL.println("\nTag non cancellabile.");
```

Caricare il programma su Arduino UNO; l'operazione si effettua avvicinando il Tag al modulo, e andrà a buon fine solo se il Tag è di tipo NFC, altrimenti il programma darà errore. In **Fig. 22** è visualizzata la conferma del buon fine dell'operazione fatta su un Tag NFC colorato.

Leggendo ora il Tag con il comando "Read Tag" e confrontando la **Fig. 23** con la risposta ottenuta in **Fig. 12** risulta evidente come il Tag che prima conteneva due record di 62 byte totali, ora contiene un record vuoto.

Tutti gli sketch qui proposti, opportunamente modificati, sono stati compressi nel file "Seed-ArduinoNFC_SKETCH.rar" e sono disponibili nella sezione download del sito www.elettronica.in.it.

RIEPILOGO DEI COMANDI DELLA LIBRERIA

Vediamo ora le piccole routine usate all'interno degli sketch precedenti, che ci torneranno utili nel momento in cui vorremo implementare le varie operazioni in un programma nostro.

Lettura di un messaggio memorizzato su un Tag

```
if (nfc.tagPresent()) {  
    NfcTag tag = nfc.read();  
    tag.print();  
}
```

Scrivere un messaggio in un Tag

```
if (nfc.tagPresent()) {  
    NdefMessage message = NdefMessage();  
    message.addTextRecord("Hello, Arduino!");  
    success = nfc.write(message);  
}
```

Cancellare un Tag

```
if (nfc.tagPresent()) {  
    success = nfc.erase();  
}
```

Formattare un Tag Mifare Classic come NDEF

```
if (nfc.tagPresent()) {  
    success = nfc.format();  
}
```

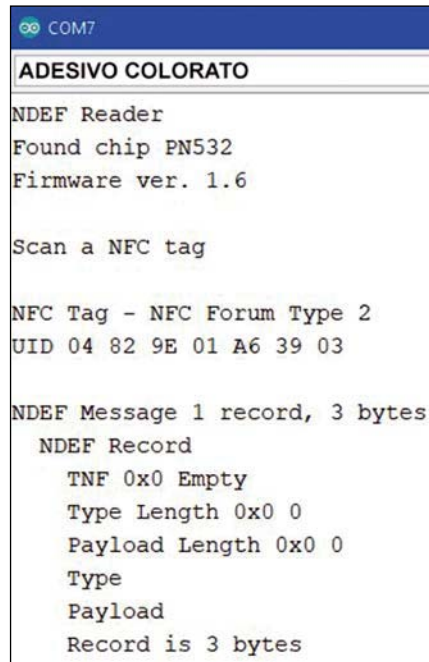


Fig. 23
Lettura del
supporto
appena
cancellato.

Resetare un Tag per riportarlo allo stato di fabbrica

```
if (nfc.tagPresent()) {  
    success = nfc.clean();  
}
```

NfcTag

La lettura di un Tag con il modulo restituisce un oggetto Tag NFC, che contiene metadati memorizzati sul Tag, come l'UID, la tipologia, dimensione della memoria. Quando viene letto un Tag NDEF, il Tag NFC contiene un "Ndef Message".

NdefMessage

Un NdefMessage è costituito da uno o più "NdefRecords". L'oggetto NdefMessage dispone di metodi di supporto per l'aggiunta di record:

```
nfcMessage.addTextRecord("hello, world");  
nfcMessage.addUriRecord("https://arduino.cc/");  
// url per aprire una pagina Internet
```

L'oggetto NdefMessage è responsabile della codifica di un messaggio NDEF in byte in modo che possa essere scritto in un Tag. Esso inoltre effettua l'operazione opposta, decodificando i byte letti da un messaggio NDEF memorizzato in un Tag.

NdefRecord

Un NdefRecord trasporta un payload, le cui informazioni sono all'interno di un NdefMessage.



PN532 E LIBRERIA SEED_ARDUINO_NFC

Per concludere questo articolo non c'è niente di meglio che mettere in pratica quanto abbiamo detto finora. Come vedremo è un circuito che si replica molto facilmente ma che può costituire la base per realizzare un sistema di controllo molto più potente. Il firmware che abbiamo preparato per il suo funzionamento è immediatamente scaricabile, in formato sorgente, dalla sezione download del sito www.elettronica.in.it. Avendo chiaro che ogni Tag dispone di un proprio UID (Identificativo univoco) ovviamente il firmware dovrà essere modificato, non solo tenendo conto delle indicazioni viste con lo studio della libreria, ma anche nei valori di UID, sostituendo quelli esistenti con quelli in vostro possesso, che potrete ottenere facilmente con il programma di lettura "Read Tag", già descritto. Il semplice circuito che vi proponiamo controlla l'accensione e lo spegnimento di tre LED di diverso colore in base al Tag che avviciniamo al modulo PN532. I colori scelti rispecchiano quelli dei tre Tag portachiavi di cui disponevamo per le prove. Oltre ad Arduino UNO e al modulo PN532 occorre:

- una breadboard con le linee di alimentazione laterale (rossa e nera oppure rossa e blu);
- LED arancione, rosso e blu 5 mm;
- 3 resistori da 220 ohm;
- 2 resistori da 10 kohm;
- 4 cavetti jumper M-M lunghi;
- 3 cavetti jumper M-M medi;
- 4 cavetti jumper M-F lunghi;
- 3 ponticelli jumper passo 11 (o 3 da 5 + 3 da 6);
- 2 ponticelli jumper passo 5;
- 2 ponticelli jumper passo 2;
- 7 Tag, possibilmente di tipologie differenti, o comunque contrassegnati anche con un semplice numero o lettera o bollino adesivo colorato, in modo che siano distinguibili tra loro.

Per la realizzazione del circuito fate riferimento allo schema di cablaggio illustrato nella **Fig. 24**. Il firmware da scaricare è **LED_Control.rar** (tre schede): la cartella ricavata dalla decompressione del file dovrà essere posizionata nella cartella degli sketch dell'IDE in vostro possesso, naturalmente in precedenza dovrà essere stata installata la libreria.

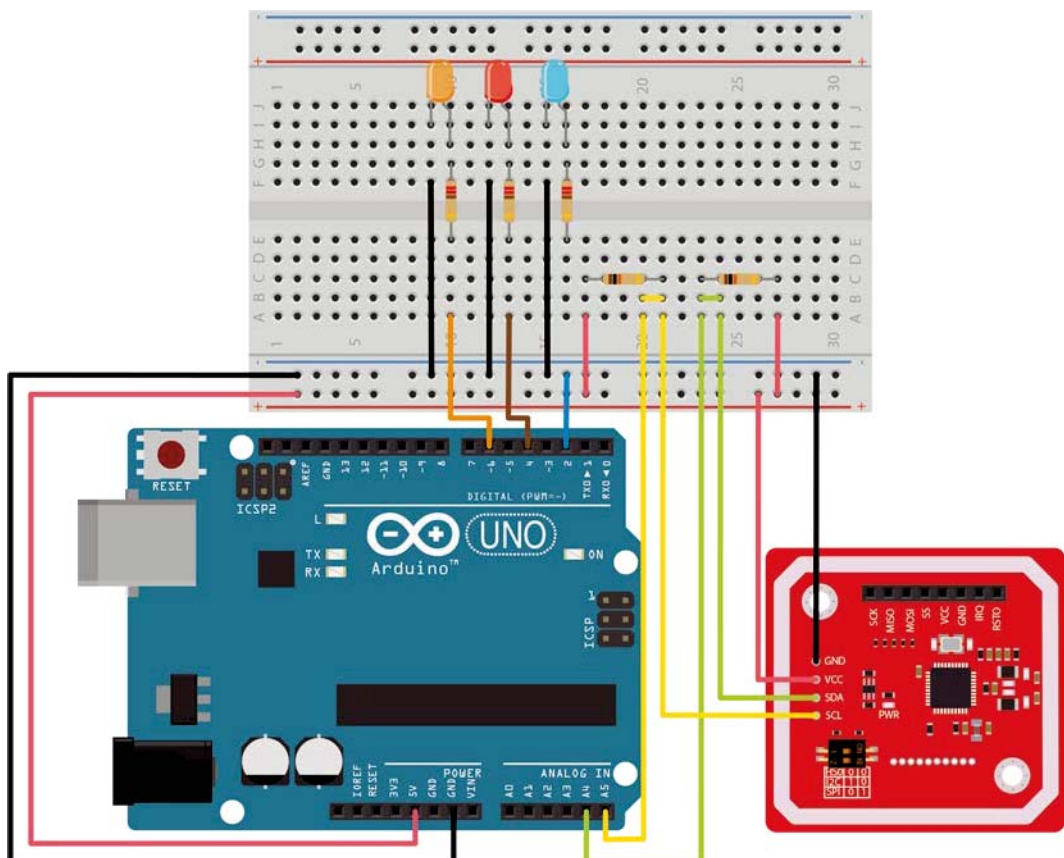


Fig. 24
Schema pratico
dell'applicazione.

PREPARAZIONE DEI TAG:

Mediante lo sketch "Read Tag" bisogna effettuare la lettura dell'UID di tutti i Tag prendendone nota; poi bisogna sceglierne 3, ed in essi va memorizzato il testo "blu", "rosso", "arancione" per replicare il nostro caso, oppure ciò che decidete, mediante lo sketch "Write Tag" opportunamente modificato per la scrittura di testi. Gli altri possono essere lasciati senza contenuto, anche se qualcuno potrebbe necessitare di essere formattato con lo sketch "FormatTag" per acquisire la compatibilità con lo standard NDEF.

MODIFICHE AL FIRMWARE

Una volta aperto lo sketch nell'IDE dell'Arduino occorrerà modificare tutte le righe contenenti gli elementi di distinzione dei Tag; in queste indicazioni daremo per scontato che replicherete il progetto come previsto da noi; coloro che sono in grado di modificare lo sketch senza particolari problemi, non avranno bisogno di indicazioni specifiche. Nella scheda principale la riga 71 dovrà contenere il valore "c = 4" per la corretta interpretazione dei testi memorizzati sui Tag con lo sketch "Write Tag". Scheda TAG_UID:

Riga 3: sostituire l'UID con quello del vostro supporto mediante il quale volete ottenere l'accensione contemporanea dei tre LED;
Riga 9: sostituire l'UID con quello del vostro supporto mediante il quale volete ottenere l'accensione contemporanea dei due LED blu e rosso;
Riga 15: sostituire l'UID con quello del vostro supporto mediante il quale volete ottenere l'accensione contemporanea dei due LED arancio e rosso;
Riga 23: questa riga non va modificata, nel momento in cui userete un Tag non previsto nelle modifiche si otterranno tre lampeggi di tutti e tre i LED in contemporanea.

SCHEDA TAG_MESSAGGIO:

Righe 3-9-15: contengono i testi memorizzati nei tre Tag, che serviranno per l'accensione dei singoli LED, naturalmente dovranno corrispondere ai testi effettivamente memorizzati nei Tag (facendo attenzione a maiuscole e minuscole) altrimenti il riconoscimento non avverrà.

A questo punto occorre caricare lo sketch sull'Arduino, aprire il Monitor Seriale ed avvicinare uno alla volta i Tag per verificare se tutto è stato fatto correttamente.

Come già detto noi abbiamo usato Tag di diverso tipo, ma possono essere benissimo tutti dello

TIPO	UID	TESTO	DEFINIZIONE
CARD	CD:D8:20:C1		riga 3 scheda TAG_UID
Ad. Neutro	04:93:AA:6A:79:4D:80		riga 9 scheda TAG_UID
Ad. Colorato	04:82:9E:01:A6:39:03		riga 15 scheda TAG_UID
Rosso	76:CF:B5:AC	rosso	riga 3 scheda TAG_messaggio
Blu	A9:1D:1A:A3	blu	riga 9 scheda TAG_messaggio
Arancione	66:6B:D5:A1	arancione	riga 15 scheda TAG_messaggio
CARD 2	C5:05:BE:23		non definita nello sketch

stesso tipo, l'importante è contrassegnarli e creare una tabella con gli elementi personalizzati, altrimenti, in caso di problemi, diventa difficile raccapezzarsi. Un esempio può essere la **Tabella 4**, realizzata sulla base dei nostri Tag. Per l'esperienza completa servono 7 Tag: tre con i messaggi di testo inseriti nello sketch, tre con gli UID inseriti nello sketch ed uno (o più) con informazioni non inserite nello sketch, tutti quelli di questa categoria provocheranno il lampeggiamento dei tre LED. Una volta verificato il buon funzionamento del sistema Vi consigliamo di commentare tutte le righe dello sketch che servono per il Monitor seriale, in quanto diventano inutili e rallentano l'esecuzione del firmware. Ovviamente questo progetto è dimostrativo ed ha lo scopo di spiegare come associare un Tag ad una qualsiasi funzione, mediante l'attivazione di un pin; a partire dai comandi essenziali è possibile realizzare dei sistemi di controllo molto più sofisticati.

CONCLUSIONI

Bene, arrivati a questo punto non ci resta che augurarvi buon lavoro nel mondo dell'RFID, utilizzando il modulo PN532 e mettendo a frutto le nozioni che vi abbiamo trasmesso in queste pagine. 

 **Tabella 4**
I Tag usati in questo progetto e le loro specifiche.



Cosa occorre?

I componenti utilizzati in questo progetto sono disponibili presso Futura Elettronica. Il lettore NFC/RFID (cod. PN532) è in vendita a Euro 14,00, altri tipi di transponder sono disponibili come prodotti correlati. Le board Arduino sono disponibili sempre sul sito www.futurashop.it. I prezzi si intendono IVA compresa.

Il materiale va richiesto a:

Futura Elettronica, Via Adige 11, 21013 Gallarate (VA)
Tel: 0331-799775 - <http://www.futurashop.it>



Nuove telecamere



Telecamera WiFi motorizzata

- Sensore d'immagine CMOS 2Mpx
- Pixel effettivi 1920x1080
- Ottica 3.6mm
- 4 LED IR
- 4 LED bianchi
- Portata LED 30 metri
- Funzione PAN/TILT con rotazione orizzontale di 320° e verticale di 90°
- Compressione video H.264/H.264+H.265X
- Microfono incorporato
- Speaker
- Funzione Motion Detection
- Supporto ONVIF
- Registrazione su SD-Card (da 8GB a 128GB)
- Potenza assorbita 6W max
- Alimentazione 12Vdc/2A (con adattatore di rete)
- Dimensioni 160x107x107 mm
- Peso 500 grammi

€ 79,00
cod. **FR769**



Prezzi IVA inclusa

Telecamera Wi-Fi con LED dual light

- Sensore d'immagine CMOS 2Mpx
- Pixel effettivi 1920x1080
- Ottica 3.6 mm
- 2 LED IR
- 4 LED bianchi
- Portata LED 30 metri
- Compressione video H.264/H.264+H.265X
- Microfono incorporato
- Speaker
- Funzione Motion Detection
- Supporto ONVIF
- Protocolli supportati RTSP, FTP, PPPOE, DHCP, DDNS, NTP, UPnP
- Registrazione su SD-Card (da 8GB a 128GB) 2.4835GHz
- Potenza assorbita 6W max
- Alimentazione 12Vdc/2A (con adattatore di rete)
- Dimensioni 230x130x115 mm
- Peso 400 grammi

€ 52,00
cod. **FR768**



Scopri la gamma completa
su www.futurashop.it



Telecamera WiFi IP Dome Fisheye 360° con IR e sensore di movimento

- Sensore d'immagine CMOS 2Mpx
- Sistema PAL/NTSC
- Pixel effettivi 1920x1080
- Portata 10 m
- Registrazione su SD-Card (max 128GB)
- Alimentazione 12Vdc/1A
- Potenza assorbita 5W max
- Dimensioni: 15,2x16,5x4 cm
- Peso 250 grammi

€ 45,00
cod. **FR766**

Telecamera WiFi da 2MPX, IR, con pannello solare e batteria ricaricabile

- Sensore d'immagine FHD 1080P CMOS 1/2.7 PS5230
- Risoluzione Video 1920x1080/15 fps
- Portata LED IR 10 metri
- Alimentazione con 2 batterie al litio tipo 18650 ricaricate tramite pannello solare integrato
- Registrazione su SD-Card (da 4GB a 64GB)
- Grado di protezione IP67

€ 119,00
cod. **FR767N**



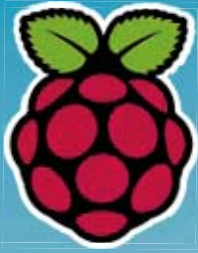
Telecamera WiFi, PIR motion con memorizzazione dati su SD Card e batteria ricaricabile



€ 98,00
cod. **FR789**



- Sensore d'immagine CMOS 2Mpx
- Risoluzione Video 1920x1080/25 fps
- Angolo di ripresa 100° wide-angle
- Portata LED IR 6 metri
- Ottica 3.6mm
- Sensibilità 0,1 LUX
- Registrazione su SD-Card (max 64GB)
- Alimentazione 2 batterie 18650 3,7V 3000Mah (incluse)
- Grado di protezione IP66
- Peso 550g
- Dimensioni 95x53x80 mm



Eva SMILE

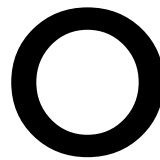


SMILE DETECTOR

di DANIEL SCAINI



Analizziamo i volti all'interno di un video o di una ripresa live e identifichiamo chi sta sorridendo mediante un algoritmo di Intelligenza Artificiale che identifica le emozioni.



Oggigiorno siamo spesso bombardati da film in cui per aprire una porta o accedere ad una zona di sicurezza occorre la scansione biometrica del soggetto; o per aprire una cassaforte è necessario il volto del proprietario seguito da una passphrase curiosa e bizzarra.

Dettagli molto interessanti che aggiungono quel tocco futuristico e suscitano notevole interesse, anche verso persone non completamente avvezze alla tecnologia.

Ma siamo davvero giunti a quel livello anche in caso di attrezzature low-cost? Non sarebbe fantastico se in ufficio, per accedere al piano, al posto del badge bastasse il volto? Ebbene, siamo già a questo livello di tecnologia, malgrado sembrano quasi incredibili i progressi fatti negli ultimi anni sotto l'aspetto del Machine learning e della Computer Vision. E ancora più incredibile è che cose del genere possano esser

fatte con hardware economico e diffuso come la scheda Raspberry Pi, la quale implementa la potenza di calcolo che basta a riconoscere in modo affidabile il volto di una persona con una precisione sorprendente, distinguendo anche emozioni come il sorriso.

Tutto ciò è possibile grazie alla disponibilità di diversi tool, primo fra questi OpenCV, il quale mette a disposizione diversi algoritmi per la risoluzione di questi problemi.

Attraverso il Deep Learning e alcuni classificatori ed encoder è possibile convertire il volto umano in una serie di punti e numeri facilmente confrontabili con elementi di riferimento.

Questo articolo si pone proprio questo obiettivo: identificare e riconoscere un viso e la sua espressione, indicandoci se è sorridente o meno.

Proprio quest'ultima feature potrebbe essere usata in un futuro per creare una passphrase per poter discernere se questo volto appartiene ad un soggetto reale o una fotografia e, in base anche a questo, abilitare o meno un pin di GPIO della nostra scheda.

IL LINGUAGGIO

Come già fatto in precedenti articoli dedicati al riconoscimento d'immagine (fascicoli n°250 e n° 251, rispettivamente riconoscimento mascherina e identificazione sesso ed età) il linguaggio di programmazione che andremo ad utilizzare è il Python; è stato scelto questo linguaggio poiché molto semplice nella sintassi, facilita l'interazione ed integrazione con altri linguaggi ed è infine orientato agli oggetti, così come Java e C++.

Per queste caratteristiche risulta molto usato dagli sviluppatori i quali lo scelgono, con ben l'84%, come linguaggio di programmazione primario.

Inoltre risulta molto usato soprattutto per fini

accademici e di ricerca, in corrispondenza di piattaforme embedded per il Machine Learning e per la Computer Graphics.

RASPBERRY PI

La macchina scelta per questa elaborazione è senza dubbio di primaria importanza. È consigliato l'uso di una Raspberry Pi 4 in modo da avere un minimo di 4GB di RAM disponibili (8GB sarebbe l'ideale) ed anche un sistema aggiornato in grado di poter supportare anche una maggiore elaborazione audio e video (**Fig. 1**).

Rappresenta la quarta generazione prodotta dalla Raspberry Foundation e non è solo adatta a chi vuole imparare a programmare, sviluppare e giocare, ma grazie a numerosi pin è particolarmente adatta anche per progetti di IoT complessi.

Guidata da un processore ARM offre prestazioni paragonabili ad un Personal Computer entry-level ma può gestire anche flussi video 4K.

Considerata ormai un caposaldo per chi appunto si cimenta in questo tipo di progetti, offre numerosi dispositivi da poter abbinare come: moduli telecamera, sensori, GPS, display LCD, ecc..

Inoltre dispone di numerosi pin, sia di alimentazione a 3,3v, sia per poter interfacciare periferiche, relay o sensori di diversa natura e gestibili tranquillamente a livello di codice Python o C++ attraverso un insieme di librerie messe a disposizione dal produttore.

A corredo di questa andremo ad utilizzare una Pi-camera rev 1.3 con 5MP questa verrà collegata direttamente sulla Raspberry Pi attraverso il connettore CSI (**Fig. 1**).

Questo modulo può offrire un'immagine chiara e nitida con una risoluzione di 5MP o registrare video a 1.080p con 30 fps.

Il sensore installato sul modulo ha un connettore

Fig. 1
La scheda Raspberry Pi 4 con 8GB RAM e il modulo fotocamera che andremo a connettere alla Raspberry Pi.



Ribbon a 15pin e attraverso la connessione in seriale con il connettore CSI offre un'ottima velocità di trasferimento dati, rendendo questo strumento particolarmente adatto a progetti nell'ambito embedded/mobile.

SSD VS SD

La configurazione predefinita della nostra piattaforma prevede una scheda microSD come supporto di memorizzazione e per far girare il proprio sistema operativo.

Ciò teoricamente porta tutte le operazioni di lettura e scrittura ad un massimo teorico di 50Mbps, mentre nella realtà, purtroppo, il valore è leggermente inferiore sfiorando i 38 Mbps nelle scritture in sequenza. Questo si traduce in un rallentamento dell'intera Raspberry Pi, poiché renderà molto più lento recuperare gli oggetti (nel nostro caso i modelli elaborati) e la visualizzazione in caso di filmati molto impegnativi.

Tuttavia sappiamo da specifiche che uno dei più grandi miglioramenti della Raspberry Pi 4 è, senza dubbio, quello di possedere due porte USB 3.0. A differenza dei suoi predecessori, i quali espongono solo porte 2.0 con una capacità di lettura/scrittura teorica di 480Mbps/60Mbps. Il nuovo modello l'interfaccia USB può arrivare teoricamente ad una capacità di 5Gbps/620Mbps. Chiaramente questi dati sono teorici, ma si percepisce già il boost nelle performance.

I tempi di apertura dei programmi migliorano a vista d'occhio, così come i tempi di scrittura e lettura su grandi moli di dati.

In grafico in **Fig. 2** possiamo notare infatti i tempi di apertura di Chromium, la versione lite del più famoso browser Chrome di Google (**Fig. 2**).

Si evince chiaramente un miglioramento stimato del 30/35%. Invece per quanto riguarda le ope-

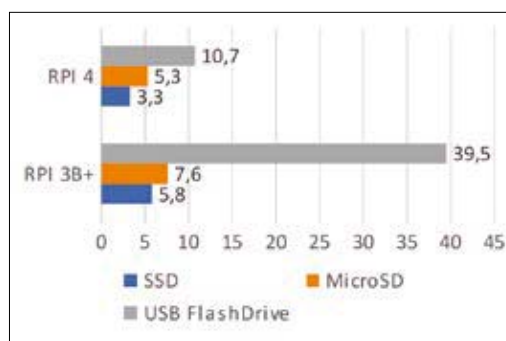


Fig. 2
A confronto il tempo impiegato nell'apertura di un'applicazione da parte degli ultimi due modelli di Raspberry Pi con i diversi tipi di supporto.

razioni di scrittura e lettura, con blocchi da 4k, il grafico proposto nella **Fig. 3** rende ancora meglio il distacco dei valori. Migliora la responsività del sistema operativo, cui è però difficile attribuire un numero.

Ovviamente questo miglioramento non riguarda tutte le operazioni: infatti non si hanno miglioramenti significativi per quanto riguarda le operazioni di boot, il caricamento delle pagine web e lo streaming video.

Quindi in caso di installazione domestica di un server web multimediale, probabilmente, questa configurazione non apporterà molti benefici al vostro sistema, ma se, come nel nostro caso, state utilizzando la piattaforma per sviluppare codice per creare programmi che hanno operazioni di I/O continue questa operazione avrà sicuramente riscontri positivi.

Nel periodo precedente la parola web fa da discriminante fra un servizio che rende fruibili pagine web o un servizio di video residenti su HDD locale. Altra cosa molto importante da tenere in considerazione è il fatto che ogni SSD e ogni MicroSD hanno propri tempi di accesso e banda utile, quindi non potremo aspettarci di replicare esattamente i grafici precedenti o di notare in miglioramenti

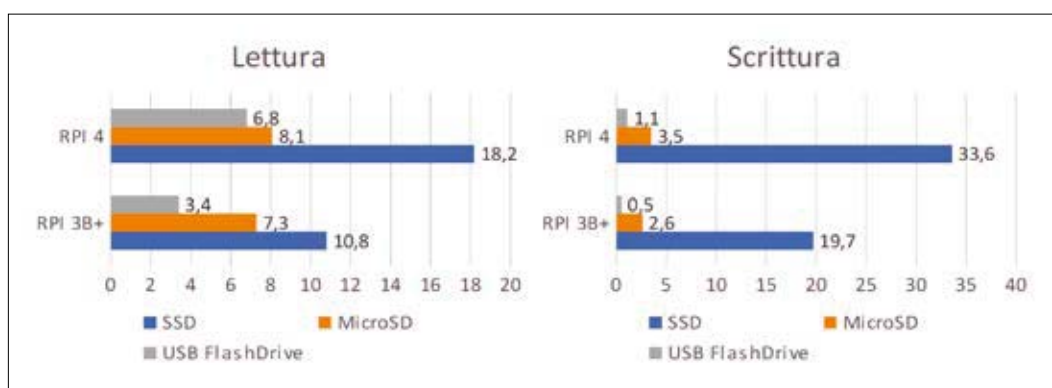
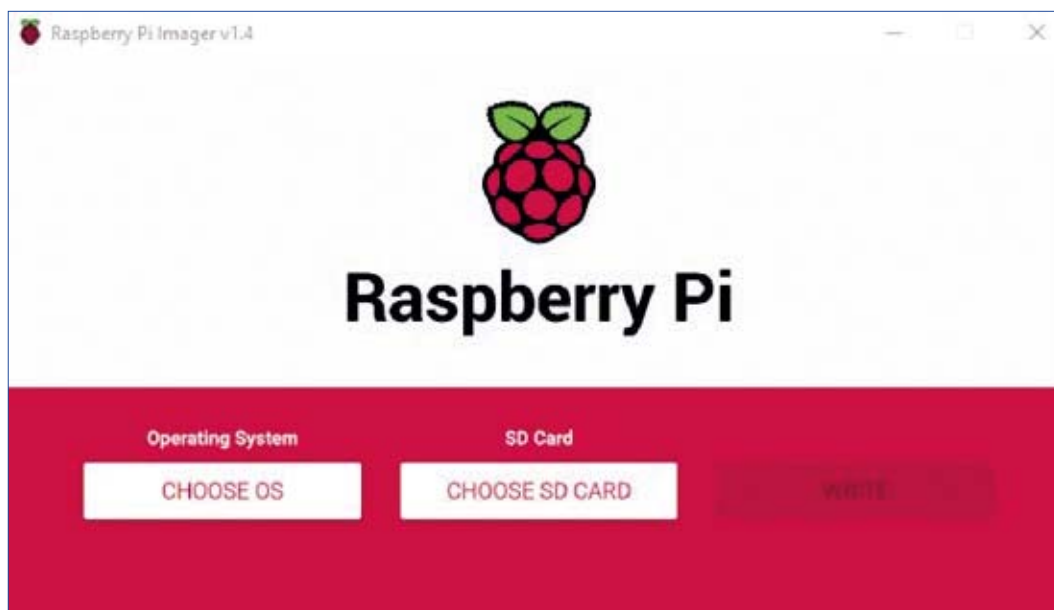


Fig. 3
A confronto, il tempo impiegato nelle operazioni di lettura e scrittura da parte degli ultimi due modelli di Raspberry con i diversi tipi di supporto di memorizzazione.

Fig. 4
Raspberry Pi Imager e scelta del primo Sistema Operativo da installare a bordo della nostra scheda.



significativi nel caso passassimo da una scheda SD molto ben fatta ad un disco di pessima qualità. Nella fase successiva andremo quindi a fare il boot da SSD in modo da cercare di sfruttare questo vantaggio. Questa operazione potrà essere ripetuta anche per gli articoli precedenti, nei quali sarà interessante notare la differenza fra prima e dopo tenendo presente che alcuni si basavano proprio su intelligenza artificiale.

CONFIGURAZIONE

Andiamo ora ad installare il sistema operativo all'interno del nostro Raspberry Pi. Sono necessarie quattro cose per compiere questo step:

- scheda microSD card con almeno 16/32GB;
- adattatore S-ATA/USB 3.0;
- SSD a interfaccia S-ATA;
- Raspberry Pi Imager.

Per quest'ultimo il sito di riferimento è quello ufficiale: <https://www.raspberrypi.org/downloads/>; nella prima sezione vedremo Imager, ovvero il tool responsabile di creare un sd card bootabile in grado di far partire il dispositivo.

Le principali operazioni che svolgeremo saranno:

- installazione sistema operativo (OS) a 32bit;
- procedura per il boot da SSD;
- installazione OS a 64bit.

Cercheremo infatti di installare un sistema operativo a 64bit per sfruttare a pieno gli 8GB di RAM messi a disposizione dalla piattaforma.

Apriamo Raspberry Pi Imager, clicchiamo su CHOOSE OS (**Fig. 4**) e selezioniamo la prima voce nella quale è indicato chiaramente 32bit. Selezioniamo la scheda MicroSD da preparare ed infine clicchiamo su WRITE.

Ora inseriamo la scheda nella nostra Raspberry Pi e andiamo ad eseguire il primo boot.

Una volta completate le prime operazioni, andiamo a modificare il firmware al fine di abilitare il boot da USB, eseguendo in ordine i seguenti comandi:

```
sudo apt update
sudo apt full-upgrade
sudo rpi-update
sudo reboot
```

La piattaforma sarà ora riavviata e potremo continuare la procedura. Andiamo quindi ad installare/aggiornare l'ultimo bootloader, precedentemente scaricato, digitando:

```
sudo rpi-eeprom-update -d -a
sudo reboot
```

Riavviamo nuovamente per permettere all'intero sistema di recepire le modifiche e ora accediamo alla configurazione:

```
sudo raspi-config
```

Verrà ora mostrata la schermata proposta nella

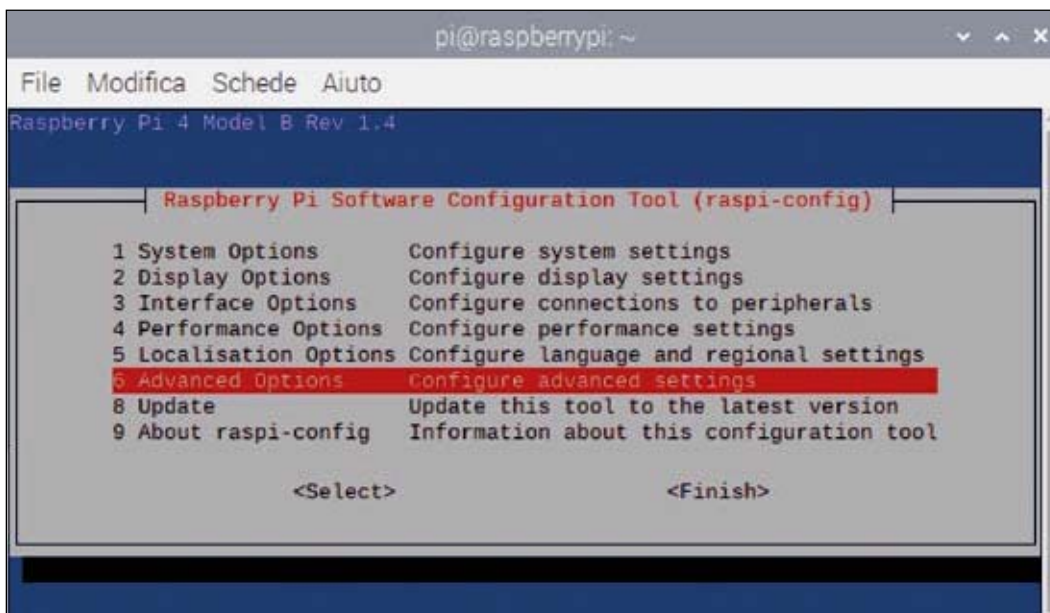


Fig. 5
Schermata di configurazione di sistema della Raspberry Pi.

Fig. 5, nella quale andremo a selezionare **Advanced Options**.

Successivamente ci recheremo su **Bootloader Version**, una volta entrati cliccheremo su **E1 Lastes** e alla domanda che ci verrà posta per il reset risponderemo **No**.

Torneremo nuovamente sulla schermata principale e da lì andremo sempre in **Advanced Options** e questa volta selezioniamo la voce appena sopra rispetto alla precedente, ovvero **Boot Order**.

Qui potranno esserci due o, come nel nostro caso, tre voci e andremo a selezionare la prima o seconda, a seconda del numero precedente di scelte (**Fig. 6**). Ora andiamo pure a cliccare su **Finish** e

riavviamo il sistema rispondendo **Yes** alla domanda posta in uscita.

La nostra piattaforma è pronta ad ospitare un boot da USB ma ancora non siamo pronti, infatti il nostro target era quello di installare un sistema 64bit per cui spegniamo la scheda ed estraiamo la microSD che andremo ad inserire nel PC.

Scarichiamo il file compresso contenente il sistema operativo x64 dalla pagina web

https://downloads.raspberrypi.org/raspios_arm64/images/raspios_arm64-2020-08-24/2020-08-20-raspios-buster-arm64.zip.

Dopo aver completato il download, eseguiamo nuovamente la procedura con Raspberry PI Imager

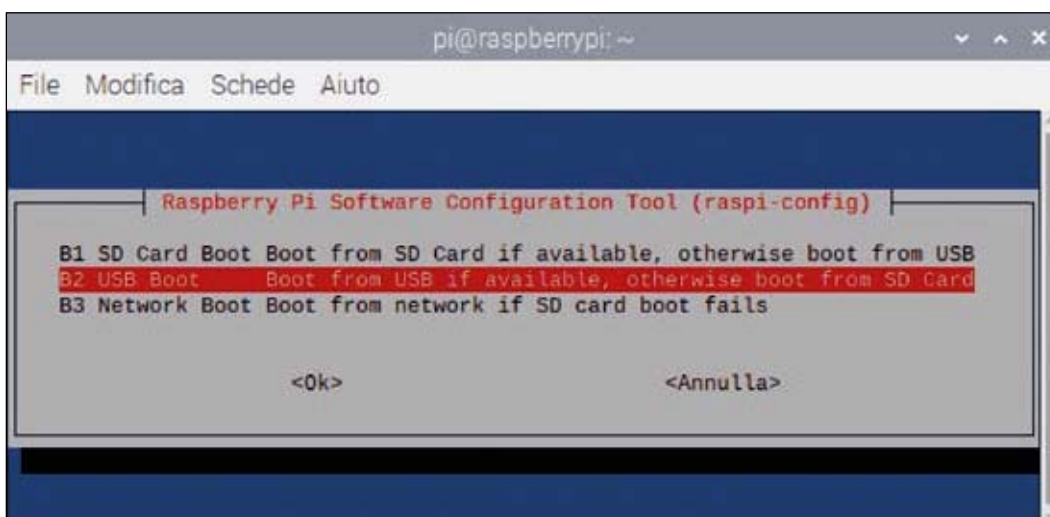
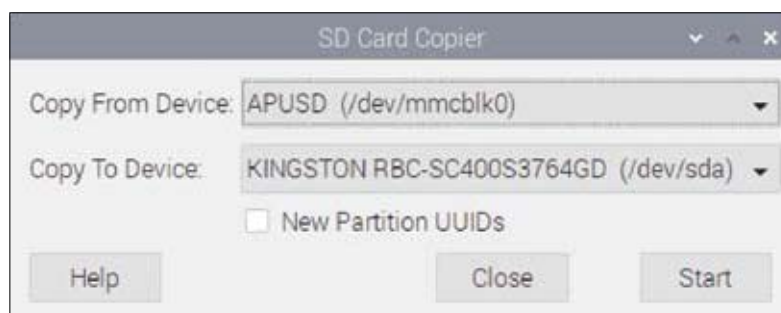


Fig. 6
Schermata di configurazione in cui scegliere la modalità di boot.

Fig. 7
Configurazione per poter effettuare la copia della partizione system su un altro dispositivo.



ma questa volta clicchiamo l'ultima voce in basso nell'elenco, ovvero **Use Custom** e selezioniamo il file scaricato precedentemente. Ora inseriamo la scheda nella nostra Raspberry Pi. Al termine delle operazioni preliminari del primo boot andiamo nel menu in alto a sinistra e apriamo **SD Card Copier** dalla scheda **Accessori**. Selezioniamo ora l'origine nella prima riga, quindi la nostra MicroSD, mentre nella seconda un SSD appositamente collegato ad una delle due USB 3.0 e clicchiamo su **Start (Fig. 7)**. Quando avrà finito apriamo un terminale e digitiamo il seguente comando per spegnere la scheda:

```
sudo poweroff
```

Terminata la procedura di spegnimento, stacciamo l'alimentatore ed estraiamo la microSD dalla Raspberry Pi, lasciando collegato solamente l'hard disk SSD e colleghiamo nuovamente l'alimentatore. Se durante il boot richiede la card microSD

non disperiamo, abbiamo solamente bisogno di un passaggio in più rispetto al normale. Questo può capitare su un Raspberry Pi 4 con 8GB di RAM. In caso ci trovassimo di fronte a questa richiesta, assecondiamo il dispositivo ed inseriamo la microSD. Chiaramente farà il boot da SD e ci riporterà al punto precedente. Ora apriamo un terminale, con il quale abbiamo già preso confidenza nei vari articoli e digitiamo il seguente comando:

```
sudo -E rpi-eeprom-config edit
```

Qui è presente tutta la configurazione di boot, sono poche righe e fra queste a noi interessa principalmente una:

```
BOOT_ORDER=0x1
```

Questa riga infatti dovrà diventare 0x4, quindi andremo a modificare 1 in 4, come anche specificato

Fig. 8
Le varie configurazioni per il bootloader.

Value	Mode	Description
0x1	SD CARD	SD card (or eMMC on Compute Module 4)
0x2	NETWORK	Network boot
0x3	RPIBOOT	RPIBOOT - See usbboot (since 2020-09-03)
0x4	USB-MSD	USB mass storage boot (since 2020-09-03)
0x5	BCM-USB-MSD	USB 2.0 boot from USB Type-C socket or USB Type-A socket on CM4 IO board. (since 2020-12-14)
0xe	STOP	Stop and display error pattern (since 2020-09-03). A power cycle is required to exit this state.
0xf	RESTART	Start again with the first boot order field. (since 2020-09-03)



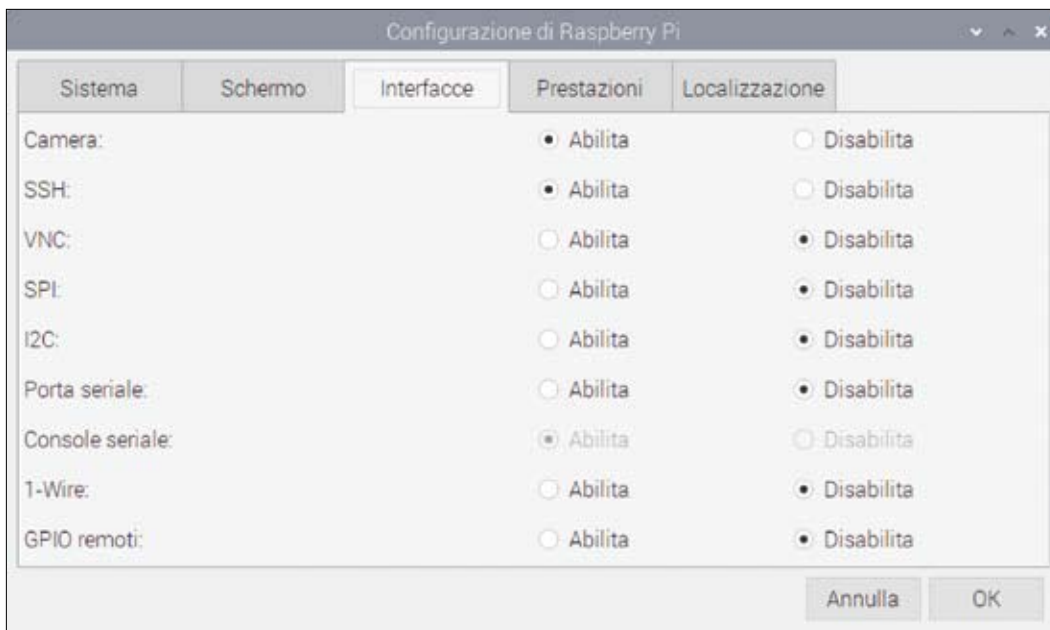


Fig. 9
Abilitiamo le impostazioni di Camera e SSH.

all'interno della referenza ufficiale riportata su www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2711_bootloader_config.md nel quale si trova la stessa tabella proposta nella **Fig. 8**. Per uscire e salvare Ctrl+X e successivamente Y. Poi ripetiamo l'operazione di spegnimento ed estrazione della microSD. Ora si dovrebbe riavviare da SSD e potremmo iniziare ad installare le librerie indispensabili per poter iniziare a programmare.

LE LIBRERIE

Come prima cosa installiamo il solito connettore (socket) per la sessione da remoto con il seguente comando:

```
sudo apt-get install xrdp -y
```

Successivamente abilitiamo alcune interfacce: in primis quella per poter connettersi in SSH, utile anche questa come il precedente tool, per avviare una sessione remota, senza avere necessariamente uno schermo e dei dispositivi di input connessi alla nostra Raspberry Pi. Altra interfaccia utile sarà quella che ci metterà a disposizione il modulo camera. Per abilitarle andiamo nel menu in alto a sinistra e selezioniamo **Preferenze**, poi **Configurazione di Raspberry Pi**.

Spostiamoci quindi nella tab **Interfacce** e qui abilitiamo le prime due (**Fig. 9**).

A differenza delle applicazioni di riconoscimento

d'immagine proposte nei precedenti fascicoli della nostra rivista, l'unico framework indispensabile è OpenCV: si tratta di una libreria ed un insieme di algoritmi scritti in C per lo streaming e la manipolazione image/video real-time per la visione artificiale disponibile ormai per tutte le piattaforme OS. Questa permette la trattazione di immagini in maniera semplice, gestendole come matrici di pixel. Ci permette di effettuare quel processo alla base della computer vision, ovvero l'Image processing, andando a modificare/migliorare le immagini o sequenze di immagini (video).

Ricordiamo anche che, come precedentemente precisato, questa libreria è installabile sia attraverso il comando `sudo apt` messo a disposizione dagli ambienti Linux, sia compilando i sorgenti.

Proprio quest'ultima sarà la nostra scelta poiché dovremmo agire su una versione già nota e compatibile con gli algoritmi che andremo a scrivere. È possibile utilizzare l'immagine preconfigurata *rpi_64bit_rpi facerec.zip* all'interno della quale sono già presenti tutte le librerie e i moduli per poter utilizzare correttamente i programmi, scaricabile dal nostro sito.

In tal caso le credenziali per accedervi sono user: *pi* e password: *12345678*.

Prima di partire installiamo i tool di appoggio utilizzando i seguenti comandi:

```
sudo apt-get update
sudo apt-get upgrade -y
sudo apt-get install build-essential cmake pkg-
```

```

config -y
sudo apt-get install libjpeg-dev libtiff5-dev
libjasper-dev libpng12-dev -y
sudo apt-get install libavcodec-dev libavformat-dev
libswscale-dev libv4l-dev -y
sudo apt-get install libxvidcore-dev libx264-dev -y
sudo apt-get install libgtk2.0-dev -y
sudo apt-get install libatlas-base-dev gfortran -y
sudo apt-get install python2.7-dev python3-dev -y
cd ~

```

Questi ci permetteranno di installare tutti quei componenti di sistema che ci renderanno autonomi nel compilare i sorgenti e compiere l'installazione. Scarichiamo i sorgenti e decompriamoli nel seguente modo:

```

wget -O opencv.zip https://github.com/Itseez/opencv/
archive/3.4.0.zip
unzip opencv.zip
wget -O opencv_contrib.zip https://github.com/Itseez/opencv_contrib/archive/3.4.0.zip
unzip opencv_contrib.zip
wget https://bootstrap.pypa.io/get-pip.py
sudo python get-pip.py

```

Come si può notare, oltre alla libreria standard di OpenCV abbiamo aggiunto anche la *_contrib*, la quale contiene diversi modelli e come dice il nome dei "contributi" dalla community, indispensabili per il riconoscimento del soggetto. Prima di partire con la vera e propria compilazione dei sorgenti, installiamo l'ambiente virtuale per poter lavorare in uno spazio protetto senza andare ad interessare tutto il sistema. Questo ci permetterà in caso di errore di eliminare l'ambiente e ripartire senza perdere nulla di quanto installato fino ad ora e senza dover cancellare e riavviare l'intero sistema:

```

sudo pip install virtualenv virtualenvwrapper
sudo rm -rf ~/.cache/pip
echo -e "\n# virtualenv and virtualenvwrapper" >>
~/.profile
echo "export WORKON_HOME=$HOME/.virtualenvs" >>
~/.profile
echo "source /usr/local/bin/virtualenvwrapper.sh" >>
~/.profile
source ~/.profile
mkvirtualenv cv -p python3
workon cv
pip install numpy

```

Compiliamo ora i sorgenti:

```

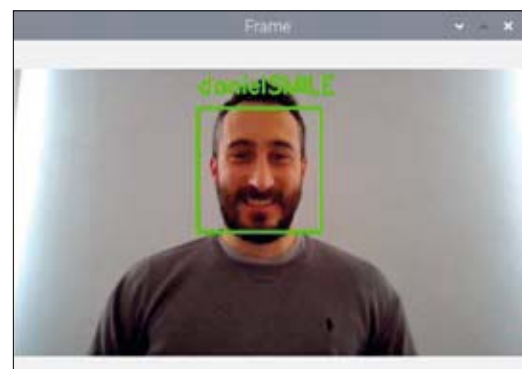
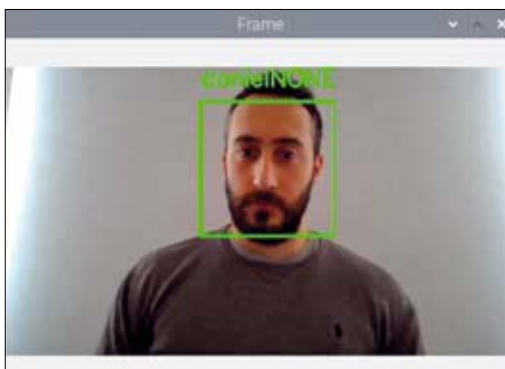
cd ~/opencv-3.4.0/
mkdir build
cd build
cmake -D CMAKE_BUILD_TYPE=RELEASE \
-D CMAKE_INSTALL_PREFIX=/usr/local \
-D INSTALL_PYTHON_EXAMPLES=ON \
-D ENABLE_PRECOMPILED_HEADERS=OFF \
-D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-3.4.0/modules \
-D BUILD_EXAMPLES=ON ..
make

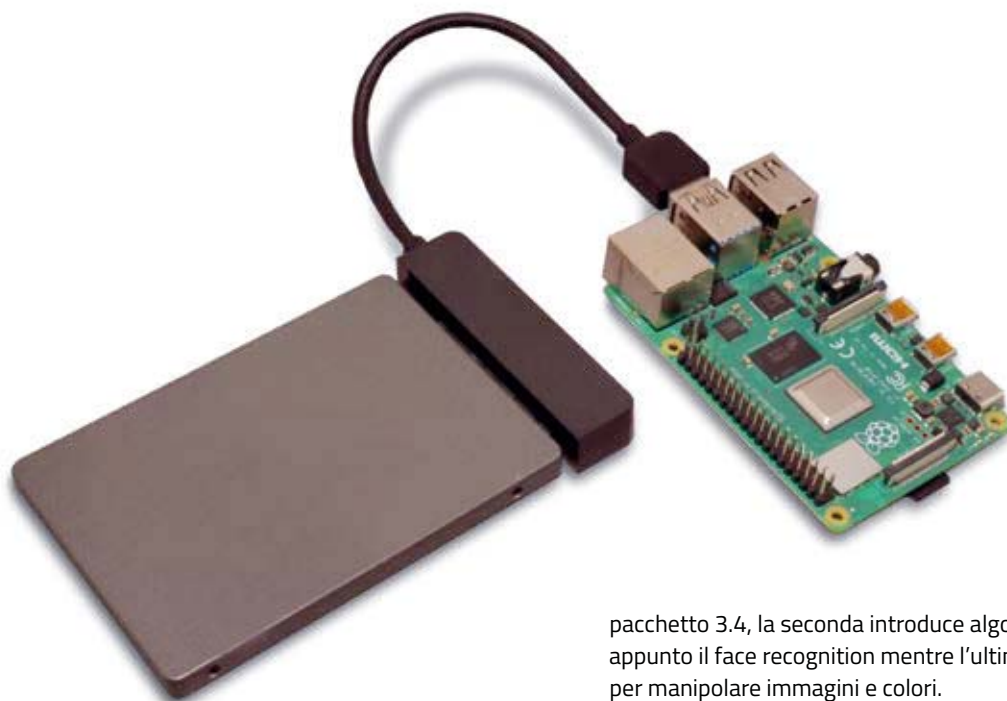
```

Fatto questo, attendiamo. Sicuramente quella di compilare in un sistema così performante con solo un core, ovvero senza specificare come nelle altre volte il *-j4* in caso di 4 core, è una scelta bizzarra e molto discussa. Questo poiché questa compilazione potrà fallire intorno al 80/83% ma nessun problema, compilando in maniera sequenziale siamo sicuri che quanto serve venga compilato ed installato correttamente. Installiamo quindi i sorgenti:

```
sudo make install
```

➔ Collegando uno schermo alla Raspberry Pi avremo il volto sorridente evidenziato dalla scritta SMILE.





←
Raspberry
Pi con il
suo SSD
connesso
tramite
l'adattatore
USB3/S-ATA.

```
sudo ldconfig
cd /usr/local/lib/python3.5/site-packages/
sudo mv cv2.cpython-35m-arm-linux-gnueabi.so cv2.so
cd ~/.virtualenvs/cv/lib/python3.5/site-packages/
ln -s /usr/local/lib/python3.5/site-packages/cv2.so
cv2.so
cd ~
```

A seconda della vostra installazione di python quest'ultima procedura potrebbe dare errore. In tal caso i path potrebbero essere con *python3.7* e solo nel primo caso *dist-package*.

Se vi trovate in questa condizione non preoccupatevi ed ignorate i precedenti comandi perché nell'ambiente virtuale, i loro vuoti, saranno colmati dalle prossime istruzioni.

Per poter accedere all'ambiente virtuale anche le prossime volte, anche dopo vari reboot, sarà sufficiente digitare:

```
source ~/.profile
workon cv
```

Mancano infine tre librerie da installare sempre all'interno dell'ambiente virtuale, ovvero:

```
pip install opencv-contrib-python
pip install face-recognition
pip install imutils
```

La prima aggiunge alcuni modelli mancanti nel

pacchetto 3.4, la seconda introduce algoritmi per appunto il face recognition mentre l'ultima serve per manipolare immagini e colori. Scarichiamo infine i sorgenti da GitHub attraverso:

```
cd ~
git clone https://github.com/danisk89/RaspberryPIFaceRec.git
cd RaspberryPIFaceRec
```

A questo punto abbiamo tutto il necessario per poter cominciare a scrivere i nostri algoritmi.

ENCODING

Se lanciassimo ora gli algoritmi di face recognition sicuramente verrà rilevato il nostro volto ma non riconosciuto poiché non abbiamo inserito alcuna referenza di noi.

Per permettere alla macchina di riconoscerci dobbiamo inserire, in modo oculato, alcune foto all'interno della cartella **dataset**.

Infatti, se entriamo nella cartella notiamo altre folder, ognuna delle quali con un nome e all'interno di ognuna abbiamo dei file immagine con delle label molto semplici (00000, 00001, ecc.); quello che dobbiamo fare è inserire una cartella con il nostro nome, volendo anche nome_cognome, e al suo interno inserire delle foto del nostro volto.

La label dell'immagine sarà composta da 5 numeri, progressivi, come nelle altre cartelle già presenti. Una volta completata questa procedura, torniamo al terminale aperto in precedenza, assicuriamoci di essere ancora all'interno dell'ambiente virtuale **cv** e lanciamo il seguente comando:

```
python encode_faces.py --dataset dataset --encodings encodings.pickle --detection-method hog
```

Ora tutte le nostre foto insieme a quelle già

presenti sono state codificate ed inserite all'interno del file **encodings.pickle**, responsabile, nella seconda parte, di associare un nome ad ogni volto scovato.

L'ALGORITMO DI MACHINE LEARNING

Come nel punto precedente se vogliamo lanciare il programma assicuriamoci sempre di essere all'interno dell'ambiente virtuale creato durante le prime fasi e digitiamo:

```
python pi_face_recognition.py --cascade haarcascade_
frontalface_default.xml --encodings encodings.pickle
```

Nell'header di ogni file c'è spiegato come poterli usare, ma esaminiamo con cura il vero algoritmo ed apriamo il file **pi_face_recognition.py** ed osserviamo la struttura:

```
from imutils.video import VideoStream
from imutils.video import FPS
import face_recognition
import argparse
import imutils
import pickle
import time
import cv2
```

Dopo aver importato i vari package andiamo a prendere in ingresso i 2 argomenti fondamentali, partendo dal fondo: il file di encoding, ottenuto dall'elaborazione precedente e il file responsabile della scoperta e della classificazione dei volti. A differenza delle precedenti pubblicazioni, in questa utilizzeremo quindi il metodo di Haar, ovvero un metodo di rilevamento di oggetti efficace proposto da Paul Viola e Michael Jones nel loro articolo "Rilevamento rapido di oggetti utilizzando una cascata potenziata di funzionalità semplici" nel 2001. Si tratta di un approccio basato sull'apprendimento automatico in cui un la funzione a cascata è addestrata da molte immagini positive e negative. Nel caso in cui al posto del modulo camera vogliamo ricevere in ingresso un'immagine o un video qui possiamo estendere la sezione con le righe di codice che possiamo trovare negli altri 2 file presenti, ovvero **pi_face_recognition_image.py** e **pi_face_recognition_video.py**.

```
ap = argparse.ArgumentParser()
ap.add_argument("-c", "--cascade", required=True,
help = "path to where the face cascade resides")
ap.add_argument("-e", "--encodings", required=True,
```

```
help="path to serialized db of facial encodings")
args = vars(ap.parse_args())
```

Carichiamo il tutto negli oggetti di Python:

```
print("[INFO] loading encodings + face detector...")
data = pickle.loads(open(args["encodings"], "rb").
read())
detector = cv2.CascadeClassifier(args["cascade"])
smile_cascade = cv2.CascadeClassifier('haarcascade_
smile.xml')
```

E inizializziamo il modulo camera:

```
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
time.sleep(2.0)
fps = FPS().start()
while True:
frame = vs.read()
frame = imutils.resize(frame, width=500)
```

Come accennato in precedenza se avessimo voluto importare un video, questo estratto era l'unico a cambiare, assumendo questa forma:

```
vs = cv2.VideoCapture(args["video"])
time.sleep(2.0)
fps = FPS().start()
while(vs.isOpened()):
ret, frame = vs.read()
frame = imutils.resize(frame, width=500)
```

Dopo una serie di trasformazioni del frame per poterlo "maneggiare" meglio, andiamo ad inserirlo all'interno della funzione che estrapola i volti:

```
encodings = face_recognition.face_encodings(rgb,
boxes)
names = []
```

Per ogni volto riconosciuto andiamo ad affinare la ricerca cercando all'interno dell'encoder se questo era stato classificato o è uno sconosciuto.

```
for encoding in encodings:
matches = face_recognition.compare_faces(data["encod
ings"],encoding)
name = "Unknown"
if True in matches:
matchedIdxs = [i for (i, b) in enumerate(matches)
if b]
counts = {}
```



```

for i in matchedIdxs:
    name = data["names"][i]
    counts[name] = counts.get(name, 0) + 1
    name = max(counts, key=counts.get)
    names.append(name)

```

Infine per ogni associazione volto-nome andiamo a costruire un riquadro di colore verde per darne riscontro anche a livello di UI:

```

for ((top, right, bottom, left), name) in zip(boxes,
names):
    cv2.rectangle(frame, (left, top), (right, bottom),
    (0, 255, 0), 2)
    y = top - 15 if top - 15 > 15 else top + 15

```

In più capiamo attraverso l'altro classificatore se il soggetto sta sorridendo oppure ha un'espressione neutra che definiremo NONE, come si intuisce dal codice seguente:

```

img_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
faceimg_gray = img_gray[top:bottom, left:right]
smiles = smile_cascade.detectMultiScale(faceimg_gray,
scaleFactor=1.7, minNeighbors=3, minSize=(15, 15))
if len(smiles) > 0:
    cv2.putText(frame, name+"SMILE", (left, y), cv2.
FONT_HERSHEY_SIMPLEX, 0.75, (0, 255, 0), 2)
else:
    cv2.putText(frame, name+"NONE", (left, y), cv2.FONT_
HERSHEY_SIMPLEX, 0.75, (0, 255, 0), 2)
E infine mostriamo il tutto a video nella finestra:
cv2.imshow("Frame", frame)
key = cv2.waitKey(1) & 0xFF
if key == ord("q"):
    break
fps.update()
fps.stop()
print("[INFO] elapsed time: {:.2f}".format(fps.
elapsed()))
print("[INFO] approx. FPS: {:.2f}".format(fps.
fps()))
cv2.destroyAllWindows()
vs.stop()

```

Questo è un semplice algoritmo per discriminare i volti e l'espressione all'interno di una ripresa live o registrata ma nulla ci vieta di ampliare questo esempio ed ottenere qualcosa di più complesso. In particolare, come accennato anche durante l'introduzione, il riconoscere un'espressione e se questa cambia spesso. Ciò potrà essere usato come modo, seppur

rudimentale, valido per riconoscere se il soggetto inquadrato è una riproduzione, una stampa oppure una persona reale.

In realtà per discriminare ciò esistono algoritmi molto più potenti e basati su keras e TensorFlow, tool già utilizzati nei progetti dei fascicoli 250 e 251, ma tali algoritmi tendono a rallentare il processo di acquisizione e quindi si opta per un processo più leggero come questo. Inoltre si potrebbe infatti abilitare uno dei pin di GPIO della nostra Raspberry Pi solamente quando viene riconosciuto il soggetto e questo cambia espressione due volte, passando quindi da una neutra (NONE), ad un sorriso (SMILE) e tornando infine ad una neutra (NONE): un po' come una passphrase o qualcosa di simile ma con l'espressione.

Per far ciò sarà sufficiente inserire una classe che tenga conto dell'espressione riscontrata e delle sue variazioni per ogni soggetto inquadrato.

CONCLUSIONI

Come avrete capito, il progetto qui descritto è la base per numerose applicazioni, realizzabili sia a fini di sicurezza (per esempio l'apertura di una porta o accesso ad una zona controllata subordinata al riconoscimento d'immagine) sia a fini statistici, come ad esempio l'identificazione di eventi che fanno sorridere o provare altre emozioni. □



Cosa occorre?

La scheda Raspberry Pi 4 è disponibile in diverse versioni da Futura Elettronica. La versione con 8GB di memoria (cod. RPI4-8GB) è in vendita a Euro 99,00, la versione con 4GB di memoria (cod. RPI4-4GB) è disponibile a Euro 69,00, la stessa versione con 2GB di memoria (cod. RPI4-2GB) è in vendita a Euro 59,00. La telecamera da 5Mpx utilizzata in questo progetto (cod. YG016) è in vendita a Euro 14,00. L'adattatore SATA-USB 3.0 per Raspberry Pi (cod. CAVOSATAUSB3) costa Euro 7,90. Per le board Raspberry Pi 4 sono disponibili anche diversi starter kit già pronti all'utilizzo disponibili alla pagina bit.ly/RPIStarterKit. I prezzi si intendono IVA compresa.

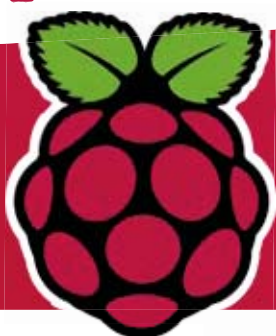
Il materiale va richiesto a:

Futura Elettronica, Via Adige 11, 21013 Gallarate (VA)
Tel: 0331-799775 - <http://www.futurashop.it>



Raspberry Pi 4

**UTILIZZALA
SUBITO CON
I NUOVI KIT**



Starter kit completo con Raspberry Pi 4 da 2 GB

Prezzi IVA inclusa.



€ 89,00

Cod. RPI4FULLG2

- » Board Raspberry Pi 4 modello B con 2GB di RAM.
- » Alimentatore 5 VDC / 3 A uscita type C e interruttore.
- » Contenitore plastico per Raspberry Pi 4 tipo B, colore bianco e rosso.
- » Cavo da HDMI (tipo A) a micro HDMI (tipo D), lunghezza 1,5 metri.
- » Micro SD card da 32GB con sistema operativo NOOBS precaricato.



Disponibile anche con Raspberry Pi 4 modello B da 4GB di RAM **COD. RPI4FULLG4 € 105,00**
e con Raspberry Pi 4 modello B da 8GB di RAM **COD. RPI4FULLG8 € 129,00**

Starter kit con Raspberry Pi 4 da 2 GB



€ 75,00

Cod. RPI4STARTERG2

- » Board Raspberry Pi 4 modello B con 2GB di RAM.
- » Alimentatore 5 VDC / 3 A uscita type C e interruttore.
- » Contenitore plastico per Raspberry Pi 4 tipo B, colore bianco e rosso.



Disponibile anche con Raspberry Pi 4 modello B da 4GB di RAM **COD. RPI4STARTERG4 € 89,00**
e con Raspberry Pi 4 modello B da 8GB di RAM **COD. RPI4STARTERG8 € 115,00**



Collegiamo in modo sicuro la nostra board Azure Sphere al dispositivo creato in Azure IoT Central. Terza puntata.

CONOSCERE

E USARE

Azure Sphere

3

di PIER CALDERAN

Come preannunciato, in questa puntata vedremo come connettere il nostro dispositivo Azure Sphere "reale" al dispositivo "virtuale" creato nell'applicazione Azure IoT Central. Se non l'avete già fatto, per prima cosa dovete creare un dispositivo all'interno dell'applicazione IoT Central. Se l'applicazione gratuita per 7 giorni è scaduta e non avete fatto l'upgrade a una sottoscrizione a pagamento, dovete rifare tutta la procedura spiegata la volta scorsa.

PROVISIONING DEL DISPOSITIVO

Una volta creato un dispositivo nell'applicazione IoT Central, apparirà la finestra "Tutti i dispositivi" con il nome del nuovo dispositivo appena creato (Fig. 1), il modello selezionato, l'ID assegnato e lo stato "Simulato No".

Tenete presente che l'ID del dispositivo è provvisorio e servirà, come vedremo a breve, a creare la prima connessione "reale". Tant'è che lo stato del dispositivo è "Registrato" e non riporta "Provisioning eseguito". Questo perché non è stato effettuato ancora il provisioning del dispositivo reale. Prima di proseguire, accertatevi di avere collegato il dispositivo a una porta USB e di avere eseguito il claim del dispositivo.

CREARE UN GRUPPO DI REGISTRAZIONI X.509

Per la connessione a IoT Central è assolutamente necessario che tutti i dispositivi Azure Sphere siano autenticati con certificazione X.509. X.509 è uno standard proposto da ITU-T e usato per definire il formato dei certificati a chiave



Fig. 1 - La pagina Tutti i dispositivi con il nuovo dispositivo “registrato”.

pubblica (PKC) e delle autorità di certificazione (CA). I certificati vengono utilizzati per la validazione dell'identità e la trasmissione di dati criptati che solo il possessore di uno specifico certificato è in grado di decifrare e leggere.

Questi certificati vengono rilasciati dalle Certification Authority (CA), che sono un soggetto terzo fidato (trusted). Queste autorità di certificazione assicurano la corrispondenza di una chiave pubblica a una determinata identità.

Uno degli usi più diffusi di X.509 è nell'ambito internet con il certificato SSL/TLS che viene usato nell'omonimo protocollo per criptare le comunicazioni tra un sito web e il browser.

Attenzione! La procedura di certificazione è molto complessa, pertanto seguite passo passo la guida seguente per non perdervi nei vari passaggi. Ma non preoccupatevi, questa procedura va fatta una sola volta.

1. Dal menu della dashboard a sinistra, aprite “Amministrazione > Connessione del dispositivo”.
2. Apparirà la pagina “Connessione del dispositivo” (Fig. 2) con le indicazioni di default.
3. Fate clic su “+ Crea gruppo di registrazioni” e compilare i campi in questo modo:
Nome: un nome qualsiasi, per esempio, “Futura”.

Connetti automaticamente i dispositivi in questo gruppo: selezionare “Attivata”. Questo consentirà ai nuovi dispositivi di connettersi automaticamente usando il tipo di attestazione (claim) scelto.

Tipo di gruppo: se non è selezionato, selezionate “Dispositivi IoT”.

Tipo di attestazione: scegliete “Certificati X.509” dal menu a discesa, ovvero la modalità predefinita di autenticazione e di connessione dei dispositivi a IoT Central.

4. Fate clic su “Salva” e uscite selezionando di nuovo “Connessione del dispositivo” dal menu.
5. Ora il certificato X.509 è nella lista “Gruppi di registrazioni” (Fig. 3).
6. Selezionate il gruppo creato con il certificato X.509.
7. Si aprirà una pagina simile alla Fig. 4 con il nome scelto per il gruppo (“Futura”).
8. Fate clic su “Gestisci primario”.
9. Si aprirà una finestra “Certificato primario” con tutti i campi vuoti (Fig. 5).
10. A questo punto, aprite il command prompt Azure Sphere in modalità amministratore e portarsi preferibilmente nella directory C:\ (consigliato).
11. Digitate il comando seguente:
`azsphere ca download --output C:\certificate.cer`



Fig. 2 - La pagina Amministrazione > Connessione del dispositivo.

Connessione del dispositivo

Per registrare e connettere i dispositivi viene usato il servizio Device Provisioning in

Ambito ID ⓘ

One00268A21

Approva automaticamente i nuovi dispositivi ⓘ

☒ On

Gruppi di registrazioni

+ Crea gruppo di registrazioni

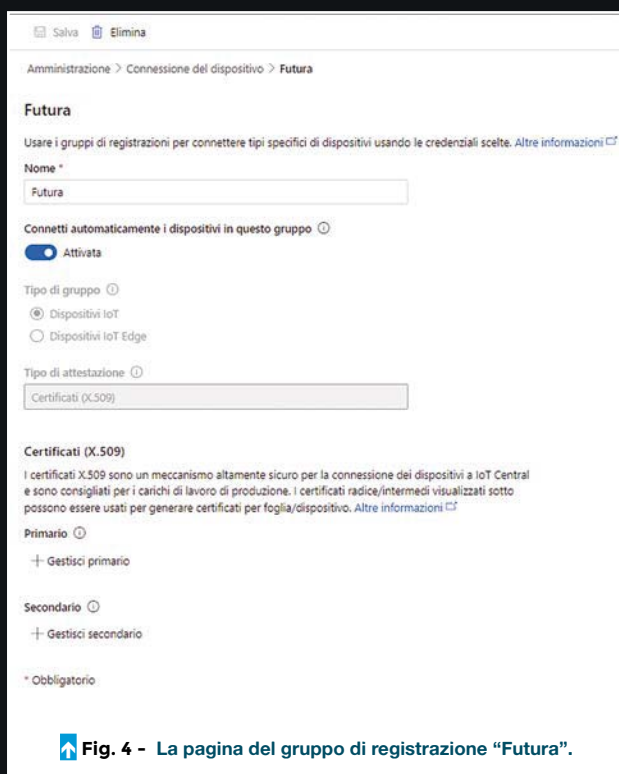
Nome	Tipo di attestazione
SAS-IoT-Devices	Firma di accesso condiviso
SAS-IoT-Edge-Devices	Firma di accesso condiviso
Futura	Certificati (X.509)

 Fig. 3 - La pagina Crea nuovo gruppo di registrazioni.

12. Verrà salvato il certificato CAcertificate.cer in C:\ (Fig. 6).

13. Tornate alla finestra "Certificato primario".

14. Fate clic sull'icona "Apri cartella" e cercate il file CAcertificate.cer in C:\.



Salva Elimina

Amministrazione > Connessione del dispositivo > Futura

Futura

Usare i gruppi di registrazioni per connettere tipi specifici di dispositivi usando le credenziali scelte. Altre informazioni ⓘ

Nome *

Futura

Connetti automaticamente i dispositivi in questo gruppo ⓘ

☒ Attivata

Tipo di gruppo ⓘ

☒ Dispositivi IoT

☐ Dispositivi IoT Edge

Tipo di attestazione ⓘ

Certificati (X.509)

Certificati (X.509)

I certificati X.509 sono un meccanismo altamente sicuro per la connessione dei dispositivi a IoT Central e sono consigliati per i carichi di lavoro di produzione. I certificati radice/intermedi visualizzati sotto possono essere usati per generare certificati per foglia/dispositivo. Altre informazioni ⓘ

Primario ⓘ

+ Gestisci primario

Secondario ⓘ

+ Gestisci secondario

* Obbligatorio

 Fig. 4 - La pagina del gruppo di registrazione "Futura".

Certificato primario

Primario ⓘ

Oggetto

Identificazione digitale

Verifica certificato

Verificheremo che la persona che carica un certificato possieda la chiave privata del certificato. Per completare la fase di verifica, è prima necessario generare un codice di verifica (tenere presente che questo codice sostituirà qualsiasi codice di verifica esistente creato in precedenza). Creare quindi un certificato di verifica X.509 con il nuovo codice. Al termine, caricare il certificato di verifica firmato Altre informazioni ⓘ

Codice di verifica ⓘ

Genera codice di verifica

Verifica

Chiudi

 Fig. 5 - La finestra Certificato primario.

- Verrà caricato il certificato e apparirà qualcosa di simile alla Fig. 7 con il codice caricato dal disco. Notare che quando il certificato primario viene caricato, vengono compilati automaticamente i campi "Oggetto" e "Identificazione digitale" e anche il messaggio "Richiede verifica".
- Nel campo "Codice di verifica" (vuoto) fate clic sul tasto "Genera codice di verifica".
- Apparirà una finestra con il codice di verifica e la nota: "assicurarsi di copiare questo codice di verifica, in quanto non verrà mostrato di nuovo".
- Copiate il codice di verifica con l'icona "Copia".
- Tornate al prompt dei comandi e digitate il seguente comando SENZA premere invio:
`azsphere ca download-proof --output validation.cer --verificationcode <codice di verifica>`
- Incollate il codice di verifica al posto di <codice di verifica>.
- Apparirà qualcosa di simile a questo:
`azsphere ca download-proof --output validation.cer --verificationcode 2A65A35F8616674665D98C8E0F9DCB411CB1AA12E214B16.`
- Premete invio e verrà salvato il file `validation.cer` in C:\ (Fig. 8).
- Tornate alla finestra "Certificato primario" e fate clic sul tasto "Verifica".
- Si aprirà una finestra di dialogo per caricare il file di verifica `validation.cer` appena salvato in C:\. Il file verrà verificato automaticamente e apparirà il messaggio "Verificata" sotto "Certificato primario".
- Chiudete la finestra con il tasto "Chiudi".
- Salvate con l'icona "Salva" e uscite.
- Il dispositivo Azure Sphere possiede ora un certificato X.509 e può connettersi a IoT Central in modo sicuro.

```
Amministratore: Azure Sphere Classic Developer Command Prompt (Deprecated)

C:\>azsphere ca download --output C:\CAcertificate.cer
Saving the requested CA certificate to 'C:\CAcertificate.cer'.
Saved the requested CA certificate to 'C:\CAcertificate.cer'.

C:\>
```

 Fig. 6 - Il prompt dei comandi con il certificato salvato.

PREPARARE L'APPLICAZIONE PER IOT CENTRAL

Ora avete tutto il necessario per la connessione sicura del dispositivo a IoT Central. È giunto il momento di vedere il dispositivo in funzione, ma prima sono necessarie alcune piccole modifiche sui file di esempio.

Per potervi connettere al dispositivo dell'applicazione IoT Central, bisogna ottenere gli indirizzi delle connessioni consentite da inserire nel file `app_manifest.json` di tutti gli esempi.

Se non lo avete già fatto, scaricate tutti gli esempi dal GitHub seguente: https://github.com/piercalderan/futura_MT3620.

Le informazioni delle connessioni consentite si ottengono avviando il tool `ShowIoTCentralConfig`. Potete trovare il tool `ShowIoTCentralConfig` nella cartella degli esempi `Futura_MT3620/Tools`. Riportiamo la procedura passo passo da fare una tantum e che sarà valida per tutti gli esempi successivi.

Tenete presente che le informazioni che otterremo verranno usate solo e soltanto dal dispositivo collegato e che abbiamo certificato.

1. Aprite il prompt dei comandi Azure Sphere in modalità amministratore.
2. Andate nella directory `Futura_MT3620/Tools`: `cd <...>/Futura_MT3620/Tools`
3. Digitate il comando seguente: `ShowIoTCentralConfig`
4. Apparirà: *Are you using a Work/School account to sign into your IoT Central Application (Y/N) ?*
Digitare "n" SENZA premere invio.
5. Apparirà: *Enter the 'ID scope' from the IoT Central App 'Administration | Device Connection' >*
6. Andate nell'applicazione IoT Central alla pagina "Amministrazione > Connessione del dispositivo".
7. Copiate con l'icona "copia" l'Ambito ID (un formato simile a "Onexxxx") (Fig. 9).
8. Lasciate l'opzione "Approva automaticamente i nuovi dispositivi".
9. Tornate al comando prompt.
10. Incollate l'Ambito ID e premete invio.
11. Apparirà: *Enter the Primary Saas Key from the IoT Central App 'Administration | Device Connection' >*.
12. Tornate alla pagina "Amministrazione > Connessione dispositivo" e fate clic sul gruppo di registrazione "SAS-IoT-Devices".
13. Si aprirà una finestra simile alla Fig. 10.
14. Copiate con l'icona "copia" la chiave primaria.
15. Tornate al command prompt, incollate la chiave primaria Saas e premete invio.
16. Apparirà: *Enter any real Device Id from the IoT Central App >*.
17. Andate alla pagina "Dispositivi" e copiate l'ID del dispositivo visibile nella colonna "ID dispositivo". Potete copiare l'ID in modo più comodo, aprendo il dispositivo e facendo clic su "Connetti" della barra in alto a destra.
Si aprirà una finestra con le connessioni del dispositivo (Fig. 11) in cui è visibile l'ID provvisorio del dispositivo. Premete invio.

 Fig. 7 - La finestra Certificato primario con il codice caricato dal disco

```

C:\>azsphere ca download-proof --output validation.cer -verificationcode FCD477F9C3B77F5852A1D2
8E633D06D21BC6F10A1FDE9CEA
Saving the requested proof of possession certificate to 'C:\validation.cer'.
Saved the requested proof of possession certificate to 'C:\validation.cer'.

C:\>

```

Fig. 8 - Il prompt dei comandi con il certificato validation salvato.

18. Tornate al command prompt, incollate l'ID del dispositivo e premete invio.

19. Apparirà l'output di *ShowIoTCentralConfig* (Fig. 12).

Connessione del dispositivo

Per registrare e connettere i dispositivi viene usato il servizio Device Provisioning in hub IoT di Azure. Altre informazioni [↗](#)

Ambito ID

One00268A21

Approva automaticamente i nuovi dispositivi

☒ On

Gruppi di registrazioni

+ Crea gruppo di registrazioni

Nome	Tipo di attestazione	Data creazione	Tipo di gruppo
SAS-IoT-Devices	Firma di accesso condiviso	28/3/2021	Dispositivi IoT
SAS-IoT-Edge-Devices	Firma di accesso condiviso	28/3/2021	Dispositivi IoT Edge
Futura	Certificati (X.509)	28/3/2021	Dispositivi IoT

Fig. 9 - La pagina "Amministrazione > Connessione del dispositivo".

A questo punto avete tutte le informazioni per poter modificare il file *app_manifest.json* del nostro esempio per poter comunicare con l'applicazione di IoT Central. Per maggior chiarezza riportiamo le parti importanti da copiare dalla finestra di output nel file *app_manifest.json*:

"CmdArgs": ["One001C85E4"], ovvero l'identificativo che abbiamo copiato dall'ambito ID.

"AllowedConnections": ["global.azure-devices-provisioning.net",

"iotc-f18c6479-0fd2-4197-80f0-337ba9198b77.azure-devices.net"], ovvero le connessioni consentite per il provisioning del dispositivo all'interno dell'applicazione. Manca ancora una cosa. Se non lo avete già fatto nelle fasi preliminari, eseguite il comando seguente dal prompt dei comandi per trovare il tenant da inserire nel file *app_manifest.json*:

azsphere tenant show-selected

SAS-IoT-Devices

Usare i gruppi di registrazioni per connettere tipi specifici di dispositivi usando le credenziali scelte. Altre informazioni [↗](#)

Nome *

SAS-IoT-Devices

Connetti automaticamente i dispositivi in questo gruppo

☒ Attivata

Tipo di gruppo

☒ Dispositivi IoT

☐ Dispositivi IoT Edge

Tipo di attestazione

Firma di accesso condiviso

Firma di accesso condiviso

I dispositivi usano token di sicurezza a firma di accesso condiviso per connettersi a IoT Central. Usare le chiavi di firma di accesso condiviso a livello di gruppo visualizzate sotto per generare le chiavi per i singoli dispositivi. Altre informazioni [↗](#)

Chiave primaria

KLDK7n89UZrGZQTGtb+1hZ/eJb/UY+PyMioD9/E3+iiU2LNI...

Chiave secondaria

CqYnDzVs7xyIpeaNs2cPZEZ9I3iVEGpIOM5EG5GccmYMuZa...

* Obbligatorio

Fig. 10 - La pagina "Amministrazione > SAS-IoT-Devices".

Connessione del dispositivo

Ambito ID

One00268A21

ID dispositivo

1nvrmiirsme

Scegliere il tipo di connessione per questo dispositivo. È possibile modificare questo valore in un secondo momento, in base alle esigenze.

Tipo di autenticazione

Firma di accesso condiviso

Le firme di accesso condiviso usano chiavi e token di sicurezza per connettersi a IoT Central. Usare le chiavi di firma di accesso condiviso dal gruppo di registrazioni predefinito mostrato sotto per registrare il dispositivo. Altre informazioni [↗](#)

Chiave primaria

aovK2SMcqlCvIr4xK/hxSVkcYeWQjboVA7GWaiSFE=

Chiave secondaria

MxvrNK2XtbLJXE0tOL7dr5Fw8YumXjRcKkTh3Mv1A=

Chiudi

Fig. 11 - La finestra "Dispositivi > Connessione del dispositivo".


```
Amministratore: Azure Sphere Classic Developer Command Prompt (Deprecated)

C:\>cd C:\Users\User\Downloads\futura_MT3620\Tools

C:\Users\User\Downloads\futura_MT3620\Tools>ShowIoTCentralConfig
Tool to show Azure IoT Central configuration for Azure Sphere applications

Are you using a Work/School account to sign into your IoT Central Application (Y/N) ?N
Enter the Scope Id from the IoT Central App 'Administration | Device Connection' >0ne00268A21
Enter the Primary SaaS Key from the IoT Central App 'Administration | Device Connection' >KLDK7
nB9UZrdZQTgtlb+1hZ/eJb/UY+PyMioDaJE3+iiUh2LNInYopoG5euzqmPdIHJikqEo/qKUgrOwB0IWHw==
Enter any real Device Id from the IoT Central App >1nvrmiirsme

Find and modify the following lines in your app_manifest.json:
"CmdArgs": [ "0ne00268A21" ],
"AllowedConnections": [ "global.azure-devices-provisioning.net", "iotc-a5ef1f75-7ff3-42ab-becf-
5fe83720377c.azure-devices.net" ],
"DeviceAuthentication": "--- YOUR AZURE SPHERE TENANT ID--- ",

Obtain your Azure Sphere Tenant Id by opening an Azure Sphere Developer Command Prompt and typi
ng the following command:
'Azsphere tenant show-selected'
```

Fig. 12 - La finestra di output del tool ShowIoTCentralConfig.

Copiate il "tenant ID" nella riga "DeviceAuthentication", ovvero l'identificativo unico e non più modificabile per quel dispositivo. Per esempio: "DeviceAuthentication": "df7ba4db-4a32-4bfd-b401-59f2846cc320".

IL FILE APP_MANIFEST.JSON

Come spiegato nella scorsa puntata, potete caricare un esempio dalla cartella Futura_samples, per esempio, Futura_MT3620_DHT22_IoT_Central. Aprite con un clic dal pannello Esplora soluzioni il file *app_manifest.json* e modificalo con tutte le informazioni raccolte. Qui di seguito, riportiamo un esempio di come dovrebbe apparire il file *app_manifest.json* dopo tutte le modifiche.

```
{ "SchemaVersion": 1,
  "Name": "Futura_MT3620_DHT22_IoT_Central",
  "ComponentId": "819255ff-8640-41fd-aea7-85d34c491d5",
```

```
"EntryPoint": "/bin/app",
"CmdArgs": [ "0ne00268A21" ], // Ambito ID
"Capabilities": {
  "AllowedConnections": [
    "global.azure-devices-provisioning.net",
    "iotc-a5ef1f75-7ff3-42ab-becf-5fe83720377c.
azure-devices.net" //connessioni consentite
  ],
  "Gpio": [ "$SAMPLE_BUTTON_3", "$SENS_DHT" ],
  "DeviceAuthentication": "df7ba4db-4a32-4bfd-
b401-59f2846cc320" // tenant
},
"ApplicationType": "Default"
```

Notare che l'esempio riporta anche l'abilitazione per la porta GPIO per quanto riguarda l'uso del pulsante 3 e del sensore DHT della scheda:

```
"Gpio": [ "$SAMPLE_BUTTON_3", "$SENS_DHT" ],
```

Tutti i dispositivi

+ Nuovo ➡ Esegui la migrazione ✖ Elimina ⏸ Approva ⏸ Blocco ➡ Esporta ➡ Importa ⏸ Sblocco ➡ Collega al gateway

Nome del dispositivo	ID dispositivo	Simulato	Stato del dispositivo	Modello di dispositivo
Futura MT3620 Device 1	1nvrmiirsme	No	Provisioning eseguito	Futura MT3620

Fig. 13 - La pagina Tutti i dispositivi con il dispositivo che riporta il messaggio "Provisioning eseguito".



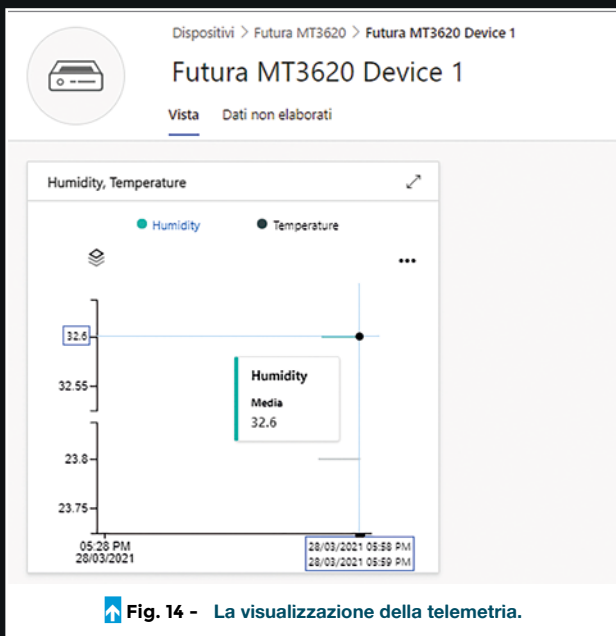


Fig. 14 - La visualizzazione della telemetria.

Attenzione! Quando verrà avviata l'applicazione da Visual Studio, verrà creato un nuovo dispositivo nella pagina "Tutti i dispositivi" con l'ID del dispositivo reale della scheda e apparirà ora "Provisioning eseguito" e non più "Registrato" (Fig. 13).

Questo a dimostrazione del fatto che il nostro dispositivo si è connesso con successo all'applicazione IoT Central e al dispositivo creato in precedenza.

COMPILAZIONE E DISTRIBUZIONE

A questo punto, avviate la compilazione e la distribuzione nel dispositivo per vedere nella finestra di Output un risultato simile a questo:

```
Remote debugging from host 192.168.35.1, port 61070
IoT Central DHT22 application starting.
Setting Azure Scope ID 0ne00268A21
Opening SAMPLE_BUTTON_3 as input
Opening DHT22 as input
[DHT] Humidity = 0.0 % Temperature = 0.0 *C (0.0 *F)
[Azure IoT] Using HSM cert at /run/daa/df7ba4db-4a32-4bfd-b401-59f2846cc320
IoTHubDeviceClient_LL_CreateWithAzureSphereDeviceAuthProvi-
sioning returned 'AZURE_SPHERE_PROV_RESULT_OK'.
IoT Central Authenticated: IOTHUB_CLIENT_CONNECTION_OK
[DHT] Humidity = 32.6 % Temperature = 23.8 *C (74.8 *F)
Sending IoT Central Message: { "Temperature": "23.80" }
INFO: IoT Central accepted the message for delivery
Sending IoT Central Message: { "Humidity": "32.60" }
```

Come potete notare nella finestra di Output appare la connessione certificata con l'ID del tenant e la risposta dell'avvenuto provisioning con il messaggio 'AZURE_SPHERE_PROV_RESULT_OK'.

Anche l'autenticazione del dispositivo con IoT Central è avvenuta con successo tramite il messaggio Authenticated: IOTHUB_CLIENT_CONNECTION_OK.

Dopodiché, premete il pulsante 3 della scheda per inviare i dati di telemetria riguardanti temperatura e umidità del sensore DHT22. Tenete presente che il sensore DHT22 accetta le richieste a intervalli di 2 secondi o più, altrimenti si verifica un errore di timing e l'applicazione si blocca.

Se l'invio avviene con successo, apparirà il messaggio "INFO: Message received by IoT Central. Result is: 0".

Se Result è 0 significa che l'operazione è andata a buon fine. Se Result è 1 ci sono stati dei problemi di connessione.

VISUALIZZARE LA TELEMETRIA

Andando nella pagina "Tutti i dispositivi", una volta fatto clic sul dispositivo, si vedrà la finestra di telemetria che avete creato già dalla volta scorsa. Nella Fig. 14 potete vedere la visualizzazione di telemetria "Temperature" e "Humidity" del dispositivo.

Avvicinando il mouse a una delle due misurazioni appare un riquadro con i dati della telemetria media rilevata e l'orario di acquisizione.

IL CODICE

Non possiamo negare che il codice di questa applicazione (come di tutte le altre applicazioni di esempio) sia complesso e molto lungo. Per brevità, al momento ci siamo limitati a descrivere solo le parti salienti del codice, riportate in forma ridotta del Listato 1.

Tenete presente che tutte le applicazioni di Visual Studio con connettività a IoT Central hanno sempre questo contenuto:

- app_manifest.json
- applibs_versions.h
- CMakeLists.txt
- CMakeSettings.json
- eventloop_timer_utilities.c
- eventloop_timer_utilities.h
- IoTCentral.md
- IoTHub.md
- launch.vs.json
- main.c
- parson.c
- parson.h
- README.md

Commentiamo ora il file *main.c* che, in pratica, riporta tutto il codice necessario al funzionamento dell'applicazione.

Tutto il codice è scritto in C e all'inizio include una serie di librerie classiche di cui non parleremo perché fanno parte dell'ambiente standard.

La libreria *applibs_versions.h* definisce le versioni della

Listato 1

```
// Futura MT3620 DHT22 IoT Central example. Copyright 2020 Pier
Calderan.
// Apre il pulsante 3 e il sensore DHT22 in ingresso. Manda i
dati di temperatura e umidità a IoT Central.
#include <signal.h>
#include <stdbool.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <stdio.h>
#include <stdarg.h>
#include <errno.h>
// applibs_versions.h definisce le versioni della struttura
dell'API da utilizzare
// per le API delle applibs seguenti
#include "applibs_versions.h"
#include <applibs/log.h>
#include <applibs/networking.h>
#include <applibs/gpio.h>
#include <applibs/storage.h>
#include <applibs/eventloop.h>
#include "DHTlib.h" // libreria per il sensore DHT22
#include <hw/sample_hardware.h> // definizione dell'hardware
della scheda
#include "eventloop_timer_utilities.h"
// Azure IoT SDK
#include <iot_hub_client_core_common.h>
#include <iot_hub_device_client_ll.h>
#include <iot_hub_client_options.h>
#include <iot_hubtransportmqtt.h>
#include <iot_hub.h>
#include <azure_sphere_provisioning.h>
#include "parson.h" // utilizzato per analizzare i messaggi
Device Twin
typedef enum {
    ExitCode_Success = 0,
    ExitCode_TermHandler_SigTerm = 1,
    ExitCode_Main_EventLoopFail = 2,
    ExitCode_ButtonTimer_Consume = 3,
    ExitCode_AzureTimer_Consume = 4,
    ExitCode_Init_EventLoop = 5,
    ExitCode_Init_MessageButton = 6,
    ExitCode_Init_DHTButton = 7,
    ExitCode_Init_TwinStatusLed = 8,
    ExitCode_Init_ButtonPollTimer = 9,
    ExitCode_Init_AzureTimer = 10,
    ExitCode_IsButtonPressed_GetValue = 11
} ExitCode;
static volatile sig_atomic_t exitCode = ExitCode_Success;
int main(int argc, char *argv[]) // Punto di ingresso principale
per questo esempio
{
    Log_Debug("IoT Central DHT22 application starting.\n");
    bool isNetworkingReady = false;
    if ((Networking_IsNetworkingReady(&isNetworkingReady) == -1)
    || !isNetworkingReady) {
        Log_Debug("WARNING: Network is not ready. Device cannot
connect until network is ready.\n");
    }
    if (argc == 2) {
        Log_Debug("Setting Azure Scope ID %s\n", argv[1]);
        strncpy(scopeId, argv[1], SCOPEID_LENGTH);
    } else {
        Log_Debug("ScopeId needs to be set in the app_manifest
CmdArgs\n");
        return -1;
    }
    exitCode = InitPeripheralsAndHandlers();
    // Main loop
    while (exitCode == ExitCode_Success) {
        EventLoop_Run_Result result = EventLoop_Run(eventLoop,
-1, true);
        // Continue if interrupted by signal, e.g. due to
breakpoint being set.
        if (result == EventLoop_Run_Failed && errno != EINTR) {
            exitCode = ExitCode_Main_EventLoopFail;
        }
    }
    ClosePeripheralsAndHandlers();
    Log_Debug("Application exiting.\n");
    return exitCode;
}
// Evento timer che controlla lo stato del pulsante
static void ButtonPollTimerEventHandler(EventLoopTimer *timer)
{
    if (ConsumeEventLoopTimerEvent(timer) != 0) {
        exitCode = ExitCode_ButtonTimer_Consume;
        return;
    }
    SendDHTButtonHandler();
}
// Apre il pulsante SAMPLE_BUTTON_3 come input
Log_Debug("Opening SAMPLE_BUTTON_3 as input\n");
sendDHTButtonGpioFd = GPIO_OpenAsInput(SAMPLE_BUTTON_3);
```

```
if (sendDHTButtonGpioFd < 0) {
    Log_Debug("ERROR: Could not open button 3: %s (%d).\n",
strerror(errno), errno);
    return ExitCode_Init_DHTButton;
}
// Apre SENS DHT come input
Log_Debug("Opening DHT22 as input\n");
DHT_SensorData* pDHT = DHT_ReadData(SENS_DHT);
if (pDHT == NULL)
{
    return false;
}
// La pressione del pulsante 3 invia la temperatura e l'umidità
del DHT22 ad Azure IoT Central
static void SendDHTButtonHandler(void)
{
    if (IsButtonPressed(sendDHTButtonGpioFd,
&sendDHTButtonState)) {
        deviceIsUp;
        DHT_SensorData* pDHT = DHT_ReadData(SENS_DHT);
        char tempBuffer[20];
        int len = snprintf(tempBuffer, 20, "%3.2f", pDHT-
>TemperatureCelsius);
        if (len > 0)
            SendTelemetry("Temperature", tempBuffer);
        len = snprintf(tempBuffer, 20, "%3.2f", pDHT->Humidity);
        if (len > 0)
            SendTelemetry("Humidity", tempBuffer);
    }
}
// Imposta un timer per eseguire il polling degli eventi dei
pulsanti
static const struct timespec buttonPressCheckPeriod = { .tv_
sec = 0, .tv_nsec = 1000 * 1000 };
buttonPollTimer = CreateEventLoopPeriodicTimer(eventLoop,
&ButtonPollTimerEventHandler,
&buttonPressCheckPeriod);
if (buttonPollTimer == NULL) {
    return ExitCode_Init_ButtonPollTimer;
}
azureIoTPollPeriodSeconds =
AzureIoTDefaultPollPeriodSeconds;
struct timespec azureTelemetryPeriod = { .tv_sec =
azureIoTPollPeriodSeconds, .tv_nsec = 0 };
azureTimer =
CreateEventLoopPeriodicTimer(eventLoop,
&AzureTimerEventHandler, &azureTelemetryPeriod);
if (azureTimer == NULL) {
    return ExitCode_Init_AzureTimer;
}
return ExitCode_Success;
}
// Invia dati di telemetria a IoT Central
// <param name = "key"> L'elemento di telemetria da aggiornare
</param>
// <param name = "value"> nuovo valore di telemetria </param>
static void SendTelemetry(const unsigned char *key, const
unsigned char *value)
{
    static char eventBuffer[100] = {0};
    static const char *EventMsgTemplate = "{ \"%s\": \"%s\" }";
    int len = snprintf(eventBuffer, sizeof(eventBuffer),
EventMsgTemplate, key, value);
    if (len < 0)
        return;
    Log_Debug("Sending IoT Central Message: %s\n", eventBuffer);
    bool isNetworkingReady = false;
    if ((Networking_IsNetworkingReady(&isNetworkingReady) == -1)
    || !isNetworkingReady) {
        Log_Debug("WARNING: Cannot send IoTHubMessage because
network is not up.\n");
        return;
    }
    IOTHUB_MESSAGE_HANDLE messageHandle = IoTHubMessage_
CreateFromString(eventBuffer);
    if (messageHandle == 0) {
        Log_Debug("WARNING: unable to create a new
IoTHubMessage\n");
        return;
    }
    if (IoTHubDeviceClient_LL_SendEventAsync(iotHubClientHandle,
messageHandle, SendMessageCallback,
/*&callback_param*/
0) != IOTHUB_CLIENT_OK) {
        Log_Debug("WARNING: failed to hand over the message to
IoTHubClient\n");
    } else {
        Log_Debug("INFO: IoT Central accepted the message for
delivery\n");
    }
    IoTHubMessage_Destroy(messageHandle);
}
```


struttura dell'API da utilizzare per le API contenute nella cartella applibs dell'SDK.

Con la direttiva `#include <hw/sample_hardware.h>` viene incluso l'header contenente tutte le definizioni hardware della scheda di cui parleremo nella prossima puntata. In questa applicazione viene inclusa anche la libreria "DHTlib.h" per il sensore DHT22.

Con `#include "eventloop_timer_utilities.h"` si include la funzionalità del timer per il polling degli ingressi. Infine nella sezione Azure IoT SDK del listato vengono incluse tutte le librerie dell'SDK di IoT Central e l'analisi dei messaggi Device Twin di cui si parlerà più avanti. Per facilitare il debug delle varie funzioni viene definita una struttura di `ExitCode` che, con `typedef enum`, enumera le condizioni di successo di uscita delle funzioni. Se la funzione ritorna 0 significa che ha avuto successo, se la funzione ritorna 1 significa che riporta un codice di errore.

Il punto di ingresso principale per tutti gli esempi è sempre la funzione `int main`. Qui, dopo il primo tentativo di connessione alla rete Wi-Fi, viene riportato lo stato della connessione. Se tutto va a buon fine, apparirà il seguente messaggio nella finestra di Output:

```
[Azure IoT] Using HSM cert at /run/daa/df7ba4db-4a32-4bfd-b401-59f2846cc320
```

```
IoTHubDeviceClient_LL_CreateWithAzureSphereDeviceAuthProvisioning returned 'AZURE_SPHERE_PROV_RE-SULT_OK'.
```

Altrimenti apparirà:

```
WARNING: Network is not ready. Device cannot connect until network is ready.
```

Quindi, viene chiamata la funzione `InitPeripheralsAndHandlers()` per inizializzare le periferiche e gli handler, ovvero l'apertura delle porte GPIO per il pulsante 3 (`SAMPLE_BUTTON_3`) e per il sensore DHT22 (`SENS_DHT`).

Nel main loop succede la magia.

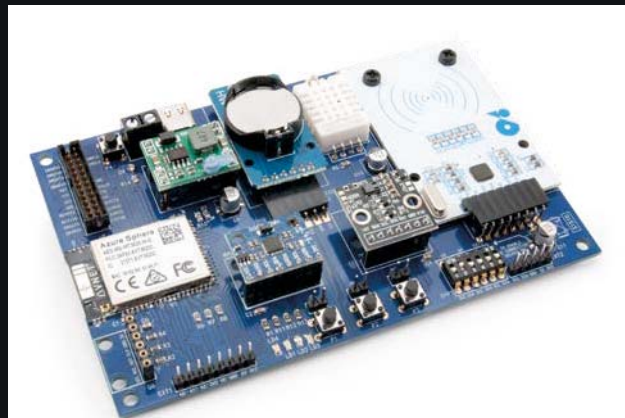
Tutte le funzioni vengono chiamate di volta in volta all'interno del ciclo `while` fintanto che il codice di uscita di ogni funzione riporta `ExitCode_Success`, altrimenti il loop termina e l'applicazione esce con il codice di errore `ExitCode_Main_EventLoopFail`.

Quando terminate il debug, l'applicazione esce chiudendo tutte le periferiche:

```
ClosePeripheralsAndHandlers();
Log_Debug("Application exiting.\n");
return exitCode;
```

Le funzioni principali richiamate dal timer degli eventi sono le seguenti:

- `ButtonPollTimerEventHandler(EventLoopTimer *timer)` // timer che controlla lo stato del pulsante



La demoboard Azure Sphere con MT3620

- `SendDHTButtonHandler()`; // funzione collegata all'handler del pulsante
- `SendTelemetry("Humidity", tempBuffer)`; // funzione di invio della telemetria di umidità
- `SendTelemetry("Temperature", tempBuffer)`; // funzione di invio della telemetria di temperatura
- `buttonPressCheckPeriod` // funzione per impostare il timer che esegue il polling degli eventi dei pulsanti.

CONCLUSIONI

Per oggi ci fermiamo qui. Con la speranza di essere stati sufficientemente chiari, vi diamo appuntamento alla prossima puntata, nella quale approfondiremo le altre parti del codice all'interno dell'applicazione. □



Cosa occorre?

La demoboard di Futura Elettronica con MT3620 (cod. FEAZURESPH) è disponibile a Euro 128,00 e include i componenti base per poter sperimentare con la tecnologia Azure Sphere. Dal sito futurashop.it è possibile acquistare anche le breakout board correlate alla demoboard. La board USB/TTL in grado di creare quattro porte seriali virtuali (cod. FT1497K) è in vendita a Euro 35,00, mentre la board adattatrice necessaria per programmare la demoboard (cod. FT1498K) è disponibile a Euro 25,00. I prezzi si intendono IVA compresa.

Il materiale va richiesto a:

Futura Elettronica, Via Adige 11, 21013 Gallarate (VA)
Tel: 0331-799775 - <http://www.futurashop.it>

Enel X Store: a Roma la prima stazione di servizio 4.0

Enel X inaugura a Roma, in collaborazione con Volkswagen Group Italia, la prima area cittadina per la ricarica ultrafast dei veicoli elettrici e primo sito HPC (High Power Charging). Si tratta di un punto Enel X che concentra la stazione di ricarica per veicoli elettrici e vari servizi Enel Smart utilizzabili ad esempio mentre si fa il pieno di corrente. L'Enel X Store è la prima area urbana ultrafast con quattro (High Power Charger), che consentono la ricarica di altrettanti veicoli contemporaneamente. Si tratta delle prime quattro stazioni ultrafast in Italia del progetto "Central European Ultra Charging", cofinanziato dalla Commissione Europea nell'ambito della call Connecting Europe Facility Transport 2017. Ciascun punto di ricarica ha una potenza fino a 350 kW e permette, alle auto elettriche compatibili con la carica a potenza elevata, di fare il pieno di energia in circa 20 minuti. Oltre ai caricatori ultrafast sono presenti anche una JuicePole, la stazione di ricarica pubblica da 22kW+22kW e un JuiceMedia, ossia lo schermo digitale integrato con due stazioni di ricarica, riservato ai test drive di Volkswagen Group Italia.

Diverse soluzioni per un'esperienza di ricarica completa, comoda e veloce che dà la possibilità, nell'attesa, di prendere un caffè nell'area bar dello store e di

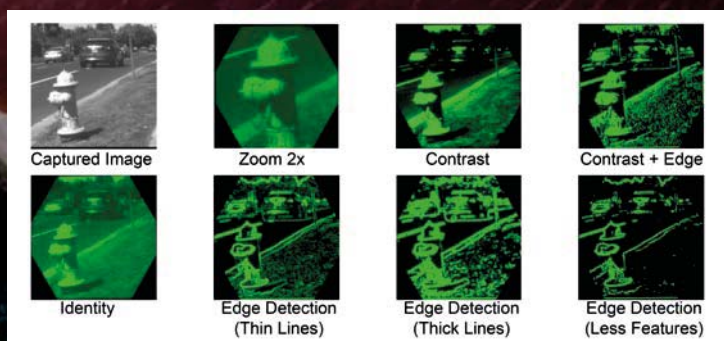
conoscere e provare le tecnologie e i prodotti sostenibili di Enel X per la casa e la mobilità elettrica.

Lo store offre anche l'opportunità di effettuare i pagamenti di bollettini, le ricariche, a breve disponibili anche per l'app JuicePass per fare il pieno alle vetture elettriche, le spedizioni e avere informazioni per l'attivazione di Enel X Pay: l'App e conto on line con carta Mastercard e IBAN italiano che consente, in maniera semplice e intuitiva, di effettuare tutte le tipologie di transazioni: dal pagamento di bollettini, tasse e tributi delle Pubbliche Amministrazioni aderenti al circuito pagoPA, ai bonifici SEPA, passando per lo scambio di denaro peer to peer fino alle donazioni per iniziative di solidarietà ad Associazioni del terzo settore. Enel X Pay permette inoltre di aprire un conto con carta abbinata per i propri figli dagli 11 ai 17 anni e di aderire al Programma Cashback di Stato senza SPID e App Io. All'interno dello store saranno esposti, a rotazione, i modelli di auto full-electric Volkswagen, Audi, SEAT, CUPRA, ŠKODA e Volkswagen Veicoli Commerciali, iniziando dalla recente Volkswagen ID.3, l'auto che per tecnologia segna l'inizio di una nuova era della mobilità elettrica.

www.enelx.com



Chip nelle lenti a contatto per realtà aumentata



Un display microLED su scala millimetrica che si adatta a una lente a contatto: è questo quello che la Mojo Vision, azienda californiana specializzata in innovative lenti a contatto per realtà aumentata, ha realizzato. Un sensore e un chip elaborano le immagini e ricompongono sul display integrato nel lato interno della lente, ciò che l'occhio deve vedere. Questo piccolo gioiello di tecnologia, che probabilmente nemmeno il genio creativo di Ian Fleming avrebbe ipotizzato per il suo 007, si adatta all'occhio senza interferire con la visione di chi lo indossa e nasce idealmente per migliorare la vista di chi ha malattie che la pregiudicano significativamente. Il display incorporato compone un'immagine che migliora il contrasto, evidenzia i bordi ed è in grado di fornire un leggero zoom.

Non è stato facile inserire nelle lenti a contatto tutto quel che serviva: batterie (inevitabilmente di piccola capacità) alloggiare su un circuito stampato flessibile che circonda l'obiettivo, processore di immagini e visualizzatore, che insieme consumano poco più di 100 microwatt ciascuno. Inoltre, l'imager e il display si trovano esattamente nel campo visivo di chi li indossa, quindi devono essere abbastanza piccoli da non proiettare un'ombra apprezzabile sulla retina. Ancora, poiché la vista umana verrebbe fortemente disturbata se ci fosse molta latenza tra ciò che osserva l'imager e ciò che produce il display, il processore di immagini deve essere incorporato nella lente a contatto stessa. Mojo Vision ha stabilito che

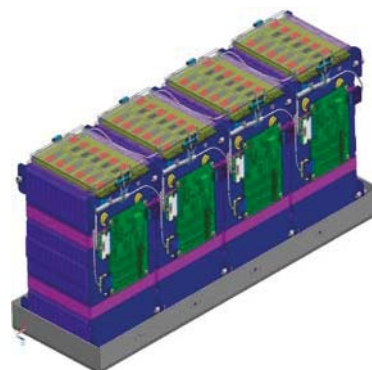
256x256 pixel monocromatici sono sufficienti; l'illuminazione che cade su ogni pixel viene digitalizzata a 4, 6 o 8 bit e il processore di immagini deve eseguire il rilevamento dei bordi, lo zoom e il miglioramento del contrasto oltre ad essere programmabile e calibrabile. Un'architettura di imaging convenzionale non andrebbe bene, perché sarebbe ingombrante e consumerebbe troppo, ad esempio perché in un imager convenzionale, ogni colonna ha il proprio convertitore analogico-digitale (ADC), che occupa spazio. L'ADC e i circuiti pixel richiedono diversi cicli di clock per raggiungere la piena risoluzione del riproduttore d'immagini, consumando energia. La soluzione Mojo Vision è stata sostituire il banco di ADC "energivori" con un set di condensatori che immagazzinano passivamente la carica di lettura dai pixel, che vengono selezionati in sequenza amplificandone le tensioni che vengono poi inviate a un ADC condiviso con una tecnica in grado di evitare errori. Ciò consente di risparmiare energia e area, risultando in un chip da 1,3 mm² che consuma 61-95 microwatt a seconda del frame-rate e della risoluzione voluti. Il processore di immagini è stato mantenuto piccolo condividendo l'hardware utilizzato per produrre gli effetti visivi e la potenza è stata ridotta limitando il numero di volte in cui i dati vengono recuperati dalla memoria integrata. L'intero circuito consuma un massimo di 111 μ W e occupa solo 0,21 mm² di superficie.

www.mojo.vision

Ioni di sodio per batterie applicabili su larga scala

Nel campo delle batterie per veicoli elettrici e applicazioni mobili il litio la fa ancora da padrone, ma per altre applicazioni esistono alternative come la batteria agli ioni di sodio. La sua densità di energia, garantisce circa la metà del litio, ma va bene in applicazioni fisse come lo stoccaggio di energia rinnovabile per le case e la rete elettrica o l'alimentazione di backup per i Data Center, dove il costo è più importante delle dimensioni e della densità energetica. Applicando economie di scala, il costo delle batterie agli ioni di sodio è inferiore di circa il 10-20% rispetto a quelle agli ioni di litio. Ma le batterie agli ioni di sodio presentano vantaggi anche perché la quantità di sodio presente sulla terra è mille volte quella di litio. Inoltre costa meno da estrarre e purificare. Altro pregio è che i catodi di ossido di sodio metallico utilizzati nelle batterie (gli anodi sono al carbonio come quelli delle Li-ion) si accontentano di metalli abbondanti come ferro e manganese. Invece i catodi delle attuali Li-ion richiedono il cobalto, raro e costoso. Tra le tecnologie emergenti spicca quella di Natron, che sta realizzando sia il catodo che l'anodo partendo dal blu di Prussia, pigmento utilizzato nelle vernici e nei coloranti, la cui struttura chimica assicura buon assorbimento e rilascio degli ioni di sodio, il che consente brevi tempi di carica e forti correnti di scarica, nonché durate fino a 50.000 cicli di ricarica. Altro produttore di batterie agli ioni di sodio è Faradion, che ha sviluppato un prototipo capace di una densità di energia fino a 200 Wh/kg.

www.faradion.co.uk

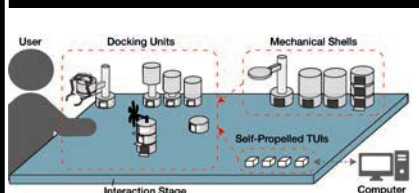


HERMITS: piccoli robot tuttfare

I robot sono ottimizzati per un'attività specifica, ma in alcuni casi, come dimostrano i ricercatori del MIT Media Lab, è possibile dotare robot generalisti di qualcosa che gli permetta lavori specializzati. I ricercatori hanno creato gli HERMITS, piccoli robot basati sui Toio della Sony, che possono svolgere attività altamente specializzate dotati di gusci meccanici specifici. I Toio sono cubi robotici a due ruote che possono localizzarsi in base a modelli a infrarossi, riuscendo a interagire tra loro e con

altri oggetti attraverso un controller centralizzato. I Toio nascono per essere modificati e i ricercatori del MIT gli hanno aggiunto un servomotore che può far fuoriuscire un perno dalla parte superiore del robot per applicarvi "carrozzerie" di tutti i tipi, anche dotate di sistemi di trasmissione meccanica che sfruttano i motori dei robot; qualcosa di simile a un elettrotensile cui si possono applicare varie teste per svolgere diverse lavorazioni. Molti gusci sono privi di elettronica e quindi ottenibili per stampa 3D. I ricercatori hanno gestito fino a 70 robot contemporaneamente utilizzando 14 Raspberry Pi, quindi con hardware economico e comune, accessibile alla comunità open source.

www.media.mit.edu



Su Giove c'è vento

Insieme a un team guidato da astronomi francesi del Laboratoire d'Astrophysique de Bordeaux, ricercatori del Southwest Research Institute (SWRI) hanno identificato per la prima volta venti incredibilmente potenti nell'atmosfera centrale di Giove. Il team ha misurato le molecole riesumate dall'impatto del 1994 della cometa Shoemaker - Levy 9 per tracciare venti superiori a 400 m/s vicino ai poli di Giove. Le caratteristiche bande rosse e bianche dovute alle nuvole vorticosi di Giove consentono agli scienziati di monitorare i venti nella bassa atmosfera del pianeta. Fino ad ora, i modelli di vento nella stratosfera senza nuvole, tra i due strati atmosferici, avevano eluso l'osservazione.

Il radiotelescopio Atacama Large Millimeter / submillimeter Array (ALMA) ha consentito agli scienziati di misurare per la prima volta i venti stratosferici in una condizione meteorologica unica nel nostro sistema solare.

I getti stratosferici potrebbero comportarsi come un vortice gigante con un diametro fino a quattro volte quello della Terra. Gli scienziati SwRI studiano l'aurora gioviana utilizzando lo spettrografo ultravioletto (UVS) guidato da SwRI a bordo della navicella spaziale

Juno della NASA, fornendo la loro esperienza nell'interpretazione di queste misurazioni. Utilizzando 42 delle 66 antenne ad alta precisione di ALMA, il team ha misurato lo spostamento Doppler delle molecole di acido cianidrico, corrispondenti a piccoli cambiamenti nella frequenza della radiazione emessa dalle molecole causati dai venti in questa regione del pianeta, così da dedurre la velocità dei venti come si potrebbe intuire la velocità di un treno in transito dalla velocità con cui il tono del suo fischio varia mentre si allontana.

www.swri.org



Un colorante alza l'efficienza delle celle di Gratzel

Un gruppo di ricercatori del Politecnico di Losanna (EPFL) ha sviluppato un colorante organico che consente alle celle fotovoltaiche organiche (DSSC) un'elevata efficienza di conversione, anche con scarsa illuminazione, semplice ed economico da realizzare. L'efficienza è del 13,5%: 0,5 punti rispetto al record certificato. Conosciuta anche come DSSC (Dye-Sensitized Solar Cell, letteralmente "cella solare sensibilizzata con colorante"), la cella di Gratzel, nasce in Svizzera nei primi anni '90 dal lavoro degli scienziati Brian O'Regan e Michael Grätzel, entrambi dell'EPFL.

Oggi questa tecnologia si è ritagliata una nicchia del mercato fotovoltaico, in virtù della semplicità ed economia di produzione, così come della flessibilità e della versatilità d'utilizzo.

Gli ultimi progressi in questo campo si devono nuovamente al politecnico elvetico, i cui ricercatori Dan Zhang e Marko Stojanovic hanno sviluppato un nuovo colorante battezzato MS5 e funzionante da sensibilizzatore può essere utilizzato singolarmente per far produrre alla cella di Gratzel una tensione a circuito aperto in pieno irraggiamento solare, di 1,24 Volt. Il colorante può anche essere impiegato insieme al colorante organico commerciale XY1b, raggiungendo un'efficienza di conversione dell'energia del 13,5%.

Si tratta di rendimenti e valori tra i più alti ottenuti nel settore delle DSSC.

actu.epfl.ch/news

Temperatura sempre sotto controllo!

Termometri a infrarossi senza contatto



€ 38,00
Cod. TP213N

Termometro IR con puntatore LASER (da -20°C a +400°C)

Termometro a infrarossi UNI-T con puntatore LASER per la misura a distanza della temperatura (da -20°C a +400°C). Dispone di display LCD con retroilluminazione, possibilità di visualizzazione in gradi centigradi o in gradi Fahrenheit, spegnimento automatico. La lettura della temperatura è istantanea. La confezione comprende: il termometro e due batterie da 1,5 V tipo AAA.

Termometro IR con puntatore LASER (da -32°C a +420°C)

Termometro a infrarossi UNI-T con puntatore LASER per la misura a distanza della temperatura (da -32°C a +420°C). Dispone di display LCD con retroilluminazione, possibilità di visualizzazione in gradi centigradi o in gradi Fahrenheit, spegnimento automatico. La lettura della temperatura è istantanea (250 ms) con la memorizzazione della soglia minima e massima in caso di scansione continua e prolungata. Memorizzazione fino a 5 set di valori. La confezione comprende: il termometro, la pila da 9 V e la borsa da cintura per il trasporto.

€ 78,00
Cod. TP224



Prezzi IVA inclusa.



€ 19,00
Cod. FR811

Termometro IR per la lettura della temperatura corporea o di superfici (da +34°C a +42,9°C)

Dispone di ampio display a LED retroilluminato che mostra la lettura in gradi centigradi o Fahrenheit (selezionabile). Gli infrarossi consentono al termometro di leggere la temperatura corporea a una distanza di 5-10 cm in meno di un secondo. L'apparecchio si spegne dopo 8 secondi di standby. Il termometro ha una precisione del display di 0,1°C con un errore massimo di $\pm 0,2^\circ\text{C}$ e un range $34,0^\circ\text{C} \sim 42,9^\circ\text{C}$. Dispone inoltre di segnalazione di allarme con soglia impostabile, cambio di colore del display per indicare le soglie di temperatura misurate. Offre la possibilità di memorizzare fino a 32 misurazioni.



I tuoi PCB a partire da

1€

Con il servizio PCB di Futura Elettronica, potrai realizzare facilmente il circuito stampato del tuo progetto!

Vai su www.futurashop.it, inserisci nella ricerca il codice "PCBPRODUCTION" e carica il tuo Gerber scegliendo le dimensioni e le caratteristiche preferite.

Prezzi IVA inclusa.



cod. PCBPRODUCTION

COLORE SERIGRAFIA



BIANCA



NERA

La serigrafia nera è disponibile solamente per PCB bianchi e gialli.

COLORE SOLDER



VERDE



ROSSO



GIALLO



BLU



BIANCO



NERO

SPESSORE (mm)



0.2

0.4

0.6

0.8

1.0

1.2

1.6

2.0

2.4

2.8

3.2

NUMERO DI LAYERS

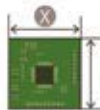
1 Layer

2 Layers

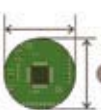
4 Layers

6 Layers

DIMENSIONI (cm)



Single Size



Pannel Size

- 5x5 - 5x10 - 5x15 - 5x20 - 5x25 - 5x30
 - 10x10 - 10x15 - 10x20 - 10x25 - 10x30
 - 15x15 - 15x20 - 15x25 - 15x30
 - 20x20 - 20x25 - 20x30
 - 25x25 - 25x30
 - 30x30

ECCO QUALCHE ESEMPIO:

PCB con 1 layer
 da 5x5 cm
 spessore 1,6 mm
 solder verde
 serigrafia bianca
€ 1 cad.

PCB da 2 layer
 da 10x10 cm
 spessore 1,6 mm
 solder verde
 serigrafia bianca
€ 6 cad.