

Elettronica In

www.elettronica.in.it

€ 6,00 - Anno XXVII - n. 255 - GIUGNO 2021



■ Realizzazione e verifica di un crossover passivo

■ Le configurazioni del timer 555

■ Monitoraggio ambientale con LoRa

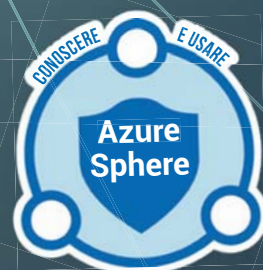
Controller per stampante 3D dual extruder

Lampadina WiFi con moduli ESP

■ Temperatura e umidità da remoto

■ Gestione elettronica dell'iniezione diesel

■ La libreria MQTT per Arduino



Tutto sotto controllo!

Vuoi stare tranquillo anche quando sei fuori casa? Niente di più facile con le telecamere WiFi motorizzate!

Istruzioni in italiano!



TELECAMERA MOTORIZZATA IP WiFi - 2MPX - ONVIF

Telecamera IP a colori motorizzata con sensore d'immagine CMOS da 2MPX. È dotata di ottica fissa da 3,6 mm e di 11 LED IR che permettono di riprendere immagini perfette anche in condizioni di scarsa luminosità. Supporta la visione da remoto tramite smartphone con sistema Android e iOS. I video possono essere registrati sulla memoria dello smartphone tramite apposita APP, oppure su micro SD Card (max. 128GB - acquistabile separatamente). Dispone di microfono e di altoparlante che permettono di parlare e di essere ascoltati da remoto. Formato di compressione video H.264/MJPEG. Push di allarme su motion detection, possibilità di visualizzare dall'APP fino a 9 telecamere contemporaneamente.



Prezzi IVA inclusa.

TELECAMERA Wi-Fi MOTORIZZATA CON LED DUAL LIGHT - AUDIO BIDIREZIONALE / SD CARD / ETHERNET

Telecamera motorizzata Wi-Fi a colori basata sulla tecnologia ONVIF. Può essere gestita da remoto, tramite APP gratuita, disponibile sia per sistemi Android che iOS. Grazie al sistema PTZ (PAN/TILT/ZOOM) di cui la telecamera è dotata, è possibile orientare, da remoto, l'inquadratura delle immagini in funzione delle proprie esigenze di ripresa. Funzione motion detection con notifica di push sul display dello smartphone. È dotata di 8 LED, dei quali 4 IR e 4 a luce bianca. Integra un microfono e uno speaker, che consentono alla telecamera di parlare e di essere ascoltati da remoto. Offre la possibilità di registrare le immagini riprese, sia sulla memoria interna dello smartphone sul quale è installata l'APP di gestione, sia sulla micro SD card inserita nell'apposito slot della telecamera. Viene fornita completa di staffa di fissaggio.



TELECAMERA IP WiFi MOTORIZZATA CON FUNZIONE SMART TRACKING MOTION 2MPX ONVIF

Mini telecamera IP a colori motorizzata dotata di ottica fissa da 3,6 mm, 6 LED IR, con funzione Smart Tracking Motion (monitoraggio del movimento). Il modulo Wi-Fi integrato consente di collegare la telecamera in remoto permettendo di visualizzare su PC e Smartphone (iOS, Android), quanto ripreso dal suo obiettivo che può ruotare sia orizzontalmente sia verticalmente. Consente inoltre di registrare video su PC oppure sulla micro SD card integrata (max 128 GB, acquistabile separatamente) e visualizzare dall'APP fino a 9 telecamere contemporaneamente. Grazie al microfono e all'altoparlante integrato è possibile comunicare con chi è vicino alla telecamera.





La crisi dei componenti

Honda, Ford, Toyota, Nissan, VW e Stellantis hanno ridotto o sospeso la produzione di auto: mancano i componenti per le centraline che governano i complessi sistemi elettronici delle nuove autovetture.

Come è successo questo? Come è possibile che la carenza di componenti abbia fermato l'automotive?

In realtà non vi è un motivo, ma una serie di fattori che sommandosi hanno creato quello che prende il nome di "Electronic Component Shortages".

Si prevede che la carenza di chip causerà una problematica diffusa su diversi fronti, dai dispositivi medici ai computer e smartphone.

I principali produttori di semiconduttori in tutto il mondo hanno interrotto la produzione.

Tra le principali ragioni di questa crisi troviamo ancora una volta il Covid-19: i principali produttori di semiconduttori in tutto il mondo hanno interrotto la produzione come conseguenza della mancanza di materie prime dovuta alle

restrizioni sulla mobilità durante il lockdown. Inoltre con il calo delle vendite di auto nella primavera del 2020, le case automobilistiche sono state costrette a rivedere i loro ordini di materiali, compresi i chip necessari per l'elettronica di bordo. Poi, nel terzo trimestre, quando la domanda di veicoli si è ripresa, i produttori di chip avevano già convertito la produzione per soddisfare la richiesta del mercato di elettronica di consumo e IT.

In ottobre un incendio in uno stabilimento di Asahi Kasei Microdevices, che produceva dispositivi utilizzati nell'industria automobilistica, ha reso introvabili alcuni componenti. A febbraio 2021 le tempeste di neve in Texas hanno bloccato la produzione di Samsung, NXP e Infineon.

Il traffico marino cinese è stato notevolmente ridimensionato, bloccando la movimentazione dei container con un innalzamento dei costi della spedizione via nave. Il trasporto aereo era, ed è, congestionato a causa della distribuzione globale del vaccino Covid-19 e inoltre la capacità di movimentazione merci si è ridotta a causa del calo dei viaggi di passeggeri. A questo va aggiunto il blocco della flotta di Boeing 777 con motori Pratt & Whitney a seguito del guasto di un motore di un aereo sopra il Colorado.

Se il vaccino ci sta portando verso la fine di questa crisi sanitaria, non ci rimane che sperare che la carenza di componenti non rallenti l'uscita dalla crisi economica di questo settore.

Boris Landoni

boris.landoni@elettronica.it



In copertina:
La nuova scheda di controllo per stampanti 3D in grado di gestire due estrusori.

ON LINE

elettronica.it

SI RINGRAZIA

ADI
Futura Elettronica
Microchip
Mouser

Elettronica In

Rivista mensile, anno XXVII n. 255

GIUGNO 2021

EDITORIALE

Direttore Responsabile

Boris Landoni

boris.landoni@elettronica.it

Redazione

Stefano Garavaglia, Paolo Gaspari,
Davide Scullino, Gabriele Daghetta

redazione@elettronica.it

Grafica

Alessia Sfulcini

alessia.sfulcini@elettronica.it

Ufficio Pubblicità

Monica Premoli (0331-752668)

monica.premoli@elettronica.it

Ufficio Abbonamenti

Tel. 0331-752668

abbonati@elettronica.it

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.it/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.it/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

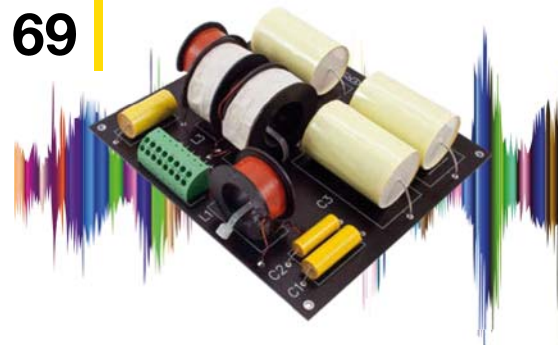


56

Audio

Realizzazione e verifica di un crossover passivo

69



19

ARTICOLI

NETWORKING

19 Temperatura e umidità da remoto

Utilizziamo Blynk per visualizzare temperatura e umidità, rilevate attraverso il popolare sensore combinato DHT11 e inviate da una Arduino MKR1000.

DIDATTICA

29 Il timer 555

Impariamo ad usare il celeberrimo integrato 555 nelle sue tre configurazioni principali: astabile, monostabile e bistabile.

Didattica

Il timer 555



29

Networking

Temperatura e umidità da remoto



19

Didattica**La libreria MQTT per Arduino****41****Tecnologia****Gestione elettronica dell'iniezione diesel****109****Wireless****Monitoraggio ambientale con LoRa****81****Didattica****Corso Azure Sphere****117****Domotica****Controllare una lampadina WiFi con moduli ESP****97****DIDATTICA****41 La libreria MQTT per Arduino**

Analizziamo la libreria MQTT che consente ad Arduino di inviare dati sfruttando anche la connessione GSM.

MONDO 3D**56 Controller per stampante 3D dual extruder**

Stampiamo a due colori con la nostra stampante 3D capace di produrre pezzi in plastica delle dimensioni di 40x40x40 cm: sostituiamo il controller con uno avanzato, montiamo il secondo estrusore e siamo operativi! Prima puntata.

AUDIO**69 Realizzazione e verifica di un crossover passivo**

Costruiamo e testiamo un semplice crossover passivo a tre vie e verifichiamo la rispondenza delle simulazioni con le misure effettuate sul prototipo.

WIRELESS**81 Monitoraggio ambientale con LoRa**

Realizziamo un sistema di rilevamento e monitoraggio di temperatura, umidità e pressione atmosferica e trasmettiamo i dati sfruttando la tecnologia LoRa.

DOMOTICA**97 Controllare una lampadina WiFi con moduli ESP**

Controlliamo una lampadina WiFi con un ESP8266 e con il più evoluto ESP32 utilizzando una libreria realizzata apposta per questi moduli.

TECNOLOGIA**109 Gestione elettronica dell'iniezione diesel**

I moderni motori diesel utilizzano elettroiniettori piezoelettrici e tanta elettronica, che ne hanno elevato l'efficienza, ma che hanno introdotto problemi ignoti con i diesel tradizionali.

DIDATTICA**117 Conoscere e usare Azure Sphere**

In questa puntata spieghiamo come inviare dati seriali ricevuti sull'UART del nostro dispositivo Azure Sphere reale al dispositivo virtuale creato in Azure IoT Central. Quarta puntata.



Soluzioni Touch per la Prevenzione dei Rischi

I primi controller touchscreen capacitivi di sicurezza certificati per applicazioni con elettrodomestici

Metti la sicurezza al primo posto, e al contempo riduci i costi e accedi a moderne soluzioni di interfaccia utente per elettrodomestici.

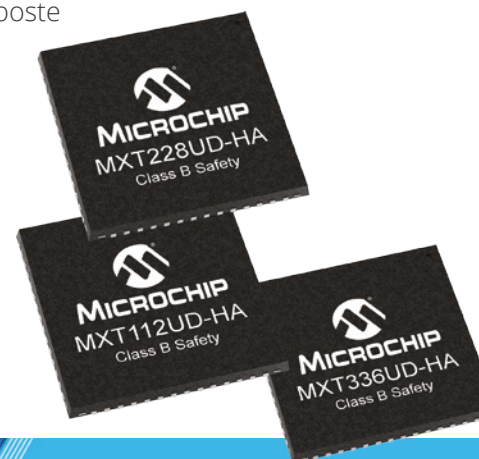
La più recente famiglia di controller per touchscreen capacitivi maXTouch® di Microchip offre firmware IEC 60730 e UL 60730 Class B-certified pre-approvato a livello di sistema, una novità sul mercato. Questi controller touchscreen pre-certificati ti aiutano ad ottenere la certificazione richiesta del tuo sistema con un singolo chip, eliminando anche la necessità di un microcontroller aggiuntivo, e riducendo così i tempi di progettazione, oltre ai costi e i rischi complessivi della progettazione stessa.

Progettati specificatamente per il segmento di mercato degli elettrodomestici, la famiglia mXT336UD-HA è dotata di funzionalità di power-on self-test e di autodiagnostica periodica integrata che garantiscono il funzionamento e le funzionalità previsti di CPU, timer, bus di comunicazione, memoria e front-end touch analogico.

Le funzioni diagnostiche integrate di questa famiglia di touch controller assicurano il corretto funzionamento del controller stesso, nonché del sensore touch connesso. Quando viene rilevata una condizione di guasto, la famiglia mXT336UD-HA avvisa immediatamente il sistema host, consentendo all'elettrodomestico un funzionamento sicuro a prova di guasto, riducendo pericolosi rischi.

Caratteristiche Salienti

- Consente il rilevamento e il tracciamento affidabili di più dita su superfici esposte a umidità, acqua, grasso e altro, persino se l'utente indossa dei guanti
- Disponibili strumenti software e hardware tra cui maXTouch Studio, maXTouch Analyzer ed Evaluation kit



microchip.com/mXT336UD



Il nome e logo Microchip, il logo Microchip e maXTouch sono marchi industriali registrati di Microchip Technology Incorporated negli U.S.A. e altri Stati. Tutti gli altri marchi menzionati appartengono ai loro titolari.
© 2021 Microchip Technology Inc. Tutti i diritti riservati.
DS00003806A, MEC2362A-ITA-02-21

Un po' transistor, un po' MOSFET: ESBT

Nel panorama dei componenti elettronici a semiconduttore destinati al controllo della potenza, si è inserito di recente l'ESBT. Sostanzialmente è una sorta di circuito integrato realizzato però in un package da transistor di potenza, contenente un BJT NPN in serie al cui emettitore si trova un MOSFET a canale N di tipo enhancement-mode.

Il suo funzionamento è il seguente: tenendo sempre polarizzata la base, il componente vede scorrere corrente nel proprio collettore solo quando l'emettitore del BJT viene effettivamente chiuso sul contatto E, cosa alla quale provvede il MOSFET. Si può quindi affermare che l'ESBT è un transistor NPN in cui l'emettitore viene commutato da un MOSFET; da qui il nome ESBT, che è l'acronimo di Emitter Switched Bipolar Transistor, ossia transistor bipolare ad emettitore commutato.

L'ESBT si comanda in tensione come un FET, ha elevatissima resistenza di gate e tempi di commutazione estremamente bassi, dovuti al fatto che per far accendere e interdire il componente non si gioca sulla polarizzazione della base, bensì sul circuito di emettitore, quindi non si deve fare i conti con la lentezza di ricombinazione degli elettroni nella base, che essendo piccola rispetto ad emettitore e collettore dispone di meno lacune di quelle occorrenti alla ricombinazione del gran flusso di elettroni che l'attraversa (nel BJT le regioni di collettore ed emettitore sono drogate con alta percentuale di impurità mentre quella di base ha uno scarso livello di drogaggio).

Piezoelettricità e piezoresistività

Due fenomeni connessi al funzionamento di vari sensori ed attuatori

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.

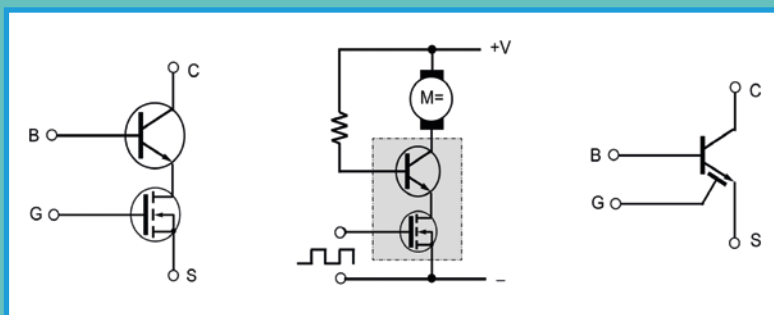
utilizzati in elettronica, nell'automazione domestica e industriale, nonché nell'automobile, sono l'effetto piezoelettrico e l'effetto piezoresistivo. Il primo vede una correlazione biunivoca tra tensione e dimensioni, mentre nell'altro è la resistenza ad essere correlata alle dimensioni dell'elemento interessato.

L'effetto piezoelettrico (il nome deriva dal greco *piezein* che significa premere) consiste nella capacità di un corpo di produrre tensione se deformato lungo una direzione; in questo caso si parla di effetto piezoelettrico diretto. Ma il fenomeno è reversibile, perché infatti applicando tensione a un corpo, questo si deforma, allungandosi o accorciandosi

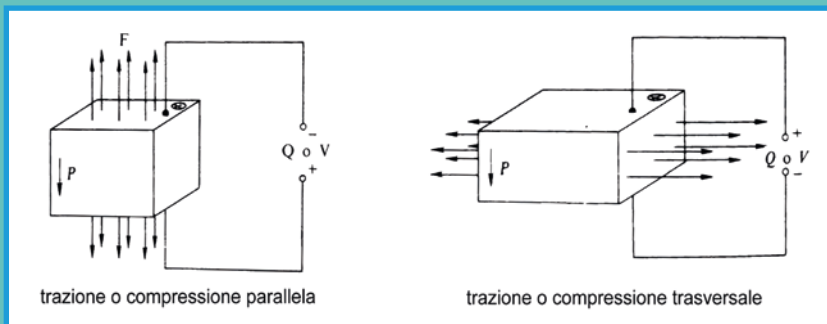
lungo una direzione in funzione della polarità della tensione applicata. Questi effetti si manifestano nei materiali piezoelettrici, che possono essere naturali, come il quarzo, o di sintesi, come alcuni composti ceramici.

L'effetto piezoelettrico si sfrutta ad esempio nei trasduttori di forza e di pressione; l'effetto piezoelettrico inverso è invece sfruttato negli attuatori piezoelettrici (come quelli dei moderni iniettori dei motori diesel) nei cicalini (segnalatori acustici) e negli altoparlanti piezoelettrici.

La tensione sviluppata da un parallelepipedo di materiale piezoelettrico e con essa la direzione di polarizzazione (P) può essere sia ortogonale, sia parallela all'asse della forza, la



↑ Circuito equivalente (sinistra), schema applicativo (centro) e simbolo grafico (destra) dell'ESBT.



↑ **Comprimendo o stirando un materiale piezoelettrico, tra le due facce ortogonali o parallele alla forza F si genera una tensione V con relativo campo elettrico.**

quale può derivare da compressione o da trazione.

Quanto all'effetto piezoresistivo, consiste nella proprietà di taluni materiali di variare la propria resistenza elettrica in base alla sollecitazione meccanica ricevuta lungo un asse e viene sfruttato tipicamente nei sensori di pressione, di peso (celle di carico) e di forza. Anche in questo caso la forza applicata può derivare da compressione o trazione.

I pin di Arduino Uno: come utilizzarli

Nelle schede Arduino i pin riportati sugli header sono quelli del microcontrollore, che per Arduino Uno rev.3 è l'ATmega 328P; le connessioni realizzabili sono ingresso e uscita, per segnali digitali o analogici, ma anche per lo scambio dati su bus. Per spiegarle analizziamo le linee di I/O partendo da quelle digitali, che possono funzionare da ingresso (Input) o uscita (Output) in base

all'inizializzazione del microcontrollore mediante istruzioni che provvedono alla scrittura di opportuni valori nei registri di controllo. Ogni I/O digitale è collegato a una serie di interruttori allo stato solido che commutano il pin verso un registro d'ingresso o di uscita della CPU.

Come ingressi, i pin di Arduino Uno possono leggere due livelli logici: 0 - da zero a 2,5 volt- e 1, da 3 a 5 volt; tra 2,5 e 3 volt non è garantito con certezza il livello che Arduino Uno identifica. Gli ingressi hanno elevata impedenza (alcuni megaohm) per non caricare i dispositivi che li pilotano, che di norma sono in grado di erogare pochissima corrente (gate logici).

Quando sono in modalità Output, i pin possono assumere i livelli logici 0 e 1, corrispondenti rispettivamente a 0 e 5 volt; esiste anche un terzo stato detto Open (l'uscita va in stato di alta impedenza, come fosse interrotta) che rende il pin come sconnesso dal circuito. Per ottenere questa condizione è sufficiente definire il pin come ingresso e non leggerlo. In uscita, i pin di Arduino Uno

hanno dei circuiti di pilotaggio CMOS (Complementary MOS) che sono simmetrici, ossia dispongono di un MOSFET N ed uno P posti in cascata, con i gate in comune: l'uscita è prelevata dai drain. Questa soluzione consente di generare dei livelli 0 e 1 non solo con carichi ad alta impedenza (e quindi bassissimo assorbimento), ma anche su carichi che assorbono un massimo di 40 mA, fino a raggiungere un totale complessivo, per tutti i vari pin di 150 mA. L'avere due transistor consente di presentare la stessa impedenza e quindi erogare (quando l'uscita è a 1 logico) o assorbire (stato logico 0) la stessa corrente, coerentemente con il comportamento che la parola simmetria fa presagire.

Con gli ingressi Analogici Arduino può leggere delle tensioni analogiche mediante gli appositi ingressi siglati A0, A1, A2, A3, A4, A5; siccome la CPU lavora con segnali logici, occorre una conversione, che nell'ATmega328P viene operata da un dispositivo chiamato convertitore analogico/digitale (A/D converter o ADC). Si tratta di un circuito che periodicamente campiona (mediante un dispositivo chiamato S&H - Sample & Hold) il segnale, ovvero lo fa passare a periodi, e mediante una serie di comparatori ne trasforma il livello nella commutazione di stati logici, che poi formano un certo numero di bit. Quindi l'A/D converter effettua misurazioni istantanee che converte in un valore binario.

Del convertitore A/D si definiscono la risoluzione e la frequenza di campionamento; la prima rappresenta il numero di livelli in cui viene diviso l'intervallo di tensioni misurabile ed è espressa in bit. Se il convertitore è a 8 bit, potrà dividere la tensione in 256 valori; se è a 16 bit, in 65.536. Maggiore è la risoluzione, migliore sarà la capacità dell'ADC di distinguere con precisione i valori di tensione assunti dall'ingresso analogico. Arduino ha un convertitore A/D a 10 bit, quindi può distinguere 1.024 gradini di tensione.

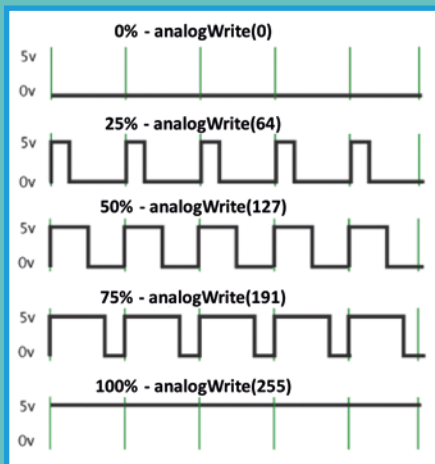
La risoluzione non va confusa con la finestra di tensione campionabile, tuttavia più cresce la differenza tra massima e minima tensione misurabile, minore è la precisione dell'ADC a parità di bit di risoluzione. La scala di rappresentazione binaria della lettura dell'A/D è lineare, quindi nel caso di Arduino Uno, che ha una risoluzione di 1.024 campioni, con tensione di riferimento AREF di 5 V, il valore binario 512 corrisponde a 2,5V, mentre 128 corrisponde a 0,625 volt e così via.



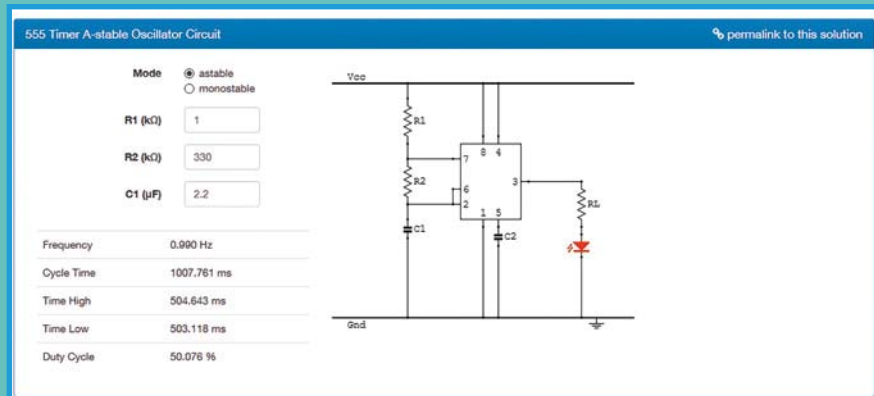
↑ **Header e pin-strip degli I/O di Arduino Uno rev.3.**

Un singolo livello è pari a 0,0048 V. Per avere la massima definizione possibile, impostate sempre un'AREF di poco superiore alla tensione da misurare e ricordate che la definizione è pari al rapporto tra AREF e 1.024. Il valore predefinito di AREF è 5V, internamente impostato; per applicare un riferimento esterno occorre inserire nello sketch l'istruzione `analogReference()` dove tra parentesi si va a scrivere il valore applicato, ovvero `INTERNAL` se si vuole il riferimento interno di Arduino Uno a 1,1 volt. La tensione di riferimento su AREF non deve superare i 5V o essere più negativa di 0,7 V sotto il potenziale di GND. Per altre versioni di Arduino il riferimento potrebbe cambiare (è differente, ad esempio, nella Mega). La tensione di riferimento su AREF può essere applicata anche mediante un comune partitore resistivo, purché alimentato a tensione stabilizzata per evitare letture imprecise. Quanto alla frequenza di campionamento, rappresenta la frequenza con cui la lettura può essere ripetuta nel tempo: più è alta, maggiore è la frequenza dei segnali che possiamo misurare. La frequenza massima di lettura è di circa 9 kHz e questo permette di campionare frequenze fino a 4,5 kHz.

Il convertitore A/D di Arduino è unico ma può essere assegnato alle sei linee analogiche A0÷A5. Sebbene abbia delle uscite considerate analogiche, Arduino Uno in realtà non genera tensioni analogiche, cioè che possono assumere qualsiasi valore compreso tra due margini, con continuità. Le uscite analogiche di Arduino sono in realtà delle PWM (Pulse



↑ Il segnale PWM consente di ottenere una tensione di valore medio proporzionale al suo duty-cycle: si ottiene con l'istruzione `analogWrite`.



↑ La web app accessibile da <https://visual555.tardate.com/> indica la frequenza dell'astabile o la durata della temporizzazione del monostabile specificando i valori dei componenti.

Width Modulation), perché producono impulsi periodici a livello logico 1 o 0 (sono quindi digitali sotto l'aspetto del livello di tensione), però la loro frequenza e larghezza può variare entro due margini assumendo qualsiasi valore intermedio.

Per comprendere il discorso va introdotto il concetto di segnale elettrico ciclico (periodico): un segnale periodico varia ciclicamente e ripete il ciclo costantemente nel tempo con una certa frequenza; la frequenza è il numero di volte che il ciclo si ripete in un secondo. Nel caso del segnale logico, il periodo o ciclo è costituito dall'alternarsi degli stati logici uno e zero; per segnali del genere si usa definire il duty-cycle (letteralmente, ciclo di lavoro) che corrisponde al rapporto tra la durata del livello alto e quella del periodo, ossia della durata del livello alto sommata a quella del livello basso.

Il trucco che permette di ottenere da un segnale digitale uno analogico sta nel giocare sul valore medio di tensione prodotto dagli impulsi: se disegniamo su un foglio di carta i singoli impulsi e sommiamo le aree definite in un periodo, otteniamo il valore medio.

Ne deriva che con un PWM a duty-cycle del 25%, il valore medio è 1/4 dell'ampiezza degli impulsi; con il 50% è la metà. Se gli impulsi hanno ampiezza di 5 volt, nel primo caso il valore medio è 1 volt e nel secondo 2,5V. Il modulo PWM di Arduino Uno è in grado di spaziare da 0 al 100% in 256 passi (il controllo del PWM viene realizzato dalla CPU dell'ATmega328 impostando valori ad 8 bit), quindi il 50% corrisponde al valore binario 128. Il tutto utilizzando l'istruzione `analogWrite()` secondo lo schema visibile nel grafico in alto in questa pa-

gina. Dalle uscite PWM è possibile ricavare una tensione pressoché continua mediante un filtro RC, i cui valori di R (resistenza) e C (condensatore) vanno scelti con la formula:

$$f = 1/(2\pi \times R \times C)$$

dove f è la frequenza del PWM espressa in Hz ed R e C sono espressi rispettivamente in Mohm e microfarad.

Simulatore per 555

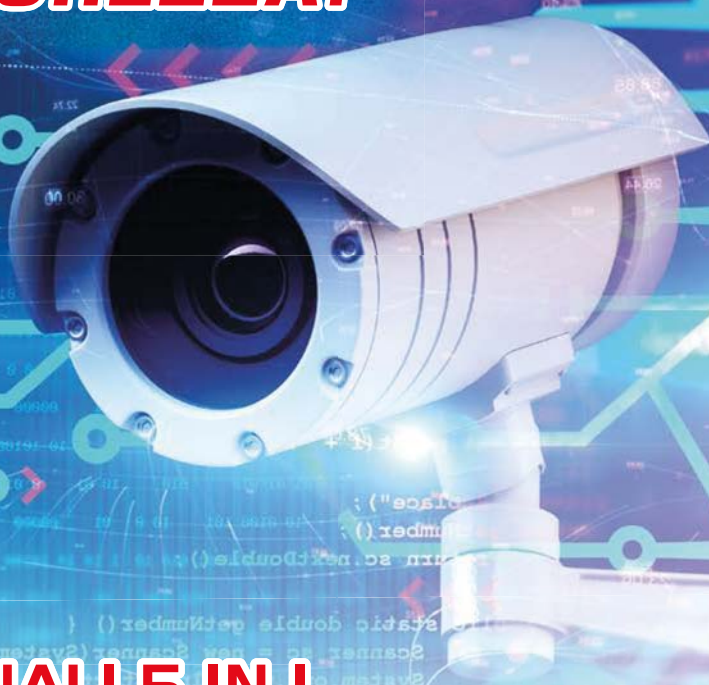
Il popolare timer integrato permette di realizzare monostabili e multivibratori astabili con pochissimi componenti; calcolarli è semplice, ma come al solito in tale compito viene aiuto un tool web come quello accessibile da visual555.tardate.com. Qui troviamo le caselle per definire se simulare il monostabile o l'astabile e il relativo schema con all'uscita un LED che diviene rosso quando l'uscita è a livello alto; inserendo i valori dei componenti di temporizzazione viene calcolata la frequenza per l'astabile o la durata del livello alto per l'astabile.

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.elettronica.in.it/webforum

PROBLEMI DI SICUREZZA? A SORVEGLIARE CI PENSA LUI!



DVR A 4 CANALI 5 IN 1



€64,00

Cod. FR810-4



Videoregistratore a 4 canali 5 in 1 (AHD / CVI / TVI / IP / Analogico), dalle caratteristiche professionali. Dispone di 4 ingressi video (BNC) che possono essere utilizzati sia in modalità HD (AHD / CVI / TVI) che in modalità analogica. È dotato di interfaccia Ethernet (10-100 Base-T) che consente di gestire fino a 16 telecamere IP. Visione remota tramite PC connesso a Internet e Smartphone (iOS, Android) con possibilità di effettuare registrazioni e backup. Funzione OSD (On Screen Display) gestibile tramite mouse. La confezione comprende: DVR, mouse, alimentatore da rete (con uscita 12 VDC - 2A) e guida rapida di installazione in italiano. Hard Disk acquistabile separatamente.



VERSIONE A 8 CANALI



VERSIONE A 16 CANALI



DISPONIBILI ANCHE NELLE
VERSIONI A 8 CANALI

COD. FR810-8 € 82,00

E A 16 CANALI

COD. FR810-16 € 148,00

Advantech e Aws (Amazon Web Services) collaborano all'integrazione fra la tecnologia di bridging edge to Cloud Wise-Edgelinek e il servizio IoT Greengrass, per connettere dispositivi IoT situati negli impianti di produzione manifatturieri allo scopo di accelerare i progetti di Proof of Concept e Proof of Value, sfruttando la potenza di questi strumenti per collegare la parte di automazione (Ot-Operational Technology) alla parte di informatica (It-Information Technology).

Wise-EdgeLink offre un software gateway completo e leggero per la conversione di protocolli che supporta oltre 200 PLC e protocolli di comunicazione per abilitare la raccolta di dati dai dispositivi legacy esistenti. Advantech WISE-Edge-

› IoT edge to cloud per la manifattura



Link semplifica l'integrazione dei dati e gestisce periodi di acquisizione dei dati definiti dall'utente per ogni interfaccia di ingresso, riducendo così

la quantità di dati trasmessi e memorizzati in cloud. Questa funzione, insieme alla funzionalità plug-and-play del software, riduce drasticamente il lavoro di programmazione e facilita lo sviluppo e l'implementazione di applicazioni.

Aws IoT Greengrass è un servizio edge runtime e Cloud IoT open-source che consente di sviluppare, implementare e gestire software per dispositivi. I dispositivi possono così raccogliere e analizzare dati più vicino alla fonte delle informazioni, rispondere autonomamente a eventi locali e comunicare fra loro in maniera sicura su reti locali. I dispositivi locali possono anche comunicare in sicurezza con AWS IoT Core ed esportare dati IoT su Amazon AWS.

www.advantech.com

› Siemens e Google Cloud: l'AI entra nella produzione industriale

La collaborazione tra il colosso tedesco e Big G mira all'implementazione su larga scala di soluzioni basate su Intelligenza Artificiale per efficientare la produzione industriale. Nell'ambito della collaborazione, Siemens intende integrare le principali tecnologie di cloud di dati e di Intelligenza Artificiale e Machine Learning di Google Cloud con le proprie soluzioni di automazione di fabbrica per aiutare i produttori a innovare per il futuro. I dati guidano i processi industriali odierni, ma molti produttori continuano a utilizzare software legacy e più sistemi per analizzare le informazioni sugli impianti, che richiedono molte risorse e frequenti aggiornamenti manuali per garantire l'accuratezza delle analisi. Inoltre, mentre i progetti di intelligenza artificiale sono stati implementati da molte aziende in "isole" in tutto l'impianto, i produttori hanno faticato a implemen-

tare l'intelligenza artificiale su larga scala nelle loro operazioni globali. Combinando il cloud di dati di Google Cloud e le funzionalità AI/ML con il portafoglio Digital Industries Factory Automation di Siemens, i produttori saranno in grado di armonizzare i dati di fabbrica, eseguire modelli AI e Machine Learning basati su cloud su tali dati e distribuire algoritmi sulla rete edge. Ciò consente applicazioni come l'ispezione visiva dei prodotti o la previsione dell'usura delle macchine sulla linea di assemblaggio. Distribuire l'Intelligenza Artificiale in fabbrica e integrarla nell'automazione e nella rete è un compito complesso, che richiede competenze altamente specializzate e prodotti innovativi come Siemens Industrial Edge. L'obiettivo della cooperazione tra Google Cloud e Siemens è rendere più semplice l'implementazione dell'IA in connessione con Industrial



Edge - e la sua gestione su larga scala -, responsabilizzando i dipendenti mentre lavorano nello stabilimento, automatizzando le attività ordinarie e migliorando la qualità generale. Siemens è tra i principali attori nel progresso dell'automazione industriale e del software industriale, mentre Google Cloud è leader nell'analisi dei dati, nell'Intelligenza Artificiale e Machine Learning. Questa cooperazione combinerà il meglio dei due mondi e porterà l'AI/ML nell'industria manifatturiera su larga scala.

press.siemens.com

› Gestire le identità digitali dall'edge al cloud

Eurotech sta collaborando insieme a Infineon, Microsoft e GlobalSign per semplificare il rollout su larga scala dei dispositivi connessi al cloud. Obiettivo di questa collaborazione è assicurare che l'identità dei dispositivi sia certificata lungo tutta la catena del valore, dall'edge al cloud.

La collaborazione assicura che l'identità dei dispositivi sia certificata lungo tutta la catena del valore, dall'edge al cloud.

La chain of trust della soluzione parte dall'integrazione dell'OPTIGA TPM (Trusted Platform Module) di Infineon installato sui gateway IoT di Eurotech.

Come produttore di componenti hardware, Eurotech estende la sicurezza dei dispositivi mediante un Initial Device Identifier,

un'identità basata su certificati IEEE 802.1AR che è legata e crittografata unicamente sul dispositivo cui è assegnata.

Questa identità attesta l'integrità del dispositivo all'interno della supply chain e fornisce una base per la connessione zero touch al cloud. Eurotech collabora con GlobalSign, autorità nella fornitura di certificati di sicurezza, e con Microsoft e il suo IoT Identity Service di Azure IoT Edge, per estendere la chain of trust alla connessione verso il cloud. Questo è possibile grazie all'integrazione di ulteriori certificati locali di sicurezza che confermano l'appartenenza di un dispositivo a un determinato cliente e all'utilizzo di queste identità per il provisioning automatico del dispositivo su Azure IoT Hub tramite il servizio Azu-

› Da Maxim e Aizip soluzioni Low-Power per il riconoscimento di persone

Maxim sta collaborando con Aizip Inc. (società focalizzata sull'intelligenza artificiale orientata alle applicazioni Internet of Things) per fornire soluzioni IoT a bassa potenza di rilevamento delle persone. In quest'ottica il microcontrollore di rete neurale MAX78000 di Maxim Integrated rileva le persone all'interno di un'immagine utilizzando il modello VWW (Visual Wake Words) implementato dalla Aizip che funziona con soli 0,7 millijoule (mJ) per inferenza: un consumo 100 volte inferiore rispetto a quello delle soluzioni software convenzionali e la soluzione di rilevamento delle persone in sistemi IoT

più economica ed efficiente disponibile.

La rete a basso consumo fornisce un funzionamento più a lungo ai sistemi IoT alimentati a batteria che richiedono il rilevamento della presenza umana, inclusi quelli per la gestione energetica degli edifici e le telecamere di sicurezza intelligenti.

Il microcontrollore MAX78000 a bassa potenza, con rete neurale accelerata, esegue inferenze AI a meno di 1/100 dell'energia delle soluzioni software convenzionali per migliorare notevolmente il tempo di esecuzione per le applicazioni AI edge alimentate a batteria.

La rete VWW a precisione mista fa parte della serie Aizip Intelligent Vision Deep Neural Network (AIV DNN) per applicazioni di immagini e video ed è stata sviluppata con gli strumenti di automazione della progettazione proprietari di Aizip, per ottenere una precisione nell'identificazione della presenza umana superiore all'85%.

Il modello Aizip d'Intelligenza Artificiale efficiente e il sistema su chip (SoC) a microcontrollore a basso consumo Maxim Integrated riducono l'energia di inferenza a 0,7 mJ, consentendo 13 milioni di inferenze da una singola batteria AA / LR6. L'estrema compressione supportata dal modello consente una visione intelligente accurata attraverso un microcontrollore (il MAX78000 di Maxim Integrated con accelerazione AI) a basso costo e limitato dalla memoria e sfruttando sensori di immagine economici.

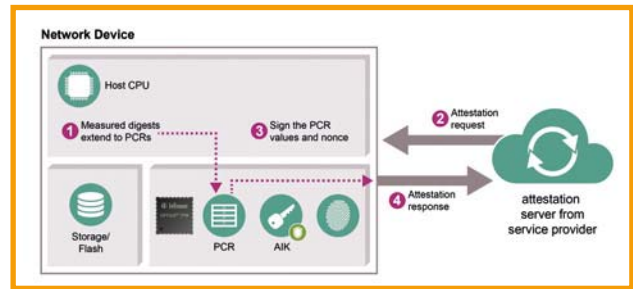
La combinazione delle soluzioni di chip a bassissima potenza di Maxim Integrated e dei modelli AI compatti di Aizip apre le porte a nuove ed entusiasmanti applicazioni nel mondo dell'IoT, che vanno oltre il riconoscimento d'immagine attraverso sistemi edge.

www.maximintegrated.com



re Device Provisioning. La soluzione riduce le complessità legate all'integrazione di identità certificate all'interno di architetture connesse in cloud e fornisce uno schema per la gestione delle identità digitali, basate su standard di sicurezza, lungo tutto il ciclo di vita del dispositivo, dalla produzione, al provisioning, alla manutenzione fino al decommissioning. L'IoT sta cambiando il modo in cui le aziende pensano e operano, permettendo di ottimizzare i processi esistenti e di aprire le porte a nuovi modelli di business e fonti di guadagno. Definendo il processo di creazione di una chain of trust si riduce il rischio di manomissioni e attacchi ai dispositivi che possono comprometterne l'identità lungo tutta la supply chain.

La sicurezza resta un elemento fondamentale per l'adozione di soluzioni cloud e una chain of trust che parta dagli edge-node e arrivi fino al cloud e che possieda caratteristiche di sicurezza



hardware, permette di identificare in modo sicuro ogni dispositivo IoT e di proteggere i dati sensibili, così come l'integrità dell'intera infrastruttura.

www.eurotech.com

ST e Mioty Alliance verso l'impiego di massa dell'IoT

STMicroelectronics, noto produttore di semiconduttori per l'intero spettro delle applicazioni elettroniche, sta ispirando le applicazioni Massive IoT di nuova generazione, che sono guidate dalla scala piuttosto che dalla velocità, attraverso il supporto per lo standard Mioty, che consente un'elevata scalabilità e la realizzazione di connessioni wireless a lungo raggio e a bassissima potenza.

Unendosi alla Mioty Alliance, che mantiene le specifiche e promuove la relativa tecnologia IoT, la STMicroelectronics ha annunciato la disponibilità di uno stack di protocollo dal partner autorizzato ST Stackforce che consente ai progettisti e agli sviluppatori di creare dispositivi Mioty utilizzando il System-on-Chip (SoC) wireless STM32WL. Mioty promette di rendere possibili nuove entusiasmanti applicazioni Massive IoT, ad esempio, per coprire una vasta area geografica; lo stack Mioty rafforza ulteriormente l'ecosistema attorno al SoC STM32WL, che supporta varie tecnologie LPWAN sub-GHz comuni come LoRaWAN, Sigfox e wM-Bus, offrendo un livello unico di integrazione che con-

sente di risparmiare spazio, potenza e time to market.

Mioty invia messaggi utilizzando una tecnica avanzata di suddivisione dei telegrammi, riconosciuta e standardizzata dall'European Telecommunication Standards Institute (ETSI). Ciò garanti-



sce trasmissioni radio estremamente brevi che consentono un funzionamento prolungato senza le sfide logistiche della sostituzione delle batterie in enormi reti IoT. Le brevi trasmissioni riducono al minimo le interferenze con i segnali vicini e consentono a molte migliaia di nodi Mioty di coesistere sulla stessa rete.

I pacchetti di dati Mioty possono viaggiare per diversi chilometri in aree edificate e per più di 15 km in linea di vi-

sta, consentendo a poche stazioni base di coprire grandi siti industriali o aree esterne come i giacimenti petroliferi. Inoltre, i dispositivi Mioty possono comunicare durante gli spostamenti fino a 120 km/h senza problemi di degrado del segnale, servendo applicazioni come la gestione delle flotte, il monitoraggio delle risorse e il rilevamento dei furti.

Lo stack di protocollo Mioty per STM32WL è stato sviluppato da Stackforce, un membro fondatore di Mioty Alliance e ST Authorized Partner. Lo stack è disponibile come libreria per l'integrazione diretta dell'applicazione o del firmware, pronto per essere aggiornato direttamente sul SoC per un utilizzo simile a un modem. STM32WL ha già dimostrato di essere una piattaforma molto potente, soprattutto per quanto riguarda il supporto per stack multiprotocollo.

L'STM32WL è incluso nel programma di longevità di 10 anni della ST, che garantisce la disponibilità a lungo termine del SoC per supportare gli sviluppatori di prodotti industriali e gli OEM.

www.st.com

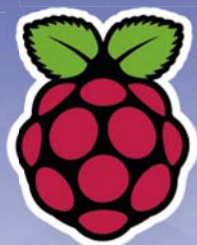
RASPBERRY PI 400!

La tastiera con il computer integrato

€ 146,00

Cod. PI400KIT-ITA

IVA inclusa



**Manuale
in italiano!**

Prezzo IVA inclusa.

Kit completo con computer Raspberry Pi 400 accessori e manuale in Italiano.



Raspberry Pi 400 è un personal computer completo che integra nella tastiera (italiana), una scheda Raspberry Pi 4 da 4Gb con processore quad-core a 64 bit, Wi-Fi, Bluetooth, doppia uscita video (micro HDMI) con una risoluzione fino a 4K, Gigabit Ethernet, USB 2.0 e USB 3.0.

Leggero, portatile, semplice da usare. È ideale per navigare sul web, creare e modificare documenti, guardare video e imparare a programmare utilizzando l'ambiente desktop del sistema operativo Raspberry Pi. I pin GPIO di Raspberry Pi rimangono comunque accessibili, permettendo di connettere componenti e prototipare progetti.

La confezione di Raspberry Pi 400 contiene tutto il necessario (ad eccezione di una TV o di un monitor): computer Raspberry Pi P400, alimentatore da rete, mouse, scheda microSD da 16GB pre-programmata con Raspberry Pi OS, cavo da Micro HDMI a HDMI A lungo 1 metro, un manuale illustrato a colori di 247 pagine completamente in Italiano.

Primo controller single-chip per touchscreen automotive fino a 45"

Microchip annuncia il suo controller touchscreen maXTouch MXT2912TD-UW, prima soluzione single-chip qualificata per il settore automobilistico destinata a display di dimensioni fino a 45 pollici LCD e OLED. Supportato dall'eccezionale rapporto segnale/rumore della tecnologia maXTouch, l'MXT2912TD-UW consente il rilevamento del tocco con più dita attraverso guanti spessi e un'ampia varietà di materiali e spessori di rivestimento, anche in presenza di umidità. Conforme alla specifica ISO 26262 per la sicurezza funzionale nei veicoli stradali, MXT2912TD-UW ha caratteristiche come autotest periodico, test del sensore tattile, test interni di Flash e RAM, controlli completi dell'integrità del percorso dei dati del segnale e test aggiuntivi del core del microprocessore. Il firmware integrato è svilup-



pato per i processi Automotive SPICE. A contorno del controller touchscreen, Microchip offre regolatori low-dropout (LDO), microcontrollori a 8, 16 e 32 bit (MCU), controller area network (CAN) e controller CAN Physical Layer (PHY) e

altro ancora. Il supporto software include lo strumento di sviluppo maXTouch Studio di Microchip e lo strumento di ispezione maXTouch Analyzer per i test di produzione.

www.microchip.com

Digitalizzatore front-end a 16 canali per segnali misti ADI



Analog Devices presenta un digitalizzatore MxFE (front-end a segnale misto) a 16 canali per applicazioni aerospaziali e di difesa, inclusi radar phased array, guerra elettronica e SATCOM (comunicazioni satellitari) di terra. Il digitalizzatore include 4 ricetrasmittitori RF a campionamento diretto SD AD9081 o quattro AD9082. Fornisce catene di segnali RF di riferimento, architetture software, progetti di alimentazione e codice di esempio dell'applicazione.

ADI ha anche introdotto una scheda di digitalizzazione per completare la piattaforma e facilitare gli algoritmi di calibrazione a livello di sistema e la dimostrazione del determinismo della fase di accensione. Le caratteristiche dell'ADQUADMFXFE1EBZ sono:

- 16 canali di ricezione RF (Rx) (32x canali Rx digitali)
- 16x canali di trasmissione RF (Tx) (32x canali Tx digitali)
- esempi applicativi in script MATLAB e in una GUI
- Distribuzione flessibile del clock.

La scheda di digitalizzazione ADQUADMFXFE-CAL fornisce opzioni di loopback del canale adiacente individuale e di loopback del canale combinato, ha un'uscita combinata dei canali Tx e Rx tramite connettori SMA e rilevatori di potenza log incorporati con AD5592R.

www.analog.com

Server rugged per AI e applicazioni Edge da Eurotech

Eurotech propone BoltCOR 32-18, un nuovo Edge server per applicazioni di intelligenza artificiale sul campo e a bordo di veicoli, con prestazioni di calcolo accelerate grazie alla GPU NVIDIA. BoltCOR 32-18 è un server 2U rugged, fanless e ventless per AI e applicazioni Edge accelerate da GPU. Supporta workload tradizionali e per GPU grazie a un processore Intel Xeon E-2276ME con 6 core fino a 4.5GHz, una RAM DDR4 da 64GB e una NVIDIA T4, con 320 Turning Tensor Core, 2560 core CUDA e 16GB di RAM GDDR6 dedicati.

Dotato di numerose opzioni di connettività, consente applicazioni di AI come Deep Learning anche in condizioni ambientali difficili, garantendo un'estrema affidabilità. Inoltre integra fino a 4 modem dual SIM 5G/LTE, oltre a fornire connessione WiFi 6 e Blue-

tooth 5.1, mentre un modulo GNSS multicostellazione con Untethered Dead Reckoning consente una precisa geolocalizzazione e un'accurata sincronizzazione temporale. Il server dispone di porte GbE multiple, con possibilità di integrare interfacce 10GbE e PoE. La capacità di storage è data da quattro slot SATA con RAID opzionale, che completano il modulo NVME interno. BoltCOR 32-18 utilizza connettori M12 ed è certificato EN50155 e EN45545 con temperatura operativa da -25 a +55°C.

www.eurotech.com/it



Kingston porta le MMC nei SoC di NXP



Kingston Digital Europe ha siglato una partnership con NXP Semiconductors, per la realizzazione dei nuovi processori applicativi i.MX 8M Plus, cui fornisce la soluzione di memoria eMMC integrata nella scheda di riferimento per lo sviluppo di applicazioni.

I produttori di dispositivi smart

che richiederanno i kit di verifica tecnica di NXP con chipset i.MX 8M Plus troveranno quindi incluso il componente eMMC prodotto dalla Kingston.

Le soluzioni di memoria e di storage di Kingston sono state incorporate anche nelle schede dei processori NXP delle precedenti generazioni i.MX 6 e

i.MX 7, rappresentando quindi una garanzia di qualità e affidabilità per gli sviluppatori.

Il nuovo processore NXP i.MX 8M Plus trova principali applicazioni nel machine learning, nel settore video e multimedia e nei dispositivi IoT industriali (IIoT).

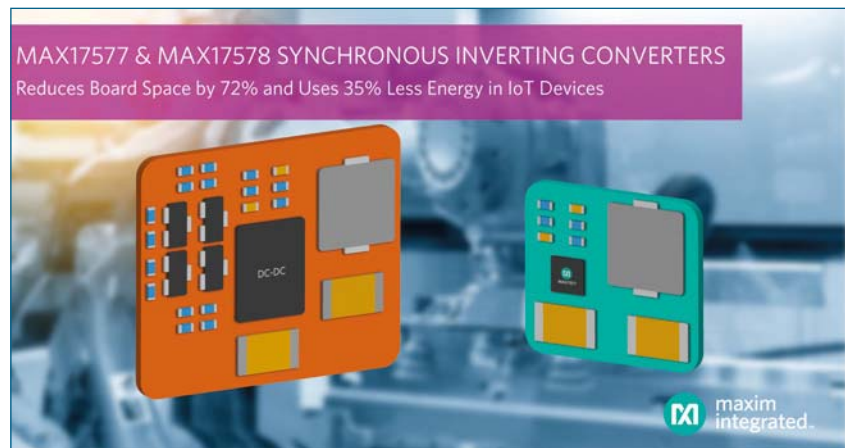
www.kingston.com

DC/DC converter sincroni Maxim per circuiti a ridotto footprint

I nuovi regolatori integrati MAX17577 e MAX17578 di Maxim Integrated sono gli step-down sincroni più piccoli e con la massima efficienza del settore.

Primi convertitori DC/DC a 60 V di Maxim con traslatori di livello integrati, consentono di risparmiare fino al 72% dello spazio sulla scheda riducendo della metà il numero di componenti e utilizzando il 35% di energia in meno rispetto alle soluzioni simili.

Entrambi i circuiti integrati consentono di ridurre le dimensioni, il calore e i costi, semplificando nel contempo la progettazione delle linee di tensione di uscita negativa necessarie per i segnali analogici all'interno dei dispositivi IoT intelligenti che si trovano nell'automazione di fabbrica, nell'automazione degli edifici e nei sistemi di comunicazione. Man mano che i dispositivi edge di rete si espandono con le capacità di intelligenza artificiale, i progettisti devono ridurre le dimensioni della soluzione e la generazione di calore; gli integrati MAX17577 e MAX17578 integrano circuiti di cambio di livello per ridurre il costo dei compo-



nenti fino al 50%, riducendo l'area del PCB a 60 mm²: una riduzione del 72 per cento nello spazio su scheda rispetto alle soluzioni concorrenti.

Questi convertitori invertitori sincroni proteggono dai transitori della tensione di ingresso con la gamma di tensioni di ingresso più ampia del settore da 4,5 V a 60 V, aumentando così la robustezza del sistema in ambienti difficili.

I modelli MAX17577 e MAX17578 generano meno calore rispetto alle soluzioni della concorrenza, migliorano l'affidabilità del sistema e forniscono agli sviluppatori fino a sei punti percentuali in più di efficienza. I nuovi DC/DC coprono una gamma di tensioni di ingresso compresa da 4,5 V a 60 V: la gamma più ampia del settore.

www.maximintegrated.com

SOM Kria: la soluzione Xilinx per lo sviluppo edge-based

Xilinx presenta la sua serie di SOM (System-on-Module) adattivi Kria, schede embedded small form factor per una rapida implementazione in applicazioni edge-based.

Insieme a uno stack software completo, portano il calcolo adattivo agli sviluppatori di software e AI. Il primo disponibile è il Kria K26 SOM,

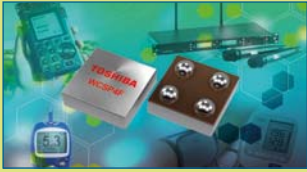
si rivolge specificamente alle applicazioni AI di visione nelle smart city e smart factory; si basa sull'architettura Zynq UltraScale+MPSoC, con un processore quad-core Arm Cortex A53, oltre 250.000 celle logiche e un codec video H.264/265. Il SOM dispone anche di 4 GB di memoria DDR4 e 245 IO, che gli con-

sentono di adattarsi praticamente a qualsiasi sensore o interfaccia. Con 1,4 tera-op di elaborazione AI, Kria K26 SOM consente agli sviluppatori di creare applicazioni AI di visione che offrono prestazioni 3 volte più elevate con una latenza e una potenza inferiori rispetto ai SOM basati su GPU.

www.xilinx.com



LDO precisi e a basso rumore da Toshiba



Toshiba Electronics ha annunciato l'introduzione della nuova serie di dispositivi TCR5RG. Questi regolatori di tensione a basso drop-out (LDO) coprono un intervallo di tensioni di uscita compreso fra 0,9V e 5,0V, che ne consente l'impiego in un'ampia varietà di applicazioni potenziali. Presentano un rapporto di reiezione del ripple di 100 dB (operando a una frequenza di 1kHz). Tutto ciò è stato ottenuto attraverso la combinazione di una topologia circuitale interna ad ampio bandgap, di un amplificatore operativo integrato ad alta velocità che supporta il funzionamento a basso rumore, e inoltre di un filtro passa-basso. L'ampia serie di LDO com-

prende un totale di 45 dispositivi, tutti in package WCSP4F, che occupa uno spazio minimo su PCB. Gli LDO supportano una corrente di uscita massima di 500 mA e presentano un rumore di tensione di uscita di 5µV (tipico), con una precisione della tensione di uscita di $\pm 1,5\%$ per tensioni comprese tra 1,8V e 2,8V (che passa a $\pm 1,8\%$ per tensioni superiori a 2,8V) e un intervallo di temperature del chip compreso fra -40°C e $+85^{\circ}\text{C}$. Grazie alla combinazione di elevate prestazioni e fattore di forma compatto, i dispositivi della serie TCR5RG sono ottimizzati per l'utilizzo in sistemi elettronici ad alta densità con vincoli di spazio.

Gli impieghi principali includono i sistemi basati su sensori, chipset IoT, sistemi RF, tecnologia indossabile, apparecchi audio portatili e unità per il monitoraggio medicale.

toshiba.com

Prima MCU certificata CMVP-livello 3 da Renesas

Il microcontrollore RX65N Renesas ha ottenuto la certificazione Cryptographic Module Validation Program (CMVP) di livello 3 secondo lo standard di sicurezza FIPS 140-2. L'RX65N è il primo MCU per uso generico al mondo a ottenere la certificazione di livello 3 e il suo utilizzo renderà più facile sviluppare dispositivi con funzioni di sicurezza robuste e affidabili, semplificando lo sviluppo e aumentando la sicurezza. L'RX65N a 32 bit offre funzioni di sicurezza, connettività e interfaccia uomo-macchina (HMI) adatte in applicazioni come dispositivi industriali e IoT. Incorpora il modulo Trusted Secure IP (TSIP), già certificato CAVP,



che comprende un motore di crittografia con supporto AES, SHA, RSA ed ECC, un generatore di numeri casuali TRNG e la gestione delle chiavi crittografate. L'RX65N è dotato di Flash a doppio banco che supporta il funzionamento in background e la funzione SWAP per consentire aggiornamenti firmware sicuri e affidabili. Renesas offre anche una "Secure Cloud & Sensor Solution" che combina l'RX65N con i sensori Renesas per un'interazione Cloud sicura.

www.renesas.com

Optoisolatore digitale ad ossido spesso da STM

È iniziata, in casa STM, la produzione in serie dell'isolatore digitale a doppio canale STISO621, destinato a sostituire i fotoaccoppiatori generici nelle applicazioni industriali. L'STISO621 permette di trasferire i dati tra due domini isolati fino a 100 Mbit/s, con distorsione dell'impulso inferiore a 3 ns, sfruttando la tecnologia di isolamento galvanico a ossido spesso 6kV di STM. Con due canali unidirezionali indipendenti, il dispositivo funziona come un'interfaccia UART che gestisce i dati in entrambe le direzioni.

Gli ingressi trigger di Schmitt su ciascun canale garantiscono un'elevata immunità ai disturbi. Le tensioni di alimentazione ai due lati reciprocamente isolati galvanicamente dell'STISO621 sono indipendenti l'una dall'altra.

Ogni lato ha un ampio intervallo di tensione, consentendo la conversione del livello tra i circuiti 3,3 V e 5,5 V. La tipica immunità ai transienti di modo comune (CMTI) di 65 kV/s protegge il lato a bassa tensione da transistori di commutazione elevati in ambienti difficili. STISO621 è adatto a un'ampia gamma di applicazioni industriali e di consumo in alimentatori, azionamenti motore, contatori, inverter, monitor della batteria, apparecchiature, isolatori fieldbus, adattatori di isolamento multicanale dimensionali critici e isolamento generale in tutti i sistemi di automazione industriale.

Il design è testato in conformità VDE0884-10 e UL 1577, le specifiche tipicamente applicate ai disposi-

tivi di isolamento per applicazioni ad alta tensione.

Per accelerare lo sviluppo dei progetti, STM rende disponibile la scheda di valutazione del prodotto EVAL-STISO62XV1.

È inoltre disponibile il progetto di riferimento EVALSTPM-3PHISO, specificamente creato da STM per i casi d'uso di sistemi di misurazione isolati trifase: combina l'STISO621 per separare i domini dell'alta tensione con l'IC front-end di misurazione STPM52 ad alta precisione della ST e il firmware specifico in esecuzione su un microcontrollore STM32 per calcolare la metrologia trifase e i dati sulla qualità dell'alimentazione. Con una tensione di isolamento massima di esercizio (VIOWM) di 1200 V di picco e un'elevata tensione di tenuta all'impulso (VIOTM), l'integrità dell'isolamento galvanico viene mantenuta nel tempo e durante qualsiasi guasto del sistema. Sono disponibili due varianti di pacchetto.

Nel package SO8 a passo stretto con pitch laterale di 4 mm, l'STISO621 garantisce una VIOTM di 4,800V. L'STISO621W in package SO8 wide-body con passo di 8 mm presenta invece una VIOTM di 6000V. Il campo di temperatura di funzionamento spazia per entrambi da -40°C a 125°C .

www.st.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.it/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



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

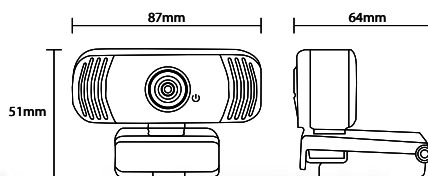
Prezzo IVA inclusa.



€24,00
Cod. FR795

WEBCAM FULL HD 1080P

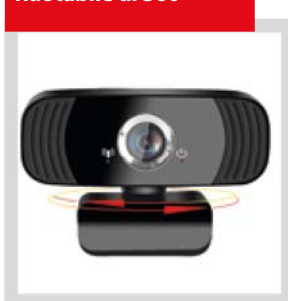
Webcam Full HD 1080P particolarmente indicata per seguire lezioni online, effettuare videoconferenze, videochat, videosorveglianza, registrazioni video, ecc. Dispone di lenti prive di distorsione, microfono omnidirezionale con riduzione del rumore, regolazione manuale della messa a fuoco, angolo di ripresa: 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



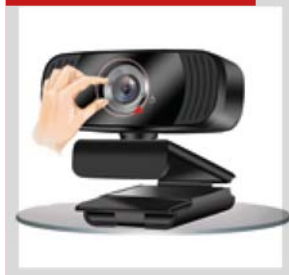
Ruotabile di 360°



Vari metodi di posizionamento



Supporta la messa a fuoco manuale



TEMPERATURA E UMIDITÀ DA REMOTO



di TIZIANO ROSSI



Utilizziamo Blynk per visualizzare temperatura e umidità, rilevate attraverso il popolare sensore combinato DHT11 e inviate da una Arduino MKR1000.

S

empre più di frequente troviamo applicazioni per smartphone che controllano l'accensione, spegnimento delle luci e/o la visualizzazione delle condizioni dell'ambiente in cui viviamo. In un certo senso si tratta di una piccola parte di sviluppo di quello che fino ad oggi siamo stati abituati a chiamare IoT (Internet delle cose). Già esistono in

commercio dispositivi e siti come Cayenne, ThingSpeak, Google, Amazon, che ci permettono di sviluppare controlli remoti, ma, come spesso accade, hanno un costo che può variare da poche centinaia a qualche o più migliaia di euro.

Le applicazioni possibili sono già tante e in continuo aumento: dal controllo dagli elettrodomestici, grandi o piccoli che siano, per arrivare all'impianto di climatizzazione alle porte blindate e alle finestre. Tutto può essere in rete e così si può non solo controllare a distanza, ma anche monitorare il buono stato

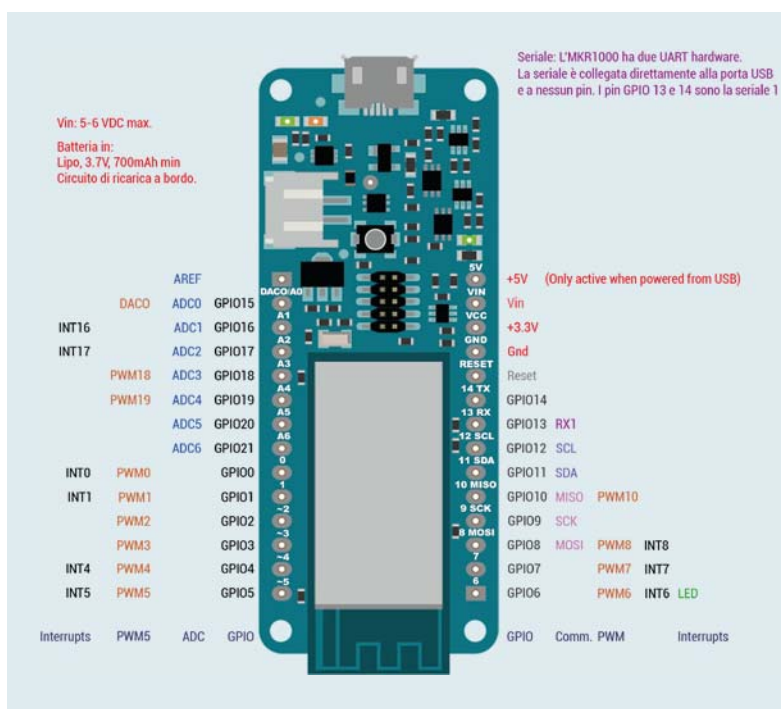
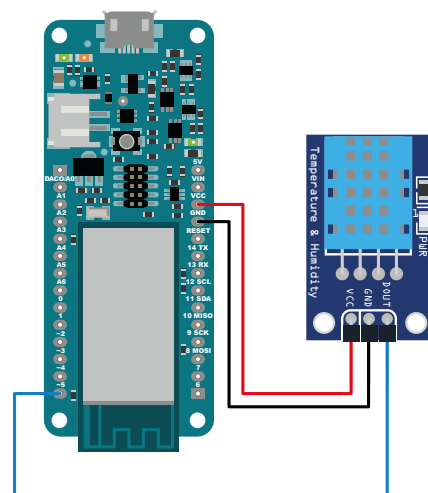


Fig. 1
Pinout del
modulo
Arduino
MKR1000.

degli apparecchi e gestirne, nel caso, la manutenzione.

Lo smartphone è ormai un telecomando tuttotfare. In questo articolo descriviamo un'applicazione che utilizza Blynk App per comandare e monitorare la temperatura e l'umidità dell'ambiente in cui viene installato il dispositivo hardware; inoltre non necessita di alcun collegamento a siti Internet in precedenza menzionati per il monitoraggio dei dati, ma semplicemente utilizza lo smartphone per visualizzare i dati di temperatura e umidità. Trattandosi di un progetto che utilizza la connessione ad Internet, quindi dobbiamo avere un hardware che sia in grado di fare ciò; a tale scopo abbiamo utilizzato la scheda Arduino MKR1000, la quale contiene un modulo WiFi che ci permette, con i dovuti accorgimenti, di collegarci al nostro router WiFi per accedere a Internet. Con Blynk si potrebbero usare altri dispositivi come Arduino Uno, ma avrebbero bisogno di una Ethernet o WiFi Shield per comunicare; altri sono già dotati di WiFi, come ESP8266, Raspberry Pi, Particle Photon o SparkFun Blynk Board. L'elenco di hardware supportati da Blynk App è lungo e continuerà a crescere. Come si può vedere in **Fig. 1** lo schema di cablaggio del nostro sistema locale è di semplice attuazione; la stessa figura propone la piedinatura della scheda utilizzata. La scheda Arduino MKR1000 acquisisce i dati tramite il sensore DHT11 e li trasferisce via Internet alla nostra

schema di CABLAGGIO



applicazione Blynk App, che chiameremo FUTURA_BLYNK_DHT11; inoltre, e per nostra comodità, i dati trasmessi possono essere visualizzati in anteprima sulla porta seriale a cui la scheda Arduino MKR1000 è collegata, utilizzando il monitor seriale dello sketch di Arduino. Tutti i componenti utilizzati possono essere reperiti su sito di Futura Elettronica, www.futurashop.it:

- Arduino MKR1000, codice prodotto MKR1000;
- Scheda con sensore di Temperatura e Umidità DHT11, codice prodotto TEMPHUM.

INSTALLAZIONE APPLICAZIONE

L'applicazione che andremo a sviluppare può funzionare sia su smartphone con sistema operativo iOS versione 9 o superiore, sia su dispositivi Android versione 4.2 o superiore. Tramite il vostro smartphone installate l'applicazione Blynk, apritela e fate clic sull'icona appena scaricata (vedi **Fig. 2**). Una volta aperta l'applicazione sul vostro smartphone potrete creare il vostro account personale seguendo le istruzioni sotto riportate, oppure tramite le indicazioni riportate nel manuale Blynk App alla sezione docs.blynk.cc/#getting-started-getting-started-with-the-blynk-app-1-create-a-blynk-account. La prima pagina di Blynk App che vi comparirà è riportata in **Fig. 3**. Il passo successivo è fare clic su Create New Account in Blynk app; compariranno due campi, ossia:

- **Email** – dovrete immettere l'account della vostra e-mail;
- **Password** – non deve essere necessariamente la stessa password del vostro account e-mail, potete utilizzare una nuova password possibilmente "sicura".

La password creata vi servirà per il login all'applicazione. Avete così creato il vostro account, quindi siete pronti per partire: fate clic sul pulsante New Project di **Fig. 3** allorché vi troverete in una schermata contenente tre campi da riempire:

- **PROJECT NAME**; digitate FUTURA_DHT11 (abbiamo utilizzato un titolo breve per essere interamente visualizzato nella toolbar);
- **CHOOSE DEVICE**; fate clic all'interno del campo, vi compariranno una lista di dispositivi. Scorgete la lista fino a trovare la scheda Arduino MKR1000, evidenziatela e fate clic su OK;
- **CONNECTION TYPE**; corrisponde al tipo di connessione che utilizzeremo. Nel nostro caso sarà WiFi.

Terminate quindi la compilazione dei tre campi e fate clic su CREATE PROJECT; riceverete quindi due e-mail il cui oggetto è:

- **Get started with Blynk**;
- **Auth Token for FUTURA_DHT11 project and device FUTURA_DHT11**: l'**Auth Token** ricevuto è composto normalmente da trentadue numeri di lettere minuscole e maiuscole e lo dovrete

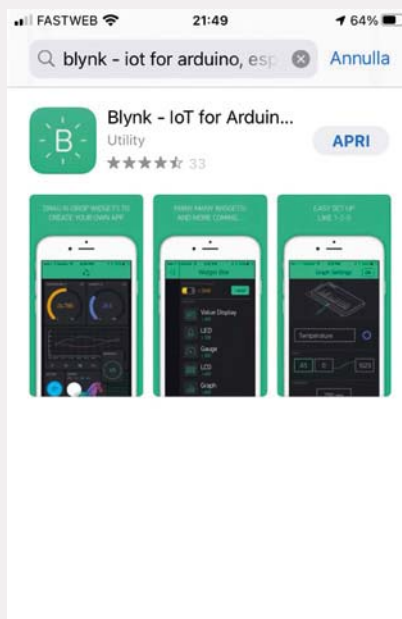


Fig. 2
Applicazione
Blynk App per
smartphone.

copiare nello sketch di Arduino come vi spiegheremo tra breve.

Il token dovrete utilizzarlo per inserirlo nella parte di autenticazione dello sketch Arduino tramite l'IDE. In entrambe le mail ricevute sono riportati alcuni link che vi permettono di accedere alla documentazione di BlynkApp come: iniziare, generare sketch per diversi dispositivi, e, soprattutto, come scaricare e installare la libreria per i dispositivi da utilizzare.

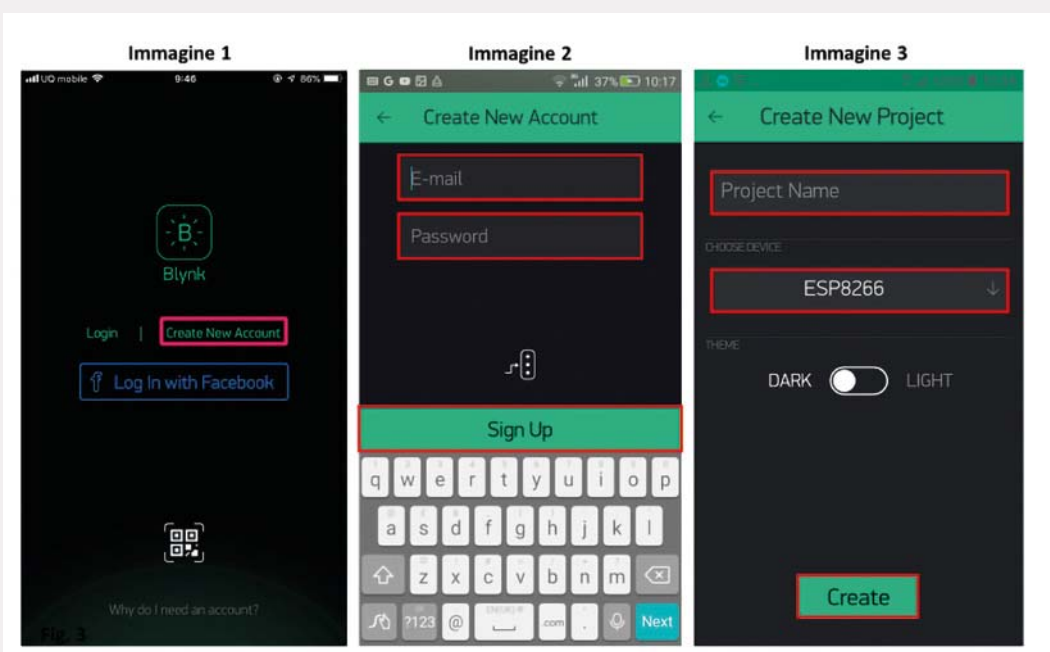
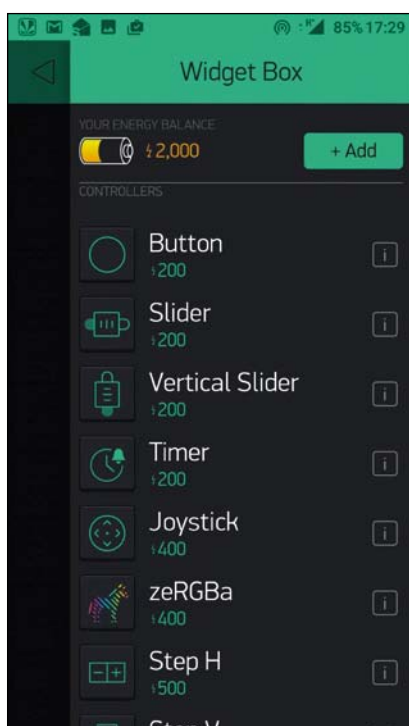
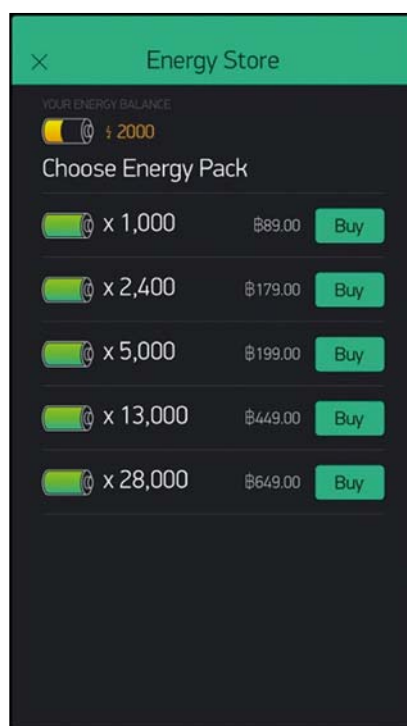


Fig. 3
Blynk App,
sequenza
immagini
per la
creazione dei
"Widget" di
Temperatura
e Umidità.

➔ **Fig. 4**
Blynk App, Widget
Box.



⬅ **Fig. 5**
Blynk App,
Energy Store.



CONFIGURAZIONE APPLICAZIONE

Eseguiti i passi sopra indicati finalmente possiamo incominciare a creare la nostra applicazione; nella pagina vuota che apparirà andremo a collocare i nostri widget (oggetti). Toccando la pagina, o spostandola da sinistra a destra, entrerete nel Widget

Box (**Fig. 4**); scorrendo verso il basso troviamo una lista completa di widget che possiamo utilizzare "gratuitamente" in base al valore totale disponibile di **YOUR ENERGY BALANCE** indicato in alto a fianco del simbolo della batteria. Il valore riportato dovrebbe essere di 2000 Energy Balance: è più

➔ **Fig. 6**
Blynk App,
applicazione
DHT11.



⬅ **Fig. 7**
Blynk App,
configurazione
widget.



che sufficiente per creare gratuitamente la nostra applicazione.

Ogni widget ha un Energy Balance Value; per la nostra applicazione saranno necessarie: due widget Gauge e un widget SuperChart.

- Gauge: è un widget di tipo indicatore numerico e visivo (se prima di selezionare qualsiasi widget volete conoscere la funzione potete avere una indicazione cliccando sulla **i** che vi trovate a fianco del widget).
- Super Chart: è una visualizzazione grafica, per quanto ci riguarda, dell'andamento della Temperatura e dell'Umidità nel tempo.

La somma dei widget selezionati non supera YOUR ENERGY BALANCE a disposizione, per cui non è necessario acquistarne.

Raggiunta una certa familiarità con Blynk App, qualora vogliate ampliare la vostra applicazione, aggiungendo widget, e il valore di "energia" che rimane è insufficiente, potete acquistarne nell'Energy Store cliccando su +Add (vedi Fig. 5).

Bene, ci siamo: possiamo incominciare a immettere i dati che ci permetteranno di visualizzare Temperatura e Umidità nella nostra applicazione chiamata **FUTURA_DHT11** (vedi Fig. 6) ma, prima, valutiamo se ci sono eventuali errori. Qualora sbagliate nella scelta dei widget o vogliate adattarli allo schermo procedete come segue:

- tenendo premuto il widget di vostro interesse, vi comparirà la modalità "modifica" in modo che possiate allargare o restringere il widget a vostro piacimento;
- se non di vostro interesse, per cancellarlo tenete premuto il widget e spostatelo verso l'alto a destra sulla barra del titolo;
- rilasciatelo e automaticamente verrà cancellato andando così anche a liberare l'"energia".

Supponendo che tutto sia corretto, procediamo alla compilazione dei dati per la visualizzazione di Temperatura e Umidità come segue:

- Fate clic sull'icona **GAUGE** in alto a sinistra per entrare nella pagina di configurazione dove possiamo dare un titolo: **Temperatura**;
- scegliete l'input dati: è importante impostare il **Virtual PIN – V6** come di seguito:
 - Blynk.virtualWrite(V5, h);
 - Blynk.virtualWrite(V6, t);
- scegliete il range minimo e massimo di lettura Temperatura (0 e 50 °C);
- date un'etichetta al dato per la temperatura espressa in gradi centigradi (°C);



Fig. 8
Blynk App,
visualizzazione
dati.

- scegliete la dimensione dei Font e il colore del widget;
- scegliete l'intervallo di lettura (vi consigliamo di lasciare **Push**).

Tutti i passi sopra trattati sono riassunti in Fig. 7. Questa operazione deve essere ripetuta per ogni widget che compone la nostra applicazione, nel nostro caso dovremo inserire anche l'umidità, facendo attenzione a scegliere il Virtual PIN appropriato a cosa dobbiamo visualizzare nel momento in cui ci collegheremo al nostro server (Fig. 8). Creata l'applicazione, più tardi vedremo come effettuare il collegamento al nostro account server di Blynk per ricevere i dati. Manca solo un altro piccolo passo e avremo raggiunto il nostro obiettivo.

ARDUINO MKR1000

La scheda Arduino MKR1000 è il nostro end device che ci permette il collegamento ad Internet, per cui se escludiamo il collegamento del PC tramite USB alla scheda per caricare l'IDE del nostro programma, non abbiamo bisogno di laptop o PC per il controllo dei dati, ma solo del nostro smartphone (vedi Fig. 9).

Iniziamo a lavorare con Arduino: per prima cosa installate la libreria Blynk App per l'IDE di Arduino, scaricandola al seguente indirizzo:



Fig. 9
Schema a blocchi
del funzionamento
Blynk App.

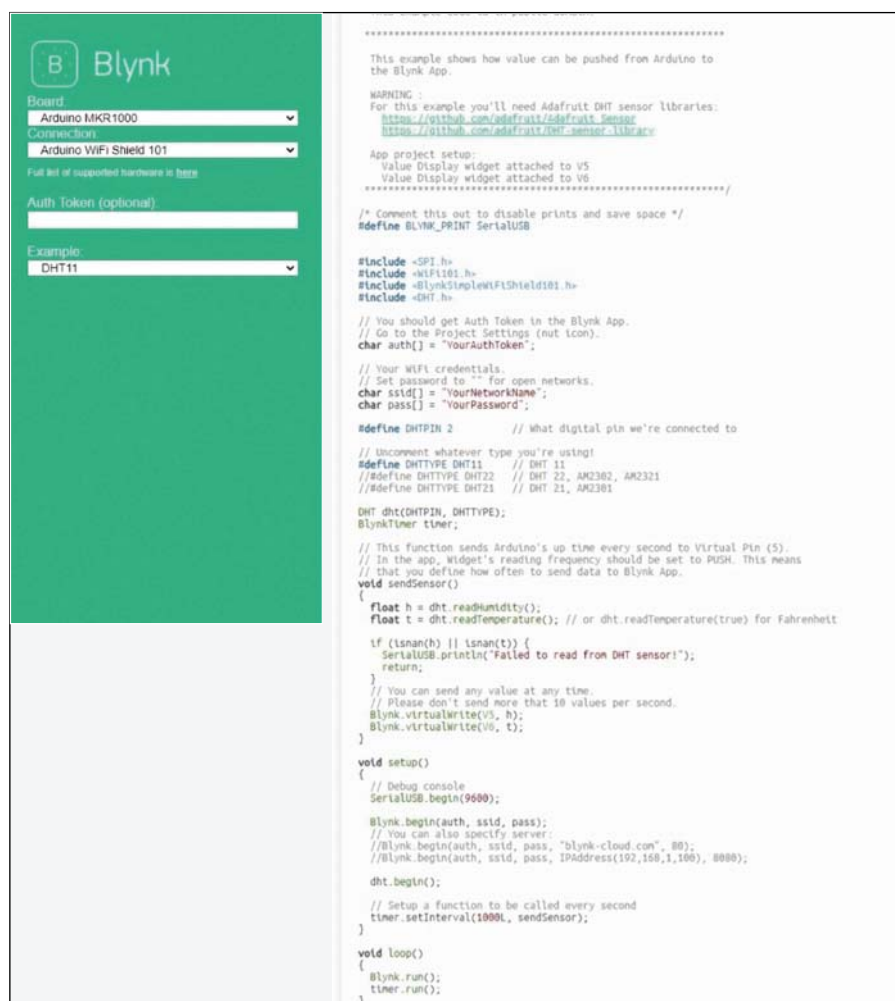
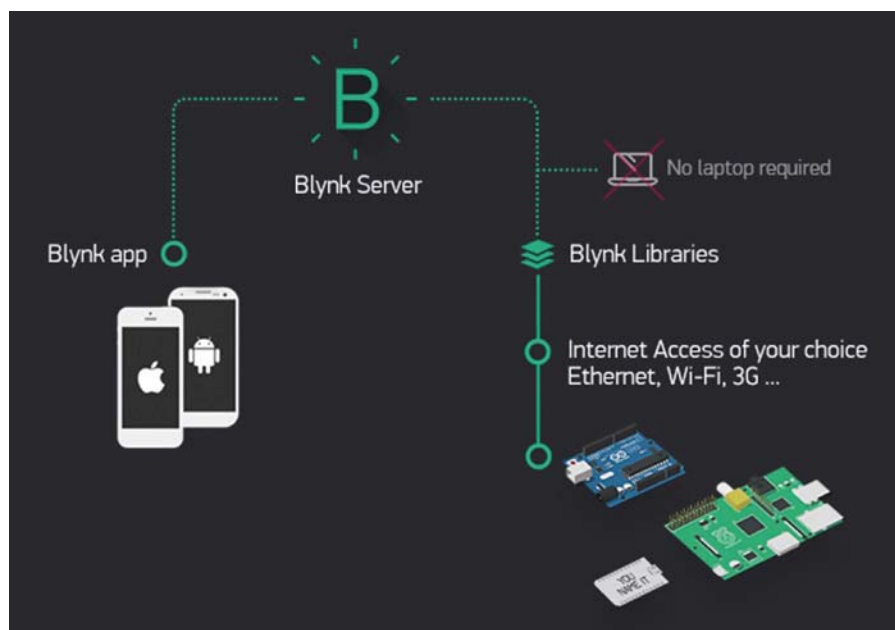


Fig. 10
Creazione listato.

Terminata l'installazione possiamo procedere a creare lo sketch da copiare nell'IDE di Arduino collegandoci al link: <https://examples.blynk.cc/>. All'interno della pagina Blynk examples nel campo Board- potrete scegliere l'hardware da utilizzare (dovrete scegliere Arduino MKR1000), mentre nel campo – Example – sceglierete il sensore o altri tool da collegare al nostro dispositivo (nel nostro caso si tratta del sensore DHT11).

Fatte le scelte appropriate all'applicazione di nostro interesse verrà automaticamente generato il listato (vedi **Listato 1**) da copiare nell'IDE di Ardui-

- **YourAuthToken:** come accennato nel capitolo CONFIGURAZIONE APPLICAZIONE la mail ricevuta con l'oggetto Auth Token for FUTURA_DHT11 project and device FUTURA_DHT11 contiene il codice (Auth Token) composto da trentadue tra numeri e lettere minuscole e maiuscole. Il codice che automaticamente si genera alla creazione della nostra applicazione è un codice identificativo necessario per connettersi all'applicazione. Ogni nuova applicazione che creerete genererà un nuovo e univoco codice.
- **YourNetworkName:** digitate le credenziali SSID



Listato 1

```

/*****
Download latest Blynk library here:
https://github.com/blynkkk/blynk-library/releases/latest

Blynk is a platform with iOS and Android apps to control
Arduino, Raspberry Pi and the likes over the Internet.
You can easily build graphic interfaces for all your
projects by simply dragging and dropping widgets.

  Downloads, docs, tutorials: http://www.blynk.cc
  Sketch generator:          http://examples.blynk.cc
  Blynk community:          http://community.blynk.cc
  Follow us:                http://www.fb.com/blynkapp
                           http://twitter.com/blynk_app

Blynk library is licensed under MIT license
This example code is in public domain.

*****/

This example shows how value can be pushed from Arduino to
the Blynk App.

WARNING :
For this example you'll need Adafruit DHT sensor libraries:
https://github.com/adafruit/Adafruit_Sensor
https://github.com/adafruit/DHT-sensor-library

App project setup:
  Value Display widget attached to V5
  Value Display widget attached to V6
*****/

/* Comment this out to disable prints and save space */
#define BLYNK_PRINT Serial

#include <SPI.h>
#include <WiFi101.h>
#include <BlynkSimpleWiFiShield101.h>
#include <DHT.h>

char auth[] = "YourAuthToken";           //digitare Auth Token ricevuto via mail

// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "YourNetworkName";
char pass[] = "YourPassword";           //digitare password WiFi

#define DHTPIN 5                         //digitare numero PIN a cui è collegato DHT11

#define DHTTYPE DHT11                    // DHT 11
// #define DHTTYPE DHT22 // DHT 22, AM2302, AM2321
// #define DHTTYPE DHT21 // DHT 21, AM2301

DHT dht(DHTPIN, DHTTYPE);
BlynkTimer timer;

// questa funzione permette di spedire i dati di Arduino ogni dieci secondi sui Virtual Pin.
// nella app, la frequenza di lettura Widget's deve essere impostata su PUSH. Questo significa
// che viene definito quanto spesso vengono spediti i dati a Blynk App.
void sendSensor()
{
  float h = dht.readHumidity();           //lettura Umidità Virtual Pin(5)
  float t = dht.readTemperature();        //lettura Temperatura Virtual Pin(6):
  digitare(true) se volete ricevere il dato in Fahrenheit

  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  // Si può spedire qualsiasi valore in qualsiasi intervallo di tempo.
  // In ogni caso è consigliato più di un valore ogni 10sec.
  Blynk.virtualWrite(V5, h);
  Blynk.virtualWrite(V6, t);

  Serial.print("Temperatura: ");           //invia la scritta "Temperatura" alla
  porta seriale                           //invia il dato della Temperatura alla
  Serial.print(t);                         //invia la scritta "°C" sulla porta seriale
  Serial.println("°C");

  Serial.print("Umidità: ");               //invia la scritta "Umidità" alla porta
  seriale                                 //invia il dato Umidità alla porta seriale
  Serial.print(h);

```



```

Serial.println("%");
}

void setup()
{
  // Debug console
  Serial.begin(9600);
  Blynk.begin(auth, ssid, pass);
  dht.begin();
  timer.setInterval(10000L, sendSensor);
  ogni 10sec.
}

void loop()
{
  Blynk.run();
  timer.run();
}

```

della vostra rete WiFi.

- **YourPassword:** digitate la password della vostra rete WiFi.
- **DHTPIN:** inserite il numero del PIN (ingresso digitale) a cui è collegato il segnale del sensore DHT11 (nel nostro caso è il PIN 5, ma potete scegliere qualsiasi ingresso della scheda Arduino MKR1000). Per i punti trattati fino ad ora potete fare riferimento alla **Fig. 11**;
- Verificate che i **Blynk.virtualWrite (V6, t)** siano gli stessi del **Listato 1**. Come si può intuire il Virtual PIN V5 corrisponde al dato Umidità, mentre il Virtual PIN V6 al dato Temperatura.

Fatte le dovute modifiche potete verificare che il firmware copiato nell'IDE di Arduino funzioni. Se non ci sono errori caricate il firmware nella scheda Arduino MKR100 tramite il collegamento con cavo USB al vostro PC e aprite il Monitor Seriale di Arduino e otterrete quanto visualizzato in **Fig. 12**. A questo punto possiamo acquisire i dati di Temperatura e Umidità visualizzati nel Monitor Seriale di Arduino sulla nostra Blynk App, cliccando sull'icona a forma di triangolo in alto a destra. L'icona si modificherà in un quadrato, indicando

così che avete effettuato correttamente il collegamento tra il vostro server Blynk App per ricevere i dati trasmessi via Internet dalla scheda Arduino MKR1000 (vedi **Fig. 13**).

CONCLUSIONI

Scopo di questa applicazione era di introdurvi nel mondo di Blynk App. A questo punto potete implementare altri widget per ottenere, per esempio, lo spegnimento e l'accensione di un LED, comandare un relè per spegnere o accendere le luci di casa o altre applicazioni; basta avere un po' di pazienza e di intraprendenza e potrete implementare molte altre interessanti applicazioni. □



Cosa occorre?

I componenti utilizzati in questo progetto sono disponibili presso Futura Elettronica. La board Arduino MKR100 (cod. MKR1000) è in vendita a Euro 45,00 mentre il modulo temperatura/umidità (cod. TEMPHUM) è disponibile a Euro 5,40. 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>



Fig. 13
Visualizzazione collegamento alla scheda Arduino MKR1000.

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.



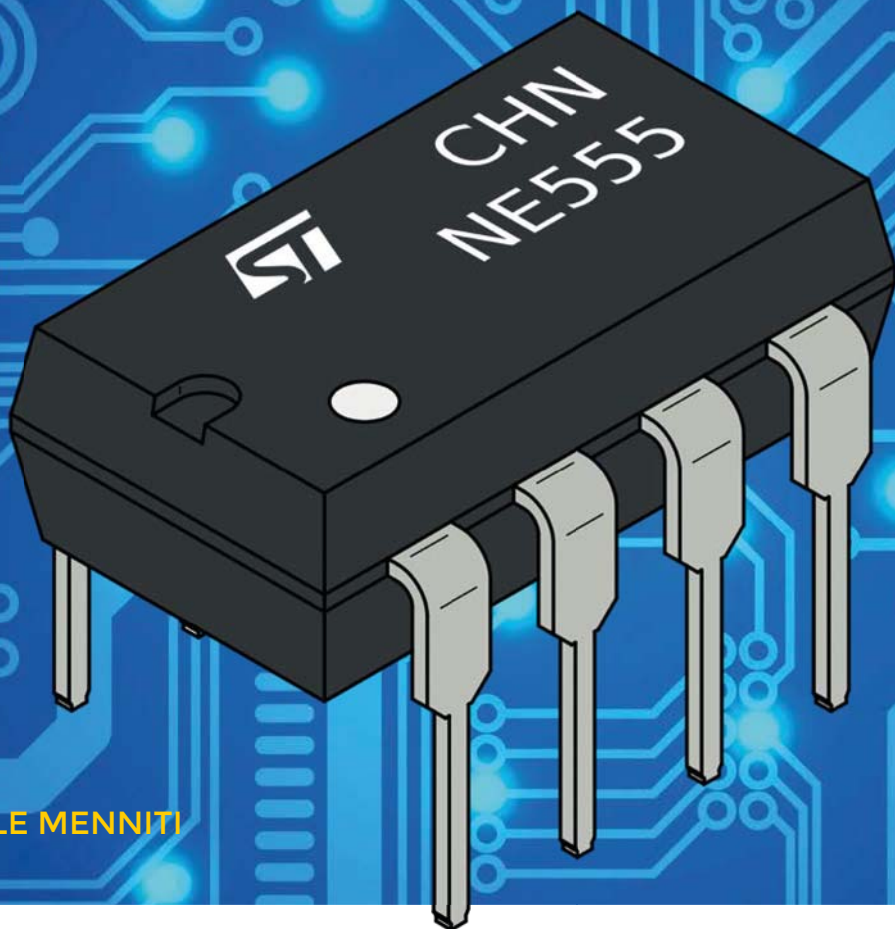
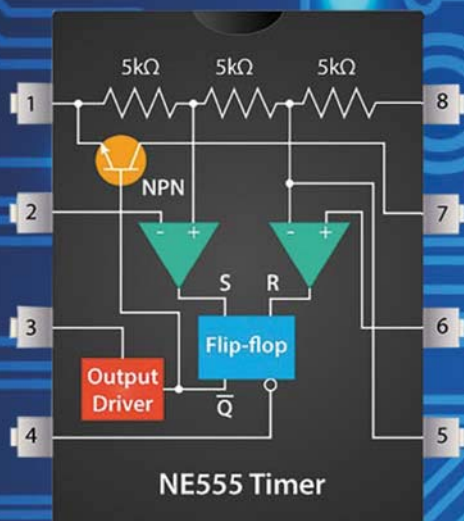
€ 19,00
Cod. FR811

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.



IL TIMER 555



del Prof. MICHELE MENNITI

Impariamo ad usare il celeberrimo integrato 555 nelle sue tre configurazioni principali: astabile, monostabile e bistabile.

Il timer 555 è un circuito integrato temporizzatore a 8 pin con più di 40 anni di progetti alle spalle. Nonostante la sua età è sempre attuale e tutt'oggi trova spazio in moltissime applicazioni, anche grazie alla sua semplicità d'uso. Il famoso 555 ha anche un fratello maggiore, il 556 che integra, all'interno del suo case, due timer.

In **Fig. 1** abbiamo riportato i principali Costruttori di questo circuito integrato e le relative sigle, alcuni non lo producono più, altri lo hanno addirittura rinnovato potenziandolo in alcune caratteristiche, come l'aumento della frequenza massima e la diminuzione dei consumi.

Il suo nome deriva dal valore dei tre resistori interni collegati ai due comparatori, che sono tutti da 5 kΩ. A seconda dei modelli può operare con tensioni tra 1,5 V e 18 V, con frequenze da 100 kHz a 3 MHz e può erogare correnti massime tra 50 mA e 200 mA. In **Fig. 2** abbiamo illustrato il suo schema interno funzionale e la piedinatura.

Fig. 1
Tabella Costruttori
e sigle del 555.

Costruttore	Sigla
Fairchild	NE555
Motorola	MC14555
National	LM555
RCA	CA555
Texas Instr.	LM555
Exar	XR555
Intersil	NE555

Esso è costituito fondamentalmente da due comparatori, un LATCH ASINCRONO SET-RESET, un buffer negato ed un transistor NPN (nei modelli più recenti è un mosfet a canale N) in configurazione interruttore. I tre resistori uguali da 5k costituiscono un partitore di tensione che ha lo scopo di fornire le tensioni di riferimento ai comparatori di SOGLIA (2/3 di VCC) e di TRIGGER (1/3 di VCC). Le uscite dei due comparatori sono applicate in ingresso ad un LATCH ASINCRONO DI TIPO SET-RESET, che è un circuito logico con due ingressi (S e R) e due uscite (Q e Q'); nel 555 è utilizzata solo l'uscita Q' (Q negato), che è collegata ad un transistor NPN e ad una porta NOT che funge da buffer invertente. In pratica se l'uscita Q' si trova a livello L (0 V) il transistor si trova in OFF

ed il pin 7 risulta flottante, mentre la porta NOT inverte lo stato di Q' ed il pin 3 (uscita del 555) si trova a livello H (VCC). Se invece l'uscita Q' si trova a livello H, il transistor va in ON ed il pin 7 viene collegato a massa, mentre la porta NOT, invertendo lo stato di Q' porta il pin 3 a livello L (0 V).

Il comparatore inferiore ha l'ingresso non invertente collegato a 1/3 VCC; quando la tensione sul pin di TRIGGER (pin 2) è maggiore di 1/3 VCC l'uscita di questo comparatore (ingresso S del LATCH) si porta a livello logico basso (L); se invece la tensione sul pin di TRIGGER è inferiore a 1/3 VCC l'uscita del secondo comparatore si porta a livello alto (H).

Il comparatore superiore ha l'ingresso invertente collegato a 2/3 VCC, questa tensione di riferimento è misurabile esternamente sul pin 5.

Quando la tensione sul pin di SOGLIA (6) è maggiore di 2/3 VCC l'uscita di questo comparatore (ingresso R del LATCH) si porta a livello logico alto (H); se invece la tensione sul pin di SOGLIA è minore di 2/3 VCC l'uscita del primo comparatore si porta a livello basso (L). Vediamo ora come vengono gestiti i pin 3 (USCITA) e 7 (SCARICA), in relazione allo stato degli ingressi SOGLIA (pin 6) e TRIGGER (pin 2). Per semplicità abbiamo riportato tale comportamento nella tabella visibile in **Fig. 3**. Come si può notare, il pin 4 (RESET del 555) se posto a livello logico basso, fissa l'USCITA (pin 3) su L e le tensioni o segnali applicati sui pin SOGLIA e TRIGGER vengono ignorati, quindi questo pin può essere utilizzato per bloccare temporaneamente l'attività del 555. Se invece viene fissato su H (collegandolo all'alimentazione del circuito) abbiamo tre possibili condizioni:

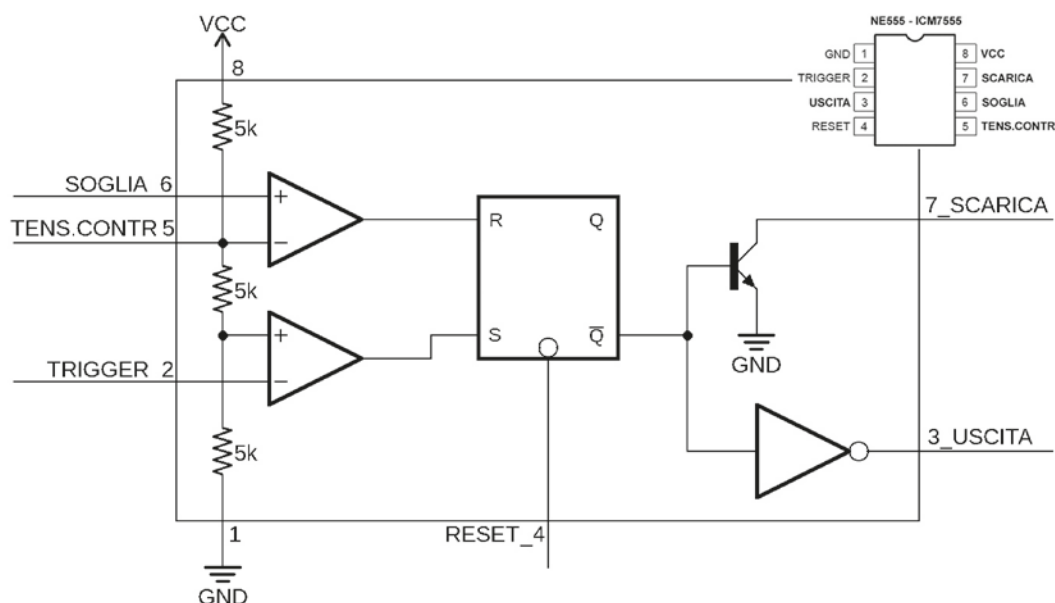


Fig. 2
Schema a
blocchi e
piedinatura
del 555.

RESET 4	TRIGGER 2	SOGLIA 6	S	R	Q'	SCARICA 7	USCITA 3
L	Irrilevante	Irrilevante	x	x	H	ON	L
H	$< 1/3 VCC$	Irrilevante	H	x	L	OFF	H
H	$> 1/3 VCC$	$> 2/3 VCC$	L	H	H	ON	L
H	$> 1/3 VCC$	$< 2/3 VCC$	L	L	non cambia	non cambia	non cambia

Fig. 3
Tabella
funzionale
del 555.

1. TRIGGER $< 1/3 VCC$: SCARICA OFF (quindi scollegata dalla massa) e USCITA H (poco meno di VCC), il livello di tensione applicato sul pin SOGLIA è irrilevante;
2. TRIGGER $> 1/3 VCC$ e SOGLIA $> 2/3 VCC$: SCARICA ON (quindi collegata a massa) e USCITA L (circa 0 V);
3. TRIGGER $> 1/3 VCC$ e SOGLIA $< 2/3 VCC$: SCARICA e USCITA non cambiano lo stato attuale.

Quindi, ovviamente, il comportamento dei due pin di uscita del 555 dipende dallo stato dei pin di ingresso. Le tre colonne centrali della tabella (S, R e Q') rappresentano il comportamento del solo LATCH interno, secondo la tipica tabella delle transizioni. I riferimenti del comparatore sono espressi in frazioni di VCC, in quanto l'alimentazione, come abbiamo detto, ha un elevato range di valori e quindi questi riferimenti variano al variare dell'alimentazione. In genere i riferimenti di tensione dei tre resistori vanno bene per qualsiasi applicazione, in tal caso conviene applicare al pin 5 (TENSIONE CONTROLLO) un condensatore da 10 nF verso massa, in modo da bloccare eventuali disturbi circuitali che potrebbero influenzare le tre tensioni di riferimento. In caso di particolari necessità si può collegare a questo pin un resistore verso il positivo di alimentazione o verso massa per ottenere rispettivamente tensioni di riferimento maggiori o minori per i due comparatori. Il timer 555 può essere utilizzato come multivibratore astabile, monostabile o bistabile. Vediamo assieme le diverse configurazioni di base.

MULTIVIBRATORE ASTABILE

La **Fig. 4** mostra la tipologia base del multivibratore astabile, in cui i tre componenti fondamentali sono i due resistori RA e RB ed il condensatore CX; il C1 (10 nF) è il filtro antidisturbo di cui abbiamo già parlato, mentre i resistori R1, R2 ed i LED LD1, LD2 servono per visualizzare il comportamento del circuito, in mancanza di un oscilloscopio. Infatti LD1 (rosso) si accende quando l'USCITA del 555 è in fase L, mentre LD2 (blu) si accende quando l'USCITA del 555 è in fase H. Da notare che i due ingressi del comparatore, TRIGGER e SOGLIA, sono collegati assieme, e ciò comporta che siano costantemente allo

stesso livello di tensione, durante il funzionamento, per cui si possono verificare solo le condizioni 1 e 2, descritte in precedenza. Al momento dell'accensione del circuito CX è scarico, quindi il TRIGGER si trova ad un livello inferiore ad $1/3 VCC$ e, come visto, l'USCITA è alta (H), con conseguente accensione di LD2. Poiché contemporaneamente il pin SCARICA è scollegato dalla massa, inizia la carica del condensatore CX, mediante la corrente che passa attraverso RA ed RB, e la tensione sugli ingressi inizia gradualmente a salire; quando essa supera i $2/3 VCC$ l'USCITA diventa bassa (L), con conseguente spegnimento di LD2 ed accensione di LD1; inoltre ora il pin SCARICA viene collegato alla massa e CX inizia a scaricarsi attraverso la sola resistenza RB, e la tensione sugli ingressi ricomincia a diminuire. Quando essa scende nuovamente sotto $1/3 VCC$ l'uscita ritorna alta (H), per cui è evidente che si ha un funzionamento a ciclo continuo, con alternanza degli stati alto e basso dell'USCITA; quindi le due condizioni sono instabili, ed è questa la ragione per la quale questo tipo di configurazione viene definita ASTABILE. Questo significa che ci troviamo in presenza di un circuito oscillatore con due stadi H-L ben definiti, e quindi parliamo di un'onda quadra. A questo punto bisogna capire come variano la durata del ciclo H-L ed il rapporto tra le due fasi H e L, nel primo caso si tratta della frequenza del segnale in uscita, nel secondo del suo Duty Cycle, definiremo le due fasi t_H e t_L .

Possiamo ricavare le formule per il calcolo di questi

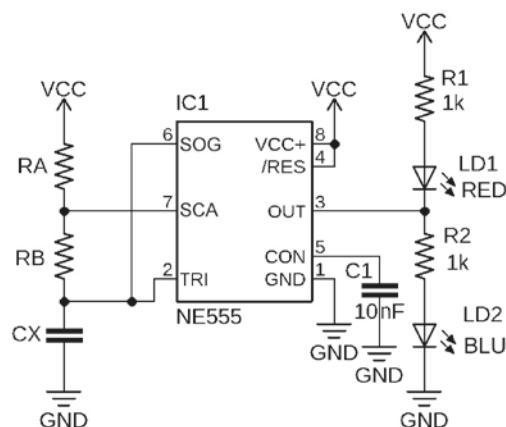
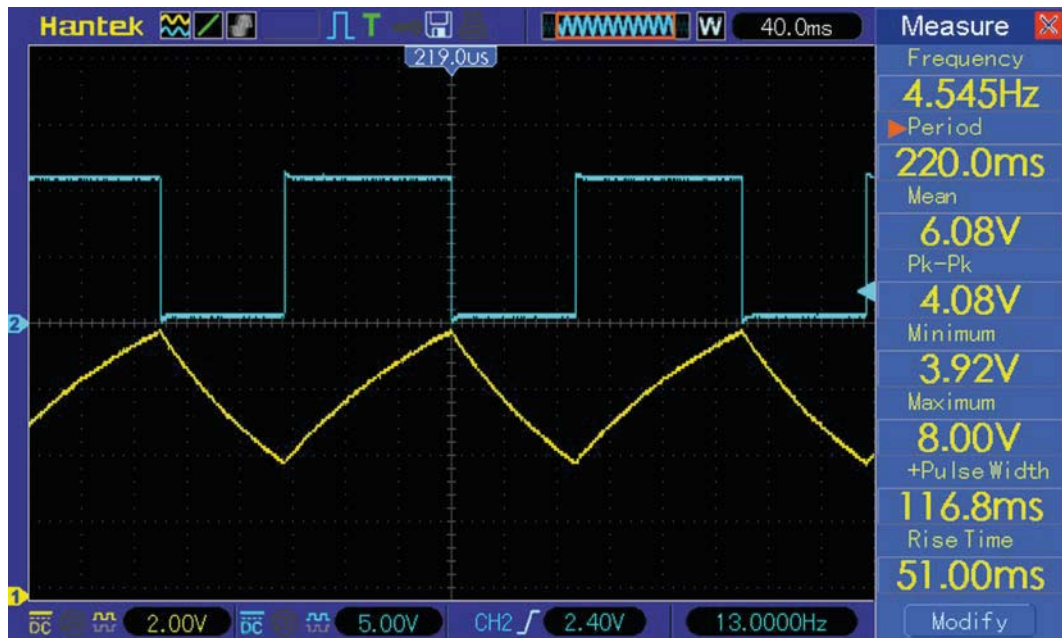


Fig. 4
Multivibratore
astabile con
555.

Fig. 5
Corrispondenza
tra segnale di
uscita e carica del
condensatore CX.



valori direttamente dal datasheet dell'integrato,
reperibile al link www.ti.com/lit/ds/symlink/lm555.pdf
 $t_H = 0,693 * (R_A + R_B) * C_X$
 $t_L = 0,693 * (R_B) * C_X$

Risulta evidente la correlazione tra queste due
formule e quanto affermato in precedenza, cioè
che la fase H dipende dalla carica del condensatore
CX attraverso i resistori RA e RB, mentre la fase L
dipende dalla scarica del condensatore CX attra-
verso il solo resistore RB. Queste relazioni sono
chiaramente visibili in **Fig. 5** in cui l'onda quadra è
prelevata sul pin USCITA e quella pseudotriangolare
sui pin SOGLIA-TRIGGER; quest'ultima rappresenta
appunto le fasi di carica (ascendente) e di scarica

(discendente) del condensatore.

Il periodo **T** del segnale generato (ciclo H-L) è dato
dalla somma delle due fasi, quindi:

$$T = t_H + t_L = 0,693 * (R_A + R_B) * C_X + 0,693 * R_B * C_X = 0,693 * (R_A + 2R_B) * C_X$$

La frequenza **f** è l'inverso del periodo quindi:

$$f = \frac{1}{t_H + t_L} = \frac{1}{0,693 * (R_A + 2R_B) * C_X} = \frac{1,44}{(R_A + 2R_B) * C_X}$$

Il Duty Cycle è dato dal rapporto percentuale tra **t_H**
e **t_H + t_L**, cioè tra la fase H e l'intero ciclo del segnale
(**T**), quindi:

$$D.C.\% = \frac{t_H}{t_H + t_L} = \left(1 - \frac{R_B}{R_A + 2R_B}\right) * 100$$

In condizioni normali il **t_H** è sempre più lungo del **t_L**
e l'onda non sarà mai perfettamente simmetrica;
come vedremo più avanti, un particolare tipo di
configurazione monostabile consente di ottenere la
condizione di Duty Cycle al 50%, mentre in un pros-
simo articolo vedremo anche una configurazione
astabile per ottenere lo stesso risultato.

Da quanto visto finora è chiaro che se in un circuito
basato sul 555 si conoscono i valori dei tre compo-
nenti passivi RA, RB e CX, è facile ricavare periodo,
frequenza e Duty Cycle. Se invece bisogna realiz-
zare un circuito ex-novo, per ottenere la frequenza
ed il Duty Cycle desiderati, occorre operare con le
formule inverse. Per esempio per ottenere un'onda
quadra con frequenza di 100 Hz e Duty Cycle al 55%,
la prima operazione da fare è stabilire la capacità

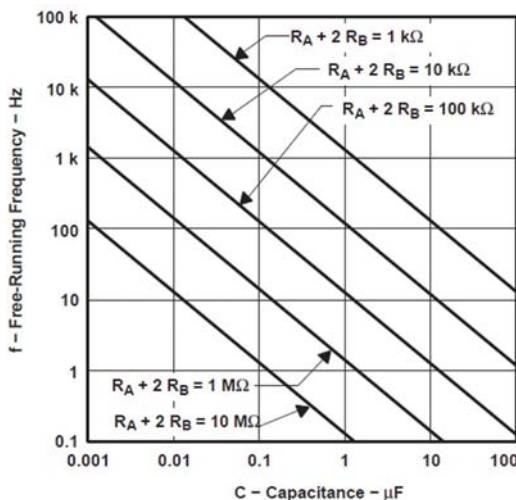


Fig. 6
Il grafico per
il calcolo di RA
ed RB.

CX da adottare, un qualsiasi valore da 1 nF a 100 µF. In linea di massima per la scelta dei componenti è consigliabile far riferimento al grafico presente sul datasheet, che abbiamo riportato in **Fig. 6**.

Il grafico presenta sull'asse X il range di valori per la capacità di CX (da 1 nF a 100 µF), sull'asse Y il range della frequenza generabile dal modello di 555 a cui si riferisce il datasheet (nel caso del modello base varia da 0,1 Hz a 100 kHz); vi sono poi cinque linee oblique, parallele tra loro, ognuna delle quali rappresenta una combinazione RA+2RB (da 1 kΩ a 10 MΩ). Vediamo come si usa questo grafico: una volta stabilita la frequenza da ottenere (nel nostro caso 100 Hz), è consigliabile scegliere una capacità che coincida con la linea della frequenza nell'area centrale del grafico (in questo caso tra 100 nF e 1 µF, scegliamo 1 µF), infine si localizza la linea obliqua più vicina al punto di incrocio tra la linea orizzontale della frequenza e quella verticale della capacità, in questo caso la RA+2RB = 10 kΩ.

Nell'esecuzione delle formule bisogna tener presente che la capacità va sempre espressa in µF (p.es. 100 nF = 0,1 µF) e la resistenza va sempre espressa in kΩ. Il tempo totale **T** del ciclo è dato dall'inverso della frequenza, quindi:

$$T = \frac{1}{100} = 0,01s$$

il t_H si ricava dalla formula del Duty Cycle, quindi:
 $t_H = D.C. \% * T = 55/100 * 0,01 = 0,0055s$

Il t_L si ricava per differenza: $t_L = T - t_H = 0,01 - 0,0055 = 0,0045s$.

Ora che abbiamo tutte le tempistiche, diventa abbastanza semplice ricavare i valori dei due resistori, per comodità trasformiamo i periodi in ms, quindi: $T = 10ms, t_H = 5,5ms, t_L = 4,5ms$. Iniziamo da RB, ricavabile dalla formula del t_L :

$$t_L: RB = \frac{t_L}{0,693 * CX} = \frac{4,5}{0,693 * 1} = 6,49k\Omega.$$

Ora che conosciamo RB possiamo calcolare RA a partire da t_H :

$$t_H: RA = \frac{t_H}{(0,693 * CX) - RB} = \frac{5,5}{(0,693 * 1) - 6,49} = 1,44k\Omega.$$

Eseguendo la formula $RA + 2RB = 1,44 + (2 * 6,49) = 14,42$, possiamo notare che il risultato è compatibile con l'intersezione delle tre linee del grafico.

Naturalmente queste formule spessissimo danno come risultato valori di RA ed RB difficilmente reperibili in commercio, come in questo caso; inoltre c'è da considerare che i condensatori in genere hanno tolleranze elevate, dell'ordine del 5-10-20%, quindi

non sarà mai facile ottenere nel circuito i valori ipotizzati. Vediamo cosa succede simulando una tolleranza del condensatore del -5% e adottando per RA un valore commerciale di 6,2 kΩ 1% e per RB 1,5 kΩ 1%. Li inseriamo nelle formule iniziali ed otterremo: Frequenza 109,3 Hz (100), T=9,15 ms (10), $t_H=5,07$ ms (5,50), $t_L=4,08$ ms (4,50), D.C.=55,4% (55); i valori tra parentesi sono quelli teorici, quindi possiamo notare un discreto discostamento; se è accettabile per i nostri scopi va bene così, in caso contrario bisogna procedere in modo diverso, misurando la capacità del condensatore con un buon capacimetro e ricalcolando i resistori con la seconda serie di formule; a quel punto occorrerà ottenere i valori ricavati dalle formule mediante l'uso di più resistori in serie/parallelo, oppure sostituire RA ed RB con due trimmer di precisione, configurati come reostati, e regolarli in modo da ottenere i valori desiderati. Torniamo al nostro circuito valutando la prima ipotesi. Per ottenere una frequenza da 100 Hz ed un D.C. del 55%, se il CX risulta avere una capacità di 0,95 µF, le formule ci restituiranno: RA=1,52 kΩ ed RB=6,84 kΩ. Nel primo caso ci servono un resistore da 1,5 kΩ ed uno da 22 Ω da mettere in serie, ottenendo 1,522 kΩ, nel secondo caso possiamo mettere in serie un resistore da 6,8 kΩ ed uno da 39 Ω, ottenendo 6,839

CALCOLO DI PERIODI, FREQUENZA E DUTY CYCLE (DEVONO ESSERE NOTI RA, RB, CX)

Fase H del segnale di USCITA

$$t_H = 0,693 * (RA + RB) * CX$$

Fase L del segnale di USCITA

$$t_L = 0,693 * (RB) * CX$$

Periodo totale T del segnale di USCITA

$$T = t_H + t_L = 0,693 * (RA + RB) * CX + 0,693 * RB * CX = 0,693 * (RA + 2RB) * CX$$

Frequenza del segnale di USCITA

$$f = \frac{1}{T} = \frac{1}{t_H + t_L} = \frac{1}{0,693 * (RA + 2RB) * CX} = \frac{1,44}{(RA + RB) * CX}$$

Duty Cycle del segnale di USCITA

$$D.C. \% = \frac{t_H}{T} = \frac{t_H}{t_H + t_L} = \left(1 - \frac{RB}{RA + 2RB}\right) * 100$$

Calcolo di RA e RB (devono essere noti CX ed i periodi H e L del segnale di USCITA)

$$RB = \frac{t_L}{0,693 * CX}$$

$$RA = \frac{t_H}{(0,693 * CX) - RB}$$

Tabella 1
Formule
per i circuiti
astabili con
555.



Fig. 7
Calcolatore
per i circuiti
astabili con 555

	A	B	C	D	E	F	G	H	I
1									
2									
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									
13									
14									
15									
16									

Formule per circuiti astabili con 555

Istruzioni: inserire i valori richiesti nei campi con sfondo bianco

I risultati appariranno automaticamente in quelli con sfondo grigio

Calcolo periodi, Freq. e DC			Calcolo RA e RB		
RA kΩ	RB kΩ	CX μF	RA kΩ	RB kΩ	CX μF
1,44	6,49	1	1,44	6,49	1
Periodo H	5,50 ms		Periodo H	5,50 ms	
Periodo L	4,50 ms		Periodo L	4,50 ms	
Periodo T	9,99 ms		Periodo T	10,00 ms	
Frequenza	100,07 Hz		Frequenza	100,000 Hz	
D.C.	55,0 %		D.C.	55,0 %	
			CX	1 μF	

kΩ. Tutti i resistori dovranno essere di precisione con l'1% di tolleranza (banda finale marrone), meglio ancora se è lo 0,5% (banda finale verde).

Se andiamo a verificare ora questi ultimi valori con la prima serie di formule otteniamo: Frequenza 99,93 Hz (100), T=10,00 ms (10), $t_H=5,50$ ms (5,50), $t_L=4,50$ ms (4,50), D.C.=55,0% (55); in pratica ci mancano solo 70 mHz di frequenza, mentre tutti gli altri valori combaciano alla perfezione, certamente una condizione eccellente.

Per comodità riepiloghiamo nella **Tabella 1** tutte le formule utili ai calcoli per i circuiti astabili con 555. Comunque il fatto di saper ora effettuare i necessari calcoli per i nostri circuiti non significa che non possiamo agevolarci la vita; allo scopo abbiamo preparato un file di Excel, con tutte le formule preimpostate, in **Fig. 7** sono visibili i riquadri con le

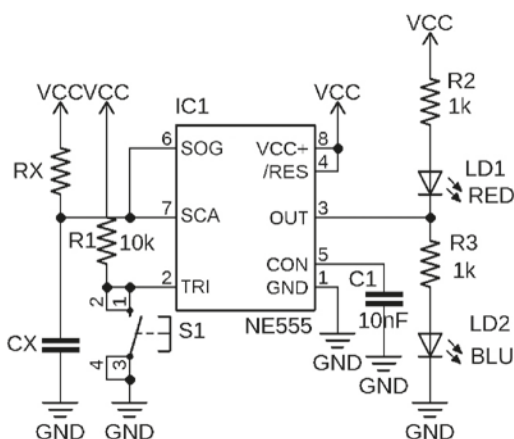
formule per i circuiti astabili.

Il file è scaricabile nella pagina di questa rivista sul sito www.elettronica.in.it.

MULTIVIBRATORE MONOSTABILE

La **Fig. 8** mostra la tipologia base del multivibratore monostabile, in cui i due componenti fondamentali sono il resistore RX, il condensatore CX e lo switch S1; il C1 (10 nF) è il solito filtro antidisturbo, il resistore R1 ha la funzione di pullup, mentre i resistori R2 e R3 ed i LED LD1 e LD2 servono per visualizzare il comportamento del circuito, come nella situazione precedente. Da notare che in questo caso l'ingresso SOGLIA è scollegato dal TRIGGER ed è invece collegato al pin SCARICA. Al momento dell'accensione il TRIGGER si trova ad un livello maggiore di 1/3 VCC a causa del resistore di pullup e, come visto, l'USCITA è bassa (L), con conseguente accensione di LD1. Contestualmente il pin SCARICA è collegato a massa, mantiene a livello L anche il pin SOGLIA ed impedisce al condensatore CX di caricarsi, in quanto la corrente che attraversa R1 finisce direttamente a massa. Nel momento in cui si preme il pulsante S1, il TRIGGER viene cortocircuitato a massa, quindi va ad un livello inferiore a 1/3 VCC, l'USCITA diventa alta (H), si spegne LD1 e si accende LD2; il pin SCARICA viene scollegato dalla massa e la corrente che attraversa RX inizia a caricare il condensatore CX e la tensione sulla SOGLIA inizia gradualmente a salire. Rilasciando il pulsante il TRIGGER ritorna a livello maggiore di 1/3 VCC. Durante tutta la fase di carica l'USCITA si mantiene alta, in quanto, come

Fig. 8
Multivibratore
monostabile
con 555.



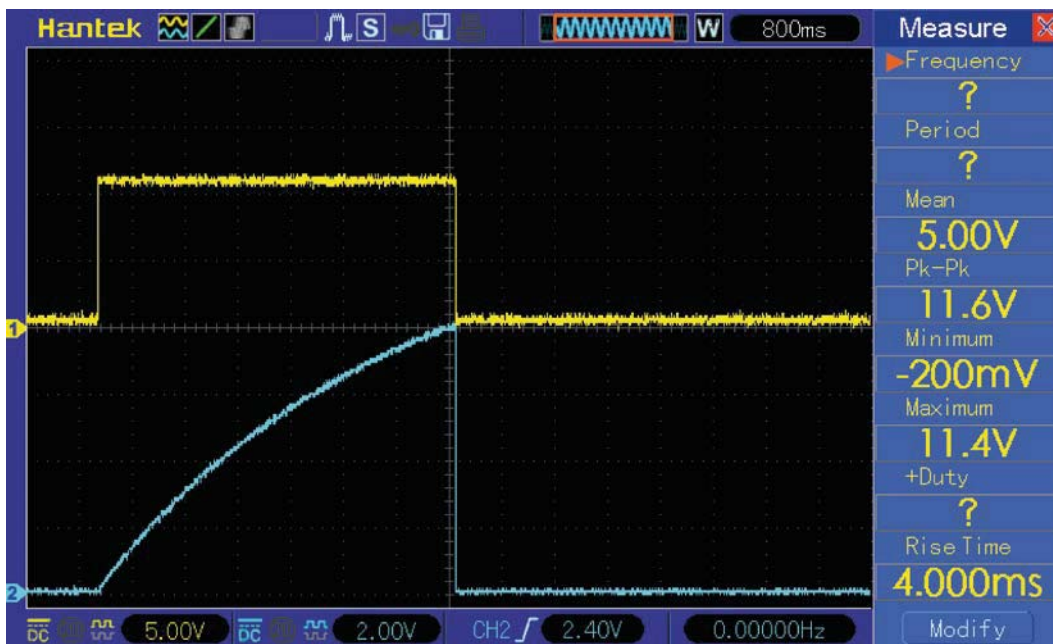


Fig. 9
La larghezza dell'impulso H in funzione della carica del condensatore CX.

abbiamo visto, la combinazione TRIGGER > 1/3 VCC e SOGLIA < 2/3 VCC non provoca modifiche di stato del LATCH. Nel momento in cui il condensatore è abbastanza carico da avere ai suoi capi (e quindi sulla SOGLIA) una tensione maggiore di 2/3 VCC, avviene la nuova commutazione dell'USCITA a livello basso (L) ed il condensatore si scarica istantaneamente attraverso il pin SCARICA, ora collegato nuovamente a massa; tale condizione resterà stabile fino ad una nuova pressione del pulsante. Riepilogando, mentre lo stato di USCITA bassa è stabile, quello di USCITA alta è instabile, in quanto dipendente dalla costante di tempo risultante dai valori di RX e CX. Ecco la ragione per la quale questo tipo di configurazione viene definita MONOSTABILE.

Naturalmente in questo caso non abbiamo da calcolare i valori di un segnale, in quanto esso si riduce ad un solo periodo H (t_H) dipendente dalla formula: $t_H = 1,1 \cdot RX \cdot CX$. La Fig. 9 illustra appunto la larghezza dell'impulso H in funzione dell'andamento della carica del condensatore. La differenza che salta subito all'occhio, rispetto alla configurazione astabile, è l'immediata scarica del condensatore nel momento in cui avviene il cambio di stato dell'USCITA; ciò è naturalmente dovuto al fatto che il condensatore è direttamente collegato al pin SCARICA e quindi viene cortocircuitato a massa senza limitazioni di corrente. Anche in questo caso, se ci si trova di fronte ad un circuito già realizzato, basta inserire nella formula i valori dei due componenti RX e CX; se invece siamo in fase di progettazione, occorre stabilire a priori il valore di uno dei due componenti (in genere

il CX) e poi ricavarsi l'altro con la formula inversa. Un esempio chiarirà sicuramente le idee. Supponiamo di avere uno schema monostabile con $R_X = 47 \text{ k}\Omega$ e $C_X = 100 \text{ }\mu\text{F}$, la formula ci dice che la durata dell'impulso è: $t_H = 1,1 \cdot RX \cdot CX = 1,1 \cdot 47 \cdot 100 = 5170 \text{ ms}$ (5,17 s). Se invece avessimo necessità di realizzare un circuito che fornisca un impulso di 50 ms, al solito dovremo scegliere la capacità basandoci sul grafico fornito dal datasheet, che è diverso da quello visto per i circuiti astabili. Infatti in Fig. 10 possiamo notare il consueto asse X con le capacità, mentre l'asse Y riporta il periodo dell'impulso e non la frequenza del segnale, come nel grafico precedente. Il range del periodo va da 10^{-5} s (10 μs) a 10 s; le linee in questo caso riguardano la sola RX, sono sempre

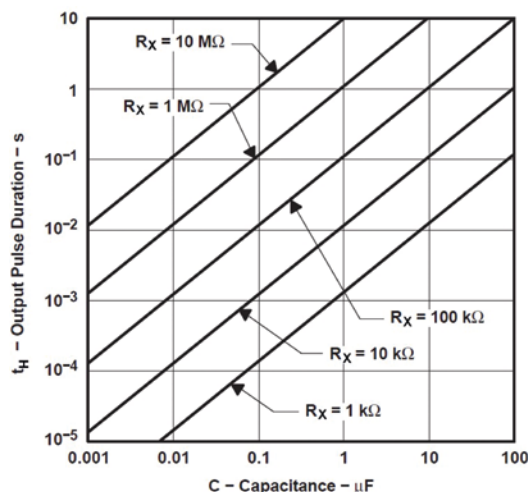
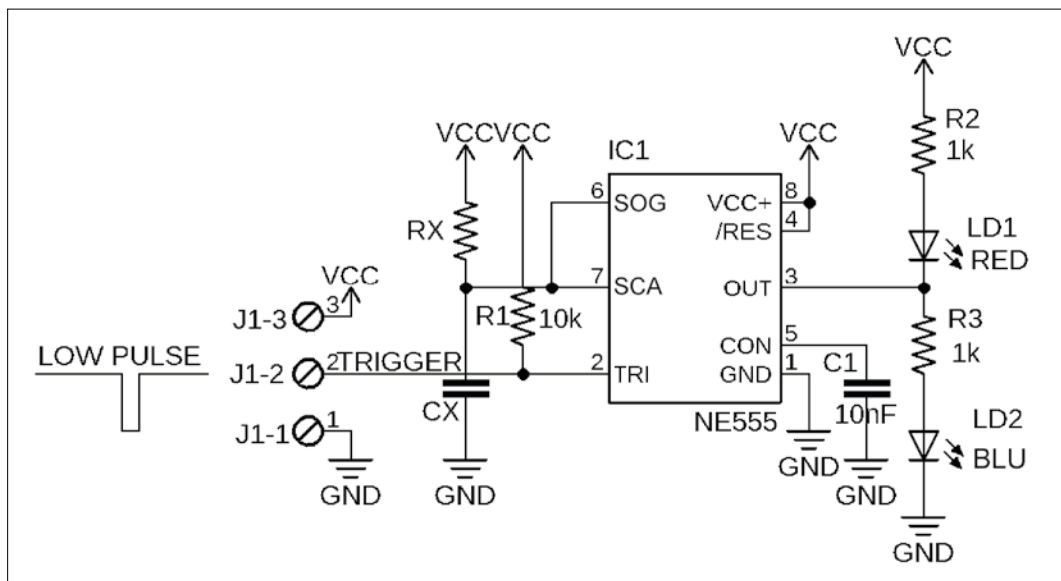


Fig. 10
Il grafico per il calcolo di RX.

Fig. 11
Circuito
monostabile
con impulso di
TRIGGER esterno.



oblique ma in direzione opposta alla precedente. Scegliamo un condensatore da 100 nF (0,1 μ F), a questo punto possiamo usare la nostra formula per ricavarci il valore di RX:

$$RX = \frac{t_H}{1,1 \cdot CX} = \frac{50}{1,1 \cdot 0,1} = 455k\Omega,$$

tale valore è compatibile con il grafico, infatti l'intersezione tra la linea verticale della capacità scelta con la riga virtuale collocata a metà tra 10^{-2} (10 ns) e 10^{-1} (100 ns) si localizza a circa metà strada tra la linea obliqua 1 M Ω e quella 100 k Ω . Naturalmente per la questione valori commerciali e precisione vale

quanto spiegato nella sezione dedicata al circuito astabile. Finora al pin di TRIGGER abbiamo fornito un impulso negativo mediante un pulsante collegato a massa, ma questo circuito potrebbe tornare utile anche se si dispone di un altro circuito in grado di generare un impulso (**Fig. 11**). Affinché l'impulso venga rilevato dal 555 è necessario che sia largo almeno 10 μ s. Con l'aiuto di un Generatore di Funzioni abbiamo creato un segnale costituito da un impulso L di circa 60 ms ed un periodo totale di circa 500 ms tra un impulso ed il successivo.

In questo caso l'eccitazione del circuito ha un andamento regolare (**Fig. 12**), ma se pensiamo p.es.

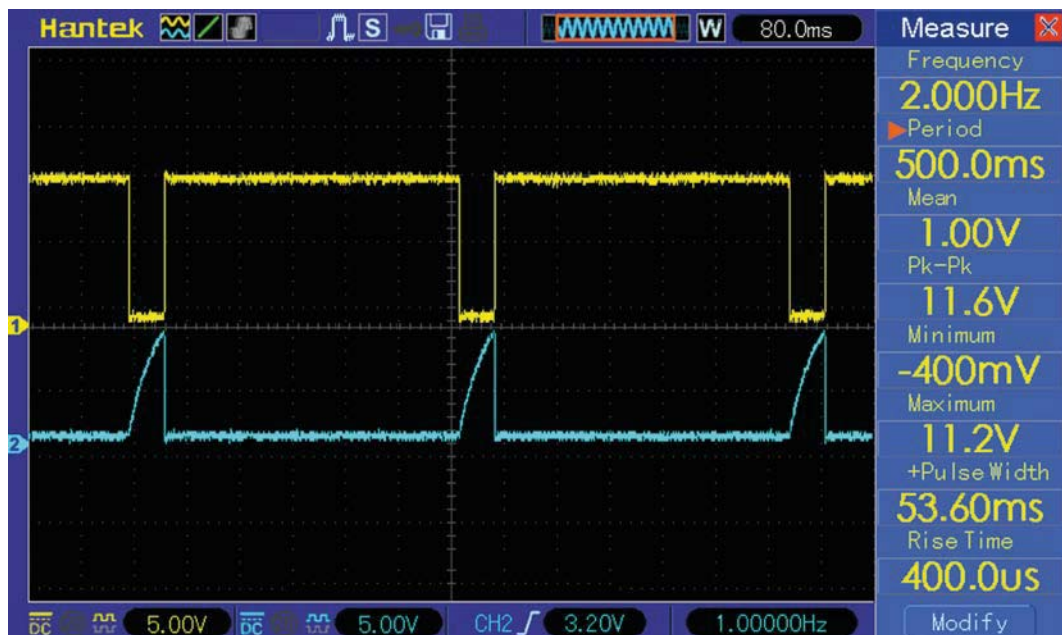


Fig. 12
Simulazione di
un impulso di
TRIGGER esterno.

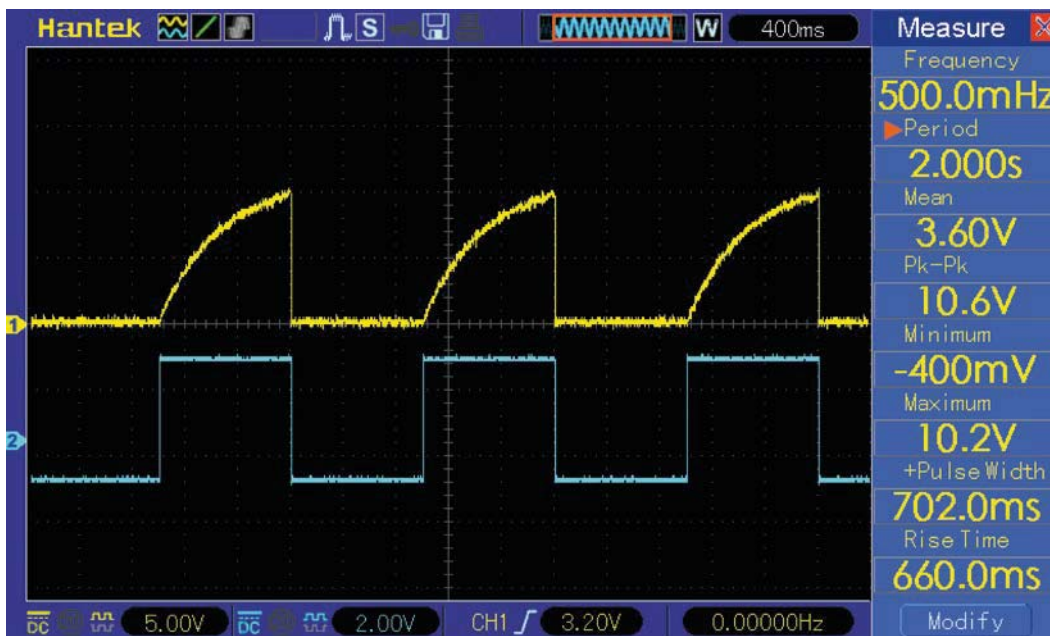


Fig. 13
Tecnica per
generare un
D.C. del 50%.

ad un circuito che genera questo impulso negativo solo al verificarsi di una data condizione, solo in quel momento avverrà il cambio di stato dell'USCITA del 555. Una caratteristica importante del circuito monostabile è data dal fatto che, una volta che il TRIGGER riceve un impulso valido e porta l'USCITA allo stato alto H, tutti gli altri eventuali impulsi che arrivano al TRIGGER durante la carica del condensatore, e quindi della fase H, saranno ignorati; questa cosa garantisce che la durata dell'impulso sia

sempre uguale.

Con la tecnica dell'impulso negativo applicato esternamente, se si ricorre ad un clock, cioè ad un segnale a frequenza costante, è possibile utilizzare un circuito monostabile come astabile, ed in tal caso si può superare la limitazione del Duty Cycle maggiore del 50%.

In questo esempio realizziamo un generatore di onda quadra con frequenza 1 Hz e D.C. al 50%. Prima di tutto va chiarita una questione importan-

	A	B	C	D	E	F	G	H
1								
2		Formule per circuiti monostabili con 555						
3		Istruzioni: inserire i valori richiesti nei campi con sfondo bianco						
4		I risultati appariranno automaticamente in quelli con sfondo grigio						
5								
6				RX kΩ	CX μF	t _H ms		
7		Calcolo periodo		47	100	5170		
8								
9				t _H ms	CX μF	RX kΩ		
10		Calcolo resistore		5170	100	47		
11								
12				t _H ms	RX kΩ	CX μF		
13		Calcolo condensatore		5170	47	100		
14								

Fig. 14
Calcolatore
per i circuiti
monostabili
con 555.

te, la formula per il circuito monostabile calcola la larghezza dell'impulso H, mentre un'onda quadra ha un ciclo composto anche dalla fase L; nel nostro calcolo dobbiamo tener conto di questa cosa, altrimenti non otterremo il risultato desiderato. Una frequenza da 1 Hz e D.C. al 50% deve essere costituita da una fase H ed una fase L di pari durata; poiché il periodo corrispondente alla frequenza di 1 Hz è di 1000 ms, va diviso equamente fra H e L.

Per ottenere un impulso t_H di circa 500 ms ricorriamo ad un resistore da 1 M Ω per RX ed un condensatore da 470 nF per CX. Per ottenere la fase L identica alla fase H bisogna applicare al TRIGGER un clock con frequenza pari al 50% di quella impostata per il monostabile, quindi 500 mHz.

La **Fig. 13** dimostra come il clock (onda quadra in basso) forzi la fase di carica di CX (segnale in alto) ad intervalli regolari, e che la fase di carica e quella in cui il condensatore è scarico sono identiche.

Come per i circuiti astabili, anche in questo caso abbiamo predisposto un foglio Excel con le tre formule di calcolo utili per questa configurazione (**Fig. 14**).

MULTIVIBRATORE BISTABILE CON TIMER 555

La **Fig. 15** mostra la tipologia base del multivibratore bistabile, in cui i due componenti fondamentali sono gli switch S1 e S2; il C1 (10 nF) è il solito filtro antidisturbo, i resistori R1 e R2 hanno la funzione di pullup, mentre i resistori R3 e R4 ed i LED LD1 e LD2 servono per visualizzare il comportamento del circuito, come nelle situazioni precedenti.

Anche in questo caso l'ingresso SOGLIA è scollegato dal TRIGGER, ma questa volta il pin SCARICA resta inutilizzato, poiché il circuito non prevede l'uso di condensatori, infatti questa configurazione non prevede alcuna temporizzazione, ma solo due stati logici stabili.

Al momento dell'accensione il TRIGGER si trova ad

un livello maggiore di $1/3 V_{CC}$, quindi l'USCITA si trova inizialmente al livello L (STOP), nel quale resta in modo stabile, in mancanza di eventi; la SOGLIA si trova ad un livello minore di $2/3 V_{CC}$, e questa combinazione, in base alla quarta riga della tabella della **Fig. 3**, non provoca commutazioni; a conferma di ciò il LED LD1 risulta costantemente acceso.

Se si preme il pulsante S2 (STOP) la SOGLIA viene portata ad un livello maggiore di $2/3 V_{CC}$, e ci si trova nella condizione prevista dalla riga 3 della tabella, quindi lo stato dell'USCITA non cambia.

Se invece si preme il pulsante S1 (START) il TRIGGER viene portato ad un livello minore di $1/3 V_{CC}$, e ci si trova nella condizione della riga 2 della tabella, quindi l'USCITA diventa alta (H), si spegne il LED LD1 e si accende LD2. Rilasciando S1 la combinazione degli ingressi ridiventa quella della riga 4, e quindi anche questo stato è stabile.

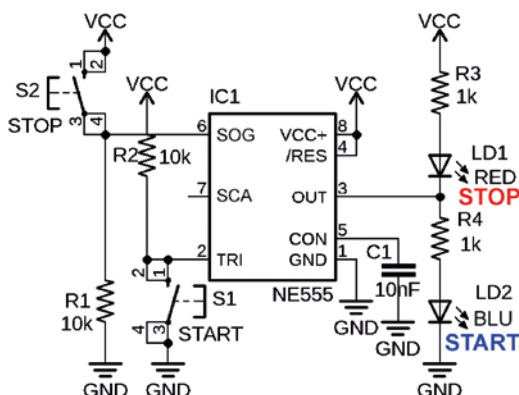
Questa è la ragione per la quale questo tipo di configurazione viene definita bistabile. Il multivibratore bistabile trova applicazione quando bisogna gestire un sistema in base al comportamento di due diversi elementi (sensori o interruttori, p.es.); in effetti è il tipico funzionamento del LATCH SET-RESET.

CONCLUSIONI

Arrivati a questo punto finisce la nostra trattazione su questo semplice ma eccellente ed economico circuito integrato; ora che è chiaro il suo funzionamento nelle sue tre possibili configurazioni, si potrà essere in grado di realizzare, con molta facilità, innumerevoli applicativi.

Stiamo comunque preparando una serie di circuiti molto interessanti che vi presenteremo nei prossimi numeri della rivista, e che naturalmente non mancheremo di spiegare in modo da approfondire ancor più le conoscenze del Timer 555. ■

Fig. 15
Multivibratore
bistabile con 555.



Cosa occorre?

Il timer 555 (cod. 555) è disponibile a Euro 0,40 direttamente dal sito di Futura Elettronica. Chi volesse realizzare il proprio applicativo può utilizzare il servizio PCBPRODUCTION disponibile sul sito www.futurashop.it per realizzare il PCB del progetto. I prezzi si intendono IVA compresa.

Il materiale va richiesto a:

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

WHAT IF



E SE POTESSIMO BERE L'ACQUA DELL'OCEANO?

Il software e le competenze di ADI stanno contribuendo a portare la tecnologia dell'osmosi inversa a più persone, rendendo potabile l'acqua degli oceani.

Analog Devices. Where what if becomes what is.

E se... Scopri di più: analog.com/WhatIf



AHEAD OF WHAT'S POSSIBLE™

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.

LA LIBRERIA MQTT PER ARDUINO



dell'ing. **MATTEO DESTRO**

Analizziamo la libreria MQTT che consente ad Arduino di inviare dati sfruttando anche la connessione GSM.

Nel precedente articolo abbiamo introdotto il protocollo di comunicazione MQTT, versione 3.1.1, di cui abbiamo discusso la storia, il suo funzionamento, il suo ambito di utilizzo e i relativi pacchetti. In questo articolo invece introdurremo la libreria MQTT, scritta in linguaggio C, che va ad integrare la nostra libreria per i motori GSM QUECTEL e SIMCOM. Ricordiamo che il protocollo MQTT è uno standard ISO/OASIS aperto con funzioni di "publish" / "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 utilizzato in applicazioni che richiedono di connettere dispositivi situati in postazioni

remote con larghezza di banda limitata per la trasmissione dei dati. Quindi, dato questo concetto, diventa necessario implementarlo in quelle applicazioni che fanno uso di motori GSM per inviare dati acquisiti sul campo magari in postazioni remote e difficilmente raggiungibili. Da qui la necessità di implementare nella nostra libreria GSM il protocollo MQTT in versione 3.1.1. In futuro verrà anche implementata la versione 5.0.0 dello stesso.

CONCETTI BASE LIBRERIA MQTT PER MOTORI GSM

La nostra libreria è composta da due file, il primo con estensione **.cpp** e il secondo con estensione **.h** e vanno ad aggiungersi agli altri file che compongono la libreria di gestione dei motori GSM Quectel e SimCom. Nel primo trovano posto tutte le funzioni di gestione del protocollo scritte in C/C++ mentre nel secondo abbiamo le definizioni di tutte le costanti usate, nonché delle variabili globali e delle strutture dati della libreria. Quindi nel file con estensione **.h** abbiamo parametrizzato tutta una serie di dati tra cui:

- Costanti stringa da salvare in EEPROM per la gestione delle connessioni MQTT e relativi pacchetti
- I codici univoci dei pacchetti MQTT
- I flag statici da utilizzare nei pacchetti MQTT
- I codici univoci di risposta da parte del Server alla richiesta di connessione fatta da Client
- Una serie di costanti per la formazione dei pacchetti MQTT
- Tutte le variabili, gli array e le strutture dati usate dalla libreria

La parametrizzazione e la realizzazione di apposite strutture dati agevola l'utilizzo della libreria. Infatti tutte le funzioni inerenti la gestione dei pacchetti MQTT prevedono soltanto come parametro da passargli un puntatore in memoria dove salvare il pacchetto dati che si sta creando. Tutte le altre informazioni necessarie alla creazione del pacchetto arrivano dalle strutture dati, da configurare preventivamente, e dalle costanti presenti in libreria.

FUNZIONI GESTIONE PROTOCOLLO MQTT 3.1.1

La libreria mette a disposizione una serie di funzioni pubbliche, che saranno poi quelle richiamate negli sketch Arduino, necessarie a gestire tutti i pacchetti MQTT più una serie di funzioni private

necessarie al corretto funzionamento della libreria ma non visibili negli sketch.

Ricordiamo che i pacchetti MQTT sono composti da almeno uno di questi tre campi:

HEADER TYPE	DESCRIPTION
Fixed Header	Presente in tutti i pacchetti MQTT
Variable Header	Presente in alcuni pacchetti MQTT
Payload	Presente in alcuni pacchetti MQTT

Le funzioni pubbliche messe a disposizione dalla libreria sono visibili nella **Tabella 1**.

La prima funzione che analizziamo è "**void InitData(void)**" la quale non necessita di nessun parametro. Lo scopo della funzione è inizializzare le variabili e le strutture dati della libreria a un valore di default. La seconda funzione dell'elenco "**uint8_t * DecodeMqttPacketReceived(uint8_t * DataPnt, uint8_t MqttCmd)**" serve per decodificare il pacchetto MQTT ricevuto.

I parametri da passare sono un puntatore in memoria che indica da dove iniziare a processare i dati presenti nell'array e il codice MQTT del pacchetto da decodificare. La funzione restituisce un puntatore in memoria che indica a quale locazione di memoria si è arrivati nel decodificare i dati presenti nell'array. I pacchetti MQTT che vengono processati da questa funzione sono i pacchetti di ACK, il pacchetto PUBLISH e la risposta al pacchetto di ping.

Passiamo ora alle funzioni per la generazione dei pacchetti MQTT e la prima che prendiamo in considerazione è "**void Packets_CONNECT(uint8_t * DataPnt)**" che come suggerisce il nome serve al Client per connettersi al Server MQTT e necessita di tutti e tre i campi sopra esposti. L'unico parametro da passare alla funzione è "**DataPnt**"; puntatore in memoria che serve ad indicare al codice da quale locazione iniziare a scrivere i dati che andranno a comporre il pacchetto (tutte le funzioni di generazione di un pacchetto MQTT necessitano unicamente di questo parametro, quindi in seguito ometteremo di evidenziarlo per ogni funzione). Solitamente si tratta di un puntatore a un array di dimensione opportuna. Nelle nostre applicazioni è l'array dell'interfaccia seriale usato per inviare i comandi AT al motore GSM. Non a caso un pacchetto MQTT viene inviato utilizzando il comando AT "**AT+CIPSEND**" per i motori **SIMCOM** e "**AT+QISEND**" per i motori **QUECTEL**.

La composizione del pacchetto MQTT "**CONNECT**" da parte della funzione in esame richiede la pre-



compilazione di alcune strutture dati:

- Struttura dati **"ConnectFlags"** per la gestione dei flag di connessione:
 - Flag **"CleanSession"**
 - Flag **"WillFlag"**
 - Flag **"WillQoS"** (2 bit)
 - Flag **"WillRetain"**
 - Flag **"Password"**
 - Flag **"UserName"**
- Gli array stringa per comporre il pacchetto.

- La variabile **"KeepAlive"** che indica il tempo che può intercorrere tra l'invio di un pacchetto MQTT al Server e il successivo. Se il Server non riceve nulla per un periodo superiore a una volta e mezza il **"KeepAlive"** questo provvederà a chiudere la comunicazione verso il Client.
- Il significato di ogni flag è stato già affrontato nella precedente puntata. A seconda dei flag settati a **"1"** il codice aggiungerà o meno le stringhe di testo inerenti al flag stesso.

void InitData(void)	Funzione per inizializzazione variabili
uint8_t * DecodeMqttPacketReceived(uint8_t * DataPnt, uint8_t MqttCmd)	Funzione per la decodifica del pacchetto MQTT ricevuto
void Packets_CONNECT(uint8_t * DataPnt)	Funzione per la generazione pacchetto MQTT di connessione al Server
void Packets_DISCONNECT(uint8_t * DataPnt)	Funzione per la generazione pacchetto MQTT di disconnessione dal Server
void Packets_CONNACK(uint8_t * DataPnt)	Funzione per la generazione pacchetto MQTT di acknowledge alla richiesta di connessione da parte del Client
void Packets_PUBLISH(uint8_t * DataPnt)	Funzione per la generazione pacchetto MQTT per pubblicazione messaggio
void Packets_PUBACK(uint8_t * DataPnt)	Funzione per la generazione pacchetto MQTT di acknowledge al pacchetto di pubblicazione messaggio (QoS = 1)
void Packets_PUBREC(uint8_t * DataPnt)	Funzione per la generazione pacchetto MQTT di risposta al pacchetto di pubblicazione messaggio (QoS = 2 – Step 1)
void Packets_PUBREL(uint8_t * DataPnt)	Funzione per la generazione pacchetto MQTT di risposta al pacchetto di pubblicazione messaggio (QoS = 2 – Step 2)
void Packets_PUBCOMP(uint8_t * DataPnt)	Funzione per la generazione pacchetto MQTT di risposta al pacchetto di pubblicazione messaggio (QoS = 2 – Step 3)
void Packets_SUBSCRIBE(uint8_t * DataPnt)	Funzione per la generazione pacchetto MQTT per richiesta "Topic" di interesse al Server
void Packets_SUBACK(uint8_t * DataPnt)	Funzione per la generazione pacchetto MQTT di acknowledge al pacchetto di richiesta "Topic"
void Packets_UNSUBSCRIBE(uint8_t * DataPnt)	Funzione per la generazione pacchetto MQTT di annullamento richiesta "Topic" di interesse al Server
void Packets_UNSUBACK(uint8_t * DataPnt)	Funzione per la generazione pacchetto MQTT di acknowledge al pacchetto di annullamento richiesta "Topic"
void Packets_PINGREQ(uint8_t * DataPnt)	Funzione per la generazione pacchetto MQTT per esecuzione PING verso il Server
void Packets_PINGRESP(uint8_t * DataPnt)	Funzione per la generazione pacchetto MQTT di risposta al PING di cui sopra

Tabella 1
Funzioni di libreria per gestione pacchetti MQTT.



Quindi gli array stringa necessari alla formazione del pacchetto MQTT in esame sono:

- Array **"ClientIdentifier"**. Questo array contiene l'identificativo univoco obbligatorio del Client (ovvero l'ID). Quindi è sempre usato per comporre il pacchetto di connessione al Server.
- Array **"WillTopic"** e **"WillMessage"**. Questi array sono due stringhe di testo le quali vengono utilizzate per comporre il pacchetto solo se il rispettivo flag **"WillFlag"** è settato a **"1"**.
- Array **"UserName"**. Questo array è una stringa di testo contenente il nome utente per la connessione al Server. Tale stringa viene usata per comporre il pacchetto solo se il rispettivo flag **"UserName"** è settato a **"1"**.
- Array **"Password"**. Questo array è una stringa di testo contenente la password da usare per la connessione al Server. Tale stringa viene usata per comporre il pacchetto solo se il rispettivo flag **"Password"** è settato a **"1"**.

Questi array devono essere pre-caricati all'inizio dello sketch per poi essere pronti quando serve, vedremo come fare negli esempi.

Tali dati sono memorizzati in EEPROM in modo permanente tramite un altro sketch di cui abbiamo già discusso più volte ovvero **"GSM_WriteEeprom"**. Il codice della funzione eseguirà quindi i seguenti passi:

- Calcolo della lunghezza massima del pacchetto MQTT. Il calcolo viene fatto richiamando la funzione privata **"uint32_t CalcPayloadLen(uint8_t MqttCmd)"**. Questa ritorna un intero a 32bit contenente la lunghezza totale del pacchetto. A questa funzione deve essere passato il codice MQTT del pacchetto che si sta creando in modo che la funzione possa calcolarne correttamente la lunghezza. Ovviamente la lunghezza del pacchetto dipende da come sono stati impostati i flag di connessione trattati precedentemente.
- Intestazione pacchetto MQTT con codice univoco seguito dai rispettivi flag di connessione come discusso prima.
- Codifica lunghezza pacchetto MQTT richiamando funzione privata **"uint8_t * EncodePacketLen(uint8_t * DataPnt, uint8_t**

PayloadLen)". Ricordiamo che la lunghezza del pacchetto MQTT deve essere calcolata con un apposito algoritmo perché è un dato a lunghezza variabile ovvero potrebbe occupare un solo byte (pacchetto non più lungo di 128 byte totali) oppure fino a 4 byte per pacchetti con lunghezza superiore a 128 byte. Per i dettagli su come si fa a codificare la lunghezza del pacchetto vi invitiamo a leggere la precedente puntata. Questa funzione ritorna un puntatore in memoria che indica da quale indirizzo continuare a scrivere i dati. I parametri richiesti sono:

- Puntatore in memoria che indica da dove iniziare a scrivere la lunghezza del pacchetto
 - La lunghezza del pacchetto precedentemente calcolata
- Inserimento dei dati statici e dinamici inerenti il campo **"Variable Header"**.
 - Inserimento dei dati statici e dinamici inerenti il campo **"Payload"** ovvero:
 - Inserimento della stringa **"Client ID"**
 - A seconda dei flag settati a **"1"** inserimento delle stringhe discusse precedentemente

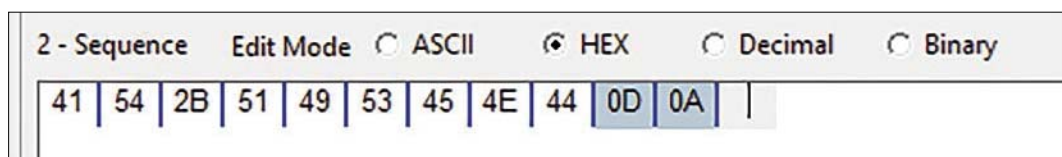
Il risultato che si ottiene da questa serie di passaggi è un array contenente tutti i dati del pacchetto MQTT pronti da essere inviati con apposito comando AT. Di seguito uno spezzone di codice dello sketch che imposta i flag discussi prima e richiama le funzioni di libreria al fine di caricare i dati nell'array **"Gsm.GSM_Data_Array"**:

```
Mqtt.ConnectFlags.Flag.FlagByte = 0x00;
Mqtt.ConnectFlags.Flag.Bit.UserName = 1;
Mqtt.ConnectFlags.Flag.Bit.Password = 1;
Mqtt.ConnectFlags.Flag.Bit.CleanSession = 1;
Mqtt.KeepAlive = 1000;

Gsm.ClearBuffer(&Gsm.GSM_Data_Array[0], (sizeof(Gsm.GSM_Data_Array)/sizeof(Gsm.GSM_Data_Array[0])));
Mqtt.Packets_CONNECT(&Gsm.GSM_Data_Array[0]);
Index = Mqtt.MessageLen;
Gsm.GSM_Data_Array[Index++] = ASCII_SUB;
```

La **Fig. 1** evidenzia il comando **"AT+QISEND"** in formato HEX al quale devono seguire i dati da inviare, in questo caso il pacchetto MQTT.

Fig. 1
Comando
AT+QISEND
(QUECTEL) in
formato HEX.



2 - Sequence		Edit Mode		☐ ASCII		☒ HEX		☐ Decimal		☐ Binary		Pos. 86 / 85									
10	52	00	04	4D	51	54	54	04	C2	03	E8	00	20	34	43	45	45	37	45		
31	36	44	42	36	31	34	34	44	43	42	39	37	45	36	46	46	45	39	34		
46	45	39	33	36	33	00	12	6D	64	65	73	74	72	6F	40	74	69	73	63		
61	6C	69	2E	69	74	00	10	38	4B	4B	53	4D	32	35	4D	42	43	53	42		
53	57	57	46	1A																	

Fig. 2
Sezione dati comando AT+QISEND contenente il pacchetto MQTT da inviare.

La **Fig. 2**, sempre in formato HEX, evidenzia la sezione dati del comando AT sopra esposto nonché il pacchetto MQTT che si vuole inviare.

Passiamo alla funzione **"void Packets_CONNACK(uint8_t * DataPnt)"** la quale serve per codificare il pacchetto di risposta alla precedente funzione nel caso in cui si stia facendo una applicazione lato Server. Questo pacchetto non necessita del campo "Payload".

Quindi molto semplicemente la funzione fa quanto segue:

- Intestazione con codice pacchetto e relativi flags statici.
- Inserimento lunghezza pacchetto. In questo caso si tratta di soli due byte di lunghezza.
- Inserimento variabile flag con la definizione del solo bit "Session Present Flag" che deve essere stato precedentemente configurato.
- Inserimento del codice di risposta al tentativo di connessione dove:
 - "0" → connessione accettata
 - "1" → connessione rifiutata. Versione protocollo non corretta
 - "2" → connessione rifiutata. ID non valido
 - "3" → connessione rifiutata. Server non disponibile
 - "4" → connessione rifiutata. User Name o Password errate
 - "5" → connessione rifiutata. Client non autorizzato

Ovviamente il codice di risposta deve essere caricato nella apposita variabile (**uint8_t ReturnCode**) prima di richiamare la funzione.

La funzione successiva è **"void Packets_DISCONNECT(uint8_t * DataPnt)"** la quale serve per chiudere la connessione. Quando il Client invia questo pacchetto MQTT non riceverà nessuna risposta da parte del Server. In questo caso è presente solo il "Fixed Header" e quindi il codice è limitato alle seguenti:

- Intestazione con codice pacchetto e relativi flags statici.
- Inserimento lunghezza pacchetto che in questo caso è zero.

Passiamo ora alla funzione **"void Packets_PUBLISH(uint8_t * DataPnt)"** la quale serve per pubblicare i messaggi contenenti i dati dell'applicazione in esame; ad esempio temperatura, umidità ecc. In questo caso abbiamo da compilare tutti e tre i campi del pacchetto MQTT. Quindi il codice della funzione eseguirà quanto segue:

- Calcolo della lunghezza massima del pacchetto MQTT richiamando la funzione privata già vista prima **"uint32_t CalcPayloadLen(uint8_t MqttCmd)"**.
- Intestazione con codice pacchetto e relativi flags dinamici precedentemente settati (per i dettagli sul significato dei flags vedere precedente puntata). I flags da compilare sono:
 - "DUP" → Flag usato per indicare se è il primo tentativo di invio del pacchetto
 - "QoS" → Flags (2 bit) che indicano il grado di garanzia di consegna del pacchetto. Valori ammessi "0", "1" e "2"
 - "RETAIN" → Flag che indica al Server se deve memorizzare il messaggio ricevuto
- Codifica lunghezza pacchetto MQTT richiamando funzione privata **"uint8_t * EncodePacketLen(uint8_t * DataPnt, uint8_t PayloadLen)"**.
- Compilazione del campo "Variable Header" tramite inserimento della voce "Topic Name". Questo viene fatto usando la funzione privata **"uint8_t * AddDataToPayload(uint8_t * DataPnt, uint8_t * ArrayPnt, uint16_t ArrayLen, uint8_t MqttCmd, uint8_t DataType)"**. Alla funzione devono essere passati i seguenti:
 - Puntatore in memoria da dove iniziare a scrivere i dati
 - Puntatore in memoria da dove iniziare a leggere i dati da usare come "Topic Name"
 - Lunghezza dell'array contenente i dati da usare per il "Topic Name"
 - Il codice del pacchetto MQTT che si sta compilando
 - Tipologia del dato con cui si sta lavorando: Byte, Integer, Char, String
- Nel caso in cui il flag "QoS" sia maggiore di zero



si deve inserire nel pacchetto anche la voce "Packet Identifier".

- Infine viene aggiunto il campo "Payload". Per caricare i dati del "Payload" si usa ancora la funzione `"uint8_t * AddDataToPayload(uint8_t * DataPnt, uint8_t * ArrayPnt, uint16_t ArrayLen, uint8_t MqttCmd, uint8_t DataType)"` passandogli ovviamente un diverso puntatore in memoria da dove iniziare a leggere e a scrivere.

Quindi prima di richiamare la funzione sopra esposta si devono impostare i flag che si trovano nella struttura dati `"PublishFlags"` e i due array stringa `"PublishTopicName"` e `"PublishPayload"`. Il primo viene caricato prelevando la stringa dalla memoria EEPROM all'avvio dello sketch mentre il secondo andrà riempito di volta in volta a seconda dell'applicazione. Sarà quindi cura dell'utente scrivere il codice necessario per caricare i dati in questo array.

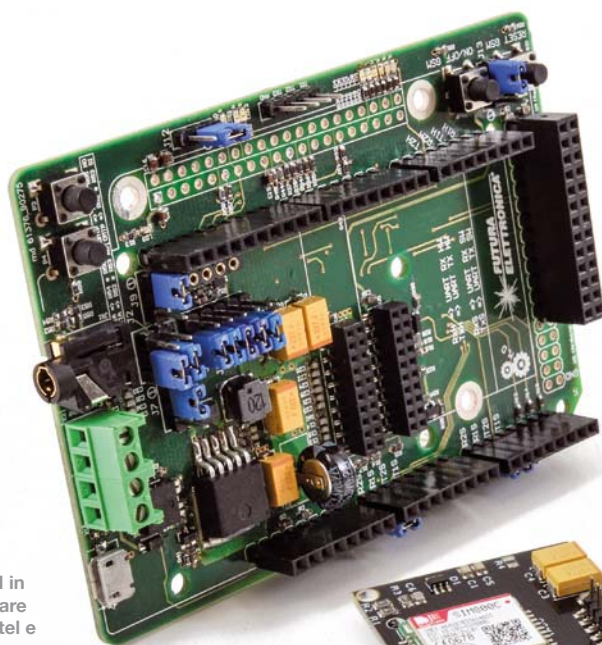
Seguono le funzioni `"void Packets_PUBACK(uint8_t * DataPnt)"`, `"void Packets_PUBREC(uint8_t * DataPnt)"`, `"void Packets_PUBREL(uint8_t * DataPnt)"` e `"void Packets_PUBCOMP(uint8_t * DataPnt)"`. Queste vengono usate quando si deve rispondere al pacchetto MQTT "PUBLISH" a seconda di come è stato impostato il campo "QoS". Quindi la prima della serie serve come risposta al pacchetto MQTT "PUBLISH" con "QoS" settato a "1" mentre le altre

tre fanno parte della risposta al pacchetto MQTT "PUBLISH" con "QoS" settato a "2". Infatti con tale impostazione del campo "QoS" la risposta al pacchetto MQTT "PUBLISH" avviene in tre fasi come ampiamente discusso nella precedente puntata. Il codice delle quattro funzioni è praticamente identico salvo che cambia la codifica del pacchetto MQTT che si sta creando (il campo "Payload" non è previsto); le operazioni eseguite in sequenza dalle quattro funzioni sono:

- Intestazione con codice pacchetto e relativi flags statici.
 - Inserimento lunghezza pacchetto. In questo caso si tratta di soli due byte di lunghezza.
 - Inserimento codice "Packet Identifier" a 16 bit.
- Per tutte queste funzioni bisogna sempre impostare la variabile `"PacketIdentifier"` prima di chiamare la funzione desiderata.

Guardiamo la funzione di libreria `"void Packets_SUBSCRIBE(uint8_t * DataPnt)"` la quale crea il pacchetto MQTT "SUBSCRIBE" con i "Topic" che si desidera ricevere dal Server. Quindi la funzione in esame esegue:

- Calcolo della lunghezza massima del pacchetto MQTT richiamando la funzione privata già vista prima `"uint32_t CalcPayloadLen(uint8_t MqttCmd)"`.
- Intestazione con codice pacchetto e relativi flags statici.
- Codifica lunghezza pacchetto MQTT richiamando funzione privata `"uint8_t * EncodePacketLen(uint8_t * DataPnt, uint8_t PayloadLen)"`.
- Inserimento codice "Packet Identifier" a 16 bit.



→ La shield GSM in grado di ospitare i moduli Quectel e SimCom

- Inserimento stringhe "Topic" di cui si desidera ricevere i dati. Al massimo si possono inserire quattro "Topic". Ovviamente è una limitazione della nostra libreria; c'è possibilità di aumentare il numero dei "Topic" gestibili ma questo comporta un aumento del consumo di memoria SRAM e di EEPROM.

Prima che si possa richiamare la funzione sopra esposta si deve impostare la variabile "PacketIdentifier" e i relativi array per gestire i "Topic". Questi fanno parte della struttura dati "Subscribe" in particolare sotto la voce "Subscribe[x].TopicFilter[]" dove "x" identifica quale dei quattro "Topic" gestibili si sta configurando. Le stringhe di testo inerenti i "Topic" generalmente si trovano in EEPROM e vengono pre-caricate all'avvio dello Sketch. Infine, sempre nella struttura dati in esame, si deve configurare il flag "EnFilter" che indica quale "Topic" si intende usare e quale no. Se settato a "1" "Topic" attivo (**Subscribe[x].Flags.Bit.EnFilter**). Alla funzione di libreria appena esposta segue **"void Packets_SUBACK(uint8_t * DataPnt)"** che, come suggerisce il nome, serve per generare l'acknowledge di risposta al precedente pacchetto. In questo caso è il Server che risponde al Client. Anche in questo caso tutti e tre i campi sono da compilare. Quindi la funzione in esame esegue quanto segue:

- Calcolo della lunghezza massima del pacchetto MQTT richiamando la funzione privata **"uint32_t CalcPayloadLen(uint8_t MqttCmd)"**.
- Intestazione con codice pacchetto e relativi flags statici.
- Codifica lunghezza pacchetto MQTT richiamando la funzione privata **"uint8_t * EncodePacketLen(uint8_t * DataPnt, uint8_t PayloadLen)"**.
- Inserimento codice "Packet Identifier" a 16 bit.
- Inserimento codici di ritorno per ogni "Topic" richiesto. In caso di "Failure" il codice ritornato è 0x80 mentre negli altri casi sarà una dei possibili valori del campo "QoS".

Anche in questo caso prima che si possa richiamare la funzione sopra esposta si devono impostare la variabile **"PacketIdentifier"** e il campo **"AckRetCode"** facente parte della struttura dati **"Subscribe[]"** e in particolare **"Subscribe[x].AckRetCode"**. Nel momento in cui non si vogliono più ricevere determinati "Topics" si deve richiamare la funzione di libreria **"void Packets_UNSUBSCRIBE(uint8_t * DataPnt)"**. Anche in questo caso tutti e tre i campi sono da compilare. Quindi la funzione in esame esegue:

- Calcolo della lunghezza massima del pacchetto MQTT richiamando la funzione privata **"uint32_t CalcPayloadLen(uint8_t MqttCmd)"**.
- Intestazione con codice pacchetto e relativi flags statici.
- Codifica lunghezza pacchetto MQTT richiamando la funzione privata **"uint8_t * EncodePacketLen(uint8_t * DataPnt, uint8_t PayloadLen)"**.
- Inserimento codice "Packet Identifier" a 16 bit.
- Inserimento stringhe "Topic" di cui non si vuole più ricevere i dati.

Prima di richiamare la funzione bisogna impostare la variabile "PacketIdentifier" e settare a "1" il flag "Disfilter", presente nella struttura dati **"Unsubscribe[x].Flags.Bit.DisFilter"**, se si vuole disabilitare il "Topics" n-esimo. Ricordiamo che al massimo si possono gestire quattro "Topics". Ovviamente le stringhe che identificano i "Topics" di interesse sono stati precaricati dalla EEPROM e posti sotto la voce **"UnSubscribe[x].TopicFilter[]"**. Anche per questa funzione appena esaminata esiste la funzione di acknowledge che invia il Server al Client ovvero **"void Packets_UNSUBACK(uint8_t * DataPnt)"**.

L'unica variabile da impostare prima di richiamare la funzione è **"PacketIdentifier"**. Quindi la funzione esegue quanto segue:

- Intestazione con codice pacchetto e relativi flags statici.
 - Inserimento lunghezza pacchetto MQTT che in questo caso è soli due byte.
 - Inserimento codice "Packet Identifier" a 16 bit.
- Analizziamo ora le due funzioni **"void Packets_PINREQ(uint8_t * DataPnt)"** e **"void Packets_PINGRESP(uint8_t * DataPnt)"**. La prima si occupa di inviare il pacchetto MQTT che esegue il ping del Server ed è quindi il Client a inviare tale pacchetto. La seconda invia la risposta alla richiesta di ping ed è il Server a inviarla al Client. Le due funzioni eseguono quanto segue:
- Intestazione con codice pacchetto e relativi flags statici.
 - Inserimento lunghezza pacchetto MQTT che in questo caso è zero byte.

Per completare la serie di funzioni pubbliche necessarie al corretto funzionamento della libreria in esame introduciamo le seguenti:

- **void EepromStartAddSetup(void)** → Funzione che si occupa di inizializzare le variabili di indirizzamento con gli indirizzi di partenza degli array stringa memorizzati in EEPROM.



- **void ReadEepromData(void)** → Funzione che si occupa di leggere i dati dalla EEPROM del microcontrollore e caricare gli appositi array. La prima funzione è propedeutica alla seconda e solitamente vengono richiamate all'inizio dello sketch durante l'inizializzazione ovvero "**void setup()**".

SKETCH DI TEST "GSM_MQTT_311_TEST"

Lo sketch che andiamo a presentare mostrerà come vengono generati i pacchetti MQTT usando la nostra libreria e quindi le funzioni appena descritte. Questo Sketch è stato pensato come strumento di studio e approfondimento di quanto trattato fino a questo punto inoltre può essere fatto girare sulle schede Arduino Mega 2560 e Fishino Mega 2560.

Le schede Arduino Uno R3 non hanno abbastanza memoria SRAM per contenere le strutture dati della libreria MQTT più le variabili della libreria GSM. Lo Sketch è formato da diversi file ognuno con una sua ben definita funzione.

Di seguito un breve elenco con una veloce descrizione:

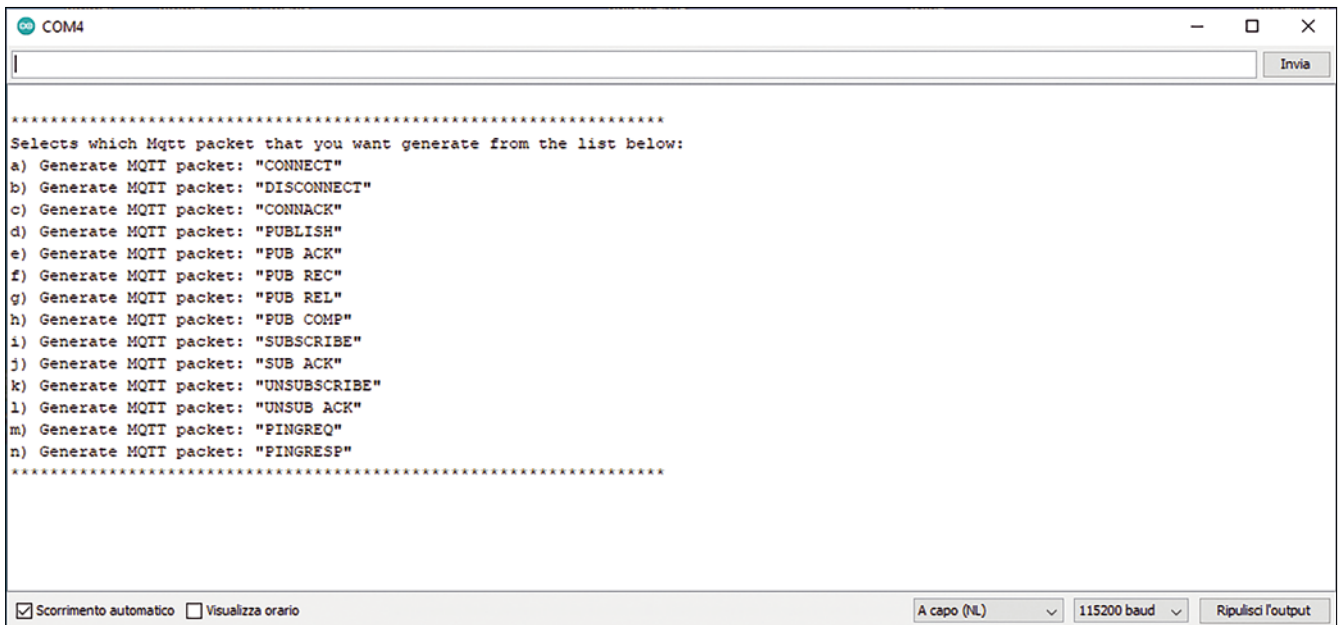
- **"GSM_MQTT_311_Test.ino"** File principale dello Sketch contenente le funzioni "**void setup()**" e "**void loop()**".
- **"MainMenu.ino"** File contenente il codice per la stampa a video del menu di selezione del pacchetto MQTT che si vuole creare.
- **"ProcessMqttConnect.ino"** File per la generazione del pacchetto MQTT CONNECT.
- **"ProcessMqttConnAck.ino"** File per la generazione del pacchetto MQTT CONNACK.
- **"ProcessMqttDisconnect.ino"** File per la generazione del pacchetto MQTT DISCONNECT.
- **"ProcessMqttPublish.ino"** File per la generazione del pacchetto MQTT PUBLISH.
- **"ProcessMqttPubAck_Rec_Rel_Comp.ino"** File per la generazione dei pacchetti MQTT PUBACK, PUBREC, PUBREL e PUBCOMP.
- **"ProcessMqttSubscribe_UnSubscribe.ino"** File per la generazione dei pacchetti MQTT SUBSCRIBE e UNSUBSCRIBE.
- **"ProcessMqttSubAck_UnSubAck.ino"** File per la generazione dei pacchetti MQTT SUBACK e UNSUBACK.
- **"ProcessMqttPingReq_PingResp.ino"** File per la generazione dei pacchetti MQTT PINGREQ e PINGRESP.
- **"SerialMonitor.ino"** File di gestione della comunicazione seriale per ricezione dei comandi stringa da PC.

Come di consueto lo Sketch è suddiviso in file per una migliore comprensione e una più facile manutenzione del codice. In testa al file principale ("**GSM_MQTT_311_Test.ino**") troviamo una serie di costanti stringa le quali sono tutte memorizzate nella FLASH del microcontrollore usando la direttiva al compilatore "**PROGMEM**". Tutte queste stringhe vengono utilizzate dal firmware per guidare l'utente nella creazione del pacchetto MQTT selezionato e come feedback alle selezioni fatte o ai dati inseriti.

Oltre alle innumerevoli costanti stringa trovano posto anche le costanti che definiscono gli stati delle macchine a stati utilizzate per la gestione del flusso di creazione del pacchetto MQTT selezionato nonché alcune costanti che identificano il massimo valore consentito per alcune variabili ecc. Queste vengono tutte definite con la direttiva al compilatore "**#define**". Come tutti gli Sketch Arduino la prima funzione che viene chiamata dal sistema è la solita "**void setup()**" la quale ci serve per inizializzare le variabili della nostra libreria MQTT chiamando l'apposita funzione "**Mqtt.InitData()**". Il suffisso "**Mqtt**" prima della funzione arriva dalla seguente dichiarazione in testa al file: "**Mqtt_GSM Mqtt**"; Seguono poi un'altra serie di funzioni di inizializzazione, sempre facenti parte della nostra libreria GSM, ovvero:

- **"Isr.EnableLibInterrupt()**" → Inizializzazione interrupt di libreria
 - **"Gsm.EnableDisableDebugSerial(true, BAUD_115200)"** → Inizializzazione seriale utilizzata dal "**Monitor Seriale**" Arduino
- Segue inizializzazione della macchina a stati "**MqttProcessPacket**" con lo stato "**ProcessMqttIdle**", delay di 1 secondo e stampa a video della schermata principale con il menù di selezione del pacchetto MQTT che si vuole creare (**Fig.3**). Come si può vedere ci sono elencati tutti i pacchetti MQTT della revisione 3.1.1 e l'utente deve solo selezionare quale pacchetto vuole creare immettendo la lettera corrispondente nella casella di testo del "**Monitor Seriale**" di Arduino e premere sul pulsante "**Invia**" posto sulla destra. Non appena si è inviata la lettera desiderata inizia il processo di composizione del pacchetto MQTT e l'utente sarà guidato passo passo nella creazione dello stesso. Questo serve per prendere familiarità con il protocollo MQTT e capire come funziona e come vengono creati i pacchetti. Infatti alla fine del processo saranno stampate una serie di informazioni e il pacchetto stesso in formato HEX. All'interno della funzione "**void loop()**", la quale





viene richiamata solo dopo avere concluso l'inizializzazione discussa prima, ritroviamo la sola macchina a stati "**MqttProcessPacket()**" la quale viene processata all'infinito.

La macchina a stati utilizzata è del tipo puntatore a funzione e quindi fatta in questo modo:

```
typedef void State;
typedef State (*Pstate)();
Pstate MqttProcessPacket;
```

Come si può vedere abbiamo creato un puntatore a funzione e lo stato iniziale è appunto la funzione "**ProcessMqttIdle**" come mostrato prima durante la discussione dell'inizializzazione dello Sketch. La funzione "**void ProcessMqttIdle(void)**" a cui stiamo puntando con la macchina a stati contiene questo codice (ne mostriamo solo un estratto, guardate il codice sorgente per la versione completa):

```
void ProcessMqttIdle(void) {
    if (TestSerialDataAvailable() == 1) {
        if (StrSerialInput.equals("a")) {
            CmdSequence = CONNECT_PACKET_REQ_CLIENT_ID;
            MqttProcessPacket = ProcessMqttConnect;
        } else if (StrSerialInput.equals("b")) {
            MqttProcessPacket = ProcessMqttDisconnect;
        } else if (StrSerialInput.equals("c")) {
            .....
            .....
        } else if (StrSerialInput.equals("m")) {
```

```
            MqttProcessPacket = ProcessMqttPingReq;
        } else if (StrSerialInput.equals("n")) {
            MqttProcessPacket = ProcessMqttPingResp;
        }
    }
}
```

Come si può vedere di volta in volta che si inserisce una differente lettera usando il "Monitor Seriale" di Arduino si va ad impostare un differente stato ovvero si va a caricare un nuovo puntatore a funzione. Ad esempio digitando la lettera "**a**" viene caricato il puntatore alla funzione "**ProcessMqttConnect**" la quale viene utilizzata per creare il pacchetto MQTT CONNECT.

Vediamo quindi cosa succede, passo passo, quando si inserisce una delle possibili lettere elencate nel menù principale.

Come prima cosa osserviamo che se si inserisce una lettera che non compare nel menù di selezione il firmware genera una stringa di errore e invita a riprovare.

Detto ciò proviamo a inserire la lettera "**a**" per iniziare a generare il pacchetto MQTT CONNECT. Di seguito la descrizione passo passo di quello che fa il firmware alla richiesta:

- Il sistema stampa a video una stringa guida chiedendo di inserire il codice ID. L'utente deve inserire il codice ID per un massimo di 32 caratteri altrimenti il sistema ritorna un errore sulla lunghezza massima dello stesso.
- Se il codice ID viene accettato dal sistema

Fig. 3
Menù di
selezione
pacchetto
MQTT.

questo ritorna una stringa guida che invita a proseguire.

Quindi digitando **"ok"** il sistema va al punto successivo altrimenti con **"exit"** si abortisce il processo.

- Digitando **"ok"** il sistema procede e stampa a video un'altra stringa guida invitando a inserire il valore in secondi della variabile **"Keep Alive"**. Si accettano solo valori numerici, in caso di valori non validi saranno stampate stringhe di errore.
- Se il valore viene accettato il sistema invita a procedere scrivendo **"ok"** come indicato precedentemente oppure **"exit"** per abortire.
- Al passo successivo si dovranno settare i flag di connessione visti nel precedente capitolo e descritti dettagliatamente nella precedente puntata. I flag possono essere settati bit a bit con la sintassi **"bx=y"** oppure tutti in una volta sola con la sintassi **"b=yyyyyyyyy"**. La lettera **"x"** indica il bit da 0 a 7 mentre la lettera **"y"** il valore che si vuole assegnare al bit. Quindi usando la prima sintassi potrebbe essere **"b6=1"**; in questo modo si indica che è necessaria la password per accedere al Server. Ovviamente con questa sintassi si dovrà procedere anche con i restanti bit che si vuole settare a 1. Invece se si vuole eseguire una impostazione globale dei flag usare ad esempio la sintassi **"b=11000010"**. In questo modo abbiamo impostato il flag **"Clean Session"** a 1 e i flag **"User Name"** e **"Password"** a 1.
- Quindi digitare nuovamente **"ok"** per procedere. Il sistema, a seconda dei flag impostati, ci proporrà di inserire altri parametri per comporre il pacchetto MQTT CONNECT. In questo particolare esempio ci verrà chiesto di inserire la stringa **"User Name"** e per fare ciò il sistema stamperà a video la stringa guida apposita. Inserita la stringa il sistema proporrà nuovamente di procedere digitando **"ok"**.
- Ora verrà proposto di inserire la **"Password"** e una volta inserita il sistema stamperà a video nuovamente la stringa che invita a procedere. Anche in questo caso digitare **"ok"**.
- Fatto ciò il sistema ha tutti i parametri necessari a comporre il pacchetto MQTT e quindi stamperà a video il risultato dell'operazione come mostrato nel **Listato 1**.

Come si vede vengono stampati tutti i dati inseriti e infine viene stampato per intero in formato HEX il pacchetto MQTT generato.

Se lo analizziamo possiamo riconoscere le seguenti:

- 10 → Codice pacchetto MQTT (nibble più significativo) e flags (nibble meno significativo).
- 37 → Lunghezza pacchetto MQTT in formato HEX.
- 0004D515454 → Stringa **"MQTT"** con in testa due byte che ne identificano la lunghezza (4 byte).
- 04 → Revisione protocollo 3.1.1.
- C2 → Flags protocollo.
- 012C → Valore **"Keep alive"** in formato HEX (valore decimale impostato a 300 secondi).
- 000E → Lunghezza stringa ID in formato HEX.
- 4D4D444454543738527373735454 → Stringa ID in formato HEX.
- 0012 → lunghezza stringa **"User Name"** in formato HEX.
- 746F706F6C696E6F40706970706F2E636F6D → Stringa **"User Name"** in formato HEX.
- 0007 → Lunghezza stringa **"Password"** in formato HEX.
- 4D696E6E693535 → Stringa **"Password"** in formato HEX.

Con questo approccio appena descritto si possono studiare tutti i pacchetti MQTT messi a disposizione dalla versione 3.1.1 del protocollo. Per alcuni pacchetti non ci sarà bisogno di inserire nessun parametro per altri sarà necessario inserire i parametri come appena visto per il pacchetto MQTT CONNECT. Se per questo pacchetto avessimo settato anche il flag **"Will Flag"** il sistema ci avrebbe chiesto di inserire la stringa di testo per il **"Will Topic"** e la stringa di testo per il **"Will Message"** e avremmo ottenuto, per esempio, il seguente pacchetto MQTT:

```
=====
Packet Code      (Type Hex):    10
Packet Length    (Type Hex):    005C
Variable Header  (Type Hex):    0004; 4D; 51; 54; 54
Protocol Level   (Type Hex):    04
Protocol Flags   (Type Hex):    C6
Keep Alive       (Type Hex):    012C
Client ID        (Type String):  MDDTT78RsssTT
Will Topic       (Type String):  Message Error
Will Message     (Type String):  Data Connection Lost
User Name        (Type String):  topolino@pippo.com
Password         (Type String):  Minni55
=====
```

Mqtt "CONNECT PACKET" generated:

```
105C0004D51545404C6012C000E4D4D4444545437
38527373735454000D4D657373616765204572726F-
7200144461746120436F6E6E656374696F6E204C6F7374001274
```



6F706F6C696E6F40706970706F2E636F6D00074D696E6E693535

Come si può vedere il pacchetto MQTT CONNECT generato è decisamente più lungo del precedente in quanto contiene due stringhe di testo in più rispetto a prima e di conseguenza la lunghezza del pacchetto non è più 0x37 (55 byte) bensì 0x5C (92 byte). Il codice che gestisce l'inserimento dei dati passo passo e la generazione del pacchetto MQTT CONNECT è sito nel file "**ProcessMqttConnect.ino**". La funzione che si occupa di gestire il meccanismo di inserimento dei dati passo passo è "**void ProcessMqttConnect(void)**", tale funzione viene richiamata dalla macchina a stati esposta prima attraverso il concetto di puntamento a funzione. All'interno della funzione stessa esiste una seconda macchina a stati realizzata con il costrutto "**switch**" dove esistono tanti "**case**" quanti sono gli stati da rappresentare. Nel **Listato 2** è possibile

vedere un estratto della funzione.

Come si vede dal codice la variabile "**CmdSequence**" contiene il valore dello stato corrente il quale viene processato per poi passare eventualmente allo stato successivo se le condizioni lo permettono.

In questo Sketch gli array stringa, che di solito vengono caricati con i dati prelevati dalla memoria EEPROM, ricevono i dati direttamente dall'interfaccia seriale connessa al "**Monitor Seriale**" di Arduino. Se osservate il codice noterete che l'array "**Mqtt.ClientIdentifier**" viene caricato con i dati ricevuti dalla seriale usando la funzione di libreria per copiare le stringhe in C ovvero "**strcpy(Mqtt.ClientIdentifier, StrSerialInput.c_str())**".

Tutte le volte che lo stato si conclude, la variabile "**CmdSequence**" viene incrementata di una unità permettendo così di processare lo stato successivo a meno che non si sia digitato la stringa "**exit**"

↓ Listato 1

Sezione di inserimento dati

```
- Inserts Client ID. Max length 32 chars
# Client ID saved. If finished type "ok" to procede or "exit" to abort!
- Inserts Keep Alive Time in seconds. Max value 65535
# Keep Alive saved. If finished type "ok" to procede or "exit" to abort!
- Sets Flags used with "CONNECT" Packet [Example to set single bit: b1=0 or b1=1; Example to set all bits:
b=00110100]
b0) Reserved
b1) "Clean Session" - Set Flag: 1 ON; 0 OFF
b2) "Will Flag" - Set Flag: 1 ON; 0 OFF
b34) "Will QoS" - Set Flag: 00; 01; 10
b5) "Will Retain" - Set Flag: 1 ON; 0 OFF
b6) "Password Flag" - Set Flag: 1 ON; 0 OFF
b7) "User Name Flag" - Set Flag: 1 ON; 0 OFF
Flags "CONNECT PACKET" set. If finished type "ok" to procede or "exit" to abort!
- Inserts UserName. Max length 32 chars:
# User Name saved. If finished type "ok" to procede or "exit" to abort!
- Inserts password. Max length 32 chars:
# Password saved. If finished type "ok" to procede or "exit" to abort!
```

Risultato ottenuto dall'algoritmo

```
=====
Packet Code      (Type Hex):    10
Packet Length    (Type Hex):    0037
Variable Header  (Type Hex):    0004; 4D; 51; 54; 54
Protocol Level   (Type Hex):    04
Protocol Flags   (Type Hex):    C2
Keep Alive       (Type Hex):    012C
Client ID        (Type String):  MMDDTT78RsssTT
Will Topic       (Type String):  #
Will Message     (Type String):  #
User Name        (Type String):  topolino@pippo.com
Password         (Type String):  Minni55
=====
```

Mqtt "CONNECT PACKET" generated: 103700044D5154540C2012C000E4D4D4444545437385273737354540012746F706F6C696E6F40706970706F2E636F6D00074D696E6E693535



per abortire il processo.
 Il quel caso lo stato viene riportato in IDLE mode.
 Arrivati quindi all'ultimo stato disponibile, ovvero **"CONNECT_PACKET_GENERATE"**, viene richiamata la funzione per la stampa dei dati ricevuti e la generazione del rispettivo pacchetto MQTT, tale funzione si chiama **"Generate_Connect_Packet()"**.
 Infine viene aggiornata la macchina a stati con il nuovo puntatore a funzione e stampato nuova-

mente a video il menù principale affinché l'utente possa effettuare una nuova scelta.

La funzione **"Generate_Connect_Packet()"**, come appena detto, stampa a video tutti i dati ricevuti che andranno a comporre il pacchetto MQTT in esame e tra le altre cose richiamerà la funzione della nostra libreria per la generazione dei pacchetti MQTT ovvero **"Mqtt.Packets_CONNECT(&Gsm.GSM_Data_Array[0])"**

Listato 2

```

void ProcessMqttConnect(void) {
    uint8_t RetValue;

    switch (CmdSequence) {
        case CONNECT_PACKET_REQ_CLIENT_ID:
            LocalReadString_FLASH((uint8_t *)STR_CMD_CONNECT_CLIENT_ID, strlen(STR_CMD_CONNECT_CLIENT_ID), TRUE, FALSE);
            CmdSequence++;
            break;
        case CONNECT_PACKET_WAIT_INSERT_CLIENT_ID:
            if (TestSerialDataAvailable() == 1) {
                // If serial data is available
                RetValue = CheckStringOkOrExit();
                if (RetValue == 1) {
                    // Typed "ok"
                    break;
                } else if (RetValue == 2) {
                    // Typed "exit"
                    CmdSequence = CONNECT_PACKET_IDLE;
                    break;
                }
            }
            if (StrSerialInput.length() > 0) {
                if (StrSerialInput.length() > ID_CLIENT_MAX_LENGTH) {
                    LocalReadString_FLASH((uint8_t *)STR_ERR_CLIED_ID_TOO_LONG, strlen(STR_ERR_CLIED_ID_TOO_LONG), TRUE, FALSE);
                    break;
                }
                memset(Mqtt.ClientIdentifier, 0, (sizeof(Mqtt.ClientIdentifier)/sizeof(Mqtt.ClientIdentifier[0])));
                strcpy(Mqtt.ClientIdentifier, StrSerialInput.c_str());
                LocalReadString_FLASH((uint8_t *)STR_CLIED_ID_OK, strlen(STR_CLIED_ID_OK), FALSE, FALSE);
                LocalReadString_FLASH((uint8_t *)STR_FINISHED, strlen(STR_FINISHED), TRUE, FALSE);
            }
            break;
        case CONNECT_PACKET_REQ_KEEP_ALIVE:
            LocalReadString_FLASH((uint8_t *)STR_CMD_CONNECT_KEEP_ALIVE, strlen(STR_CMD_CONNECT_KEEP_ALIVE), TRUE, FALSE);
            CmdSequence++;
            break;
        .....
        .....

        case CONNECT_PACKET_GENERATE:
            Generate_Connect_Packet();

            MqttProcessPacket = ProcessMqttIdle;
            PrintMainMenu();
            break;
        default:
            break;
    }
}

```



alla quale viene passato l'indirizzo di memoria dal quale iniziare a salvare i dati.

In questo caso si parte a scrivere dall'indice zero dell'array. Il contenuto dell'array viene poi stampato a video usando la seguente funzione

"PrintHexArray(&Gsm.GSM_Data_Array[0], Mqtt.MessageLen)";

Quanto appena descritto per il file **"ProcessMqtt-Connect.ino"** lo rivediamo anche negli altri file che gestiscono la generazione dei pacchetti MQTT più complessi ovvero che necessitano una serie di parametri per la composizione dello stesso.

In particolare:

"ProcessMqttPublish.ino"

"ProcessMqttSubscribe_UnSubscribe.ino"

"ProcessMqttSubAck_UnSubAck.ino"

I restanti file sono meno complessi dal punto di vista del codice ed alcuni non richiedono l'interazione da parte dell'utente come ad esempio la generazione del pacchetto MQTT DISCONNECT (lettera "b"), PINGREQ (lettera "m") e PINGRESP (lettera "n").

Di seguito i rispettivi pacchetti MQTT generati:

```
=====
Packet Code      (Type Hex):    E0
Packet Length    (Type Hex):    0000
```

```
=====
Mqtt "DISCONNECT PACKET" generated: E000
=====
```

```
=====
Packet Code      (Type Hex):    C0
Packet Length    (Type Hex):    0000
=====
```

```
=====
Mqtt "PINGREQ PACKET" generated: C000
=====
```

```
=====
Packet Code      (Type Hex):    D0
Packet Length    (Type Hex):    0000
=====
```

```
=====
Mqtt "PINGRESP PACKET" generated: D000
=====
```

Altri invece richiedono semplicemente l'inserimento del parametro **"Packet Identifier"** come richiesto per i pacchetti MQTT PUBACK (lettera "e"), PUBREC (lettera "f"), PUBREL (lettera "g") e PUBCOMP (lettera "h"). Di seguito i risultati ottenuti per i quattro pacchetti MQTT tutti con parametro **"Packet Identifier"** settato a 3333 (0x0D05 se espresso in Hex):

↓ Listato 3

Sezione di interazione con l'utente

- Sets Flag used with "CONNACK" Packet [Example to set single bit: b0=0 or b0=1
b0) Session Present - Set Flag: 1 ON; 0 OFF
b17) Reserved - Bits 1-7 are reserved and MUST be set to 0
Flag "SP" set. If finished type "ok" to proceed or "exit" to abort!
- Insert Connect Return Code. Code values available:
 - "0)" Connection Accepted
 - "1)" Connection Refused, unacceptable protocol version
 - "2)" Connection Refused, identifier rejected
 - "3)" Connection Refused, server unavailable
 - "4)" Connection Refused, bad user name or password
 - "5)" Connection Refused, not authorized
- # Return Code saved. If finished type "ok" to proceed or "exit" to abort!

Risultato finale

```
=====
Packet Code      (Type Hex):    20
Packet Length    (Type Hex):    02
Connect ACK Flags (Type Hex):    00
Return Code      (Type Hex):    04
=====
```

```
Mqtt "CONNACK PACKET" generated: 20020004
```



```

Packet Code      (Type Hex):    40
Packet Length    (Type Hex):    02
Packet Identifier (Type Hex):    0D05
=====

```

Mqtt "PUBACK PACKET" generated: 40020D05

```

Packet Code      (Type Hex):    50
Packet Length    (Type Hex):    02
Packet Identifier (Type Hex):    0D05
=====

```

Mqtt "PUBREC PACKET" generated: 50020D05

```

Packet Code      (Type Hex):    62
Packet Length    (Type Hex):    02
Packet Identifier (Type Hex):    0D05
=====

```

Mqtt "PUBREL PACKET" generated: 62020D05

```

Packet Code      (Type Hex):    70
Packet Length    (Type Hex):    02
Packet Identifier (Type Hex):    0D05
=====

```

Mqtt "PUBCOMP PACKET" generated: 70020D05

Per quanto riguarda la generazione del pacchetto MQTT CONNACK (lettera "c") verrà richiesto di settare il flag "**Session Present**" e di inserire il codice di ritorno tra quelli disponibili. Il risultato dell'operazione, dove si è impostato come "**Return Code**" il valore 0x04, è visibile nel **Listato 3** (il significato del codice 0x04 è "**Connection Refused, bad user name or password**"). Per quanto riguarda gli altri pacchetti MQTT lasciamo al lettore il piacere di sperimentare le varie combinazioni possibili al fine di studiare e capire come vengono generati i pacchetti MQTT e loro codifica.

CONCLUSIONI

Si conclude anche il secondo appuntamento sull'argomento del protocollo MQTT. Nella prossima ed ultima puntata affronteremo una applicazione reale dove ci interfacceremo con il Server ThingsSpeak per inviarli dei dati che verranno poi raccolti in modalità grafica come già fatto per altre applicazioni. Invitiamo i lettori interessati a sperimentare, tra-

mite lo Sketch proposto in questo numero, le varie combinazioni che si possono ottenere nella codifica dei pacchetti MQTT soprattutto per i pacchetti PUBLISH e SUBSCRIBE.

Ricordiamo inoltre che lo sketch gira su un micro-controllore con delle risorse limitate e che quindi gli array usati per contenere le stringhe necessarie sono stati così dimensionati:

- Array per il "**Client ID**" massimo 32 byte
- Array per la stringa "**Will Topic**" massimo 32 byte
- Array per la stringa "**Will Message**" massimo 32 byte
- Array per la stringa "**User Name**" massimo 32 byte
- Array per la stringa "**Password**" massimo 32 byte
- Array per la stringa "**Publish Topic**" massimo 64 byte
- Array per la stringa "**Subscribe Topic Filter**" massimo 64 byte per un totale di 4 filtri possibili (256 byte)

Anche la lunghezza dell'array usato con l'interfaccia seriale per l'invio dei comandi AT è stato maggiorato a 512 byte, quindi come si può vedere ci vogliono svariate risorse per quanto riguarda la memoria SRAM da usare.

Di conseguenza lo Sketch trattato non è compatibile con le schede Arduino Uno R3 o lo schede Fishino Uno per mancanza di adeguate risorse in merito alla memoria SRAM. □



Cosa occorre?

Lo shield universale GSM (cod. WWGSM SHIELD) è in vendita presso Futura Elettronica a Euro 54,90. Viene fornito montato ad esclusione degli strip line (compresi) che vanno saldati manualmente. Lo shield si può interfacciare con uno dei seguenti moduli GSM: interfaccia con M95 (cod. FT1128M) disponibile a Euro 29,00, modulo cellulare miniaturizzato con SIM800 (cod. FT1308M) in vendita a Euro 29,00.

Il materiale va richiesto a:

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



Taglia e incidi con **Cricut MAKER**



Prezzo IVA inclusa

Macchina da taglio intelligente



€ 439,00

cod. CRICUTMAKER

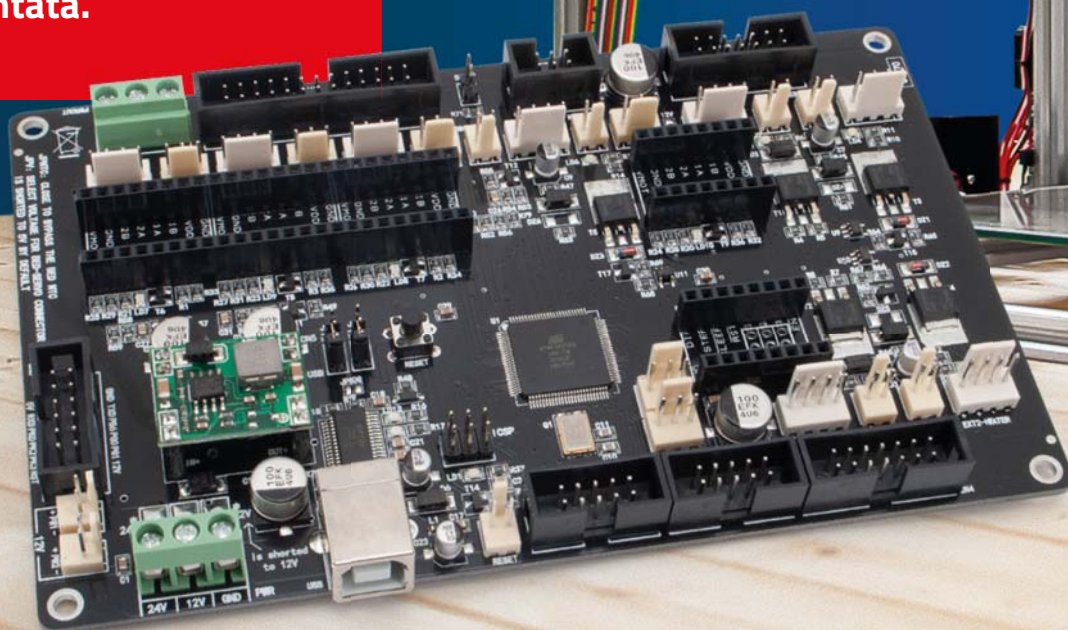
Dai vita alla fantasia e alla creatività con Cricut Maker l'innovativa macchina da taglio. Grazie a Design Space, il software di progettazione gratuito, potrai realizzare praticamente qualsiasi progetto fai-da-te, dall'arte 3D all'arredamento per la casa, gioielli, progetti di carta e molto altro ancora 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.

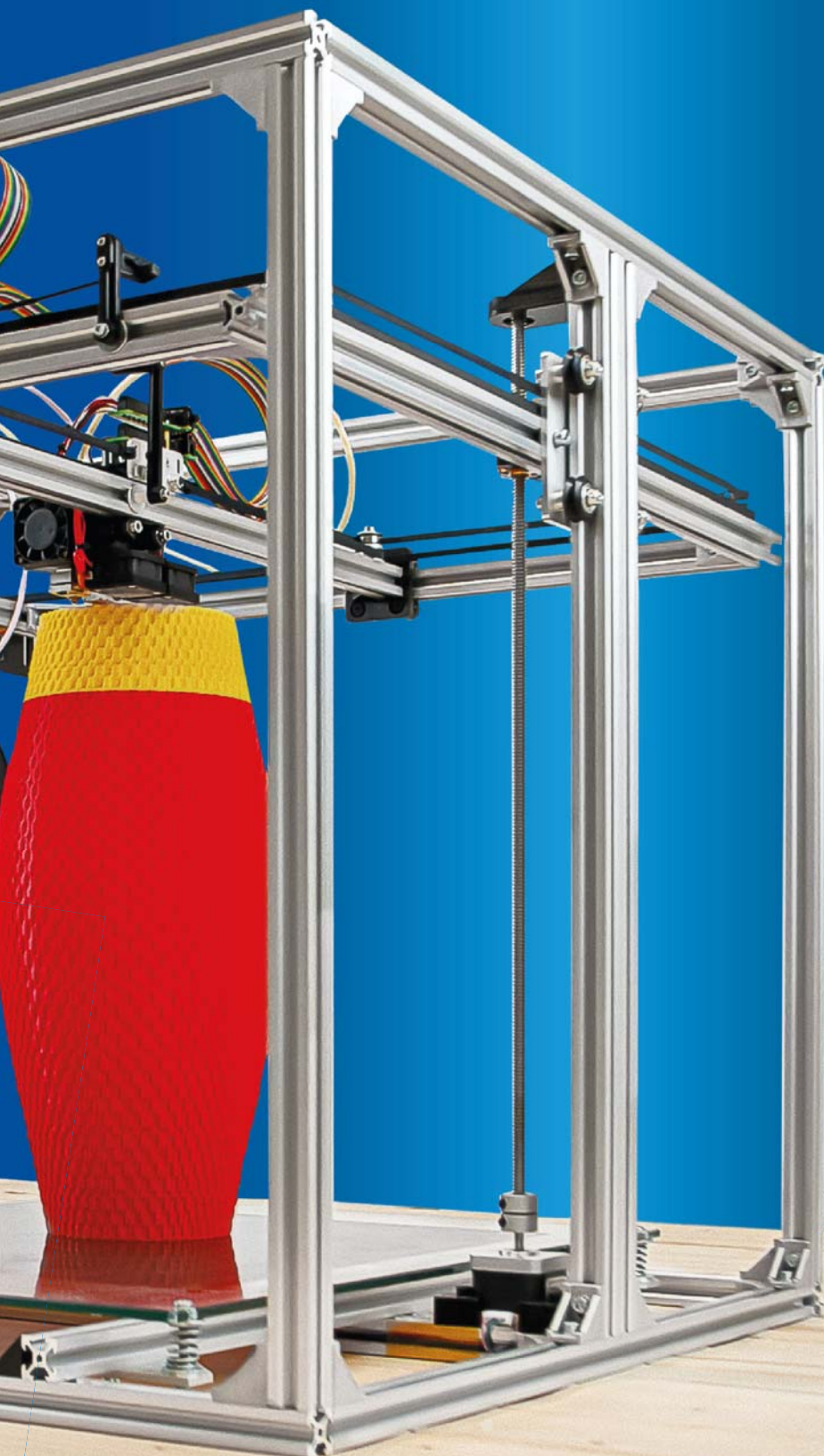


CONTROLLER 3D4040 DUAL EXTRUDER

di DAVIDE SCULLINO

Stampiamo a due colori
con la nostra stampante 3D
capace di produrre pezzi in
plastica delle dimensioni di
40x40x40 cm: sostituiamo il
controller con uno avanzato,
montiamo il secondo
estrusore e siamo operativi!
Prima puntata.





P

robabilmente conoscerete il nostro impegno nello sviluppare e proporre soluzioni per la stampa 3D e ricorderete la nostra più recente

creazione, vale a dire la stampante 3D4040, capace di stampare oggetti in materiale plastico delle dimensioni di 40x40x40 cm (64.000 cm³) sfruttando la tecnologia FDM (Fused Deposition Modeling). Questa macchina nasce per la stampa a singolo estrusore, ma è aggiornabile in modo da montare una doppia testa di stampa in luogo della singola "di serie".

In questo articolo vi spiegheremo come fare per ottenere stampe a due colori e lo faremo descrivendovi le modifiche da fare, che riguardano la testa di stampa e la scheda controller. Quest'ultima andrà sostituita con quella che descriveremo nei prossimi paragrafi, sviluppata per espandere le funzionalità della 3D4040; si tratta infatti di una scheda evoluta e dotata di più funzioni di quella essenziale con cui viene equipaggiata la stampante, la quale nasce dalla 3Drag. Vedrete, ad esempio, che questo nuovo controller può pilotare e gestire il piatto riscaldato e, chiaramente, il doppio estrusore.

IL NUOVO CONTROLLER

La scheda controller originale della stampante 3D4040 è la medesima della 3Drag (3DCONTR-DRIVER commercializzato da Futura Elettronica,

su www.futurashop.it) e quindi non può gestire un doppio estrusore, in quanto ha driver soltanto per uno, sia relativamente al motore passo-passo, sia per quanto riguarda il riscaldatore dell'estrusore. Inoltre non ha ingressi a sufficienza per gestire i termistori di piatto riscaldato ed estrusori. Ecco quindi che abbiamo pensato di progettare la nuova scheda controller che qui vi descriviamo analizzandone lo schema elettrico.

Per quanto riguarda il controllo del piatto riscaldato, anche in questo caso rimane necessario l'utilizzo di un relé allo stato solido per alimentare l'elemento riscaldante del piatto, onde evitare surriscaldamenti delle piste e del connettore di alimentazione posto sulla scheda; infatti vista la potenza elettrica richiesta (dell'ordine dei 200 watt) il riscaldatore assorbe una corrente di valore tale da sollecitare eccessivamente il MOSFET T5 (che non ha dissipatore) e il circuito stampato (nonché il connettore) di alimentazione della scheda controller.

La nuova scheda controller (**Fig. 1**) è stata progettata per gestire la 3D4040 con doppio estrusore, quindi ha in più uscite ed ingressi dedicati a gestire il doppio estrusore; è basata anch'essa su un microcontrollore ATmega 2560 e condivide buona parte dell'hardware della RAMPS (RepRap Arduino Mega Pololu Shield). Il processore ATmega2560 dispone di ben 256 kB di memoria di programma

e lavora con un clock di 16 MHz, quindi è potente quanto basta anche a svolgere le funzioni aggiuntive che abbiamo implementato e può ospitare il firmware Marlin da noi modificato per la stampante 3D4040 con doppio estrusore e relativi feedback.

Anche in questo caso la scheda può essere programmata direttamente dall'IDE Arduino e dispone, allo scopo, di una connessione USB per collegarla al computer tramite un cavo USB-A/USB tipo B; la stessa porta permette il controllo da PC durante le stampe nella modalità real-time.

Andiamo dunque ad analizzare tutti i dettagli del controller, iniziando proprio dalle differenze nella sezione USB, il cui interfacciamento è gestito da un classico converter UART/USB FTDI FT232RL, con una differenza: sono stati aggiunti dei TVS sulle linee dati data più data meno per proteggere da eventuali disturbi che potrebbero introdursi dai fili di collegamento; questa è una protezione in più, utile se la stampante lavora in un ambiente elettricamente disturbato.

Altra particolarità riguardante l'interfaccia USB è la possibilità, tramite i ponticelli JPRX e JPTX, di assegnare i canali dati, rispettivamente TXD ed RXD del convertitore U3 agli I/O del microcontrollore dedicati PE0 e PE1, ovvero alle linee TXD ed RXD direttamente là dove si voglia utilizzare l'UART (seriale fisica) interno del micro.

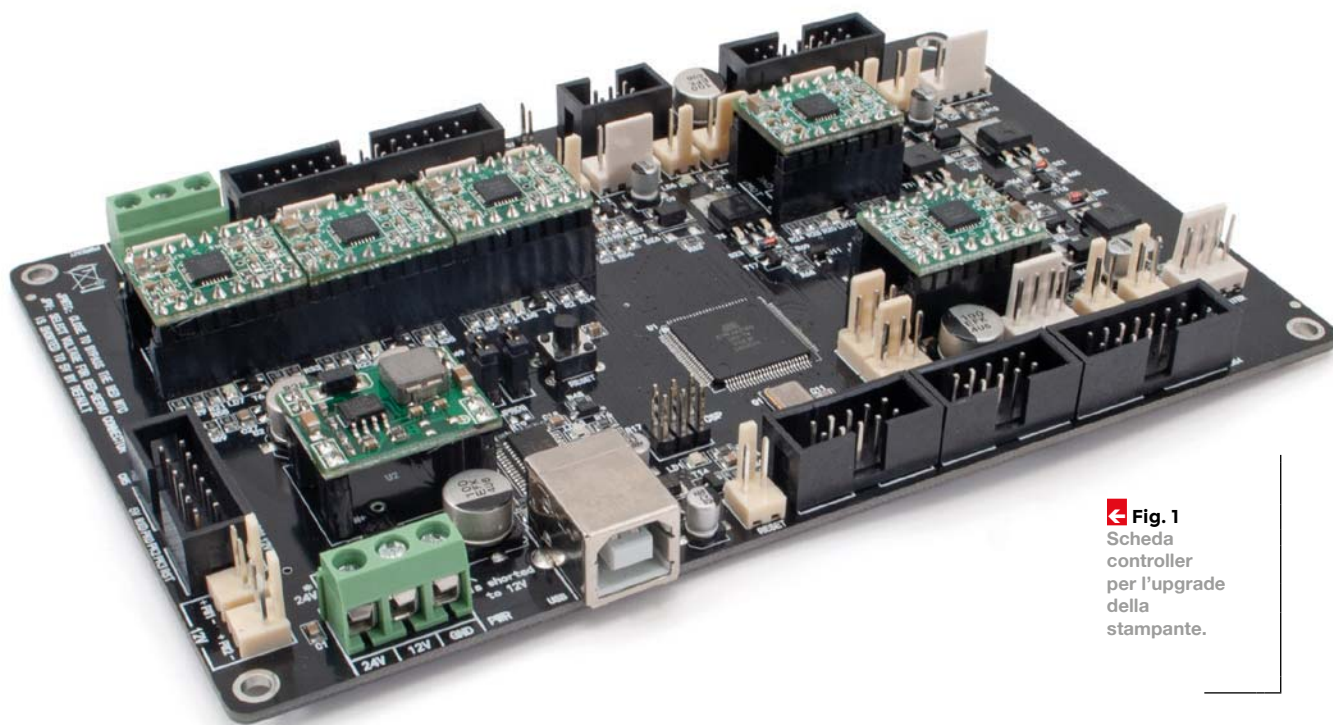


Fig. 1
Scheda controller per l'upgrade della stampante.

Questa possibilità è molto utile perché consente, se occorrono per altre applicazioni, di liberare le linee PE0 e PE1 dell'ATmega2560, ovvero creare ed assegnare a tali linee un'interfaccia seriale virtuale mediante la specifica libreria di Arduino.

Procediamo con l'analisi dello schema elettrico analizzando le connessioni: in particolare abbiamo gli ingressi EXT1-THERM ed EXT2-THERM che sono dedicati a leggere la resistenza assunta dai termistori degli estrusori (notate che ognuno fa capo ad un partitore resistivo con diodi di protezione da extratensione e polarità inversa, oltre che con filtro dei disturbi) e rilevano la temperatura di estrusione in modo da poterla regolare accuratamente con il PID all'interno del firmware; il micro legge le sonde di temperatura (NTC) che riportano il feedback sull'attività dei riscaldatori che sono connesse a EXT1-THERM per l'estrusore 1 ed EXT2-THERM per l'estrusore 2.

Le sonde consentono al controller di stabilizzare la temperatura degli estrusori in modo che se eccede il valore settato, il microcontrollore agisce sulla rispettiva uscita di controllo del riscaldatore e riduce la corrente nel filamento riscaldatore, mentre in caso contrario la aumenta.

Le uscite per il controllo delle resistenze riscaldatrici degli estrusori sono EXT1-HEATER ed EXT2-HEATER: rispettivamente per primo e secondo estrusore.

L'ingresso BED-THERM è dedicato al termistore che fornisce la retroazione per la temperatura del piatto riscaldato, che viene comandato attraverso un MOSFET connesso all'uscita BED HEATER; come accennato, c'è bisogno di un dispositivo esterno che può essere un SSR o un ulteriore MOSFET esterno (lo schema di cablaggio in queste pagine propone il controllo attraverso l'FT1357 della Futura Elettronica).

Il piatto riscaldato è comunque un accessorio e la sua assenza non pregiudica il funzionamento della macchina; il controllo della temperatura raggiunta dal piatto serve, in abbinamento al client di stampa, a far avviare la stampa stessa, solo una volta che il piatto ha raggiunto la temperatura di regime. Notate che se non viene installato il piatto riscaldato bisogna chiudere il ponticello JPNTC; in caso contrario va lasciato aperto. Tale jumper fornisce all'ingresso PK6 una tensione continua di riferimento che permette di prescindere dall'esistenza del riscaldatore.

I termistori NTC utilizzati per rilevare la temperatura dell'estrusore e del piatto riscaldato sono entrambi da 100 kohm a 25 °C.

I transistor vengono pilotati tutti con un segnale PWM, così da variare la potenza ceduta agli utilizzatori e controllare con la minima perdita nei MOSFET il calore generato dalle resistenze dei riscaldatori (idem per la velocità della ventola di ogni raffreddatore del pezzo), però con bassissime perdite, dato che i MOSFET lavorano in modo on-off. Anche l'uscita BED HEATER fornisce un segnale di comando PWM a bassissima frequenza.

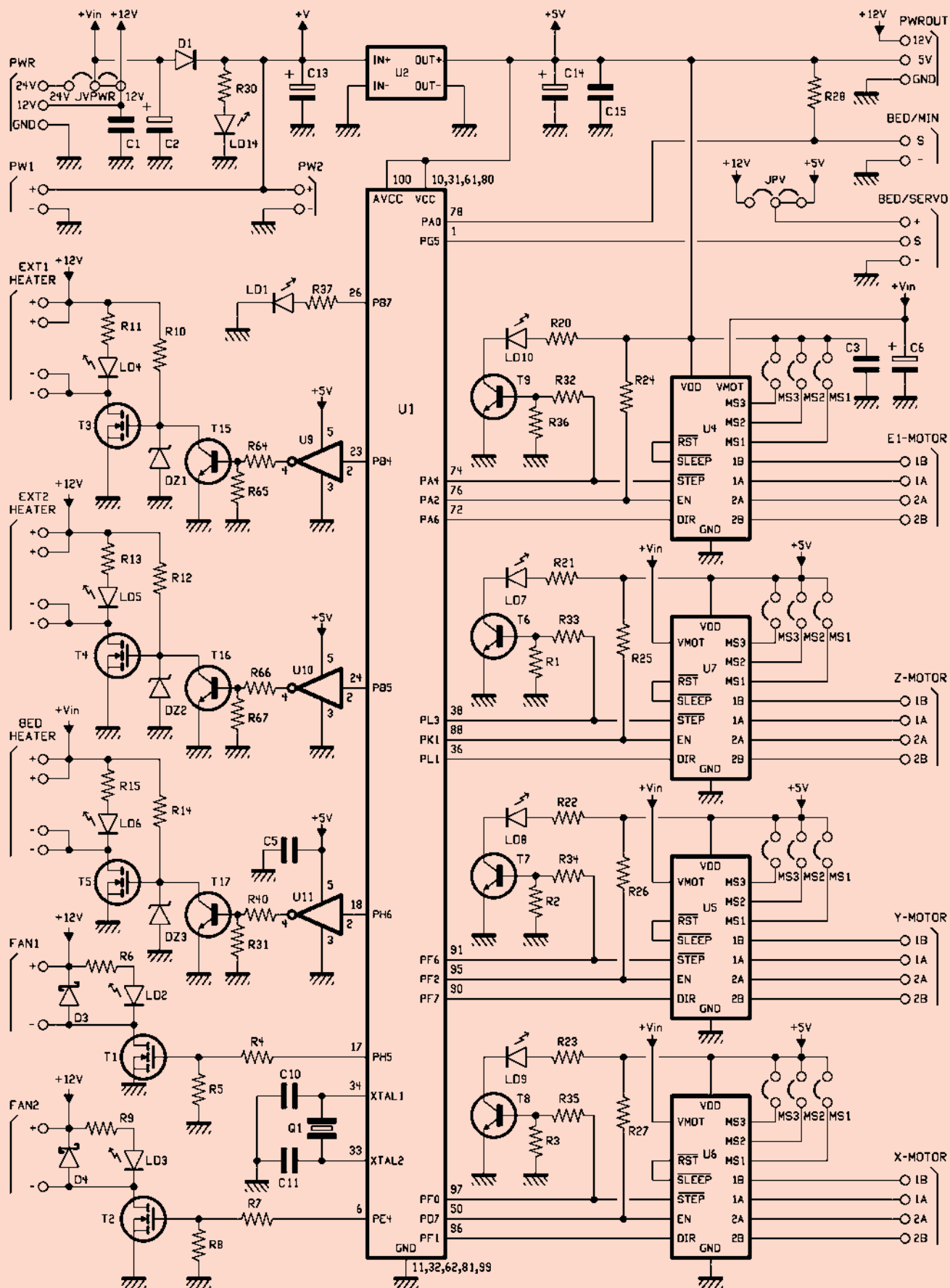
Ciascuna delle uscite riservate ai riscaldatori e alle ventole è provvista di un LED che pulsa ripetendo il relativo segnale PWM: la frequenza è circa 4 Hz per i riscaldatori di estrusori e piatto (permettendo di verificare ad occhio lo stato di funzionamento) mentre il PWM di ciascuna ventola di raffreddamento è a frequenza più alta e il relativo LED sembrerà sempre acceso.

Le uscite di potenza per riscaldatori e ventole hanno ciascuna un MOSFET BUK6215-75C, capace di erogare una corrente di drain che raggiunge 57 A e sopportano una Vds in stato di interdizione, pari a 75 V; la bassissima Rdson (15 milliohm max a 15 A

CARATTERISTICHE TECNICHE

 Tensione di alimentazione: 12 e 24 Vcc	 Velocità di stampa massima: 350 mm/s
 Corrente assorbita: max 6A escluso piatto riscaldato	 6 motori passo-passo (200 step)
 Numero estrusori: 2	 Risoluzione meccanica asse Z: 0,625 micron
 Dimensioni di stampa: 400x400x400 mm	 Risoluzione meccanica assi X e Y: 0,0125 mm (12 micron)
 Materiali utilizzabili: ABS e PLA	 Ingombro: 690x630x610 mm
 Movimentazione: tecnologia Core XY	 Firmware: Marlin personalizzato per 3D4040
 Diametro filamento: 1,75 mm	 Software: compatibile con software OpenSource (consigliato RepetierHost)
 Diametro ugello fornito: 0,4 mm	 Sistemi operativi supportati da RepetierHost: Windows - Mac - Linux
 Diametro ugelli opzionali: da 0,3 mm a 0,8 mm	





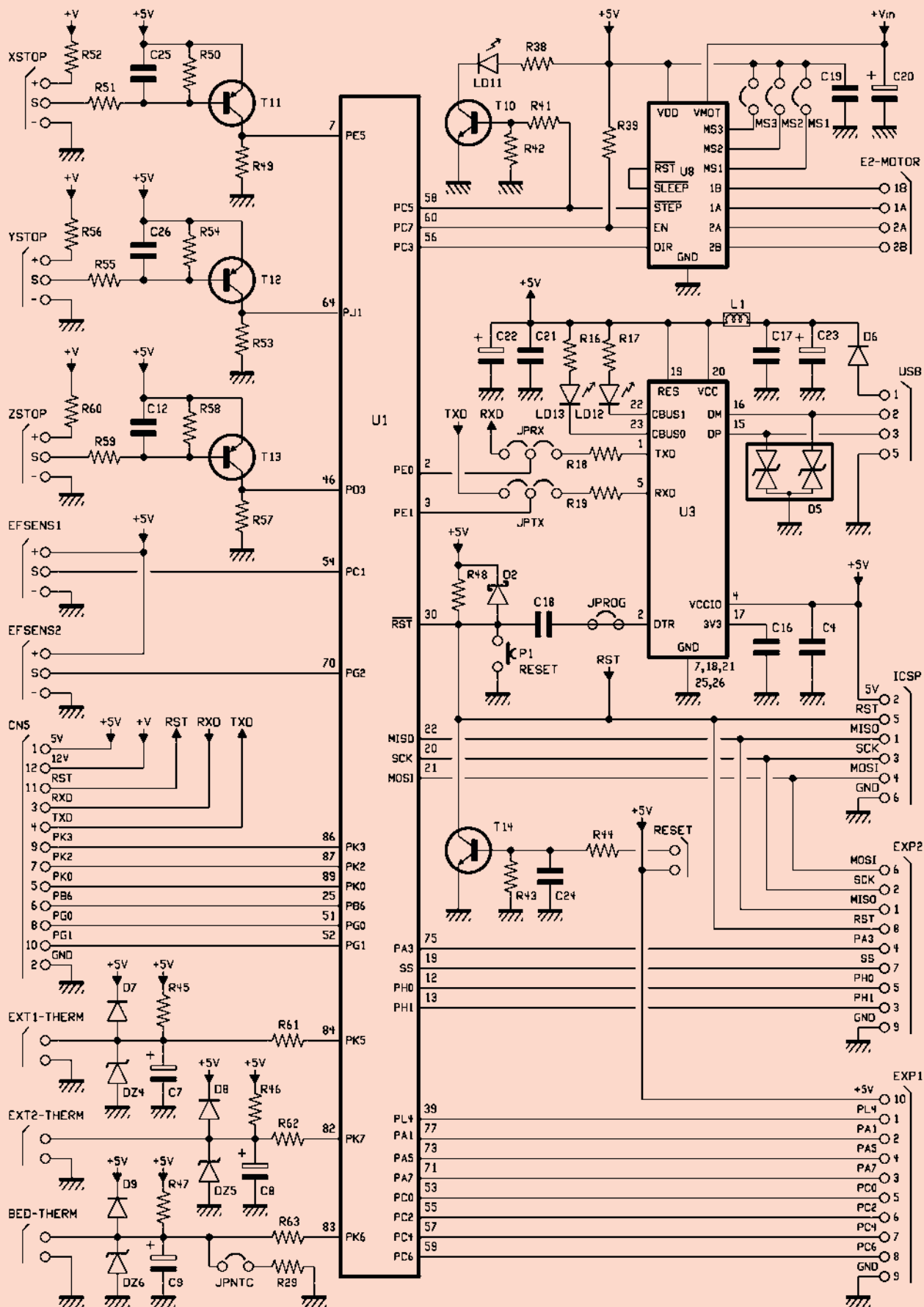


Tabella 1
Connessioni
connettore CN1.

PIN	COLLEGAMENTO
1	+12 V
2	+12 V
3	+12 V
4	+12 V
5	PIN-- EXT1 - HEATER
6	PIN-- EXT1 - HEATER
7	PIN-- EXT1 - HEATER
8	PIN-- EXT1 - HEATER
9	EXT1 - THERM
10	GND
11	+12 V
12	FAN1-
13	GND
14	+12 V
15	S DI Y STOP
16	GND
17	S DI X STOP
18	GND
19	X MOTOR 2A
20	X MOTOR 1A
21	X MOTOR 2B
22	X MOTOR 1B
23	Y MOTOR 1A
24	Y MOTOR 2A
25	Y MOTOR 1B
26	Y MOTOR 2B

di Id) consente di minimizzare la potenza dissipata e perciò di rinunciare al dissipatore, anche in virtù delle correnti molto ridotte che devono erogare (nell'ordine dei 2÷2,5 ampere per ogni estrusore, molto meno per le ventole). Fa eccezione T5, dedicato al piatto riscaldato, che come già accennato piloterà l'elemento esterno cui è demandata l'alimentazione del relativo riscaldatore. T3, T4 e T5 vengono pilotati sul gate con il segnale logico invertito da una porta NOT e amplificato da un transistor NPN, la cui funzione è scaricare rapidamente la capacità di gate quando il MOSFET

Tabella 2
Connessioni
connettore CN2.

PIN	COLLEGAMENTO
1	GND
2	+BED - SERVO
3	S BED - SERVO
4	S BED - MIN
5	+12 V
6	+5 V

deve spegnersi.

Diamo ora uno sguardo agli ingressi: Ystop, Xstop e Zstop sono gli ingressi degli switch finecorsa per i rispettivi assi, tale e quale che nel controller precedente; ma qui abbiamo introdotto una novità, ossia EFSENS1 ed EFSENS2, che sono sempre ingressi per microswitch e sono stati inseriti per rilevare quando il filamento finisce. Praticamente sono da collegare ai sensori di fine filo posti vicino alla guida-filo, ognuno dei quali incorpora un microswitch a levetta che appoggia sul filamento che lo attraversa: finché c'è il filamento la levetta viene mantenuta premuta, mentre quando finisce, la levetta ritorna in posizione di riposo e lo switch commuta. Questi finecorsa vengono rilevati dalle linee, rispettivamente, PC1 e PG2 del microcontrollore.

Per quanto riguarda i finecorsa dello spostamento degli assi, Xstop viene letto da PE5, Ystop da PJ1 e Zstop da PD3. Per ciascun finecorsa sono stati previsti tre contatti: +, S e -. Questa soluzione è stata scelta per poter collegare sia finecorsa di tipo elettromeccanico (microswitch e vari) sia rilevatori ottici, composti da un fotodiodo e un LED ad infrarossi affacciati tra loro in una cavità: quando il carrello arriva, una linguetta fissata ad esso passa nella cavità e blocca la luce che dovrebbe raggiungere il fotodiodo, determinando l'attivazione del finecorsa. Nel nostro caso, siccome utilizziamo degli switch finecorsa li colleghiamo ognuno tra S e - (massa) così la condizione di arrivo a fine corsa coincide con il livello alto sul rispettivo input.

Nel caso si voglia utilizzare finecorsa alimentati, va notato che il polo + del connettore riceve la tensione a valle del diodo di protezione D1 sulla linea di alimentazione generale della scheda ossia +V (sarà circa 12V o 24V in base alla tensione applicata alla morsettiera PWR, come spiegato fra breve). FAN1 e 2 sono le uscite per il controllo delle due ventole di raffreddamento del filamento relative ai due estrusori: FAN1 viene pilotata dalla linea PH5 e FAN2 dalla PE4 dell'U1.

Altra novità di questa scheda è la presenza di un quinto driver per motori stepper, perché in questo caso c'è da gestire il motore del secondo estrusore; i driver sono gli stessi della precedente unità. Dunque, nello schema elettrico vediamo cinque uscite per il comando di stepper-motor, che sono:

- E1-MOTOR (associata a U4) per il comando del motore spingi-filamento dell'estrusore 1;
- E2-MOTOR (associata a U8) per il comando del motore spingi-filamento dell'estrusore 2;

- X-MOTOR (associata a U6) per il comando del motore dell'asse X;
- Y-MOTOR (associata a U5) per il comando del motore dell'asse Y;
- Z-MOTOR (associata a U7) per il comando del motore dell'asse Z.

Il circuito monta cinque moduli driver per motori passo-passo prodotti dalla Futura Elettronica (cod. 3DDRIVER); ciascuno è basato su un integrato A4988 della Allegro, che può essere impostato per definire sia la direzione di rotazione dell'albero del motore, sia il numero di gradi che il rotore del motore deve compiere alla ricezione di ogni comando; in altre parole, possiamo decidere se quando gli forniamo un impulso di comando, il modulo deve far ruotare l'albero di uno step alla volta, oppure di 1/2, 1/4, 1/8 o 1/16, in base all'accuratezza che si desidera ottenere. Supporta, perciò, la modalità microstep.

Il motore si muove, ad ogni impulso che l'ATmega invia su /STEP, in base all'impostazione di MS1, MS2, MS3; nel nostro caso, avendo impostato il massimo fattore di divisione, per un passo completo occorrono 16 impulsi, quindi con i nostri NEMA 17 un giro completo richiede al microcontrollore 3.200 impulsi.

Notate che la linea /STEP di ciascun controller è stata collegata ad un transistor NPN, che attraverso un resistore pilota un LED, il quale pulserà analogamente alla corrispondente linea di comando del microcontrollore in modo da fornire un'indicazione visiva dell'attività del relativo azionamento; questo permette, nel caso durante il test (o l'uso) un motore non giri malgrado il relativo LED pulsi, vorrà dire che il problema andrà cercato nel motore o nei relativi cablaggi.

Resta inteso che siccome il LED pulsa alla stessa frequenza degli impulsi di comando, potremo vederlo lampeggiare solo quando il motore corrispondente girerà a velocità molto ridotta, dato che già a 25 Hz l'occhio umano vede il diodo sempre illuminato.

Procediamo con le novità di questa scheda controller andando alle uscite: abbiamo la linea PG5 che pilota il contatto S del connettore BED/servo; questo connettore è multifunzione per futuri sviluppi e permette sia di gestire un eventuale driver per il controllo PWM della temperatura del piatto riscaldato, sia la gestione di un servocomando un servomotore per ulteriori sviluppi. Ad esso corrisponde l'ingresso PA0.

Vediamo poi l'ingresso BED/MIN, che è un ingresso

PIN	COLLEGAMENTO
1	+12 V
2	+12 V
3	+12 V
4	+12 V
5	PIN-- EXT1 - HEATER
6	PIN-- EXT1 - HEATER
7	PIN-- EXT1 - HEATER
8	PIN-- EXT1 - HEATER
9	EXT1 - THERM
10	GND
11	+12 V
12	FAN1-
13	GND
14	+12 V

normalmente aperto con pull-up che permette di fornire un feedback generico, anche questo per implementare applicazioni future, come ad esempio il livellamento assistito del piatto di stampa.

Va notato che per rendere più versatile la scheda, tutte le linee dei singoli connettori sinora descritti sono anche raggruppate in comodi connettori maschi per flat-cable, secondo quanto esposto nelle tabelle dalla **Tabella 1** alla **Tabella 4**.

Altra cosa interessante di questa scheda riguarda lo stadio di alimentazione, perché qui abbiamo predisposto due ingressi (sulla morsettiera PWR) sia per 24V che per 12 volt gestiti con un jumper (JVPWR), che in realtà è già chiuso per impostazione predefinita sul PCB sull'ingresso 12 volt; volendo dare l'alimentazione dall'ingresso 24 volt, bisogna quindi tagliare la pista sotto il jumper del

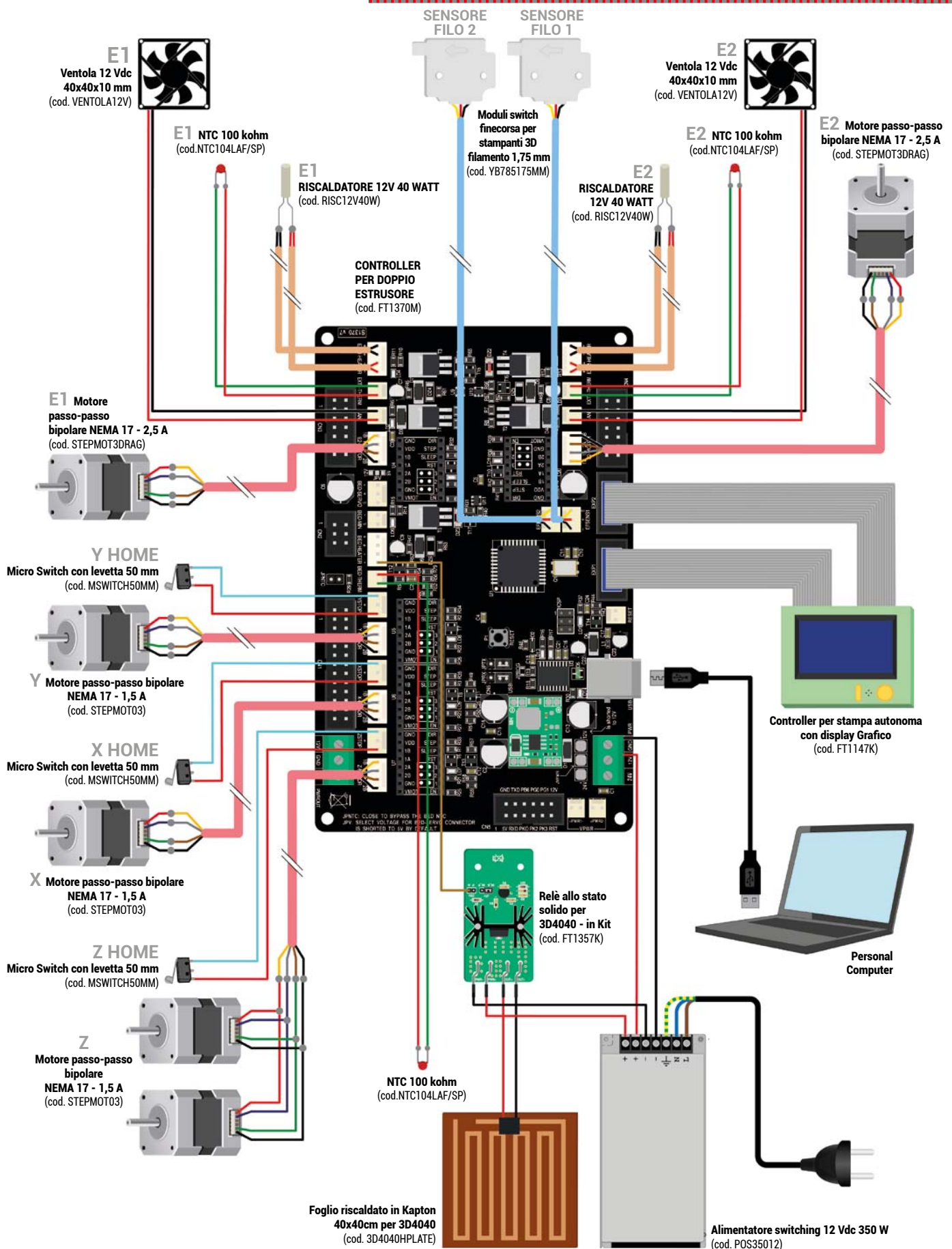
PIN	COLLEGAMENTO
1	+12 V
2	+12 V
3	+12 V
4	+12 V
5	PIN--EXT2 - HEATER
6	PIN--EXT2 - HEATER
7	PIN--EXT2 - HEATER
8	PIN--EXT2 - HEATER
9	EXT2-THERM
10	GND
11	+12 V
12	FAN2-
13	GND
14	+12 V

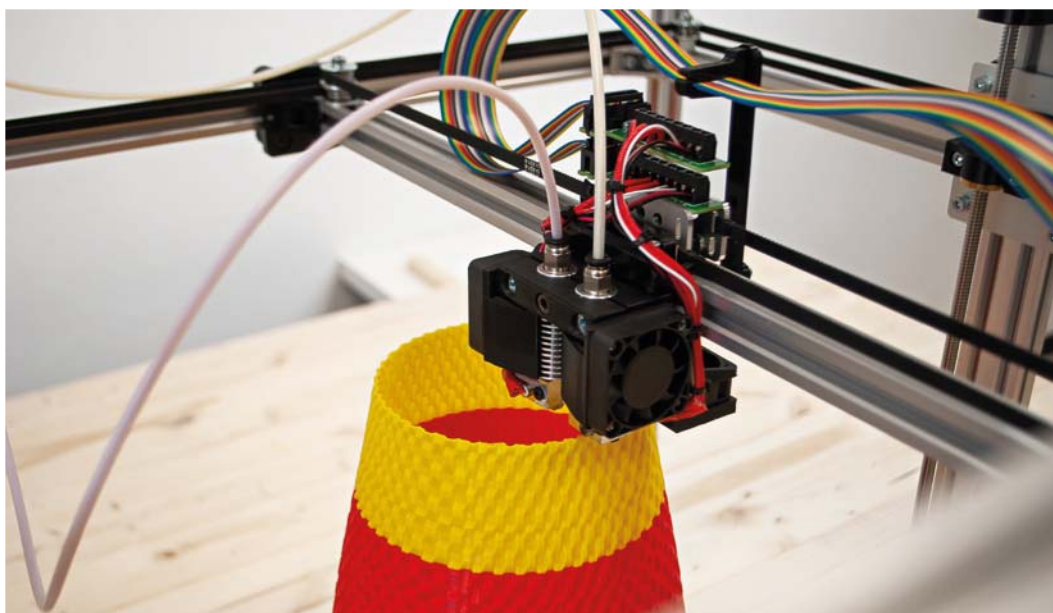
Tabella 3
Connessioni
connettore
CN3.

Tabella 4
Connessioni
connettore
CN4.



schema di CABLAGGIO





La testa di stampa a due estrusori.

12 volt e chiudere il jumper del 24 V. Abbiamo anche adottato un nuovo regolatore di tensione (U2), che non è più un semplice integrato ma un Modulo DC/DC Step Down 5V/3A MP1584EN (sempre commercializzato dalla Futura Elettronica con il cod. MP1584ENDCDC) che ricava 5V stabilizzati a prescindere dal valore della tensione d'ingresso, purché compresa fra 5 e 25V. L'uscita del modulo alimenta la linea 5V che fornisce tensione alla logica della scheda ed al polo 5V del connettore PWROUT dove arriva anche il +12V dell'ingresso PWR; tale connettore è utile per fornire alimentazione ad eventuali circuiti aggiunti alla stampante. Concludiamo l'analisi circuitale con i connettori CN5, EXP1 ed EXP2 (espansioni, riservati a eventuali sviluppi futuri ed al pannello di controllo opzionale stand-alone) ed ICSP, da utilizzare per l'eventuale programmazione in-circuit e comunque usato in fabbrica per la scrittura del bootloader del microcontrollore. Il connettore ICSP riporta l'alimentazione a 5 volt, nonché le linee MISO, MOSI ed SCK dell'SPI, oltre al RESET del microcontrollore. Le stesse linee le ritroviamo, insieme agli I/O dell'ATmega2560 sul connettore EXP2; ad esempio, sfruttando le linee del bus SPI (MISO, MOSI ed SCLK) potete collegare un lettore di SD-Card, mentre con sette degli I/O rimasti liberi potete collegare il pannello di controllo stand-alone dotato di display (FT1147K).

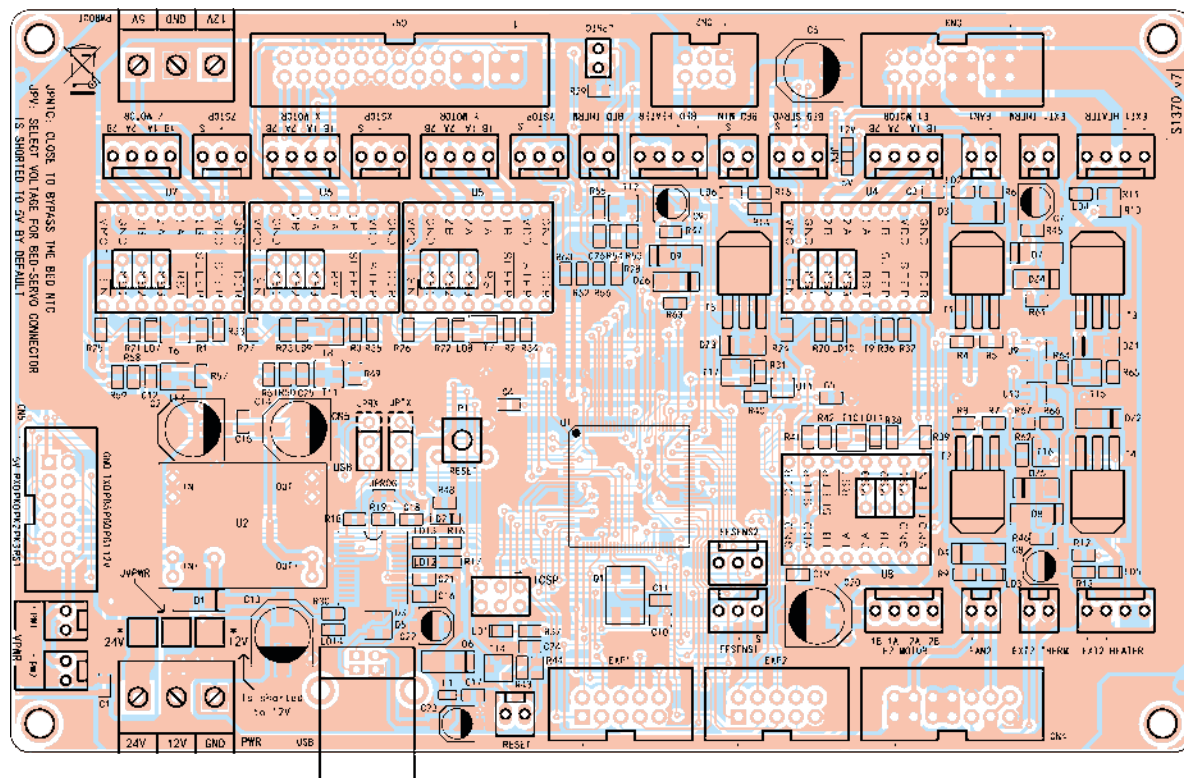
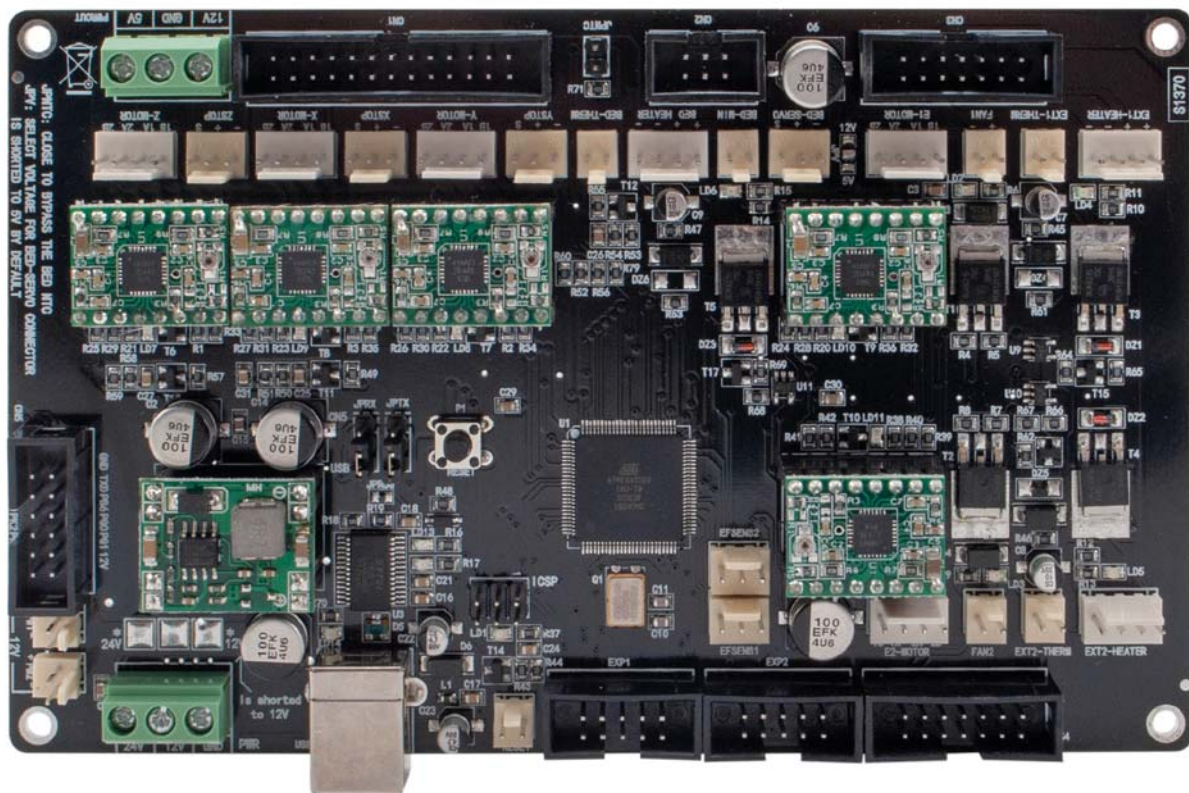
REALIZZAZIONE PRATICA

Dopo questa dettagliata descrizione dello schema elettrico del circuito della scheda controller, pas-

siamo adesso alla realizzazione della stampante, che consta di una meccanica e dell'elettronica necessaria, praticamente tutta racchiusa nella scheda controller.

Partiamo dunque dalla realizzazione di quest'ultima, esponendo qui di seguito alcune note costruttive inerenti alla scheda e poi, nella seconda ed ultima puntata vedremo come modificare la meccanica della stampante per applicare la seconda testa di stampa con relativo motore passo-passo alimentatore del filamento. Per chi non se la sentisse o non avesse i mezzi per realizzare la scheda controller, ricordiamo è stata prevista la versione già montata e pronta con il bootloader e il firmware caricati.

Chi vuole autocostruirla potrà farlo attenendosi alle indicazioni fornite qui di seguito: per prima cosa bisognerà preparare il circuito stampato mediante fotoincisione, scaricando i file delle tracce rame dal nostro sito www.elettronica.in.it insieme ai file del progetto. Inciso e forato il PCB si potranno montare tutti i componenti partendo dagli SMD, che sono la gran parte; allo scopo sono necessari degli strumenti e delle attrezzature specifici per il montaggio superficiale, soprattutto di componenti con pin ad elevata densità come il microcontrollore ATmega2560 che deve essere saldato mediante una stazione ad aria calda dopo aver spalmato con della pasta flussante le piazzole corrispondenti. Per i componenti restanti si può procedere con un saldatore da 25-30 W e del filo di stagno molto sottile, aiutandosi con il flussante per facilitare adesione e posizionamento delle saldature. Bis-



Elenco Componenti:

R1, R2, R3, R24, R25, R26, R27, R28, R31, R36, R39, R42, R43, R49, R53, R57, R65, R67: 10 kohm (0805)
R4, R7, R18, R19: 10 ohm (0805)
R5, R8, R29: 100 kohm (0805)
R6, R9, R11, R13, R15: 1.8 kohm (0805)
R10, R12, R14, R30, R32, R33, R34, R35, R40, R41, R44, R45, R46, R47, R48, R61, R62, R63, R64, R66: 4,7 kohm (0805)
R16, R17: 470 ohm (0805)
R20, R21, R22, R23, R37, R38: 330 ohm (0805)
R50, R51, R52, R54, R55, R56, R58, R59, R60: 1 kohm (0805)
C1, C3, C12, C16, C17, C18, C19, C21, C24, C25, C26: 100 nF ceramico (0805)
C4, C5, C15: 1 µF 6,3 VL ceramico
C2, C6, C13, C14, C20: 100 µF 25 VL elettrolitico (E)
C7, C8, C9: 10 µF 35 VL elettrolitico (B)
C10, C11: 22 pF ceramico (0805)
C22, C23: 4,7 µF 35 VL elettrolitico (B)
D1, D7, D8, D9: GFIM (SMA)
D2: BAT42W (SOD-123)
D3, D4, D6: MBRA140TRPBF (DO-214AC)
D5: PGB102ST23WR (SOT-23)
T1, T2, T3, T4, T5: BUK6215-75C (SOT-428)
T6, T7, T8, T9, T10, T14, T15, T16, T17: BC817 (SOT-23)
T11, T12, T13: BC807 (SOT-23)
U1: ATMEGA2560-16AU
U2: Modulo DC Step Down 5V/3A MP1584EN
U3: FT232RL (SSOP28)
U9, U10, U11: SN74LVC1G14DBVR (SOT-23)
LD1, LD12, LD14: LED verde (0805)
LD2, LD3, LD4, LD5, LD6, LD7, LD8, LD9, LD10, LD11, LD13: LED rosso (0805)
P1: Microswitch
Q1: Quarzo 16 MHz (C7S)
DZ1, DZ2, DZ3: BZV55-C10
DZ4, DZ5, DZ6: non montare
L1: BK0603HS330-T (0603)
PWR, PWROUT: Morsetto 3 poli P 5,08 mm
USB: Connettore USB-B
BED-THERM, BED-MIN, FAN1, EXT1-THERM, EXT2-THERM, FAN2, RESET, PW1, PW2: Connettore maschio Board to wire 2 poli
ZSTOP, XSTOP, YSTOP, BED-SERVO, EFSENS1, EFSENS2: Connettore maschio Board to wire 3 poli
Z-MOTOR, Y-MOTOR, X-MOTOR, BED-HEATER, E1-MOTOR, E2-MOTOR, EXT1-HEATER, EXT2-HEATER: Connettore maschio Board to wire 4 poli
CN2: 3+3 POD Connector for CS
EXP1, EXP2: 5+5 POD Connector for CS
CN5: 6+6 POD Connector for CS
CN3, CN4: 7+7 POD Connector for CS
CN1: 13+13 POD Connector for CS
ICSP: Pin strip male 2x3 way
JPNTC: Pin strip male 2 way
U4, U5, U6, U7, U8: Pin strip female 8 way
JPRX, JPTX: Pin strip male 3 way

Varie

- Jumper
- circuito stampato S1370_v7 (157x99 mm)

gna prevedere strip femmina per il montaggio dei moduli driver e dei vari strip per DC/DC, connessioni ecc.

Ottenuta o realizzata la scheda, si procede con l'inserimento dei moduli driver (che possono essere realizzati autonomamente o acquistati già pronti presso Futura Elettronica, www.futurashop.it) nei rispettivi zoccoli basati su connettori S.I.L. a passo 2,54 mm.

Per poter utilizzare la scheda bisogna caricare il bootloader (cioè il firmware di base) nell'ATmega 2560, utilizzando allo scopo una delle tecniche disponibili, vale a dire collegando un programmatore per microcontrollori Atmel al connettore siglato ICSP e operando da Personal Computer, ovvero ricorrendo alle tecniche di programmazione standalone attraverso l'impiego di una scheda Arduino come programmatore.

Ricordiamo che se si acquista la scheda controller nella versione montata e collaudata, il firmware bootloader è già caricato in fabbrica.

CONCLUSIONI

A questo punto ci fermiamo con la descrizione; proseguiremo nella seconda e conclusiva puntata di questo articolo dedicato alla stampa a due colori con la 3D4040, nella quale vedremo come modificare la meccanica della stampante per applicare la seconda testa di stampa con il relativo motore passo-passo e il meccanismo alimentatore del filamento. Spiegheremo altresì come impostare il client di stampa per fargli gestire la nostra macchina affinché possa stampare a due filamenti. Voi intanto fatevi trovare pronti, realizzando la scheda controller e tenendola disponibile per l'installazione sulla macchina! ■



Cosa occorre?

Il materiale utilizzato in questo articolo è disponibile presso Futura Elettronica. Il kit di espansione (cod. DUAL3DKIT) per aggiungere un estrusore alla stampante 3D 3D4040 è in vendita al prezzo di Euro 140,00. Il kit comprende un sensore finefilo (cod. YB785175MM, disponibile anche separatamente a Euro 9,00), il motore stepper con spingifilo, la testina riscaldante con sensore di temperatura e tutte le minuterie e accessori per trasformare la 3D4040 (disponibile in kit a Euro 999,00) in una stampante 3D con doppio estrusore. La board di controllo, completa di driver, è disponibile separatamente con il cod. FT1370M a Euro 119,00. 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>



HDMI
↓
VGA



€ 11,50

Cod. **BOXHDMITOVGA**

CONVERTITORE DA HDMI A VGA 1080P PLUG AND PLAY

Permette di convertire un segnale digitale HDMI (compatibile con HDMI 1.3) 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**

REALIZZAZIONE E VERIFICA DI UN CROSSOVER PASSIVO



dell' Ing. **VINCENZO MENDOLA**

Costruiamo e testiamo un semplice crossover passivo a tre vie e verifichiamo la rispondenza delle simulazioni con le misure effettuate sul prototipo.

C

hiunque si sia, a vario titolo, occupato di impianti audio di una certa complessità, prima o poi si è sicuramente imbattuto nei crossover e chi si è anche occupato di autocostruzioni, avrà sicuramente approfondito la propria conoscenza di questi filtri, per accrescere le nozioni al riguardo, in modo da poter scegliere un

modello adatto al proprio sistema, oppure per determinare i componenti necessari per l'autocostruzione dello stesso. Lo scopo di questo articolo è quello di fornirvi qualche suggerimento per stimolarvi a fare qualche test in prima persona e magari provare ad autocostruire una parte del vostro impianto audio, fornendovi qualche idea e qualche indicazione di massima.

Ma cosa sono e a cosa servono i crossover?

Precisiamo innanzitutto che ne esistono di due tipologie: i

crossover passivi ed i crossover attivi. I crossover più comuni, quelli più largamente diffusi ed utilizzati, sono di tipo passivo.

CROSSOVER PASSIVO

All'interno di un sistema di diffusione audio a più vie, in cui viene utilizzato un unico finale di potenza che si occupa di amplificare l'intero spettro del segnale da riprodurre, vengono utilizzati i crossover passivi, che hanno lo scopo di dividere lo spettro audio generalmente in 2 o 3 sottobande, ognuna delle quali ricade all'interno dell'intervallo di frequenze che il singolo diffusore è in grado di riprodurre: il woofer si occupa delle basse frequenze, generalmente tra poche decine di hertz fino a pochi kilohertz (50 Hz – 2 KHz); il midrange è in grado di riprodurre da poche decine di hertz a vari kilohertz (100 Hz – 6 KHz), ed infine abbiamo il tweeter, che può riprodurre le frequenze a partire da pochi kilohertz fino all'estremo superiore delle frequenze udibili dall'essere umano (2.5 KHz – 20 KHz).

Esistono inoltre i subwoofer, in grado di riprodurre una gamma stretta di frequenze molto basse (28 Hz – 150 Hz), oppure i super tweeter, utilizzati per riprodurre la parte più alta delle frequenze udibili ed in grado persino di andare oltre i 20 KHz. Questi sono ovviamente dati generici ed indicativi, dipende molto dalle caratteristiche costruttive del singolo modello per cui è sempre necessario fare riferimento alla scheda tecnica dello specifico componente per conoscere l'intervallo di frequenza in cui questo è utilizzabile.

Un'altra grandezza a questo associata che è importante conoscere, se non indispensabile, per il corretto dimensionamento dell'impianto e quindi anche del crossover, è l'impedenza caratteristica. Esiste uno standard internazionale (IEC 60268-5) che definisce le modalità con cui vanno effettuate le misure per determinare i parametri principali di un diffusore acustico. Purtroppo, però, non sempre il costruttore fornisce la curva dell'impedenza ma soltanto il suo valore nominale.

Ricordiamo che lo spettro dei segnali udibili da parte dell'essere umano, con una normale funzionalità uditiva, comprende le frequenze tra i 20 Hz ed i 20 KHz; l'intervallo 300 Hz – 3400 Hz è la cosiddetta "banda fonica", utilizzata in telefonia in quanto racchiude la maggior parte del contenuto spettrale della voce umana e corrisponde all'intervallo di frequenza nel quale la sensibilità dell'orecchio umano è maggiore, caratteristica che si è affermata a causa dei vantaggi evolutivi associati, legati all'indiscussa importanza della comunicazione verbale e della sua

comprensione.

Gli animali, che a loro volta hanno realizzato diverse forme di comunicazione sonore non verbali, sono in grado, a seconda della specie, di udire frequenze inferiori a quelle dell'orecchio umano (ad esempio gli elefanti e le balene) e superiori, come ad esempio i canidi, di cui fa parte anche il cane, in grado di sentire suoni molto acuti, tipici delle loro prede più piccole, che a noi risultano completamente non udibili. In questo caso la funzionalità dell'estensione dello spettro uditivo è duplice: da una parte per motivi prettamente comunicativi con membri della stessa specie, dall'altro come utile supporto durante la predazione.

Finora abbiamo visto il caso più semplice e comune di crossover, quello passivo, che è anche generalmente quello meno visibile in quanto nella stragrande maggioranza dei casi è posto all'interno della cassa acustica, per cui, se non si è a conoscenza dei dettagli di funzionamento dei sistemi di riproduzione audio, se ne potrebbe ignorare completamente l'esistenza. Da un lato è collegato ai singoli diffusori, e, come abbiamo visto, ha la funzione di suddividere l'intero spettro del segnale audio da riprodurre, nell'intervallo di frequenze riproducibili dal singolo elemento. Verso l'esterno sono visibili solamente i morsetti di collegamento della cassa acustica all'amplificatore: le connessioni tra il crossover ed i singoli diffusori, sono realizzate internamente. E' doveroso precisare che esistono anche diffusori a singola via, solitamente (ma non sempre) piuttosto economici, che rappresentano generalmente un compromesso tra costo, dimensioni e qualità: sono in grado di riprodurre al meglio solamente un intervallo limitato dello spettro udibile, per cui non necessitano di alcun tipo di filtro, in quanto sono le loro stesse caratteristiche fisiche a definire le frequenze riproducibili, caratteristiche che di fatto esse stesse determinano un filtraggio, con ovvi risvolti sulla qualità dell'audio emesso, specialmente ai due estremi dello spettro audio.

Anche in questo caso esistono delle eccezioni e si possono trovare diffusori a banda piuttosto larga, dal costo tutt'altro che modesto. Non esistendo la necessità di filtrare il segnale audio, il collegamento con l'amplificatore è fatto direttamente, senza l'utilizzo di alcun tipo di crossover.

Abbiamo appena visto che sono le caratteristiche elettromeccaniche dei diffusori a determinare le frequenze che possono essere riprodotte: i woofer hanno dimensioni maggiori; dovendo riprodurre le frequenze più basse hanno anche masse fisse e mobili maggiori, maggiore inerzia, un cono di

diametro generalmente tra i 20 ed i 25 cm e a volte anche maggiore, tutte caratteristiche che non gli permettono di riprodurre frequenze molto superiori al chilohertz o poco più (ed in alcuni casi anche inferiori); i tweeter invece, che devono muoversi ad altissima frequenza, hanno masse mobili contenute e leggere, con dimensioni complessive piuttosto contenute; i midrange, dovendo lavorare all'interno delle frequenze che non sono riproducibili dalle precedenti tipologie, hanno dimensioni a queste intermedie.

Anche in questo caso abbiamo semplificato molto ma il concetto di massima dovrebbe essere chiaro.

CROSSOVER ATTIVO

Il crossover attivo si utilizza quando il segnale audio, prima di essere amplificato in potenza per essere applicato al sistema di diffusione acustica, viene filtrato selettivamente, in maniera attiva, mediante l'uso di circuiti elettronici contenenti amplificatori operazionali. A differenza dei filtri passivi, i filtri attivi possono modificare più facilmente la loro frequenza di taglio, generalmente regolando appositi trimmer, non è necessario pertanto cambiare i componenti; permettono di realizzare pendenze di filtraggio maggiori di quelli passivi, anche queste regolabili, e possono ottenere ordini elevati più facilmente, senza la necessità di utilizzare componenti di grandi dimensioni rispetto a quelli passivi che solitamente non superano il terzo, o, in casi particolari e abbastanza rari, il quarto ordine. I crossover passivi generalmente vengono utilizzati nella tipologia del secondo ordine che offre un buon compromesso tra complessità realizzativa, dimensioni e costi.

UN PO' DI TEORIA

Per semplificare, ricordiamo che l'ordine di un filtro è dato dal numero di componenti reattivi (condensatori o induttori) indipendenti, cioè il loro numero minimo una volta che si sia ridotta la rete in maniera tale che non siano presenti componenti dello stesso tipo, in serie o in parallelo.

Un filtro passa basso del primo ordine sarà realizzato con un solo induttore, un passa-alto semplicemente con un condensatore; un filtro del secondo ordine sia passa-basso sia passa-alto, avrà un induttore ed un condensatore, mentre un filtro del terzo ordine avrà due induttori ed un condensatore nella configurazione passa-basso mentre sarà costituito da due condensatori ed un induttore nella versione passa-alto (**Fig. 1**, **Fig. 2**, **Fig. 3**). Per i nostri test abbiamo realizzato un crossover passivo del secondo ordine a tre vie.

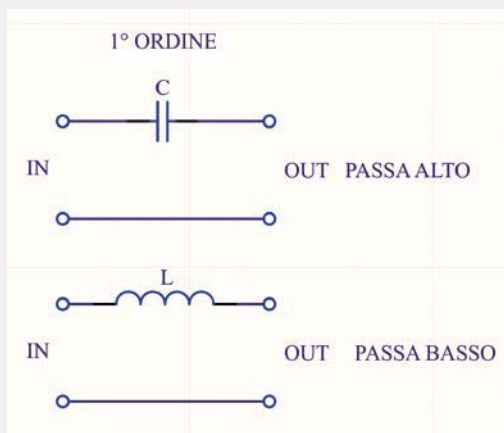


Fig. 1
Filtri passa alto e passa basso del 1° ordine.

Per ottimizzare le caratteristiche elettriche del filtro, abbiamo utilizzato solamente condensatori a film, più precisamente in polipropilene, i migliori dal punto di vista delle caratteristiche elettriche per applicazioni audio, per via del loro basso fattore di dissipazione e dei bassi valori dei parametri parassiti; per i test abbiamo utilizzato dei condensatori prodotti da un'azienda italiana, la ICEL, che ha a catalogo un gran numero di tipologie di condensatori a film.

Per quanto riguarda gli induttori, sono stati scelti del tipo avvolto in aria, per eliminare potenziali rischi di saturazione del nucleo ferromagnetico, pur accettando delle dimensioni maggiori (la permeabilità relativa μ_r dell'aria è di gran lunga inferiore a quella dei materiali ferromagnetici). Date le dimensioni non trascurabili di questi ultimi, si sono realizzati dei fori sul circuito stampato, in posizione opportuna da facilitare l'inserimento di una fascetta di nylon per il fissaggio dell'induttore; si consiglia di applicare uno strato di silicone in prossimità degli induttori, con lo scopo da tenere i rocchetti su cui sono avvolti in posizione, garantendo comunque una certa elasticità

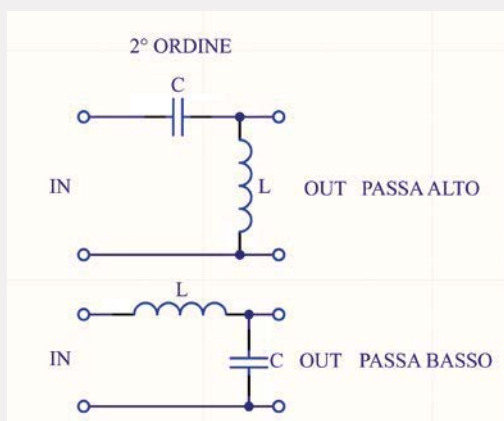
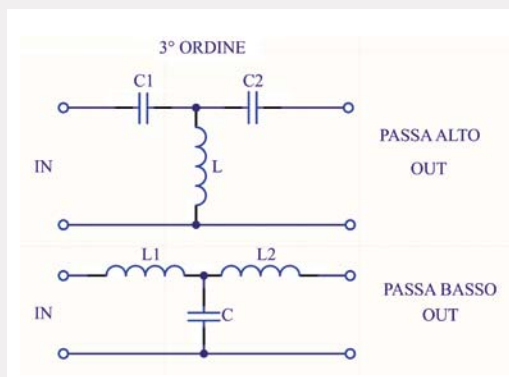


Fig. 2
Filtri passa alto e passa basso del 2° ordine.

Fig. 3
Filtri passa alto
e passa basso del
3° ordine.



per assorbire senza strappi eventuali piccole oscillazioni durante l'utilizzo ad alto volume d'ascolto. Dal punto di vista della configurazione progettuale, ci siamo limitati a dei filtri "standard": nella letteratura sono presenti esempi di configurazioni circuitali più complesse, utilizzate, tra le altre cose, al fine di equalizzare e compensare, per quanto possibile, la variazione dell'impedenza del diffusore connesso, al variare della frequenza.

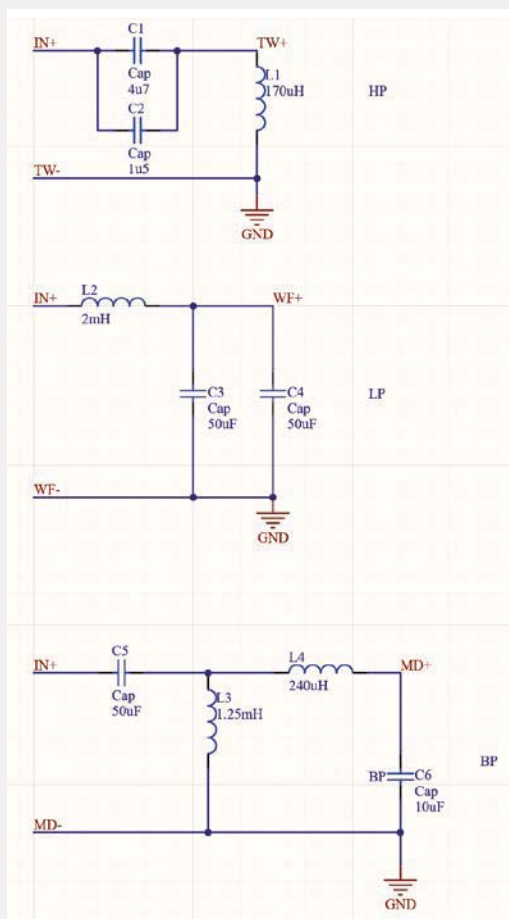


Fig. 4
Schema elettrico
del crossover.

SIMULAZIONE

In **Fig. 4** è visibile lo schema elettrico del crossover con i valori nominali dei componenti utilizzati.

In **Fig. 5** si può vedere il modello utilizzato per la simulazione con LTSPICE della risposta in frequenza nell'intervallo 20 Hz – 20 KHz ed il dettaglio della curva tra 100Hz e 20KHz con in evidenza le frequenze d'incrocio (**Fig. 6**).

Per ottenere un risultato reale, ricordatevi di "chiudere" le tre uscite del crossover con il carico corretto (4 ohm nel nostro caso); questa che stiamo facendo è chiaramente un'approssimazione, necessaria in quanto non abbiamo il modello dell'impedenza del diffusore, per cui siamo costretti a semplificare, considerandola una semplice resistenza corrispondente al suo valore nominale.

Come si può vedere dalle simulazioni, la frequenza di incrocio "bassa" (woofer-midrange) risulta essere 500Hz, mentre quella "alta" (midrange-tweeter) è pari a 4,3KHz; l'attenuazione non è perfettamente simmetrica: abbiamo circa 5 dB per la prima e 3,6 dB per la seconda; questi valori sono stati scelti tenendo conto delle caratteristiche di risposta in frequenza dei diffusori utilizzati per i test e dovranno essere da voi adattati in base agli elementi che sceglierete di collegare.

Ricordiamo che questi appena visti sono comunque dei valori puramente teorici, dato che stiamo analizzando un modello del nostro filtro, costituito da componenti reali che hanno una tolleranza di +/- 10%, perdite, parametri parassiti, etc., rispetto a quelli ideali che stiamo simulando; lo stesso diffusore, di cui non abbiamo il modello reale, ha nella realtà un'impedenza che, per semplificare, abbiamo dovuto approssimare con dei valori resistivi, 4 ohm nel nostro caso, trascurando del tutto la parte reattiva, che varia con la frequenza (ricordiamo che l'impedenza è $Z=R+jX$, dove R è la parte reale, costante, mentre la reattanza X è a sua volta uguale ad

$$X_C = \frac{1}{\omega C} = \frac{1}{2\pi f C}$$

nel caso di reattanza capacitiva ed a

$$X_L = \omega L = 2\pi f L$$

per la reattanza induttiva, funzione della frequenza f. Il modulo dell'impedenza è uguale a

$$|Z| = \sqrt{(R)^2 + (X)^2}$$

mentre la fase è uguale a

$$\varphi = \tan^{-1} \frac{X}{R}$$

L'impedenza Z è quindi funzione della frequenza, pertanto è evidente l'approssimazione fatta prece-

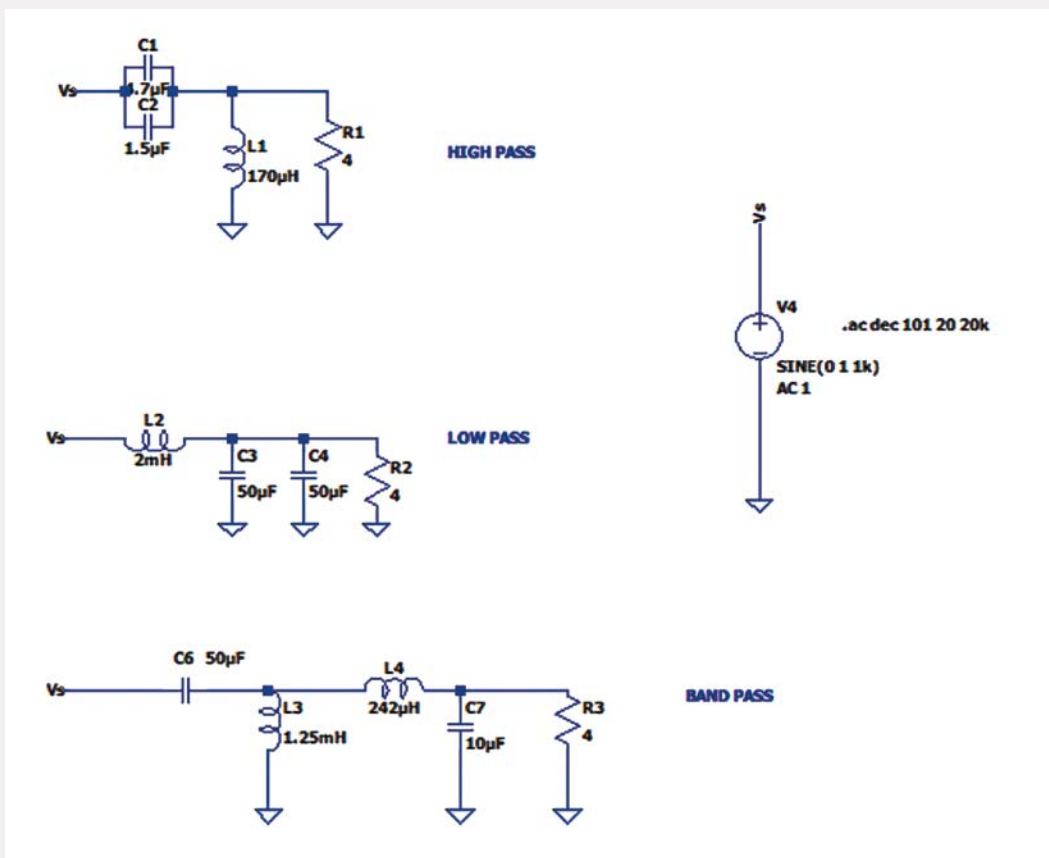
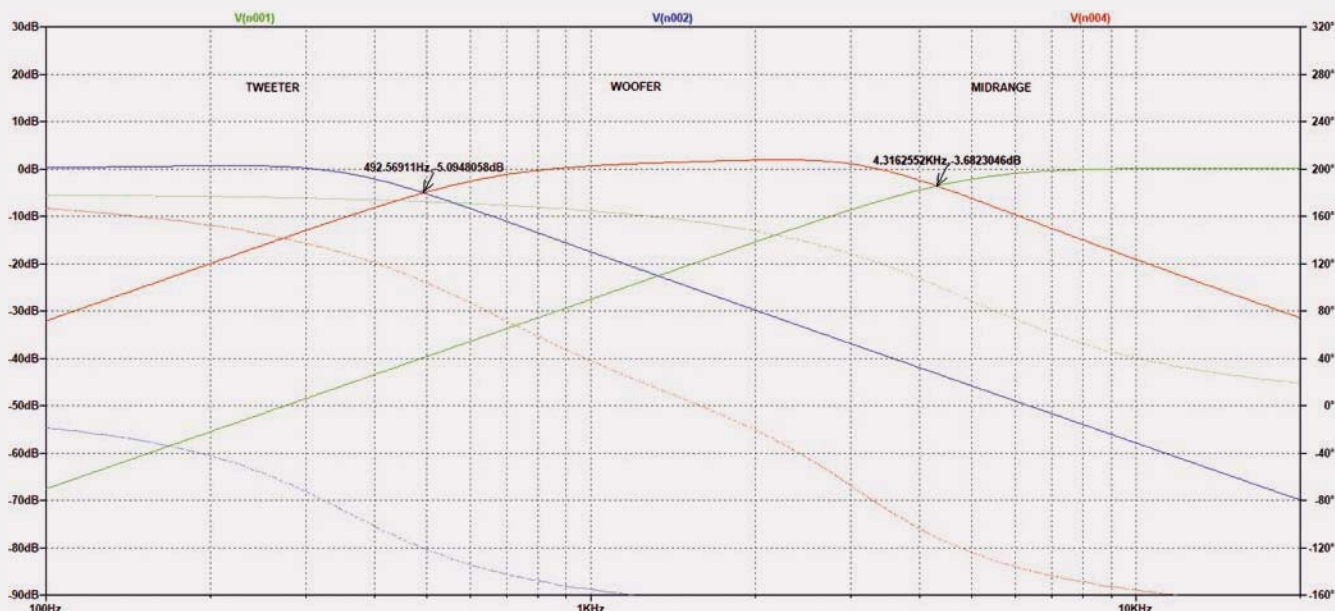


Fig. 5
Modello
utilizzato per
simulare il
crossover
con LTSPICE.

dentemente, comunque valida per i nostri scopi. In **Fig. 7** vediamo l'andamento dell'impedenza d'ingresso e delle singole sezioni del filtro (modulo e fase) in funzione della frequenza, sempre simulata

con LTSPICE della Analog Devices. Ricordiamo che anche in questo caso stiamo analizzando lo stesso modello di prima, con gli stessi limiti e le stesse semplificazioni.

Fig. 6
Dettaglio tra
100Hz e 20KHz
della risposta
in frequenza
del crossover
con evidenziate
le frequenze
d'incrocio.



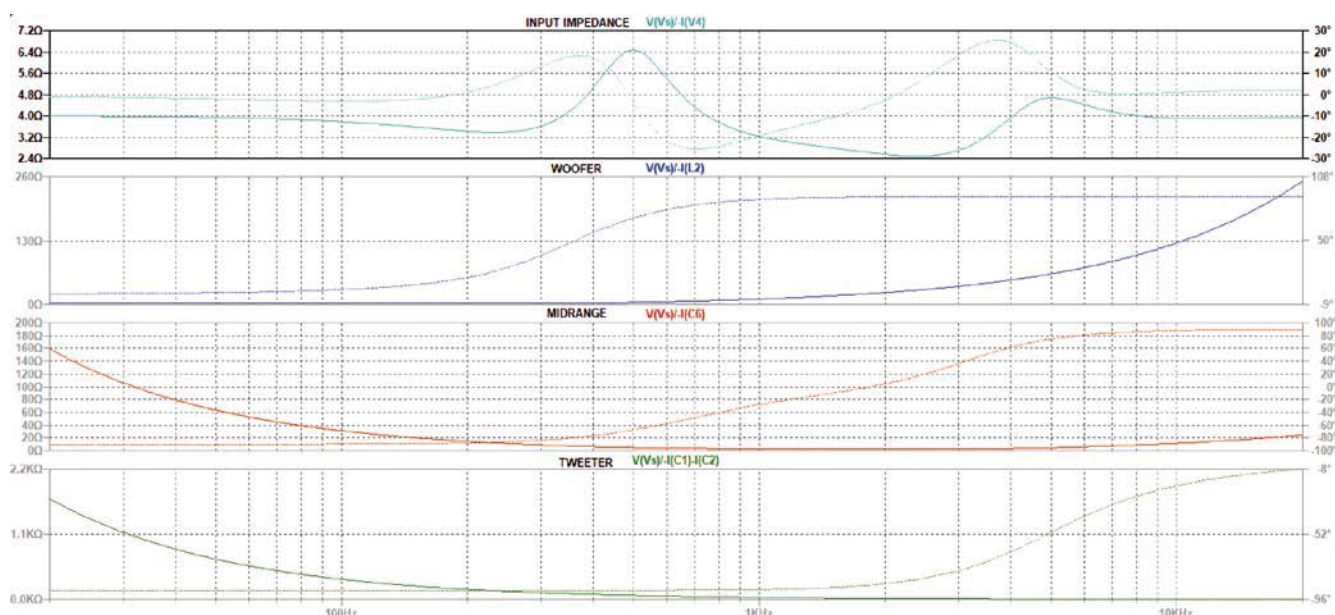


Fig. 7
Simulazione dell'impedenza (modulo e fase) del crossover in funzione della frequenza nell'intervallo 20Hz - 20KHz.

MISURE E CONFRONTO CON LA SIMULAZIONE

Possiamo ora verificare di quanto si scosta la nostra simulazione, dal prototipo che abbiamo realizzato (Fig. 8), utilizzando un generatore di segnali ed un oscilloscopio. I generatori di segnali (o di funzioni), non sono adatti per essere direttamente collegati a carichi a bassa impedenza come quello della nostra rete ma sono progettati per essere collegati a dispositivi ad alta impedenza o ad altra strumentazione pertanto hanno un valore dell'impedenza di uscita selezionabile tra "alta impedenza" oppure

pari al valore standard di 50 ohm; ovviamente non sono in grado di fornire in uscita valori di corrente e di potenza apprezzabili. Qualcuno potrebbe pensare di utilizzare un piccolo amplificatore in classe AB, interposto tra il generatore ed il carico, come ad esempio il diffusissimo LM386, magari in quanto già disponibile nel proprio laboratorio, essendo questo in grado di pilotare carichi di 4 ohm presentando inoltre una risposta in banda audio "piatta" (Fig. 9). Il problema è che questo amplificatore è in grado di erogare una potenza molto limitata e si rischierebbe di introdurre un'apprezzabile distorsione già

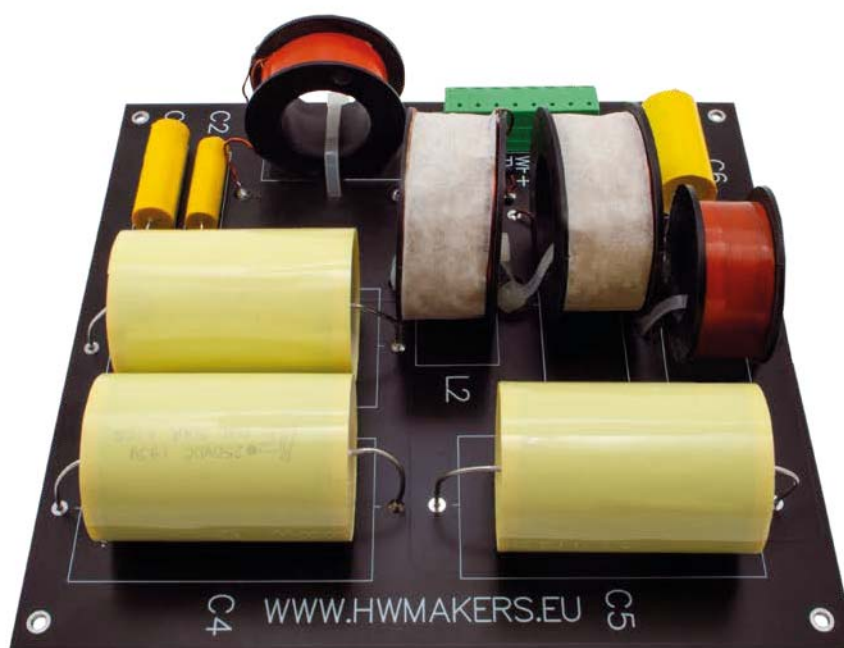
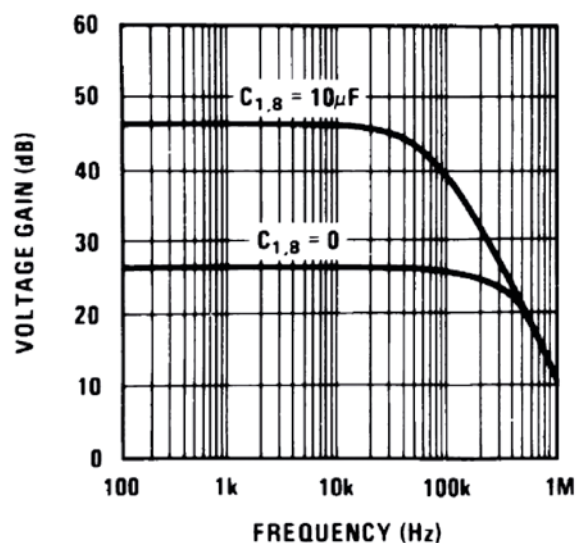
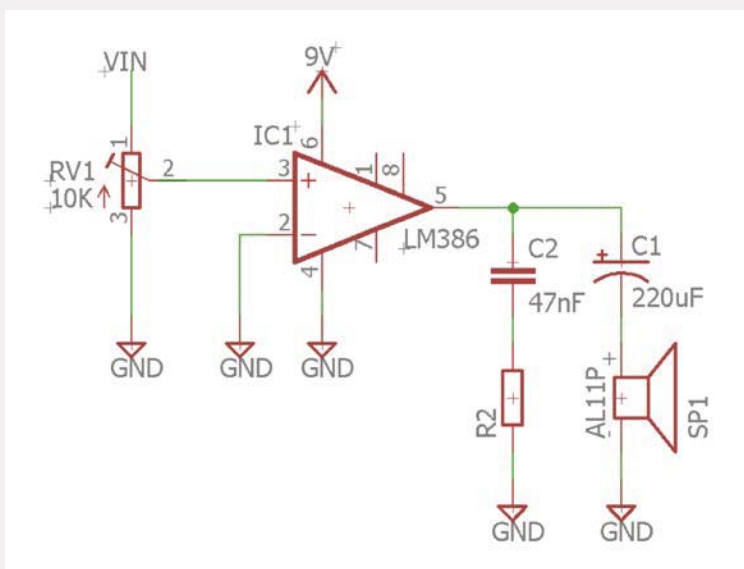


Fig. 8
Il nostro prototipo montato.



a basse potenze, meglio quindi utilizzarlo esclusivamente per applicazioni in cui è richiesta una potenza veramente modesta, un carico costituito da un unico, piccolo diffusore e non sono richieste distorsioni bassissime. Utilizziamo quindi per i test un amplificatore, autocostruito o di tipo commerciale, con una distorsione sufficientemente contenuta (la più bassa possibile) e una risposta "piatta" in banda passante, in grado di erogare almeno 10W RMS, in modo da essere certi di operare nelle condizioni di funzionamento ottimali che garantiscono, tra le altre cose, una distorsione trascurabile e un

riscaldamento minimo, quindi uno scostamento dei parametri di amplificazione trascurabile. Vediamo ora come procedere per effettuare le misure vere e proprie: dopo aver collegato il generatore di funzioni o di segnali all'ingresso dell'amplificatore appena visto, si collega la sua uscita all'ingresso del crossover e le singole uscite di quest'ultimo ad una resistenza, preferibilmente di tipo corazzato (anche se di fatto la potenza applicata al momento è comunque notevolmente inferiore al watt), facilmente reperibili nel valore di 4 ohm con tolleranza dell'1% (per i nostri scopi vanno bene

Fig. 9
Schema elettrico
amplificatore
con LM386 in
configurazione
 $G=20$ e risposta
in frequenza.

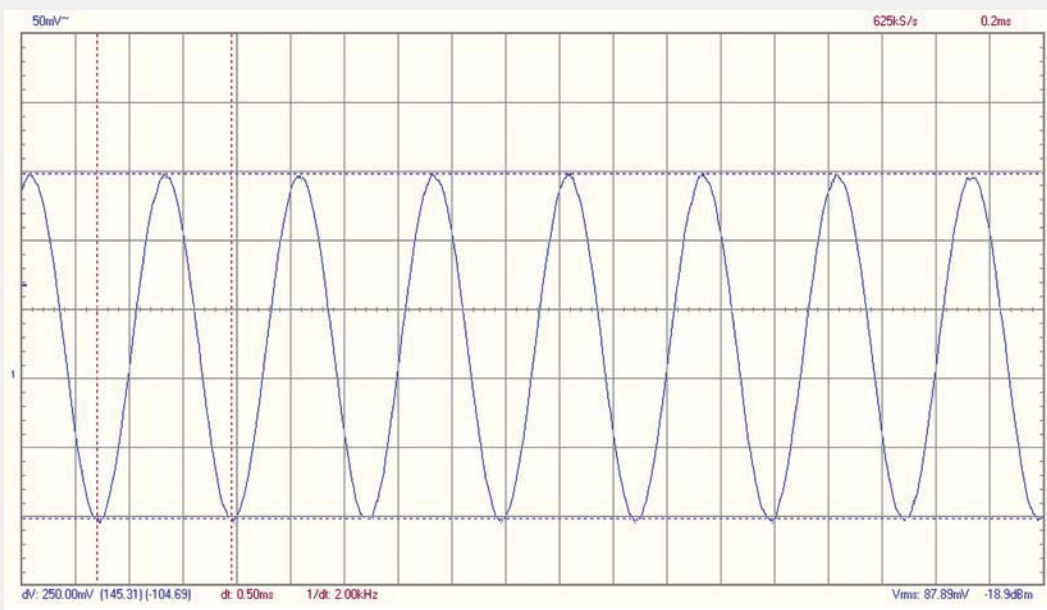


Fig. 10
Segnale
sinusoidale di
test 250mVpp
2KHz
misurato
su di un
resistore di
4 ohm.

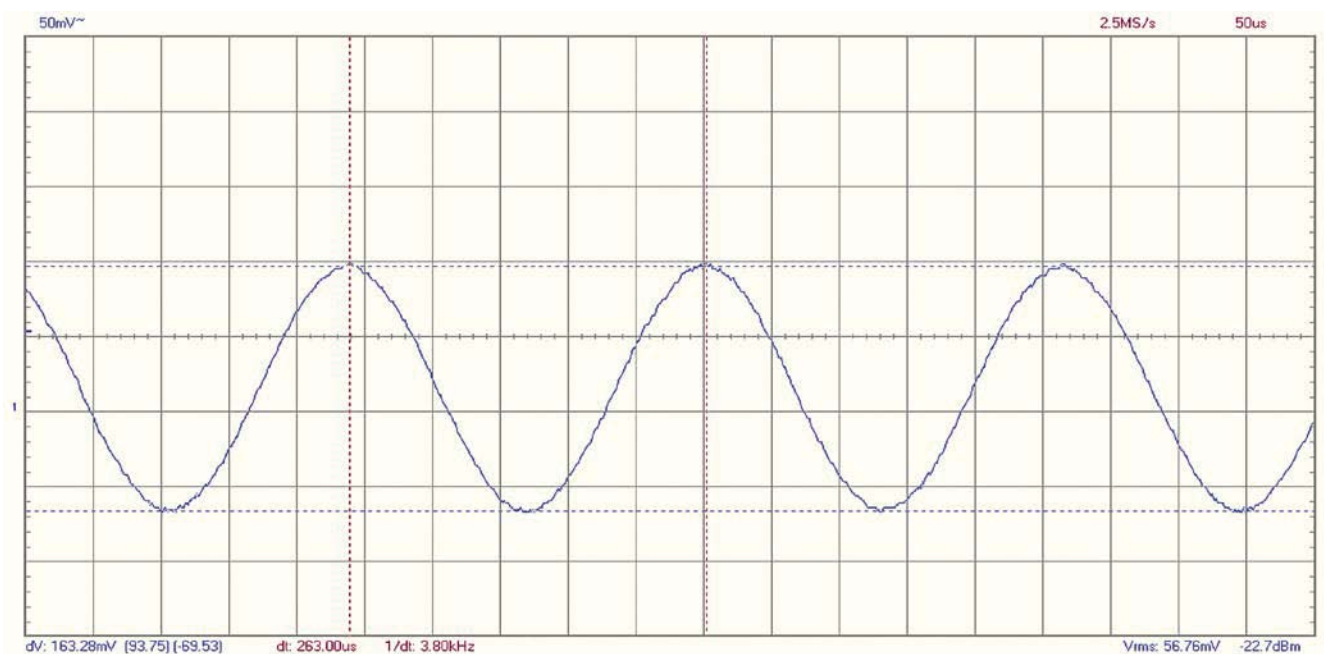
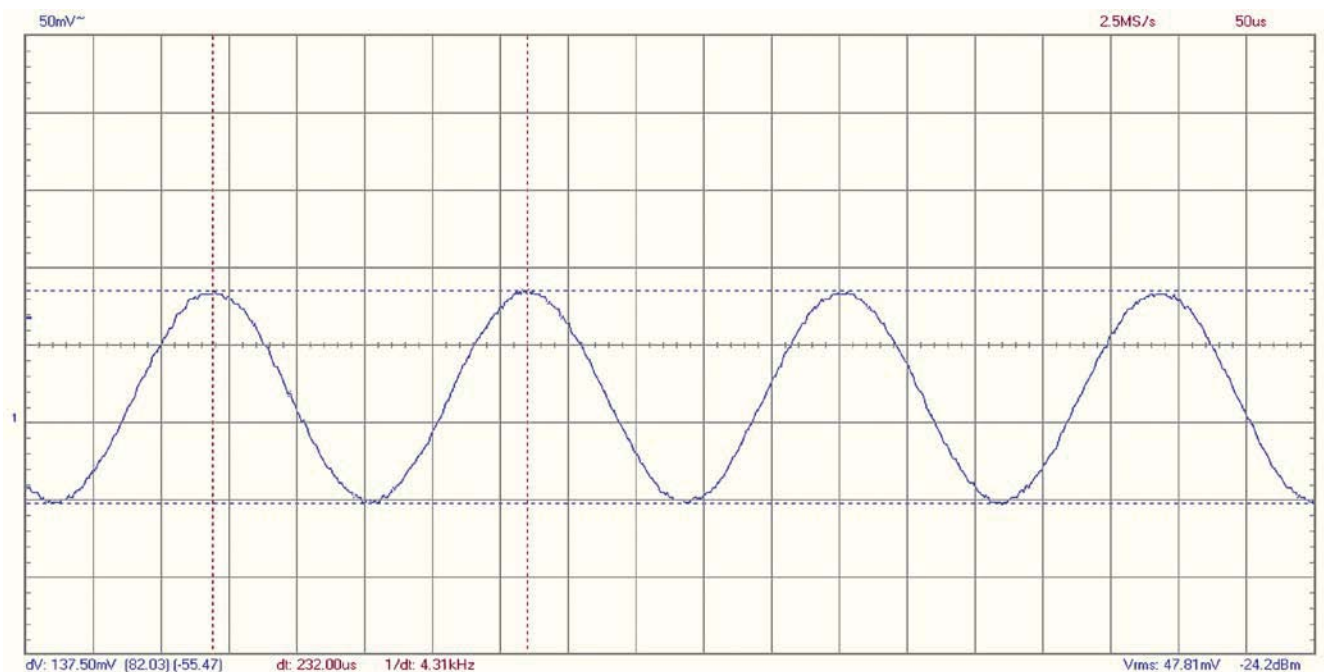


Fig. 11
Attenuazione a 4,3KHz con resistenza di 4ohm in uscita allo stadio midrange, maggiore di quella prevista dalla simulazione; l'attenuazione prevista di -3,68dB si ha in realtà a 3,8KHz.

anche tolleranze superiori); questo tipo di resistenze sono munite di contatti grandi e robusti, molto comodi per le connessioni da effettuare durante i test. Essendo di grande potenza, manterranno una temperatura costante durante i test, non alterando i risultati della misura; sono realizzati a filo ma per il nostro utilizzo possiamo considerare trascurabili gli effetti induttivi parassiti. Iniziamo ora con il verificare le frequenze d'incrocio: per prima cosa, colleghiamo direttamente all'uscita dell'amplifica-

tore, il nostro carico resistivo da 4 ohm e applichiamo all'ingresso dell'amplificatore un segnale sinusoidale con un'ampiezza picco-picco di 100mV in maniera da avere, ad esempio, 250mV in uscita alla frequenza di centro banda del midrange di 2KHz (**Fig.10**). Il valore della tensione applicata all'ingresso può anche essere differente ma deve rispettare il range ammesso dall'amplificatore utilizzato e deve essere tale da non causare distorsione del segnale amplificato, stessa cosa ovviamente per il segnale

in uscita.

I valori di ampiezza che stiamo considerando sono picco-picco, molto comodi da usare utilizzando la scala quadrettata graduata dell'oscilloscopio per un'immediata lettura; nulla vieta di usare i valori RMS, nel qual caso conviene ovviamente impostare l'ampiezza del generatore in "RMS" invece che "picco-picco". Siamo finalmente pronti per effettuare le misure. Si collega ora l'uscita dell'amplificatore all'ingresso del crossover e le tre uscite del crossover alle 3 resistenze da 4 ohm. Si aumenta la frequenza impostata sul generatore fino a raggiungere il valore d'ampiezza previsto dalla simulazione alla frequenza teorica d'incrocio superiore. La simulazione prevede un'attenuazione di 3,68dB a 4,3KHz, pari ad un'ampiezza di 0,65 volte rispetto a quella di centro banda, nel nostro caso pari a $250\text{mV} \times 0,65 = 162,5\text{mV}$. Ricordiamo che la conversione dei valori di amplificazione di tensione in decibel viene fatta nel modo seguente:

$$A_v\text{dB} = 20 \log_{10} A_v \text{ e } A_v = 10^{\frac{(-)A_v\text{dB}}{20}}$$

e ricordandosi del segno meno nel caso di conversione di valori di attenuazioni da decibel a valore lineare.

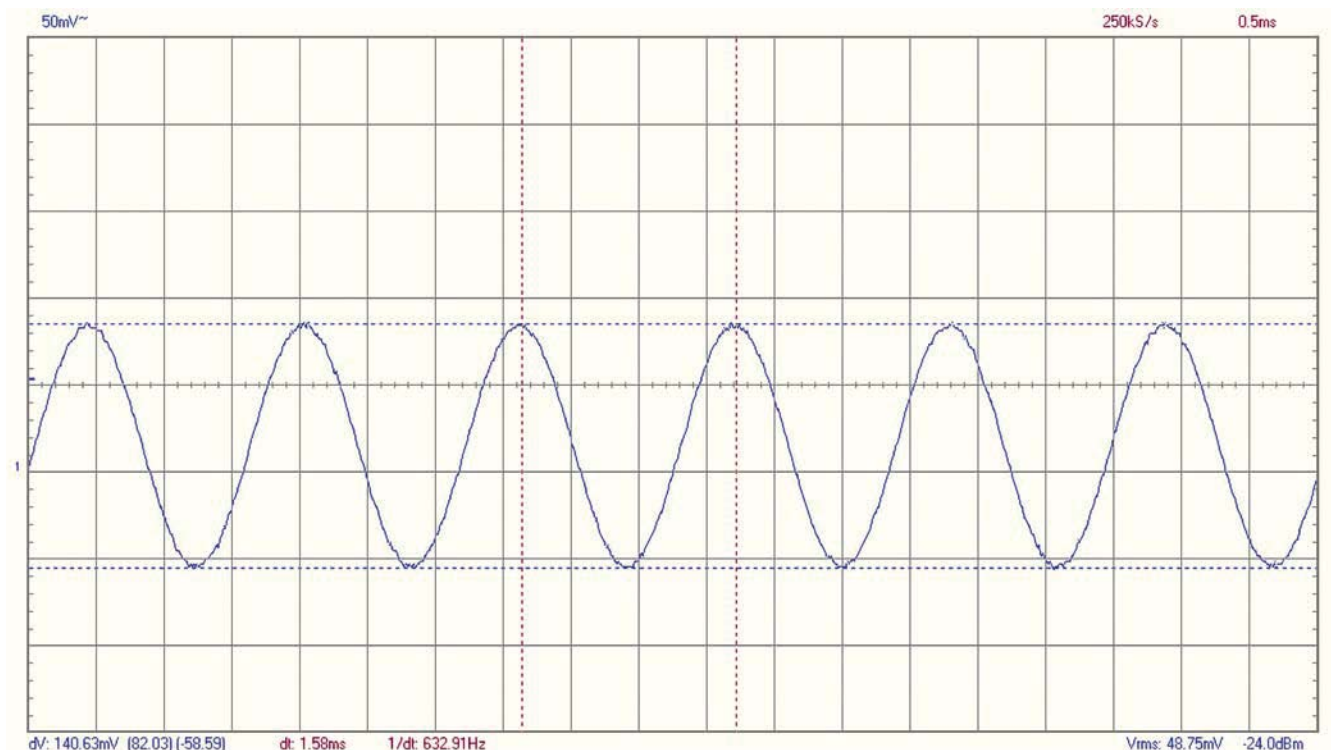
In **Fig. 11** vediamo che in realtà a 4,3KHz l'attenuazione è maggiore di quella prevista, pari a -5.2dB, ed il valore di -3,68dB si ha in realtà a 3,8KHz.

Proviamo ora a vedere cosa succede in corrispondenza della frequenza "bassa" pari al valore d'incrocio prevista dalla simulazione. In questo caso dovremmo avere -5dB, corrispondenti a 0,56 volte il valore a centro banda di 250mVpp, cioè 140mVpp, alla frequenza di 490Hz. La **Fig. 12** ci mostra invece che anche in questo caso la frequenza per cui abbiamo l'attenuazione prevista è leggermente spostata ed è pari a 630Hz.

Procediamo con le misure in modo da poter realizzare un grafico su scala logaritmica delle ampiezze in funzione della frequenza, cioè una rappresentazione grafica della risposta in frequenza del nostro filtro.

Una volta terminato, togliamo le resistenze e sostituiamole con il carico reale, cioè con i nostri diffusori, woofer, midrange e tweeter. Otteniamo la **figura 13**, in cui è visibile il confronto tra il valore teorico ottenuto dalla simulazione con LTSPICE (in arancione), le misure corrispondenti all'utilizzo dei carichi da 4 ohm (in marrone) ed infine il valore delle misure ottenute sui diffusori acustici (in verde). Si può subito notare che i valori ottenuti dalle simulazioni sono diversi dal caso reale e questo tanto più il carico si avvicina a quello reale. Il carico resistivo presenta un inviluppo paragonabile a quello della simulazione, scalato a causa dell'attenuazione presente in banda passante.

Fig. 12
Anche la frequenza d'incrocio bassa risulta essere "spostata" rispetto alla simulazione.



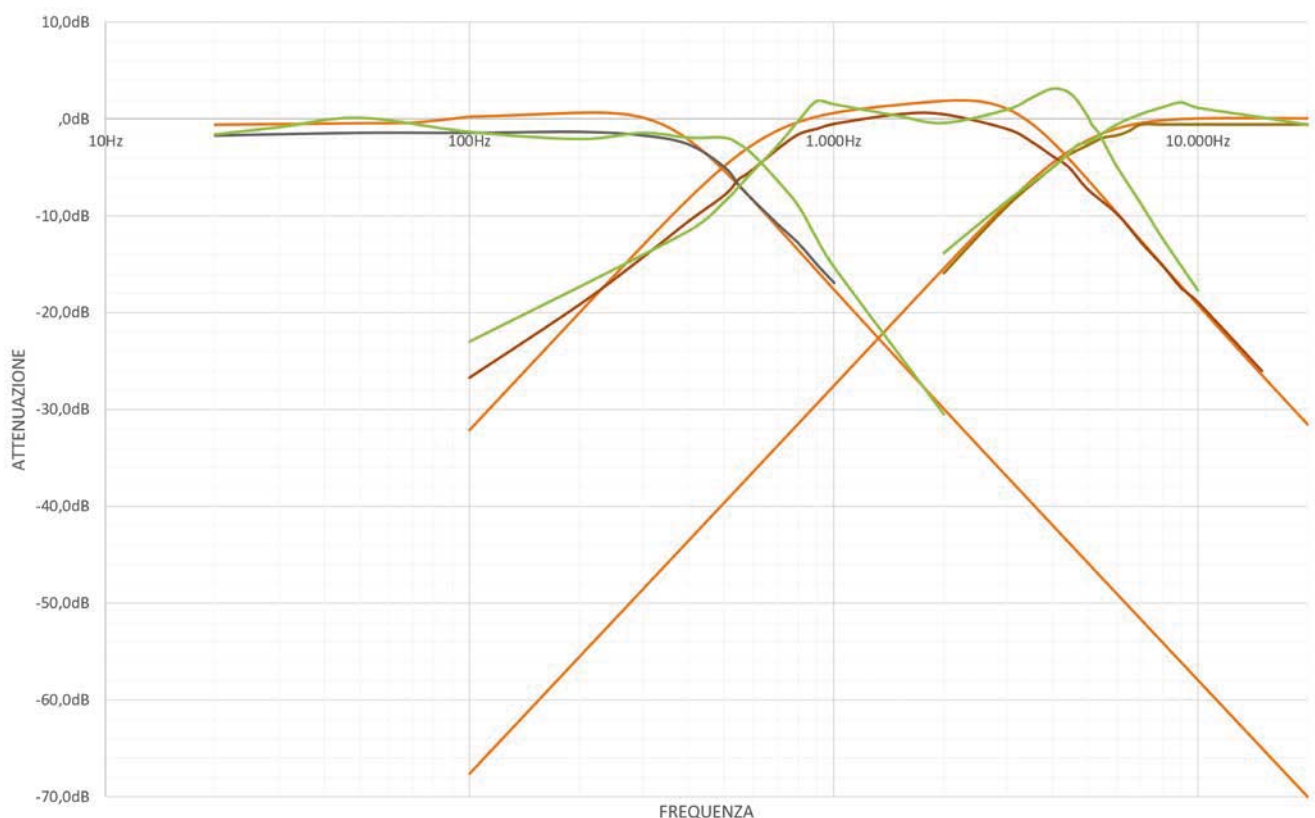


Fig. 13
Risposta in
frequenza del
crossover. In
arancione la
simulazione,
in marrone
il carico da
4 ohm e
in verde il
carico reale.

Le frequenze d'incrocio in questo caso risultano essere non molto diverse da quelle della simulazione e sono 550Hz@-7dB e 4.3KHz@-4dB, contro i valori previsti di 500Hz@-5dB e 4.3KHz@-3.7. Analizzando invece cosa succede con il carico reale, cioè sostituendo le resistenze da 4 ohm con l'impedenza dei diffusori, otteniamo un andamento che si discosta sia dalla simulazione, sia dai risultati ottenuti con i carichi puramente resistivi, a riprova del fatto che il diffusore acustico è tutt'altro che un oggetto lineare, puramente resistivo. In questo caso le frequenze d'incrocio risultano essere spostate in maniera considerevole da quanto previsto e presentano un'attenuazione inferiore: 630Hz@-2.3dB e 5.35KHz@-1.8dB. L'andamento della risposta in frequenza è tutt'altro che "piatto".

CONCLUSIONE

Risulta chiaro che il crossover andrebbe progettato tenendo conto del carico reale che verrà connesso, il diffusore audio, in quanto la modellizzazione tramite resistenze delle loro impedenze porta a risultati diversi; ciò nonostante, in mancanza delle curve d'impedenza reali dei diffusori e di un valido modello, l'utilizzo della simulazione è comunque d'aiuto per un dimensionamento di massima del

nostro filtro, che per qualcuno potrebbe anche essere soddisfacente così com'è, nonostante le approssimazioni fatte; per chi vuole ottenere il miglior risultato, si passa ora al test reale, collegando temporaneamente il crossover all'esterno della cassa acustica che si sta realizzando, per poter decidere, in base al nostro orecchio e alla nostra sensibilità e gusto personale, i ritocchi da effettuare al filtraggio e l'eventuale introduzione degli elementi atti a compensare, per quanto possibile, l'impedenza dei singoli elementi acustici. □



Cosa occorre?

Per realizzare il PCB del proprio crossover è possibile utilizzare il servizio PCBPRODUCTION disponibile sul sito www.futurashop.it

Il materiale va richiesto a:

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

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)



MOUSER
ELECTRONICS

Scopri il mondo di ARDUINO e le sue infinite potenzialità!

Progetta, costruisci e divertiti con i moduli per Arduino predisposti per essere inseriti sui mattoncini della LEGO®. Puoi utilizzarli per robotica, con Arduino o Raspberry Pi, ecc. Nella confezione sono comprese, oltre ai sensori e agli accessori, anche una board Arduino Uno compatibile e una scheda Shield sperimentale dove puoi collegare in tutta semplicità i moduli sensori e attuatori. Una dispensa a corredo vi accompagna con gli sketch per testare i vari moduli con Arduino e la Sensor Shield.

*29 MODULI PER ARDUINO
COMPATIBILI CON LEGO®*

€ 79,00

Cod. ARDULEGOKIT

I MODULI PRESENTI NELLA CONFEZIONE:

- Modulo con display 7 segmenti a 4 cifre
- Modulo con sensore di fiamma
- Modulo con LED RGB 10 mm
- Modulo sensore di temperatura e umidità
- Modulo touch sensor
- Scheda sensore umidità terreno
- Scheda sensore livello liquidi
- Modulo con LED rosso
- Modulo con LED verde
- Modulo con LED giallo
- Modulo con sensore effetto di Hall
- Modulo con LED RGB SMD
- Modulo con sensore luce ambiente
- Modulo con pulsante rosso
- Modulo con pulsante giallo
- Modulo con interruttore a pulsante
- Modulo con buzzer senza elettronica
- Modulo sensore di vibrazione
- Modulo con sensore di temperatura
- Modulo con potenziometro
- Modulo con fotoresistenza
- Modulo con interruttore reed
- Modulo con ricevitore IR a 3 pin
- Modulo con buzzer con elettronica
- Modulo con encoder rotativo
- Modulo sensore scala di grigi
- Modulo con microswitch

Prezzi IVA inclusa



Starter kit V6 con Arduino Uno REV3



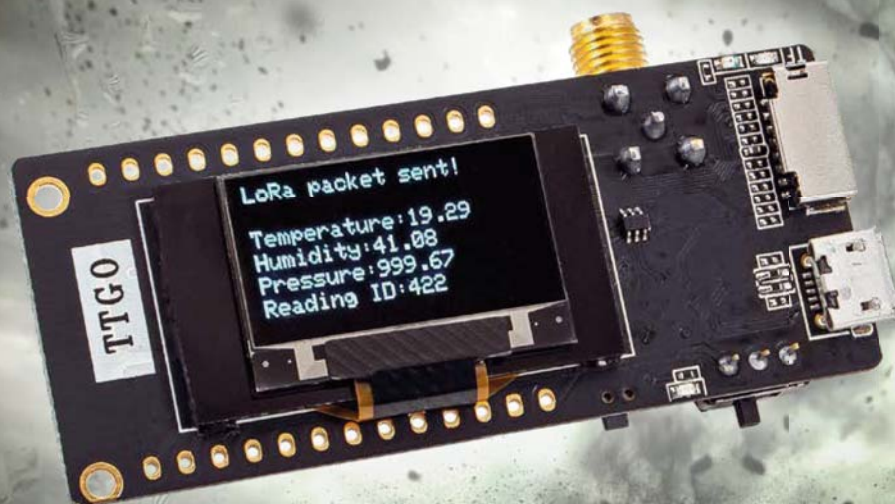
**IL PIÙ
COMPLETO!**

€ 65,00

Cod. ARDUKITV6

Acquista anche il kit
completo con tutti i
componenti necessari per
realizzare gli esperimenti
descritti nel libro "l'ABC
di ARDUINO".

WIRELESS



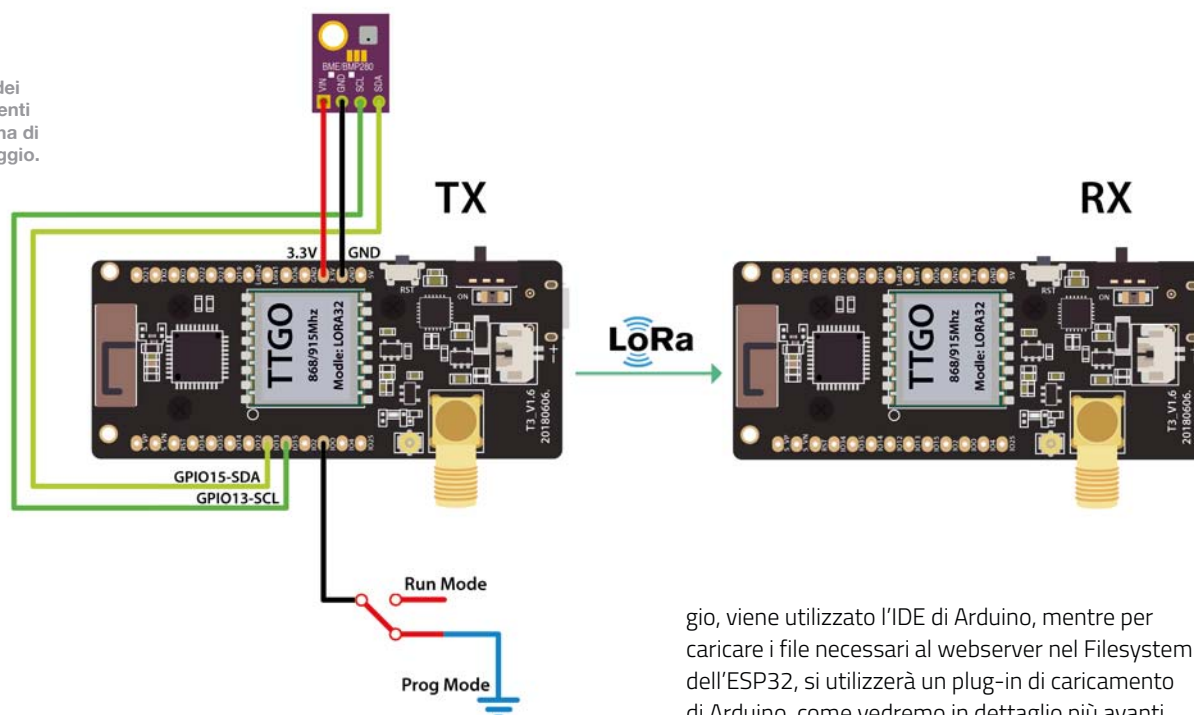
MONITORAGGIO AMBIENTALE CON LoRa

di **FULVIO DE SANTIS**

Realizziamo un sistema di rilevamento e monitoraggio di temperatura, umidità e pressione atmosferica e trasmettiamo i dati sfruttando la tecnologia LoRa.

I cambiamenti climatici che stiamo vivendo in questi ultimi anni hanno sensibilizzato molte persone, e anche noi abbiamo presentato diversi progetti finalizzati al monitoraggio di questi eventi. Grazie alla disponibilità di piattaforme di sensori a basso costo e alla presenza di reti di comunicazione dati wireless affidabili ad alta velocità, molti dispositivi possono facilmente scambiarsi i dati misurati mediante la creazione di reti di sensori. Oggi, il costo medio di ottimi affidabili sensori è così basso che la realizzazione di semplici stazioni meteorologiche private è alla portata di molti. Inoltre, un sistema basato sull'Internet of Things (IoT) contribuisce notevolmente alla riduzione dei costi e all'efficienza migliorando i servizi e la comunicazione interattiva dei dispositivi nel sistema di comunicazione wireless. Per questo, è molto importante anche migliorare le tecnologie emergenti

Fig. 1
Schema dei
collegamenti
del sistema di
monitoraggio.



di connettività wireless per l'IoT in cui vengono utilizzate bande di frequenze ISM (Industrial, Scientific and Medical) popolari come 433 MHz, 868 MHz, 915 MHz, 2,4 GHz e 5 GHz. È anche degno di nota il fatto che anche i sistemi basati sull'IoT possono ridurre il costo dell'hardware rispetto alle tecnologie tradizionali utilizzate in precedenza.

IL PROGETTO

In questo articolo viene presentato il progetto di un sistema IoT di rilevamento e monitoraggio dei principali parametri ambientali come temperatura, umidità relativa e pressione atmosferica.

Il sistema è basato su dispositivi IoT Lora32 in grado di trasmettere i dati di questi parametri su lunghe distanze in modalità wireless senza l'utilizzo di reti WiFi e visualizzarli mediante un webserver su smartphone, tablet o computer.

L'hardware del sistema è composto da due schede di sviluppo TTGO ESP32-Paxcounter LoRa32 OLED T3_V2.1 1.6 e un sensore BME280 per il rilevamento dei parametri ambientali. Una delle schede Lora32 collegata al sensore, trasmette i dati di temperatura, umidità relativa e pressione atmosferica all'altra scheda ricevente Lora32 che a sua volta rende disponibili i dati ricevuti ai dispositivi che si connettono alla scheda ricevente e quindi al server di appoggio. Per quanto riguarda il software, per programmare le schede TTGO LoRa32 con il codice di gestione del sistema di monitorag-

gio, viene utilizzato l'IDE di Arduino, mentre per caricare i file necessari al webserver nel Filesystem dell'ESP32, si utilizzerà un plug-in di caricamento di Arduino, come vedremo in dettaglio più avanti. Sarà inoltre necessario includere alcune librerie nell'IDE di Arduino per la gestione del sensore BME280 e le periferiche delle schede Lora32. In **Fig. 1** viene mostrato lo schema elettrico dei collegamenti dei dispositivi che compongono il sistema di monitoraggio dei parametri ambientali. Il trasmettitore LoRa32 TX invia le letture del sensore BME280 alla scheda ricevitore LoRa32 RX ogni 10 secondi; sul display OLED della scheda trasmittente si visualizzano le letture in tempo reale dei parametri; il ricevitore LoRa32 RX riceve le letture e le rende disponibili per la visualizzazione tramite un webserver con cui è possibile monitorare le letture aggiornate rilevate dal sensore. Il trasmettitore e il ricevitore Lora32 dotate di antenna possono comunicare da un minimo di 100 metri fino a diverse centinaia di metri di distanza a seconda della loro posizione e degli ostacoli fisici interposti. Quindi, con questo progetto è possibile monitorare i parametri ambientali rilevati da sensori distanti dall'abitazione, ubicati ad esempio in giardino, negli orti o nelle serre. Per visualizzare le letture su uno smartphone, il ricevitore LoRa RX esegue al suo interno un webserver asincrono e i file della pagina web vengono salvati nel filesystem dell'ESP32 denominato **SPIFFS** (SPI Flash File System) che significa "file di sistema della memoria flash (contenuta nell'ESP32) indirizzabile mediante bus di comunicazione seriale SPI". Il ricevitore LoRa RX, oltre alle letture dei parametri ambientali, rende disponibili la data e l'ora in cui

sono state ricevute le ultime letture e anche l'**RSSI** (**R**eceived **S**ignal **S**trength **I**ndication), ovvero, il valore in dbm della potenza del segnale RF ricevuto. Per ottenere la data e l'ora viene utilizzato il protocollo **NTP** (**N**etwork **T**ime **P**rotocol). Il display OLED della scheda ricevente visualizza il suo indirizzo IP che inserito in un browser dello smartphone consente di accedere al webserver e quindi di visualizzare i parametri ambientali.

LA SCHEDA DI SVILUPPO

TTGO LORA 868 MHZ con ESP32 e OLED è una scheda di sviluppo basata sul modulo LoRa ESP32 PICO-D4 con chip ESP32 integrato. Come è visibile nello schema nelle seguenti pagine, la scheda include un display OLED SSD1306, un connettore per schede SD, un connettore per la ricarica di una batteria Li-Po e un connettore SMA per il collegamento di un'antenna SMA esterna (**Fig. 2**). L'OLED integrato nella scheda è un modello SSD1306 da 0,96 pollici di colore blu. Il display OLED non richiede retro-illuminazione. Inoltre, i suoi pixel consumano energia solo quando sono accesi, quindi il display consuma meno energia

rispetto ad altri sistemi di visualizzazione. Il display OLED utilizzato in queste schede Lora32 comunica con l'ESP32, integrato nel modulo ESP32 PICO-D4, utilizzando il protocollo di comunicazione I2C. La scheda integra il chip convertitore seriale UART- USB CP2104 utilizzato per il collegamento della scheda Lora32 al computer. Le schede possono essere alimentate con batteria ricaricabile LiPo e ricaricabili grazie alla funzionalità di carica batterie della scheda TTGO.

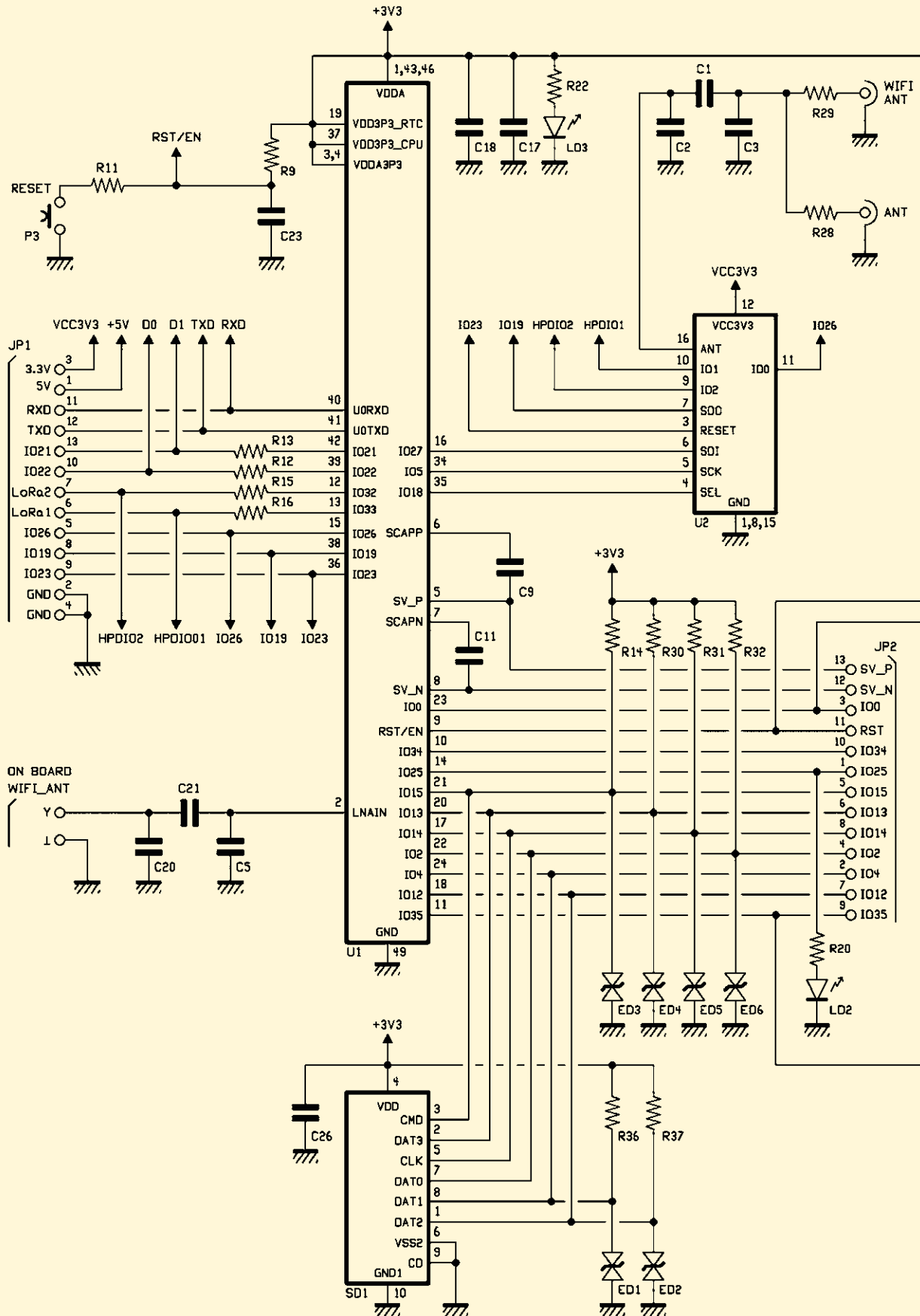
LA TECNOLOGIA LORA

LoRa è una tecnologia di comunicazione dati wireless che utilizza una tecnica di modulazione radio che viene generata dal chip ricetrasmittitore LoRa32 integrato in questa scheda. I maggiori vantaggi della tecnica di modulazione radio della tecnologia LoRa rispetto ad altre tecniche di modulazione sono:

- Comunicazione a lungo raggio (è possibile raggiungere facilmente la distanza di chilometri tra le schede in spazio aperto);
- Elevata immunità alle interferenze;
- Basso consumo energetico.



Fig. 2
TTGO LORA
868 MHZ
con ESP32 e
OLED.



SPECIFICHE TECNICHE DELLA SCHEDA TTGO LORA 868 MHZ CON ESP32 E OLED

- **Processore:** Xtensa 32-Bit LX6 Dual Core
- **Clock:** da 80 a 240 MHz (regolabile)
- **Memoria ROM:** 448 KB
- **Memoria SRAM:** 520 Kb
- **8 KB di SRAM in RTC FAST Memory** e può essere utilizzato per l'archiviazione dei dati; vi si accede dalla CPU principale durante l'avvio RTC dalla modalità Deep-sleep.
- **8 KB di SRAM in RTC SLOW Memory** e accessibile dal coprocessore durante la modalità di deep-sleep.
- **1 Kbit di eFuse:** 256 bit sono usati per il sistema (indirizzo MAC e configurazione del chip) e i restanti 768 bit sono riservati alle applicazioni del client, inclusa la crittografia flash e l'ID chip.
- **Memoria flash esterna:** 32 MB
- **Interfacce:** SD, UART (3 canali), SPI (3 canali), SDIO, I2C (2 canali), I2S (2 canali), IR LED PWM (2 canali) e motore PWM (3 canali);
- **Tipi di GPIO:** Digital IO (36), ADC 12 bit (16 canali), DAC 8 bit (2 canali), sensore capacitivo (10 canali); preamplificatore LNA;
- **WiFi:** 802.11 b / g / n: frequenza 2412 GHz – 2484 GHz
Sicurezza WiFi: WPA / WPA2 / WPA2-Enterprise / WPS;
Crittografia WiFi: AES / RSA / ECC / SHA;
- **Bluetooth:** 4.2 BR / EDR e BLE (Bluetooth Low Energy)
- **Display OLED:** blu da 0,96" con controller grafico SSD1306
- **Frequenza operativa:** 868M/915MHz
- **Massima Potenza di Trasmissione:** +20dBm
- **Sensibilità del ricevitore:** -139dBm
LoRa&62.5KHz&SF=12&146bps
-136dBm@LoRa
&125KHz&SF=12&293bps
-118dBm@LoRa
&125KHz&SF=6&9380bps
-123dBm@FSK&5KHz&1.2Kbps
- **Errore della Frequenza:** +/-15KHz
- **Spazio FIFO:** 64Byte
- **Velocità di trasmissione dati:** 1.2K-300Kbps in modulazione FSK
0.018K-37.5Kbps in modulazione LoRa
Modalità di Modulazione: FSK, GFSK, MSK, GMSK, LoRa TM, OOK
Interfaccia di comunicazione seriale: SPI
- **Corrente di Sleep:** 0.2uA SLEEP
1.5uA IDLE
- **Tensione di alimentazione:** 3,3V, +5V
- **Corrente ammessa dai pin GPIO:** 20-40mA
- **Corrente assorbita in Trasmissione:** 120mA a +20dBm
90mA a +17dBm
29mA a +13dBm
- **Temperature di funzionamento:** da -40° a +85°
- **Funzione RSSI digitale**
- **Correzione Automatica della frequenza**
- **Controllo Automatico di Guadagno**
- **Rapido wake-up e frequency hopping**
- **Gestione del pacchetto dati configurabile**

Con la tecnica della tecnologia LoRa è possibile scambiare piccoli pacchetti di dati, data la ridotta larghezza di banda, con bassi consumi di energia. Ciò è l'ideale per progetti basati sull'Internet of Things. LoRa utilizza frequenze senza licenza disponibili in tutto il mondo.

Le bande di frequenze più utilizzate sono:

- 868 MHz per l'Europa
- 915 MHz per il Nord America
- 433 MHz per l'Asia

Poiché queste bande sono prive di licenza, chiunque può usarle liberamente senza costi e senza dover richiedere una licenza. Come vedremo più avanti, nel codice della scheda Lora32 RX occorre selezionare la banda di frequenza in funzione della nazione in cui si utilizzano le schede Lora32. La scheda Lora32 dispone di varie porte GPIO per collegare periferiche, un pulsante RST di reset, un interruttore di accensione/spegnimento della scheda, un connettore micro-USB, un connettore per schede SD e un connettore per il collegamento di una batteria al litio.

La **Fig. 3** mostra la disposizione e la funzione dei pin della scheda.

Si noti che nella fila di pin di sinistra sono contrassegnati con una croce due pin di alimentazione, GND e +5V, sostituiti rispettivamente da GPIO2 e GPIO4, in quanto erroneamente indicati in alcune immagini di datasheet della scheda.

Particolare attenzione va posta anche ai pin di controllo SPI, OLED e SD card indicati in basso, ai lati della scheda ed elencati di seguito:

Il display OLED comunica con l'ESP32 utilizzando il protocollo di comunicazione I2C.

I pin di comunicazione OLED-ESP32 sono indicati di seguito:

OLED	ESP32
SDA	GPIO 21
SCL	GPIO 22

Inoltre, il modulo TTGO LORA32 comunica tramite protocollo di comunicazione SPI con la memoria flash dell'ESP32 tramite i seguenti pin GPIO:

ESP32 Pico-D4	ESP32
CS	GPIO 18
SCK	GPIO 5
MOSI/SDI	GPIO 27
MISO/SDO	GPIO 19
DIO/IOO	GPIO 26
RESET	GPIO 23

Di seguito i pin GPIO dell'ESP32 per la comunicazione con l'interfaccia SD:

SD card	ESP32
SD3-CS	GPIO13
CMD-MOSI	GPIO15
CLK-SCK	GPIO14
SD0-MISO	GPIO2
DATA2	GPIO12
DATA2	GPIO4



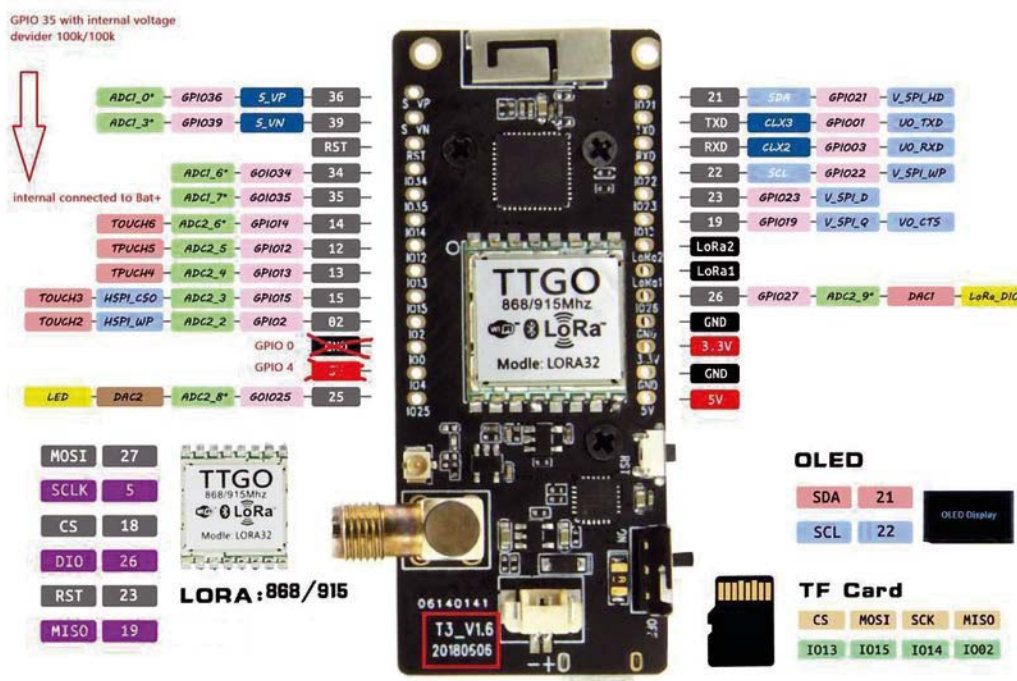


Fig. 3
Pin della
scheda TTGO
LoRa32
T3_V2.1.6

Nella progettazione del codice occorre definire questi pin per il corretto funzionamento della scheda, come vedremo nella fase di programmazione delle schede Lora32.

IL CARICABATTERIE INTEGRATO

La scheda TTGO Lora 32 T3_V2.1.6 ha la funzionalità di carica di una batteria del tipo Li-Po ad una cella. La batteria può essere ricaricata mediante il collegamento USB o applicando una tensione di alimentazione di +5V alla scheda.

La massima tensione di carica della batteria al litio è di circa 4,2 V, mentre la tensione della batteria carica è di 3,7 V.

In questa scheda è possibile utilizzare qualsiasi batteria a 3,7 V che preveda una tensione di carica di 4,2 V. La batteria si collega alla scheda con un connettore femmina a due pin del tipo JST 2P-1.25, ovvero con passo dei pin di 1,25 mm.

Durante l'inserimento del connettore della batteria al connettore della scheda, occorre prestare attenzione alla disposizione dei pin "+" e "-" della batteria riferendosi alla serigrafia riportata sulla scheda in prossimità del connettore, altrimenti si potrebbe danneggiare il chip LoRa.

Un led blu integrato nella scheda segnala lo stato di carica della batteria: quando la batteria è carica, il LED blu è spento, mentre si accende quando entra in funzione il caricamento della batteria.

IL MINI PAXCOUNTER

"Pax" sta per "persone approssimativamente" e deriva dal gergo aeronautico.

Integrato nelle schede TTGO LORA 868 MHZ con ESP32 e OLED, il Mini Paxcounter è un dispositivo dotato di funzionalità Wi-Fi e Bluetooth, è compatibile con il protocollo LoRaWAN e può essere alimentato a batterie.

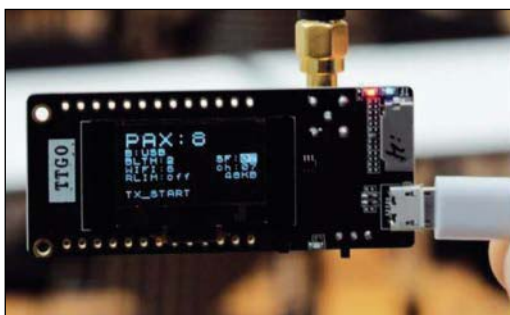
Molte persone hanno uno smartphone con sé e, a meno che siano impostati in modalità aereo, ogni tot secondi lo smartphone monitora la presenza di stazioni Wi-Fi o di dispositivo Bluetooth nei dintorni inviando piccoli pacchetti di dati.

Il Mini Paxcounter è in grado di rilevare queste attività e servirsene per fare il conteggio del numero di dispositivi (persone) che emettono segnali a radio frequenza. Per fare ciò, il Mini Paxcounter deve impostare il controller di rete in modalità di monitoraggio cosicché tutti i pacchetti di dati rilevati vengano trasferiti al livello dell'applicazione.

A differenza della modalità promiscua, in cui solo i dati provenienti dai dispositivi a cui il client è attualmente connesso vengono inoltrati in una WLAN, tutti i dati di rete che possono essere ricevuti vengono effettivamente considerati in modalità monitor. Il Paxcounter non acquisisce i dati dell'utente, ma solo l'indirizzo di controllo dell'accesso ai media (MAC address).

Senza l'uso di telecamere o di altri sistemi, con il

Fig. 4
Mini Paxcounter.



Paxcounter è possibile stimare il numero approssimativo di persone presenti in un dato momento e luogo e, oltre a visualizzare questo numero di persone sul display OLED della scheda Lora32, può anche rendere disponibile questa informazione tramite **TTN** (The Things Network).

In **Fig. 4** è mostrata la visualizzazione su OLED della funzionalità Paxcounter.

Se vi interessa approfondire l'argomento andate al sito <https://github.com/cyberman54/ESP32-Paxcounter> (in tedesco).

IL MODULO ESP32 PICO-D4

L'ESP32 PICO-D4 è un modulo SiP (System-in-Package) basato su ESP32. Progettato con la tecnologia a bassissima potenza, fornisce funzionalità Wi-Fi e Bluetooth a 2,4 GHz, integra una memoria flash da 4 MB accessibile mediante comunicazione seriale SPI con l'ESP32.

L'ESP32-PICO-D4 integra in un unico package tutti i componenti periferici alimentabili senza soluzione di continuità, compreso un oscillatore a cristallo, memoria flash, condensatori di filtraggio e colle-

gamento RF. Con le sue dimensioni ultra ridotte, prestazioni elevate e basso consumo energetico, l'ESP32-PICO-D4 è adatto a qualsiasi applicazione con spazio limitato o alimentate a batteria, come i dispositivi elettronici indossabili, apparecchiature mediche, sensori e altri dispositivi IoT.

IL SENSORE BME280

L'integrato **BME280** è un sensore digitale che abbiamo avuto modo di usare in molte applicazioni. Le sue piccole dimensioni e il suo basso consumo consentono l'implementazione in dispositivi alimentati a batteria come nel caso delle schede Lora32 impiegate in questo progetto.

Il BME280 raggiunge prestazioni elevate in tutte le applicazioni che richiedono la misurazione di umidità e pressione, come ad es. nelle applicazioni emergenti di controllo domotico, navigazione interna, fitness, che richiedono un'elevata precisione. Il sensore di umidità ha un tempo di risposta estremamente rapido per applicazioni veloci ed elevata precisione complessiva su un ampio range di temperature.

Il sensore di pressione integrato rileva la pressione barometrica assoluta con una precisione estremamente elevata, alta risoluzione e basso rumore.

Il sensore di temperatura è stato ottimizzato per il minimo rumore e la massima risoluzione.

Oltre ad essere utilizzato per la stima della temperatura ambiente, il suo segnale di uscita viene utilizzato per la compensazione termica dei sensori di pressione e umidità.

Il sensore BME280 può comunicare con microcontrollore mediante interfacce SPI e I2C.

Fig. 5
Installazione
scheda ESP32.



L'alimentazione del sensore va da 1,71 V a 3,6 V, mentre la tensione di alimentazione VDDIO per le interfacce IO va da 1,2 a 3,6 V.

Quando il sensore è disabilitato, il consumo di corrente scende a 0,1 µA. Il chip BME280 può essere utilizzato in tre modalità di alimentazione:

- modalità risparmio energetico (sleep mode)
- modalità normale
- modalità forzata

Al fine di adattare la velocità trasmissione dei dati, il rumore, il tempo di risposta e il consumo di corrente ai requisiti dell'applicazione, è possibile selezionare una varietà di modalità operative come il funzionamento a sovra-campionamento, modalità di filtraggio e velocità di trasmissione dati.

Le misurazioni possono essere attivate da un host o eseguite automaticamente ad intervalli temporali regolari.

PROGRAMMAZIONE DELLE SCHEDE LORA32

Come già accennato, per programmare le schede Lora32 utilizzeremo l'IDE di Arduino e installeremo alcune librerie. Occorre innanzitutto installare il driver che consente la comunicazione seriale USB della scheda con il computer.

Per questo, la scheda Lora32 utilizza il chip convertitore seriale UART-USB integrato nella scheda CP2104. Collegate la scheda Lora32 al computer tramite un cavo USB-micro USB.

Se il sistema operativo del computer non riconosce automaticamente il chip CP2104, occorre scaricare il driver dal sito web www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers e decomprimolo in una cartella.

Su Windows si installa eseguendo "CP210xVCPInstaller_x86.exe" o "CP210xVCPInstaller_x64.exe", a seconda del sistema operativo a 32 o 64 bit; per Mac, si installa il file DMG nell'archivio caricato. Per i sistemi operativi windows XP e Vista occorre installare il driver dal sito www.silabs.com/community/interface/knowledge-base.entry.html/2017/01/10/legacy_os_software-bgvU.

Dopo aver scollegato e ricollegato la scheda, nell'elenco delle periferiche di comunicazione in Windows dovrebbe apparire una porta COM identificabile come "Silicon Labs CP210x da USB a UART Bridge".

Il prossimo step è l'installazione del software IDE di Arduino dal sito ufficiale www.arduino.cc (in questo articolo si fa riferimento alla versione software Arduino 1.8.12).

All'avvio del programma, dall'IDE di Arduino occorre installare la scheda ESP32 quindi, da File > Impostazioni, alla voce "URL aggiuntive per il Gestore schede" inserire il link: https://dl.espressif.com/dl/package_esp32_index.json.

Poi, con *Strumenti* > *Scheda* > *Gestore Schede*, digitate "esp32" nella barra di ricerca; si visualizzerà un'immagine come in **Fig. 5**.

Selezionate "ESP 32 by Espressif Systems" e cliccate su "Installa" per caricare il firmware di gestione della scheda ESP32 nell'IDE di Arduino. Completata l'installazione della scheda ESP32, da *Strumenti* > *Scheda*, selezionate la scheda "TTGO LoRa32-OLED V1", poi selezionate la porta COM (su Windows) o *ttyUSB* (su Mac) corrispondente al driver "Silicon Labs CP210x USB to UART Bridge VCP".

IL FILESYSTEM SPIFFS

Riguardo al webserver da installare nella scheda Lora32 RX ricevente, occorre caricare nel Filesystem (poi vedremo cos'è), il file con il codice HTML relativo alla pagina web e l'immagine di sfondo del webserver. Per fare questo è necessario installare in Arduino il plug-in "ESP32 Filesystem Uploader" con il quale è possibile caricare i file necessari nel Filesystem.

Per meglio comprendere ciò che andremo a fare nei prossimi step, riteniamo utile una breve esplicitazione del Filesystem definito anche dall'acronimo SPIFFS.

Il chip ESP32 integrato nel modulo ESP32 PICO-D4 a sua volta integrato nella scheda di sviluppo Lora32, contiene uno SPIFFS (Serial Peripheral Interface Flash File System), ossia un piccolo file di sistema appositamente creato per microcontrollori con chip di memoria flash, a cui l'ESP32 accede mediante un bus di comunicazione seriale SPI, così come avviene con la memoria flash.

L'SPIFFS consente di accedere alla memoria flash come in un normale filesystem del computer, ma più semplice e più limitato. E' possibile leggere, scrivere, aprire/chudere ed eliminare file.

Con l'SPIFFS dell'ESP32 è possibile creare file di configurazione, salvare dati permanentemente, creare file per memorizzare piccole quantità di dati invece di utilizzare una scheda SD, salvare file HTML e CSS per creare un webserver, salvare immagini, figure e icone, ecc...

E' anche possibile scrivere codice HTML per webserver anche direttamente sullo sketch di Arduino, ma con l'SPIFFS si può scrivere comodamente il codice HTML e i CSS in appositi file e salvarli nel filesystem dell'ESP32.



Fig. 6
Installazione della
release esptool.py.



INSTALLAZIONE DEL PLUG-IN "ESP32 FILESYSTEM UPLOADER"

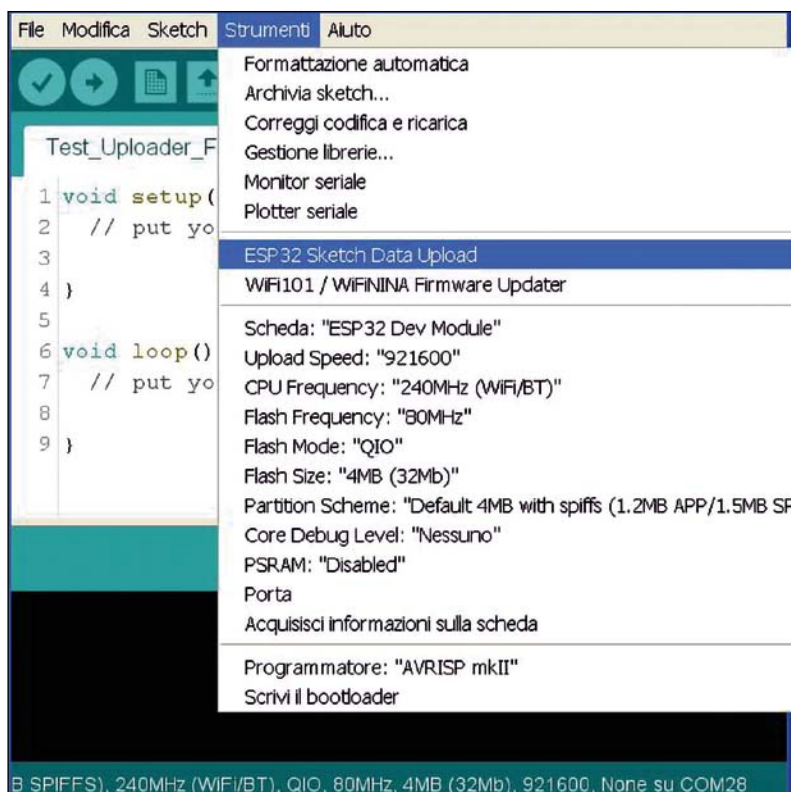
Il plug-in "ESP32 Filesystem Uploader" per l'IDE di Arduino che abbiamo su citato, consente di caricare nel software mediante l'IDE di Arduino file contenuti nella cartella di un computer direttamente nel file system.

Questo rende molto agevole e semplice lavorare con i file.

Detto questo, possiamo procedere all'installazione di questo plug-in eseguendo i seguenti step:

- 1) Aprite il sito github.com/me-no-dev/arduino-esp32fs-plugin/releases/ e scaricate il file "ESP32FS-1.0.zip" indicato in **Fig. 6**.

Fig. 7
Plug-in ESP32
Filesystem Upload.



- 2) Nella directory di Arduino aprite la cartella "tools". Ad esempio: "*C:/Program Files/Arduino/tools*", oppure "*D:/Arduino 1.8.12/tools*" se avete installato una versione portatile di Arduino.
- 3) Decomprimate la cartella ESP32FS-1.0.zip che avevate inserito nella cartella tools. Dovreste avere una struttura di cartelle simile a: *C:/Program Files/Arduino/tools/ESP32FS/tool/esp32fs.jar*.
- 4) Riavviate L'IDE di Arduino e verificate se il plugin è stato installato correttamente selezionando Strumenti -> ESP32 Filesystem Upload come mostrato in **Fig. 7**.

Installato il plug-in **ESP32 Filesystem Uploader** nell'IDE di Arduino che più avanti utilizzeremo per caricare dei file necessari alla creazione del webserver, dobbiamo installare anche le seguenti librerie che potete scaricare tra i file della rivista:

Libreria Lora
Libreria BME280
Libreria OLED
Librerie webserver
Libreria NTPClient

Una volta scaricati tutti i file .zip delle librerie dovete inserirle nell'IDE di Arduino selezionando per ognuno di essi *Sketch -> #include Libreria* -> *Aggiungi libreria da file .ZIP*, e installarle con *Strumenti -> Gestione librerie*, comando che apre il Gestore delle librerie presenti in Arduino in cui dovrete cercare le librerie inserendo il loro nome nella barra di ricerca del Gestore e installarle.

IL FIRMWARE DELLA SCHEDA TRASMETTITORE LORA32

Come abbiamo visto nello schema elettrico del progetto mostrato in **Fig. 1**, la scheda Lora32 TX è collegata al sensore BME280 da cui riceve via bus I2C i dati di temperatura, umidità e pressione e li trasmette via radio ogni 10 secondi alla scheda RX Lora32 RX. Aprite o, se già aperto, riavviate Arduino e aprite lo sketch del trasmettitore che potete scaricare dal sito della rivista.

Vediamo insieme alcuni passaggi fondamentali. Dopo l'inclusione delle librerie LoRa, OLED e BME280, vengono definiti i pin GPIO dell'ESP32 utilizzati per comunicare con il modulo LoRa, del display oled e del sensore BME280:

```
//define the pins used by the LoRa transceiver module
#define SCK 5
```

```
#define MISO 19
#define MOSI 27
#define SS 18
#define RST 23
#define DIO0 26
//OLED pins
#define OLED_SDA 21
#define OLED_SCL 22
//BME280 definition
#define SDA 13
#define SCL 15
```

Successivamente vengono create alcune variabili contenenti il messaggio LoRa di temperatura, l'umidità, pressione e l'ID della lettura:

```
//packet counter
int readingID = 0;
int counter = 0;
String LoRaMessage = "";
float temperature = 0;
float humidity = 0;
float pressure = 0;
```

Nella funzione *setup()*, sono richiamate diverse funzioni create in precedenza nel codice per inizializzare il display OLED, il BME280 e il modulo ricetrasmittitore LoRa:

```
void setup() {
  //initialize Serial Monitor
  Serial.begin(115200);
  startOLED();
  startBME();
  startLoRA();
}
```

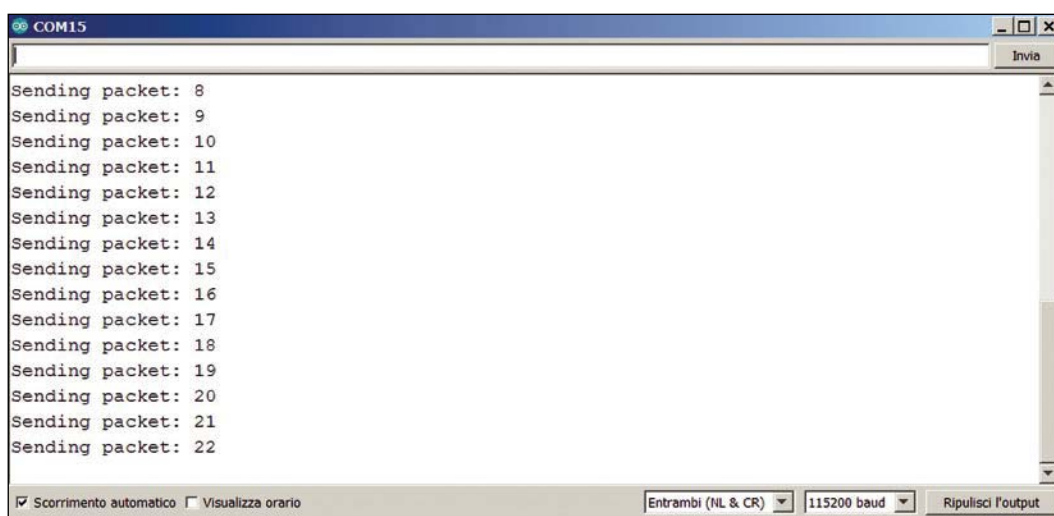
Nella funzione *loop()* sono richiamate le funzioni *getReadings()* e *sendReadings()* che consentono di ottenere le letture dal sensore BME280 e di inviarle via radio tramite il ricetrasmittitore LoRa. In particolare la funzione *getReadings()* effettua le letture dei parametri ambientali rilevati dal sensore inserendole nelle rispettive variabili, in pratica, applica i metodi *readTemperature* ecc... sull'oggetto *bme*.

Mentre la funzione *sendReadings()* viene utilizzata per inviare le letture tramite il protocollo LoRa, concatenando tutte le letture su un'unica variabile *LoRaMessage*.

Ogni lettura è separata da un carattere speciale in modo che il destinatario possa identificare facilmente ogni valore.



Fig. 8
Invio pacchetti
dal trasmettitore
Lora32.



Quindi, invia il pacchetto al ricevitore Lora32 RX utilizzando queste linee di codice:

```
LoRa.beginPacket ();
LoRa.print (LoRaMessage);
LoRa.endPacket ();
```

La funzione *loop()* viene ripetuta ogni 10000 millisecondi (10 secondi). Pertanto, anche le nuove letture del sensore vengono inviate dopo questo tempo ma, se volete, è possibile modificare questo tempo di ritardo agendo sull'istruzione:

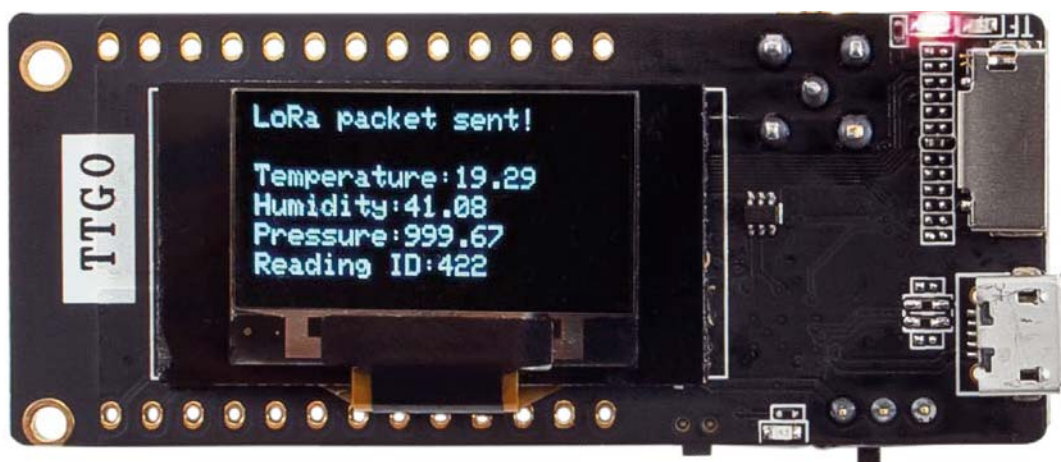
```
delay (10000);
```

CARICARE IL FIRMWARE NELLA SCHEDA TRASMETTITORE LORA32

Aprire l'IDE di Arduino e selezionare la scheda su cui installerete il codice con *Strumenti -> Scheda -> TTGO LoRa32-OLED V1*, poi con *Strumenti ->*

Porta selezionate la porta COM (su Windows) o ttyUSB (su Mac) corrispondente al driver "Silicon Labs CP210x USB to UART Bridge VCP". Predisponete la scheda in modalità di programmazione commutando su "Prog mode" lo switch mostrato in **Fig. 1** e premete il pulsante RST di reset della scheda; infine, nell'IDE di Arduino cliccate sul pulsante Carica per avviare la compilazione e l'upload del codice. Terminato il caricamento del codice, commutate lo switch in "Run mode"; aprete il monitor seriale impostato su un baud rate di 115200 b/s e premete il pulsante RST. Sul monitor seriale vedrete un'immagine simile a quella riportata in **Fig. 8** che riporta la sequenza di pacchetti di dati dei parametri ambientali che il trasmettitore Lora32 TX stà inviando via radio. Contemporaneamente, sul display OLED del trasmettitore Lora32 TX si vedranno le letture aggiornate dei parametri ambientali, come mostrato in **Fig. 9**.

Fig. 9
Parametri
ambientali
sull'OLED del TX..



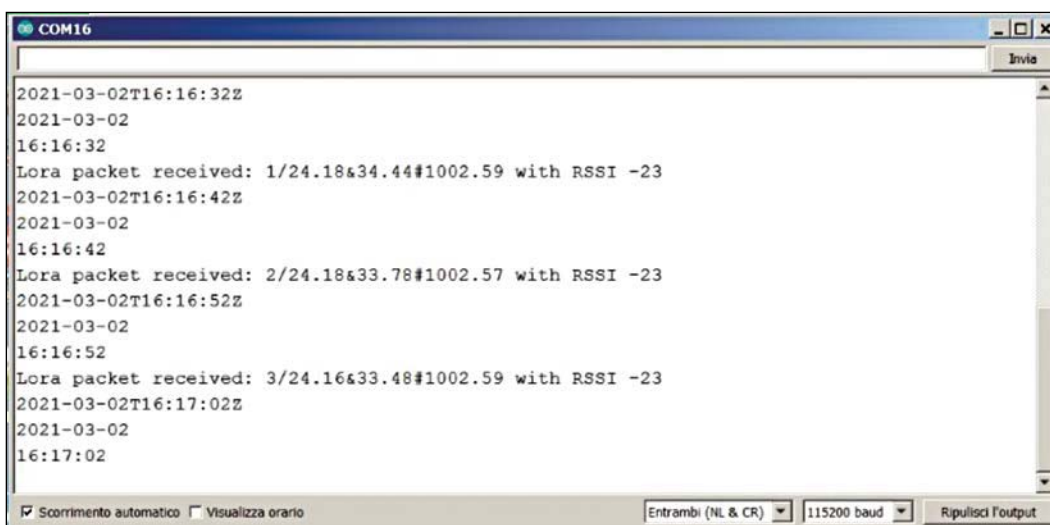


Fig. 10
Sequenza
di pacchetti
di dati dei
parametri
ambientali
ricevuti dal
ricevitore
Lora32 RX.

PROGRAMMAZIONE DELLA SCHEDA RICEVITORE LORA32

Il ricevitore LoRa32 RX riceve i pacchetti LoRa dei dati dei parametri ambientali trasmessi dal trasmettitore Lora32 TX e rende disponibili in un webserver asincrono le letture ricevute, la data e l'ora delle ultime letture ricevute e l'RSSI (indicatore della potenza del segnale RF ricevuto). Inoltre, una volta avviata la scheda ricevitore Lora32 RX, sul suo display OLED si visualizzerà il suo indirizzo IP (dell'ESP32) che inserito in un browser di uno smartphone consentirà di visualizzare le suddette informazioni. Prima di procedere al caricamento del codice della scheda Lora32 RX occorre creare un webserver.

REALIZZAZIONE DEL WEBSERVER

Per la realizzazione del Webserver di questo progetto sono necessari tre file: lo sketch di Arduino, il file HTML per la creazione della pagina web e l'immagine di sfondo del display dello smartphone che accede al server (file scaricabili dal sito della rivista). Il file HTML e l'immagine saranno salvati nel Filesystem SPIFFS dell'ESP32 della scheda ricevitore Lora32 RX, ovvero, grazie al plug-in installato, "prelevati" da una cartella denominata "data" che deve essere creata all'interno della cartella contenente lo sketch di Arduino del codice della scheda Lora32 RX, come vedremo più avanti. Nel file HTML sono state inserite porzioni di codice CSS per gli stili della pagina web, e di codice JavaScript che consentono l'aggiornamento automatico delle letture del sensore nella pagina web. Nel codice HTML troverete la presenza dei "placeholders" (segnaposti).

I segnaposto sono inseriti tra i caratteri speciali %, come nel caso di **%TIMESTAMP%**, **%TEMPERATURE%**, **%HUMIDITY%**, **%PRESSURE%** e **%RSSI%**, che saranno sostituiti con i valori effettivi dal codice Arduino (il segnaposto specifica un breve testo che descrive il valore atteso di un campo di input, ad esempio un valore o una breve descrizione).

Il codice degli stili CSS viene inserito tra i tag `<style>` e `</style>`. Il codice JavaScript verrà inserito tra i tag `<script>` e `</script>`.

In fondo alla porzione di codice CSS degli stili, trovate `"background-image: url(winter.jpg);"` che imposta l'immagine di sfondo della pagina web del server. Un'immagine del tipo della `winter.jpg` potete trovarla e scaricarla dal sito www.pexels.com/search/winter/ o anche se ne scegliete un'altra di vostro gradimento, vi consigliamo di salvarla sempre come `winter.jpg` altrimenti la dovrete sostituire al nome `winter.jpg` nel codice del file `index.html` e nello sketch di Arduino della scheda ricevitore Lora32 RX. Salvate l'immagine che avete scelto nella stessa cartella dove avete salvato il file HTML `index.html`.

Per imparare o approfondire il linguaggio HTML, CSS e JavaScript, un buon sito web è W3Schools (in inglese).

IL FIRMWARE DELLA SCHEDA RICEVITORE LORA32

Il firmware del modulo ricevitore è scaricabile gratuitamente dal sito della rivista.

Come sempre analizziamo i passaggi principali del codice che inizia includendo le librerie necessarie per creare il webserver asincrono, accedere al filesystem ESP32 (SPIFFS), comunicare con il chip



Fig. 11
Indirizzo IP della
scheda Lora32 RX.



LoRa, controllare il display OLED, ottenere data e ora da un server NTP.

Come per il trasmettitore vengono definiti i pin utilizzati dal modulo ricetrasmittitore LoRa e dell'OLED.

Successivamente trovate le linee di codice che dovrete modificare per inserire le credenziali della vostra rete WiFi per connettervi all'ESP32.

La funzione *processor()*, che troverete più avanti, attribuisce i valori ai segnaposto (holdplaces) presenti nel file HTML *index.html* che è stato

creato. La funzione ha come argomento il segnaposto e restituisce una stringa che sostituirà quel segnaposto. Ad esempio, se trova il segnaposto *TEMPERATURA*, restituirà la variabile String della temperatura.

La funzione *Setup()* inizializza il display OLED, la comunicazione LoRa, la connessione alla rete Wi-Fi, l'NTPclient e il webserver con il filesystem SPIFFS.

Riguardo al webserver asincrono, la libreria *ESPAsyncWebServer* consente di configurare il percorso in cui il server ascolterà le richieste HTTP in arrivo. Ad esempio, quando viene ricevuta una richiesta sul percorso dell'URL, viene inviato il file html che si trova nel filesystem SPIFFS dell'ESP32. Ad esempio:

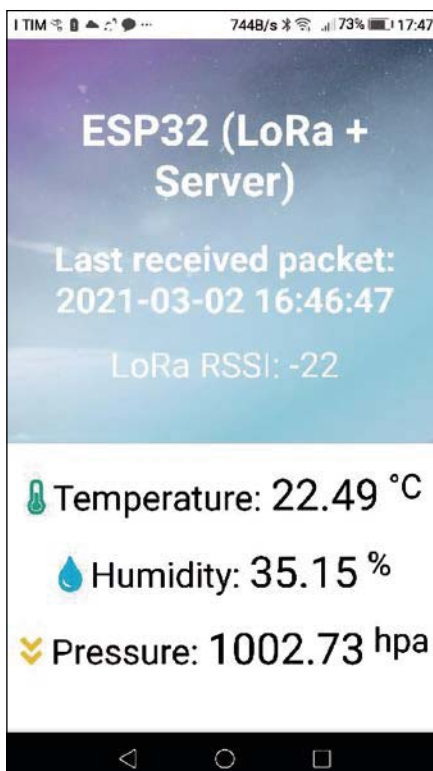
```
server.on("/", HTTP_GET, [](AsyncWebServerRequest *request)
{
    request->send(SPIFFS, "/index.html", String(), false,
processor);
});
```

Come accennato in precedenza, sono stati inseriti degli script Javascript al file HTML che consentono l'aggiornamento della pagina web ogni 10 secondi effettuando una richiesta sugli URL */ temperature*, */ humidity*, */ pressure*, */ timestamp*, */ rssi*.

Per fare ciò, vengono inviate le variabili in formato char di temperatura, umidità, pressione, data e ora e rssi, ed è questo il motivo per cui viene utilizzato il metodo *.c_str()*, come ad esempio per la variabile *temperatura*:

```
server.on("/temperature", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", temperature.c_str());
});
```

Fig. 12
Monitoraggio
ambientale su
smartphone.



Poiché abbiamo incluso un'immagine nella pagina web, il server riceverà una "request" per l'immagine. Quindi viene inviata l'immagine salvata nell'SPIFFS dell'ESP32. Infine si avvia il server. Un client **NTP** viene creato per ottenere l'ora da Internet. In base alla località in cui vi trovate (timezone), per avere l'ora esatta dovrete impostare l'offset GMT in secondi nell'argomento della funzione `timeClient.setTimeOffset()`. Nel `loop()` troviamo il richiamo alla funzione `getLoRaData()` che riceve il messaggio LoRa e lo scompone per ottenere le diverse letture. La funzione `getTimeStamp()` ottiene l'ora e la data da Internet nel momento in cui riceve il pacchetto.

CARICAMENTO DEI FILE HTML

Nell'IDE di Arduino aprite lo sketch del ricevitore Lora32 RX, aprite quindi la cartella degli sketch con *Sketch > Apri cartella dello Sketch* e create una cartella con nome *data*. All'interno inserite il file HTML *index.html* e il file dell'immagine di sfondo che utilizzeremo.

Ora, con *Strumenti -> ESP32 Sketch Data Upload* i file *index.html* e l'immagine saranno caricati nel filesystem SPIFFS dell'ESP32 pronti per essere utilizzati dal server.

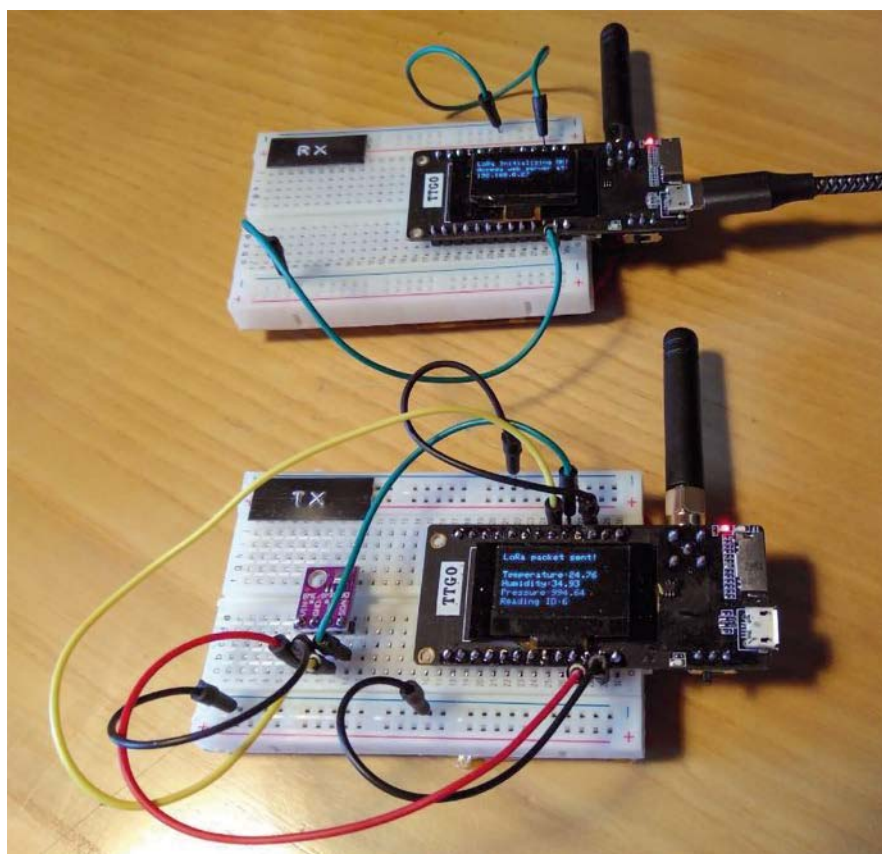
CARICARE IL FIRMWARE NELLA SCHEDA RICEVITORE LORA32

Il caricamento del codice sulla scheda ricevitore è uguale a quella del trasmettitore. Ricordate di commutare lo switch in "Run mode" finito l'upload. Aprite il monitor seriale impostato su un baud rate di 115200 b/s e premete il pulsante RST. Sul monitor seriale vedrete un'immagine simile a quella riportata in **Fig. 10** che mostra la sequenza di pacchetti di dati dei parametri ambientali che il ricevitore Lora32 RX riceve ogni 10 secondi via radio dal trasmettitore Lora32 TX.

Contemporaneamente, sul display OLED della scheda Lora32 RX viene visualizzato il suo indirizzo IP, come mostrato in **Fig. 11**.

A questo punto il progetto è completo e possiamo vederne il risultato inserendo l'indirizzo IP visualizzato sull'OLED della scheda Lora32 RX in un browser del vostro smartphone, tablet o computer. Apparirà sul display dello smartphone un'immagine delle letture dei parametri ambientali, come mostrato in **Fig. 12**.

È possibile verificare che le schede Lora32 sono in grado di comunicare stabilmente per distanze di oltre cento metri in spazi aperti.



CONCLUSIONI

L'obiettivo di questo progetto impostato come tutorial, è stato quello di dimostrare che con un sistema a basso costo basato su schede Lora TTGO ESP32 LoRa32 è possibile realizzare il controllo via radio a lunga distanza di parametri ambientali e visualizzarli sul display di uno smartphone mediante un webserver. Non vi rimane che personalizzare il progetto con altri sensori. Buon lavoro. □



Le schede Lora TTGO ESP32 al lavoro



Cosa occorre?

I componenti utilizzati in questo progetto sono disponibili presso Futura Elettronica. La board con ESP32 e modulo LoRa (cod. LORAE32) è in vendita a Euro 44,00, il sensore (cod. GY-BME280) è disponibile a Euro 8,50. 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>



IMPARA L'ELETTRONICA FACILMENTE CON TOTEM MINI LAB



€ 112,00
cod. TOTEMMINILAB

Piccolo ma potente laboratorio di elettronica ideale per sperimentare e apprendere le basi dell'elettronica. Pienamente compatibile con l'ambiente di programmazione Arduino, Totem Mini Lab comprende una varietà di strumenti di misura e blocchi elettronici pronti all'uso per realizzare esperimenti. Pensato per essere utilizzato insieme al sistema di costruzione Totem consente agli utenti di costruire solidi banchi da lavoro con breadboard integrate. Inoltre, il laboratorio è compatibile con schede di espansione I/O, sensori e audio che possono espandere ulteriormente le capacità del Mini Lab. L'utilizzo di Totem Mini Lab è semplice e divertente, ideale sia per ragazzi sia per adulti!

COMPLETALO CON LE SCHEDE DI ESPANSIONE!



ESPANSIONE AUDIO

Con questo pannello sarà possibile avere disponibili e facilmente accessibili i seguenti componenti: amplificatore audio in classe AB con 1 W di potenza, altoparlante integrato e alimentazione a 12 volt, VU-meter con barra a 10 LED, generatore di funzioni in grado di generare segnali fino a 1 MHz.

€ 39,50
Cod. TOTEMAUDIOSIDE



ESPANSIONE I/O

Contiene componenti elettronici che solitamente sono difficili da inserire sulle breadboard. Con questo pannello sarà possibile avere disponibili e facilmente accessibili i seguenti componenti: 3 interruttori a pulsante con indicatori LED, 3 potenziometri da 10 kOhm, 1 encoder rotativo, 3 pulsanti, 1 LED RGB, 1 presa Jack audio da 3,5 mm, 1 Relè da 10 A / 250 V, con LED di stato.

€ 37,00
Cod. TOTEMIOSIDE



ESPANSIONE SENSORI

Con questo pannello sarà possibile avere disponibili e facilmente accessibili i seguenti componenti: un microfono preamplificato, un display OLED 128x64 con interfaccia I²C, un sensore umidità DHT11, un termistore NTC da 10 kohm, un buzzer e un piccolo driver per motori DC.

€ 39,50
Cod. TOTEMSENSORSIDE

CONTROLLARE UNA LAMPADINA WiFi CON MODULI ESP

di ANDREA CHIAPPORI



Controlliamo una lampadina WiFi con un ESP8266 e con il più evoluto ESP32 utilizzando una libreria realizzata apposta per questi moduli.

A

bbiamo analizzato, nella puntata precedente, il protocollo utilizzato dall'applicazione *SmartLife* (e altre similari) per controllare una lampadina WiFi ed abbiamo scritto su PC un programma C# per testarlo, grazie al quale siamo in grado di effettuare le operazioni principali come accendere e spegnere la lampadina, cambiare tonalità del colore e della luce bianca. Questo programma, disponibile sul sito di Elettronica In, non è fine a se stesso, ma nasce come punto di riferimento da cui partire per effettuare il porting su piattaforme a microprocessore come i noti moduli ESP che hanno uno stack TCP/IP e sono WiFi. Sarebbe stato impensabile eseguire le varie prove di analisi direttamente su questi moduli anche solo per il tempo di programmazione che, seppur molto ridotto, avrebbe reso ogni singolo test più complicato. Ora però possiamo iniziare.

SCHEMA A BLOCCHI DELL'ENCODING

Prima di addentrarci nella scrittura della libreria è doverosa una breve analisi dei blocchi che compongono l'encoding dei comandi da inviare alla lampadina. Abbiamo visto, nel precedente articolo, come l'app per gestire la lampadina su cui ci siamo focalizzati (*Tuya Smart*) usi una codifica AES128 per trasformare una stringa di testo in chiaro in tanti blocchi di 16 byte che poi vengono uniti ed inseriti nel payload del pacchetto TCP da inviare alla lampadina. A questo pacchetto va però, inserito una CRC di controllo che va calcolata ed inserita in fondo (prima di un suffisso che è sempre uguale per tutti i pacchetti di questo tipo).

Quindi avremo uno schema di funzionamento complessivo come indicato dalla **Fig. 1**

Non scenderemo nei dettagli del pacchetto e della codifica perchè già trattate nell'articolo precedente, ma ci occuperemo di scrivere una libreria per permettere ai moduli hardware di interfacciarsi alla lampadina senza che l'utilizzatore finale debba essere per forza a conoscenza di tutti i passaggi interni. Ad esempio, il primo blocco si riferisce alla stringa di testo che contiene il comando da inviare alla lampadina dopo essere stato codificato, ma sarebbe impensabile dover usare quella stringa come parametro di ingresso di una funzione, perchè renderebbe tutto poco chiaro e poco gestibile.

Dovremmo scrivere funzioni di alto livello che poi, internamente, si occupino di generare la stringa di comando, di codificarla tramite codifica AES128 e di costruire il pacchetto così formato aggiungendogli in fondo i 4 byte di CRC, prima di inviarlo nella rete verso la lampadina.

Riportiamo in **Fig. 2** la lista di comandi analizzati nell'articolo precedente e che useremo per la scrittura delle funzioni.

Da notare come il *comando 5*, usato per rappresentare il colore, necessita di una stringa contenente sia le componenti RGB sia quelle HSV quindi dovremo creare anche un modulo software per convertire il colore tra i due formati e chi userà la libreria potrà ad esempio scegliere di impostare il colore nella modalità preferita, senza dover passare direttamente la stringa di testo.

Queste funzioni di conversione sono già state utilizzate anche nel programma C# quindi abbiamo tutto per iniziare a scrivere la libreria eseguendo il porting sia su un ESP8266 che su un ESP32.

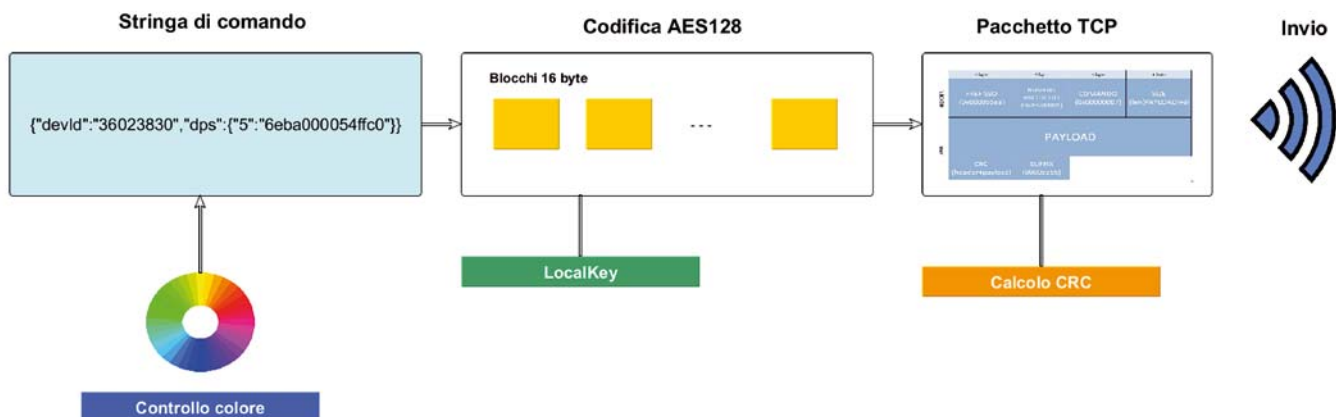
MODULI ESP

Entrambi questi moduli sono WiFi e sono dotati di uno stack TCP/IP che li rende molto versatili. Esistono diverse versioni, sono economici, hanno dimensioni ridotte, possono comportarsi sia come *AccessPoint*, sia come *Station*, non hanno bisogno di un controllore esterno perchè ne hanno uno già integrato, possono essere programmati anche dall'IDE di Arduino e per tutti questi motivi sono diffusissimi in campo IOT.

Su questi moduli è già stato scritto molto nelle pagine di questa rivista e non ci soffermeremo sulle loro caratteristiche, noi li useremo all'interno di una scheda di sviluppo chiamata *NodeMCU* (è forse quella più usata, ma ne esistono di tante versioni simili) che si interfaccia al PC tramite connettore USB (connettore tipo micro B) e presenta un regolatore di tensione (per far funzionare il modulo ESP a 3.3V) ed un chip di conversione seriale (tra USB e i pin del modulo) ed ovviamente il modulo stesso. Per scrivere la nostra libreria non ci servirà nient'altro che questa scheda e la lampada WiFi.

Se non avessimo già installato la toolchain di sviluppo per ESP8266, dovremmo farlo impostando

Fig. 1
Schema a blocchi del processo da seguire per scrivere la nuova libreria.



dall'IDE di Arduino il campo *Additional Boards Reference* (nella finestra di *Preferences*) con questa stringa http://arduino.esp8266.com/stable/package_esp8266com_index.json per il modulo ESP8266 e questa https://dl.espressif.com/dl/package_esp32_index.json per il modulo ESP32. Successivamente occorre installare le toolchain di sviluppo dalla finestra *Boards Manager*, come in **Fig. 3**, in modo da scaricare su PC le librerie native, i file con le piattaforme disponibili e qualche esempio per iniziare a programmare su questi moduli. Il modulo ESP32, rispetto al cugino ESP8266 integra al suo interno anche il protocollo Bluetooth BLE e ci permetterà di ampliare il raggio di applicazioni possibili.

Noi useremo il modulo ESP in modalità Station, in modo da farlo connettere al nostro router, sul quale sarà già stata connessa la lampadina WiFi che controlleremo. Il codice per il collegamento al router è già presente negli esempi forniti con la toolchain e si presenta come nel **Listato 1**.

LIBRERIA TUYALAMP

Inizieremo scrivendo una libreria per il modulo ESP8266, ma in buona parte dei casi una libreria scritta per ESP8266 può funzionare anche sul più evoluto ESP32, occorre, tuttavia prestare particolari accorgimenti nel caso in cui vengano usate funzioni di librerie native che hanno subito modifiche da

 **Listato 1**

```
#include <ESP8266WiFi.h>
const char* ssid    = "STATION";
const char* password = "pwd1234";
void setup()
{
    Serial.begin(115200);
    Serial.print("Connecting to ");
    Serial.println(ssid);
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED)
    {
        delay(500);
        Serial.print(".");
    }
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
}
```

COMANDO	DESCRIZIONE	PARAMETRI	ESEMPIO
1	Accensione	true	"1":true
	Spegnimento	false	"1":false
2	Modalita'	white colour scene music	"2": "white" "2": "colour" "2": "scene" "2": "music"
3	Luminosita' del bianco	25 - 255	"3": 243
4	Temperatura del bianco	0 - 255	"4": 23
5	Settaggio colore	"RRGGBB0HHHSSVV" RR: 0 - 255 GG: 0 - 255 BB: 0 - 255 HHH: 0 - 360 SS: 0 - 100 VV: 25 - 100	"5": "004e000078ff5e"

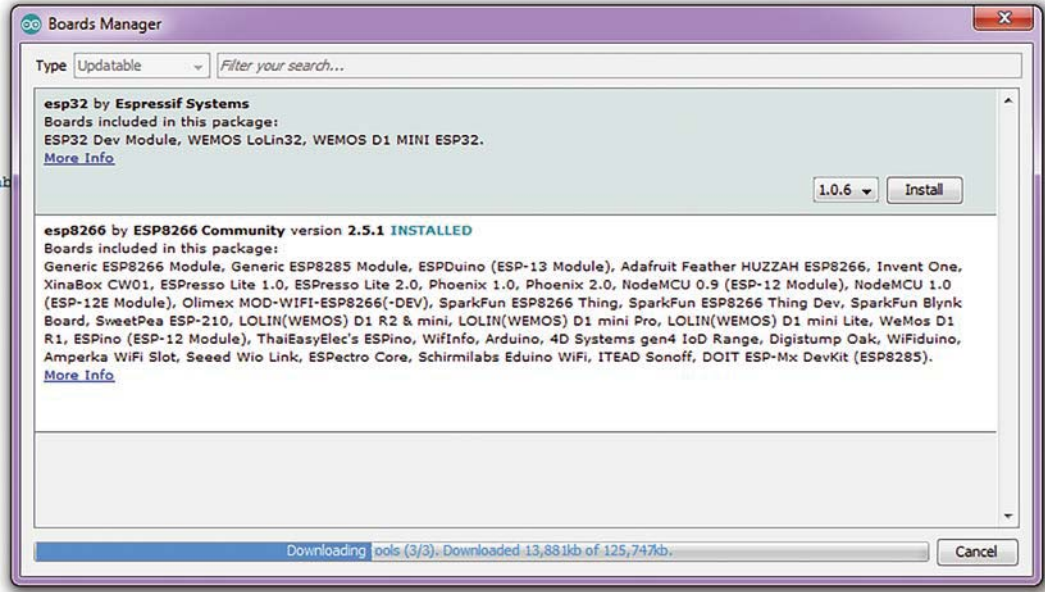
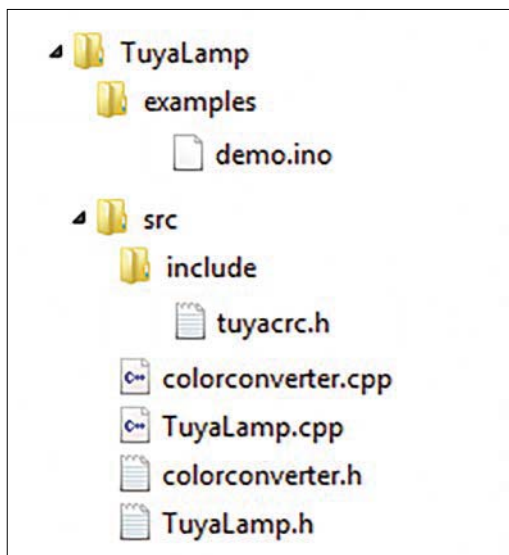


Fig. 2
Lista dei comandi che verranno gestiti dalla libreria.

Fig. 3
Finestra per il download della toolchain per ESP8266 e ESP32.

Fig. 4
Struttura della
libreria per i moduli
ESP8266 e ESP32.



un modulo all'altro ed in tal caso intervenire con le direttive `#if defined()` specificando il diverso comportamento. Ad esempio, per connettere il modulo alla lampada ed inviare pacchetti TCP, alcune funzioni come `connect()` sono contenute in librerie diverse (*WifiClient.h* per ESP8266 e *WiFi.h* per ESP32) e presentano un numero di parametri differenti, altre funzioni invece, non sono nemmeno comuni. Prima di tutto inseriamo una nuova cartella sotto la cartella *Libraries* dove sono contenute le altre librerie, creando una struttura come in **Fig. 4**.

In particolare il nome della cartella avrà il nome della libreria, conterrà un file descrittivo *library.properties* con alcune informazioni e due cartelle. Sotto *src* inseriremo i file che costituiscono la libreria vera e propria che sarà scritta in C++, si tratta di *TuyaLamp.cpp* e *TuyaLamp.h* (dal nome della libreria). All'interno della stessa cartella potrebbero esserci altri file di codice di supporto che potranno essere scritti in C (file con estensione .c) o anche in C++ (file con estensione .cpp) e che quindi conterranno altre classi, come ad esempio la classe *colorconverter* che, nel nostro caso, useremo per effettuare le conversioni da un formato di colore all'altro. Potrebbe esserci, anche se non è obbligatorio, anche una cartella *include* che contiene file.h con funzioni ausiliarie; nel nostro caso troviamo il file *tuyacrc.h* in cui abbiamo scritto una tabella per il calcolo CRC del pacchetto TCP. Infine parallelamente alla cartella *src* potrà esserci una cartella *examples* contenente file con estensione .ino, quindi veri e propri sketch in cui mostriamo come utilizzare la nostra libreria.

Rivediamo la lista dei comandi presenti in **Fig. 2** e

consideriamo che tutto si basa sulla codifica di una stringa di testo di questo tipo:

```
{ "devId": "3602383098f4abc8b73c", "t": 1613242568, "dps": {  
  "5": "ff00000167ffff" } }
```

In cui *devId* è il codice univoco che rappresenta la particolare lampadina, *t* è il timestamp che, però, può essere sempre lo stesso e non dobbiamo preoccuparci di aggiornarlo, *dps* è la lista di comandi e "5" è il comando per settare il colore secondo lo schema in **Fig. 2**.

Possiamo già pensare all'interfaccia della nostra libreria scrivendo il file .h che sarà come mostrato nel **Listato 2**.

Commentiamo brevemente i vari metodi che compaiono nell'interfaccia della classe *TuyaLamp*, senza entrare nel dettaglio del codice interno anche perché è pressoché lo stesso di quello presente nel programma in C# visto precedentemente, salvo alcuni accorgimenti dovuti al fatto che stiamo scrivendo codice per microprocessori che, se pur potenti, hanno risorse di memoria comunque limitate. Per questo motivo, se per esempio in C# potevamo usare variabili di tipo intero per rappresentare le singole componenti RGB, qua conviene usare byte per ottimizzare spazio. Il codice completo è comunque scaricabile dal sito della rivista.

Partendo dall'alto abbiamo le funzioni private come *encryptMessage()* che effettua la codifica AES128 del messaggio di testo suddividendolo in blocchi da 16 byte (128 bit da cui appunto AES128) e trasformandoli in altrettanti blocchi della stessa dimensione, ma codificati usando il modulo *AES128* che appartiene ad una libreria esterna e che, se non già presente all'interno della cartella *Libraries*, occorrerà scaricare preventivamente. Ogni blocco verrà codificato usando la chiave di crittazione ricavata dal sito *Tuya* come visto nell'articolo precedente, e che abbiamo a disposizione avendola ricevuta dal costruttore (*localkey*). Questi blocchi verranno uniti in un unico buffer interno che costituirà parte del payload del pacchetto finale. Ovviamente più lunga sarà la stringa di comando, maggiori saranno i blocchi e più lungo sarà il buffer che però avrà sempre una lunghezza multipla di 16 byte; nel caso molto probabile in cui la stringa di testo avrà una lunghezza non multipla di 16 caratteri, sul blocco finale verrà fatto un padding di byte a zero.

La funzione *createPacket()* utilizza il buffer appena creato e lo inserisce all'interno di un altro ancora più grande aggiungendo alcuni byte di comando, prefisso e suffisso (già esaminati in passato) e su questo

calcola la CRC di controllo utilizzando la funzione *tuyacrc()* che a sua volta sfrutta un buffer contenuto nella cartella *include* vista prima.

La funzione *preparePacketFromCommand()* è solo un wrapper che usa *encryptMessage()* e *createPacket()* e nasce per eseguire i primi test passandogli direttamente l'intera stringa di comando vista sopra, poco pratica, ma utile in fase di debug. A tal scopo viene usata *commandBuilder()* che costruisce in automatico la stringa di test inserendo al suo interno il *deviceId*, il timestamp e la stringa di comando vero e proprio; a seconda del tipo di comando (modalità colore, modalità bianco, cambio componenti colore, cambio luminosità e temperatura del bianco) si comporterà in modo diverso per comporre correttamente la stringa di testo che rappresenta il comando.

Infine, tra le funzioni private, la *connect()* e la *sendPacket()* usano il modulo esterno *WiFiClient* (occorre includere la libreria *WifiClient.h* o la *Wifi.h* nel caso di ESP32, che sono di sistema e si trovano già nella toolchain scaricata in precedenza) e si occupano rispettivamente di connettersi alla lampadina ed inviare il pacchetto costruito dalla linea di comando.

Entrambe sono chiamate da *cmdActivator()* che prende in ingresso proprio la stringa di comando completa e crea il pacchetto TCP prima di inviarlo effettivamente alla lampadina.

Passiamo ora alle funzioni pubbliche che userà l'utente finale e che chiameranno tutte la *cmdActivator()* dopo aver interpretato il comando. *TurnOn()* e *TurnOff()* non hanno bisogno di commenti ed usano il comando di tipo 1, passando parametro *true* o *false* rispettivamente.

SetColor() si presenta in tre forme diverse e sfruttando l'overloading del C++, ha tre comportamenti diversi a seconda del numero e del tipo dei parametri usati; senza parametri abbiamo il passaggio alla modalità colore ritrovando il colore impostato in precedenza, con tre parametri di tipo *byte* imposteremo il colore RGB corrispondente, mentre con il primo parametro di tipo *short* (la tonalità Hue è rappresentata da un valore che varia da 0 a 360, quindi un byte non basterebbe) e altri due byte (*Saturation* e *Value* sono rappresentati da un valore che varia da 0 a 100) imposteremo il colore secondo lo schema HSV. Mentre la prima usa il comandi di tipo 2 al quale passa il parametro "*colour*", le altre due

↓ Listato 2

```
class TuyaLamp
{
    Public:
        TuyaLamp(String ip, String deviceId, String localKey);
    private:
        int _port;
        String _ip;
        String _deviceId;
        ...
        AES128 _aes128;
        WiFiClient _client;
        ColorConverter _colorUtility;
        // functions
        void encryptMessage(String plainText, byte outputBuffer[]);
        uint32_t tuyacrc(byte arrbytes[], int len);
        void createPacket(byte encBuffer[], byte outpacket[]);
        void preparePacketFromCommand(String command, byte packet[], int len);
        String commandColor(byte r, byte g, byte b);
        String commandColor(short h, byte s, byte v);
        String commandBuilder(String timestamp, String cmdType, String cmdValue);
        int connect();
        int sendPacket(byte packet[], int len);
        void cmdActivator(String command);
    public:
        void TurnOn();
        void TurnOff();
        void SetColor();
        void SetColor(byte r, byte g, byte b);
        void SetColor(short h, byte s, byte v);
        void SetWhite();
        void SetWhiteTemp(int temp);
        void SetWhiteBright(int brl);
        void SetWhite(int temb, int brl);
}
```

usano il comando di tipo 5 e per scrivere la stringa nel giusto formato con tutte le componenti di colore (sia RGB che HSV) viene usato l'oggetto `_colorUtility` che è un'istanza della classe `ColorConverter` creata in precedenza e contenuta nei file `ColorConverter.cpp` e `ColorConverter.h` visti prima.

Discorso analogo per le due `SetWhite()`, questa volta le componenti in gioco sono solo due, luminosità e temperatura del bianco; chiamando la `SetWhite()` senza parametri passeremo alla modalità del bianco col comando di tipo 2, mentre usandola con i due parametri verranno gestiti i comandi di tipo 3 e 4. Per permettere una regolazione separata tra lumi-

nosità e temperatura e dal momento che i comandi sono indipendenti, sono state create anche altre due funzioni specifiche: `SetWhiteBright()` e `SetWhiteTemp()`. Chiudono la libreria la funzione `DebugEnable()` per abilitare la stampa di messaggi di debug su seriale e la funzione `ShowBuffer()` che ci permette di visualizzare l'ultimo pacchetto codificato e che, in caso di problemi iniziali, potrà essere confrontato con quello in uscita dal programma C# per capire meglio gli errori in fase di sviluppo.

APPLICAZIONI PER ESP8266

Iniziamo dal modulo ESP8266 e vediamo qualche possibile applicazione che utilizzi questa libreria. La prima è di sicuro la più semplice, ed è nata in fase di test ed in parallelo con lo sviluppo della libreria stessa; ovvero la possibilità di impartire i comandi da seriale per controllare la lampadina in modo da verificare il corretto funzionamento di ogni metodo creato. Dopo aver inizializzato l'oggetto `TuyaLamp` passandogli l'IP della lampadina, il suo `deviceId` e la `localKey` e dopo aver connesso ESP8266 al nostro router, aspettiamo nel loop principale che venga premuto un tasto dal terminale seriale ed associamo una funziona ad ogni tasto. Nel caso delle funzioni che richiedono parametri, come ad esempio la `SetColor()` con i tre parametri a byte per impostare il colore usando componenti RGB, dobbiamo prevedere il caricamento (sempre via seriale e convertendo i caratteri inseriti in numeri) di tre variabili indipendenti che verranno visualizzate a schermo premendo il tasto 'h'(help) e che verranno usate nelle funzioni che usano parametri. Il numero di queste variabili è tre proprio perchè il numero massimo di parametri usati dalle nostre funzioni è tre. Una parte del codice del loop principale è visibile in figura nel **Listato 3**.

Ovviamente prima di tutto occorrerà richiamare la nostra libreria usando l'include

```
#include <TuyaLamp.h>
```

ed istanziare l'oggetto con:

```
TuyaLamp lamp("192.168.1.5", "3602383098f4abc8b73c",  
"c7e961ddad19566e");
```

Il codice completo è comunque disponibile sul sito della rivista.

Altra possibile applicazione potrebbe essere l'utilizzo del modulo in abbinamento con un sensore di distanza ad ultrasuoni (come il famoso HC04, già ampiamente utilizzato) installato nel garage in

↓ Listato 3

```
void loop()
{
    if (Serial.available() > 0)
    {
        // get incoming byte:
        char charReceived = Serial.read();
        Serial.print("pressed:");
        Serial.println(charReceived);
        if (recVar1==true)
        {
            var1[i1] = charReceived;
            i1++;
        }
        ...
        if (charReceived == 'x')
        {
            Serial.println("x");
            recVar1 = true;
        }
        ...
        if (charReceived == 0x0D)
        {
            if (recVar1 == true)
            {
                recVar1 = false;
                var1[i1] = '\0';
                i1=0;
                v1 = atoi(var1);
                Serial.print("x is ");
                Serial.println(v1);
            }
            ...
        }

        // --- turn on ---
        if (charReceived == 'a')
        {
            lamp.TurnOn();
        }
        // --- color hsv ---
        if (charReceived == 'v')
        {
            lamp.SetColor((short)v1, (byte)v2, (byte)v3);
        }
        // --- color rgb ---
        if (charReceived == 'u')
        {
            lamp.SetColor((byte)v1, (byte)v2, (byte)v3);
        }
    }
}
```



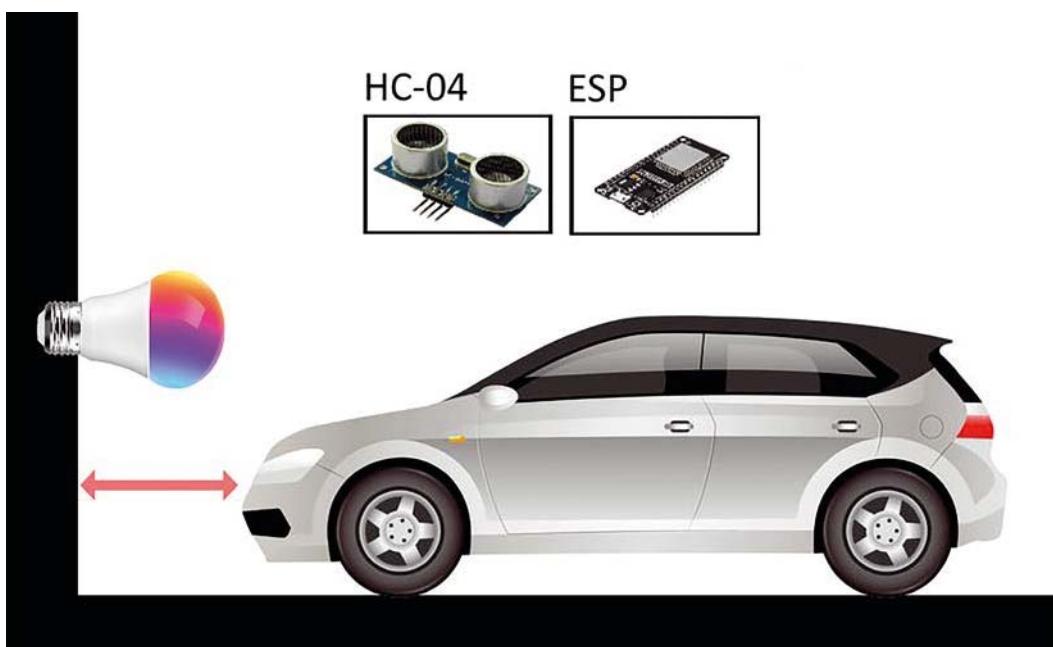


Fig. 5
Possibile
applicazione
utilizzando
sensore di
distanza.

Listato 4

```
#include <TuyaLamp.h>
#define THRESHOLD 100
#define SUP_LIMIT 10
#define INF_LIMIT 10
#define MAX_THRESHOLD (THRESHOLD + SUPLIMIT)
#define MIN_THRESHOLD (THRESHOLD + INFLIMIT)
...
TuyaLamp wifiLamp("192.168.1.5",
"3602383098f4abc8b73c", "c7e961ddad19566e");
void setup(void)
{
  Serial.begin(115200);
  initSonar();
  wifiLamp.SetColor((byte)0, 255, 0);
}
void loop(void)
{
  pulse_10us();
  float distance = getDistance();
  Serial.print(distance);
  Serial.println("cm");
  delay(250);
  if (distance > MAX_THRESHOLD)
  {
    // green
    wifiLamp.SetColor((byte)0, 255, 0);
  }
  else if ( distance < MIN_THRESHOLD )
  {
    // red
    wifiLamp.SetColor((byte)255, 0, 0);
  }
  else
  {
    // do nothings
  }
}
```

prossimità di zone "limite" in modo da segnalare con un colore rosso quando ci stiamo avvicinando troppo ad oggetti posti di fronte a noi e con un colore verde il via libera, come una sorta di semaforo. Occorrerà utilizzare un controllo con isteresi in modo da ridurre il rumore intorno alla distanza limite. Successivamente potrebbe rimanere acceso in modalità bianco per un tempo prefissato per darci possibilità di lasciare l'auto e chiudere il cancello. Il modulo ESP8266 ha ovviamente anche varie modalità di *sleep* per abbassare i consumi e nel caso il garage fosse lontano dal nostro router potremmo sempre configurare il modulo in modalità *AccessPoint* e connettere la lampada alla sua rete. Potere trovare un frammento di codice nel **Listato 4** ed il codice completo nel sito della rivista.

APPLICAZIONI PER ESP32

Proviamo ora la stessa libreria sul modulo ESP32 e sfruttando la sua capacità di gestire anche il protocollo Bluetooth, possiamo collegare un controller della PlayStation3 (*Sixaxis*) che con i suoi tasti e leve analogiche ben si presta per gestire la lampadina in tutte le sue potenzialità, regolandola nelle varie tonalità in modalità HSV o con le singole componenti di colore in modalità RGB o ancora modificando l'intensità e la temperatura del bianco. Per la gestione del controller *Sixaxis*, le cui caratteristiche sono elencate nel box presente in queste pagine, useremo una libreria già esistente, la "*PS3 Controller Host*" (scaricabile anche dall'IDE di Arduino

Grazie ad una libreria dedicata e al modulo Bluetooth a bordo di un ESP32 possiamo controllare le nostre applicazioni in modo semplice e senza fili tramite il controller della Playstation3 (denominato anche *Dualshock3*) che ci mette a disposizione una serie di tasti digitali ed analogici da riuscire a soddisfare ogni nostra esigenza.

Basti pensare che quasi ogni tasto (compreso i quattro con la forma di triangolo, quadrato, croce e cerchio) possono essere usati sia in modo digitale (evento di pressione e di rilascio) sia in modo analogico perchè forniscono in uscita un valore di tipo byte in base alla pressione con cui vengono premuti.

Inoltre la libreria ci fornisce anche informazioni sullo stato dell'accelerometro a 3 assi e del giroscopio contenuti all'interno del controller ed è in grado di attivarne la vibrazione e l'accensione dei led sulla parte frontale.

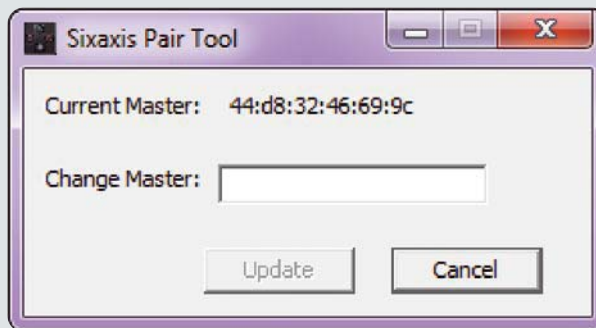
Per sfruttare queste potenzialità serve, però, sapere l'indirizzo MAC del bluetooth del controller e per fare ciò si può usare un programma gratuito per pc

(*SixaxisPairTool*) che, dopo aver collegato il controller al PC tramite cavetto USB, visualizzerà a schermo l'indirizzo desiderato. Brevemente un accenno su come usare questa libreria tramite un esempio minimale mostrato in questo box in cui abbiamo usato l'IDE di *Atmel-Studio* con il plugin *VisualMicro*.

Dopo aver incluso il file *PS3Controller.h* possiamo usare direttamente l'oggetto *PS3* nel *setup()* collegando, tramite i metodi *attachOnConnect()* e *attach()* le funzioni di callback che verranno automaticamente richiamate dalla libreria sul manifestarsi rispettivamente della prima connessione del controller al modulo ESP32 e di ogni altro cambiamento sui pulsanti e le leve.

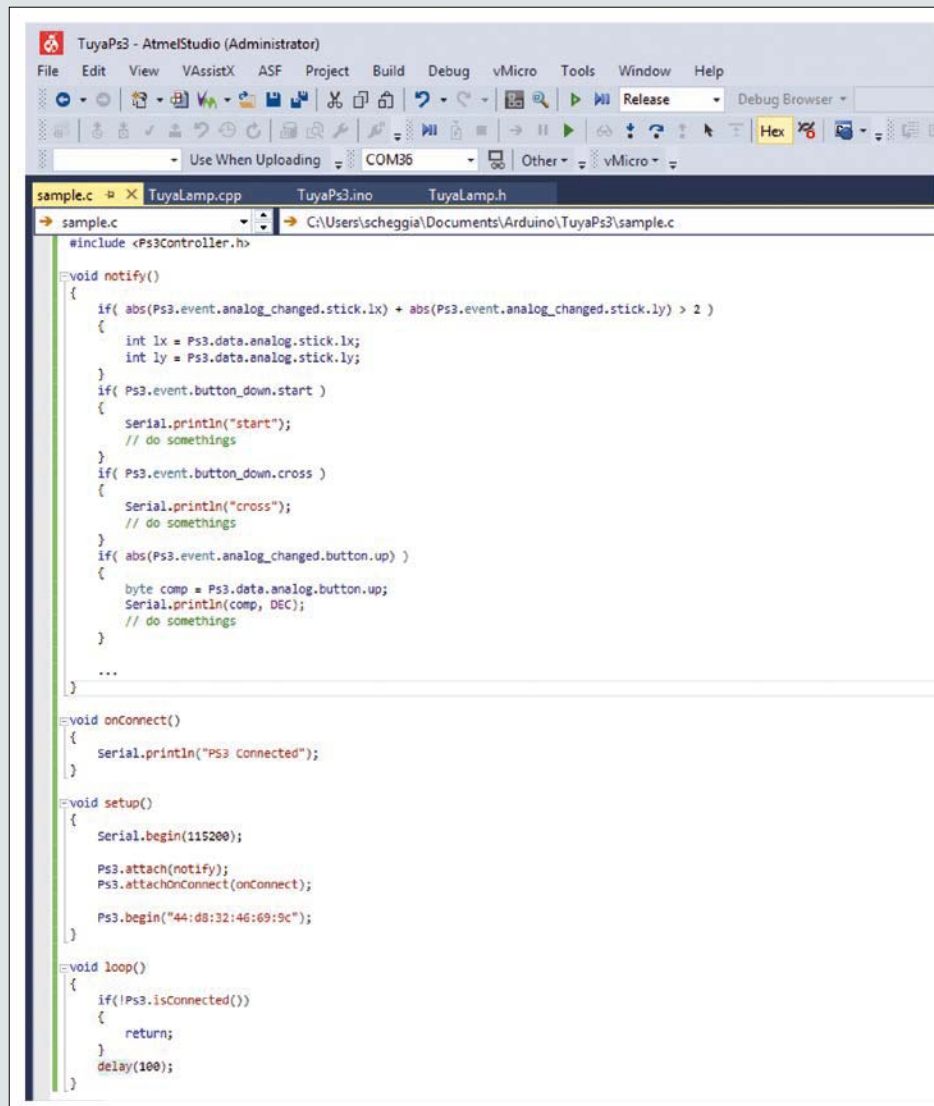
Su quest'ultima callback (nel nostro esempio *notify()*) verrà controllato quale evento ha scatenato il cambiamento potendo controllare davvero tutto e nel caso ci comporteremmo di conseguenza scrivendo il codice desiderato.

Ovviamente dopo aver impostato le callback possiamo avviare la libreria col metodo *begin()* e a quel punto il modulo



ESP32 rimarrà in attesa del collegamento del controller che avverrà dopo aver premuto il tasto centrale del controller (tasto *PS*).

Come avrete notato il metodo *begin()* ha bisogno dell'indirizzo MAC bluetooth che lo rende univoco esattamente come ogni altra interfaccia di rete, in questo modo il modulo accetterà la connessione solo dal quel determinato controller.



dalla finestra *Library Manager* che permette di gestire gli eventi di pressione e di rilascio di ogni singolo tasto e levetta, ma anche di rilevare i valori dell'accelerometro interno e dei motorini per la vibrazione. A noi, in questo caso, interessa qualche pulsante digitale per selezionare tre diverse modalità di regolazione (due per i colori, HSV e RGB, e una per il bianco) ed i controlli analogici per variare le singole componenti di colore e del bianco, approfittando anche del fatto che nella modalità di colore HSV (rappresentato proprio da un angolo giro) avere un controllo analogico a levetta come quello del *Sixaxis* è l'ideale.

Il modello HSV viene infatti, rappresentato proprio da un angolo giro, o meglio da un cilindro; se ci spostiamo lungo il perimetro del bordo abbiamo il colore pieno in tutte le sue 360 tonalità di colore (Hue), se, a parità di tonalità, ci avviciniamo verso il centro abbiamo un valore diverso di saturazione (Saturation) e se poi ci muoviamo in altezza otterremo un diverso valore di intensità (Value).

Inoltre il cambiamento di tonalità (Hue) del colore appare più evidente ai nostri occhi rispetto alla variazione di una singola componente RGB. Occorre però un calcolo aggiuntivo per questa rappresentazione, perchè la libreria del controller ci fornisce le coordinate x e y della levetta analogica, ma a noi interessa l'angolo per calcolare il valore Hue e quanto siamo distanti dall'origine (levetta ferma) per calcolare il valore di saturazione (Saturation). Per quanto riguarda Value dovremmo usare un altro tasto o levetta analogica. Per ricavare l'angolo useremo l'arcotangente in questo modo:

```
int deg = round( atan2 (-ly, lx) * 180 / 3.14159265 );
```

in cui *lx* ed *ly* sono i valori delle coordinate della levetta del controller forniti dalla libreria; sono di tipo *byte* e variano da -128 a 127, quindi nel caso in cui *deg* fosse negativo dovremo normalizzarlo eseguendo una somma di questo tipo:

```
deg = 360 + deg;
```

Per ricavare Saturation calcoleremo il modulo di questo vettore usando teorema di Pitagora, dopo aver normalizzato le coordinate della levetta ed infine moltiplicarlo per 100 dal momento che Saturation varia da 0 a 100.

```
float mod = sqrt(pow(lx/128.0, 2) + pow(ly/128.0, 2));
byte hsv_sat = mod * 100;
```

Listato 5

```
...
void loop()
{
    if(!Ps3.isConnected())
    {
        return;
    }
    int c = Ps3.data.analog.button.cross;
    if (c > 0)
    {
        if (_colorMode == COLOR_HSV)
        {
            Serial.println("CH");
            wifilamp.SetColor((short)hsv_hue, hsv_sat, hsv_val);
        }
        else if (_colorMode == WHITE)
        {
            Serial.println("WT");
            wifilamp.SetWhiteTemp(wTemp);
        }
        else if (_colorMode == COLOR_RGB)
        {
            Serial.println("CR");
            Serial.println((int)_rgb);
            if (_rgb == RED)
            {
                red = Ps3.data.analog.button.left;
            }
            else if (_rgb == GREEN)
            {
                green = Ps3.data.analog.button.up;
            }
            else if (_rgb == BLUE)
            {
                blue = Ps3.data.analog.button.right;
            }
            wifilamp.SetColor((byte)red, green, blue);
        }
        else
        {
        }
    }
    int d = Ps3.data.analog.button.down;
    if (d > 0)
    {
        Serial.println("WL");
        wifilamp.SetWhiteBright(wLight);
    }
}
```

Infine per Value basterà convertire il valore ottenuto da un altro tasto analogico in modo che vari da 0 a 100, in questo modo:

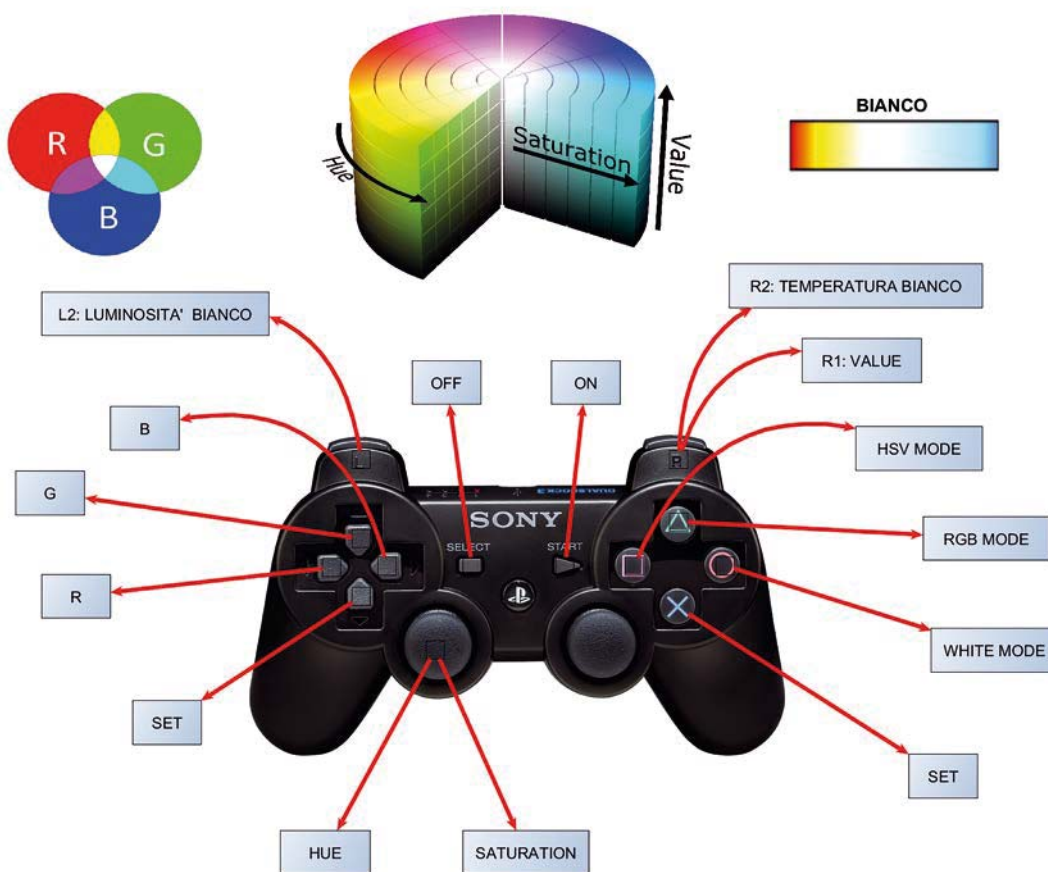
```
byte hsv_val = 100 - (ry * 100/255);
```

in cui *ry* è il valore analogico fornito dalla libreria e normalmente vale 100.

Abbiamo testato il funzionamento completo della lampadina associando i tasti come in **Fig. 6**, ma ovviamente è sempre possibile effettuare scelte diverse modificando le funzioni nel file principale, in particolare nella gestione degli eventi all'interno della funzione *notify()*.

A livello di funzionamento abbiamo scelto di rendere

Fig. 6
Associazione tasti
per il controllo della
lampadina.



effettive le modifiche solo sulla pressione dei tasti *SET* in modo da non appesantire lo stack TCP inviando di continuo pacchetti sulla rete ad ogni rilevazione di un nuovo evento, mentre useremo stringhe di debug seriali per testare ogni cambiamento. In particolare per quando riguarda la modalità del bianco, useremo il tasto *Down* per impostare il valore di luminosità al valore regolabile dal pulsante analogico *L2*, mentre per impostare il valore di temperatura agiremo sul pulsante analogico *R2*. Per la modalità HSV useremo solo il tasto *Croce* per settare il colore che varieremo con la levetta di sinistra (Hue e Saturation) ed il pulsante *R1* (Value), mentre per quanto riguarda la modalità RGB useremo sempre lo stesso pulsante, ma questa volta le componenti varieranno dalla pressione dei pulsanti *Left*, *Up* e *Right*, sulla sinistra. Presentiamo nel **Listato 5** il loop dell'applicazione principale, il codice completo è scaricabile dal sito della rivista.

CONCLUSIONI

Dal momento che il modulo ESP32 supporta anche Bluetooth 5.0, detto anche BLE da Low Energy, ed è lo stesso utilizzato dai vari dispositivi wearable che

ci circondano, potremmo anche pensare di associare una fascia cardio al modulo e collegare anche la lampadina facendo in modo di farla accendere e spegnere al ritmo del nostro battito cardiaco, mentre eseguiamo esercizi di ginnastica o pedaliamo su una cyclette; ovviamente anche l'intensità del colore potrà cambiare a seconda dello sforzo eseguito. Allo stesso modo tante nostre applicazioni potrebbero avere, è il caso di dirlo, una nuova luce ed avere anche un risvolto più domotico. L'utilizzo di questa nuova libreria potrà essere insomma molto vasto e limitato solo dalla nostra fantasia, a voi scoprirlo. ■

Cosa occorre?

La lampadina smart utilizzata in questo articolo è disponibile presso Futura Elettronica (cod. LEDWIFIBULB) a Euro 19,90. La breakout board con ESP8266 (cod. NODEMCUESP) è venduta a Euro 9,50, la scheda con ESP32 (cod. YB555) costa Euro 13,50. 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>

Sostituire il telecomando TV? Nulla di più facile!

Il tuo telecomando TV è rotto o non funziona più bene?
È arrivato il momento di sostituirlo con i telecomandi universali a infrarossi compatibili con le TV delle maggiori marche prodotte dal 2000 ad oggi. Dispongono di tutte le funzioni del telecomando originale. Subito utilizzabili!



PRONTI ALL'USO



Cod. **TVIR-SAMSUNG**



Cod. **TVIR-SONY**



Cod. **TVIR-PHILIPS**



Cod. **TVIR-LG**



Cod. **TVIR-PANASONIC**

€ 5,90
OGNI PEZZO

DA PROGRAMMARE



€ 9,00
cod. **TVFREEDOM2**

2 in 1

DISPONIBILE ANCHE 1 in 1
cod. **TVFREEDOM1**

€ 8,50

Telecomando universale, programmabile da PC, per controllare qualunque dispositivo di qualsiasi marca e modello. Include tutte le funzioni dell'originale, configurazione facile e veloce. Mai obsoleto grazie ad aggiornamenti gratuiti. Oltre 260.000 modelli in continuo aggiornamento, può comandare fino a due dispositivi audio/video: TV, TDT, SAT, DVD, BLURAY, proiettori, soundbar, VCR, media center e impianti Hi-Fi delle migliori marche. Richiede il download su PC-Windows di software gratuito per la selezione e la programmazione del telecomando.



Where are you?

Localizzatori GPS/GSM/GPRS per tutte le esigenze in grado di monitorare in tempo reale l'auto, la moto, la bicicletta, i bagagli e tanto altro.



LOCALIZZATORE GPS/GSM/GPRS CON ASCOLTO AMBIENTALE

Segui in tempo reale, tramite browser e APP gratuita, il percorso di veicoli e persone! Oltre a visualizzare i movimenti del tuo veicolo in tempo reale, potrai controllare la velocità, la direzione e i punti di sosta. Il localizzatore permette anche di gestire una flotta di autoveicoli con un unico account. Dispone di batteria ricaricabile agli ioni di litio da 6.000 mA che fornisce un'autonomia prolungata nel tempo (fino a 30 giorni in standby). Grazie al potente magnete di cui è dotato, è possibile fissarlo su superfici metalliche.



cod. FR763

€ 74,00



MICRO LOCALIZZATORE SATELLITARE GPRS/GSM

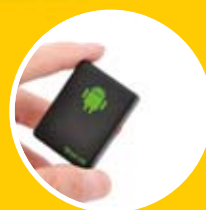
Tutto sotto controllo con il nuovo micro localizzatore GPRS/GSM. Per attivarlo basta inserire una SIM di qualsiasi operatore; inviando un SMS potrai attivare l'ascolto ambientale oppure ricevere un link con le coordinate che indicano la posizione in tempo reale del localizzatore. Dispone di batteria agli ioni di litio ricaricabile. Ideale per tenere sotto controllo la posizione di autoveicoli, persone, cose, animali domestici, bagagli, ecc. La confezione comprende: il micro localizzatore, il cavetto USB-micro USB per la ricarica della batteria e le istruzioni in italiano.

Dimensioni: 43,2x32x14 mm.
Peso: 20 grammi.



cod. A8TRACK

€ 14,00



LOCALIZZATORE TASCABILE GPS/GSM/GPRS

Ideale per la localizzazione personale e veicolare. Può inviare le proprie coordinate (latitudine e longitudine) via SMS verso telefoni cellulari, oppure, tramite tecnologia GPRS, ad un computer opportunamente configurato. Dispone di microfono integrato e batteria ricaricabile.



cod. TK102

€ 54,00

di DAVIDE
SCULLINO

GESTIONE ELETTRONICA DELL'INIEZIONE DIESEL

I moderni motori diesel utilizzano elettroiniettori piezoelettrici e tanta elettronica, che ne hanno elevato l'efficienza, ma che hanno introdotto problemi ignoti con i diesel tradizionali.

I motori delle automobili moderne sono tutti ad iniezione elettronica: non soltanto quelli a benzina ma anche i diesel, giacché il Common-Rail (rimasto ormai l'unico sistema di alimentazione per il motore a gasolio in grado di soddisfare i requisiti della norma antinquinamento EURO IV) è una soluzione di iniezione elettronica del combustibile, seppure ad alta e ad altissima pressione (fin a 2.200 bar). La tanta e forse troppa elettronica presente, se da un lato ha consentito di elevare il rapporto potenza/cilindrata e di adeguarsi alle sempre più stringenti normative antinquinamento, dall'altra ha reso il diesel e alcune sue parti meno durature e più delicate, tanto che se un tempo il punto forte del diesel rispetto al motore a benzina era nell'essere puramente meccanico, oggi anch'esso ha bisogno dell'elettronica per funzionare.

In queste pagine esporremo questo concetto analizzando un aspetto divenuto rilevante negli ultimi anni, con la massiccia adozione degli iniettori ad attuatore piezoelettrico da parte di tutti i costruttori di motori diesel.

ELETTROINIETTORI DIESEL: NECESSARI MA DELICATI

Un esempio della maggior delicatezza del diesel moderno, pur reso più efficiente dalla gestione elettronica (che permette di controllare la combustione con maggiore precisione dei sistemi meccanici e pneumatici) si può vedere negli iniettori di combustibile, che sono composti da una valvola a differenza di pressione e di una a spillo come quelli di un tempo, puramente meccanici, ma dove l'afflusso del gasolio non è più dettato dalla mandata della pompa d'iniezione a distributore o in linea, bensì viene regolato da un'elettrovalvola che la ECU (Engine Control Unit o Electronic Control Unit che dir si voglia...) attiva mediante un opportuno impulso elettrico amplificato da un transistor di potenza (BJT o MOSFET) ogni volta che deve avvenire un'iniezione nel cilindro. Le stringenti normative antinquinamento hanno elevato il livello dei requisiti che tali elettrovalvole devono avere, soprattutto in fatto di tempo di risposta all'impulso elettrico di comando, perché nei diesel Common-Rail dall'EURO IV in poi sono richieste più iniezioni per ogni ciclo di combustione:

mediamente una o più pre-iniezioni, un'iniezione principale e una o più post-iniezioni, l'ultima delle quali viene prolungata nella fase di rigenerazione del filtro antiparticolato (FAP o DPF che sia) almeno nei motori che non hanno il post-iniettore fuori dalla camera di combustione.

Considerato che il regime di giri massimo di un diesel a quattro tempi può arrivare a 4.500 giri/minuto e che nel motore siffatto c'è una combustione ogni due giri dell'albero motore (quindi due fasi attive anche ogni 2.250 giri/min.) e alla luce del fatto che ogni giro consiste in due fasi (aspirazione/compressione e combustione/scarico) ogni ciclo di iniezioni deve durare sempre meno di $\frac{1}{4}$ del regime massimo di giri: teoricamente meno di 110 microsecondi. Nella pratica per collocare 4 o 5 iniezioni in un ciclo di combustione, queste arrivano a durare pochissime decine di microsecondi.

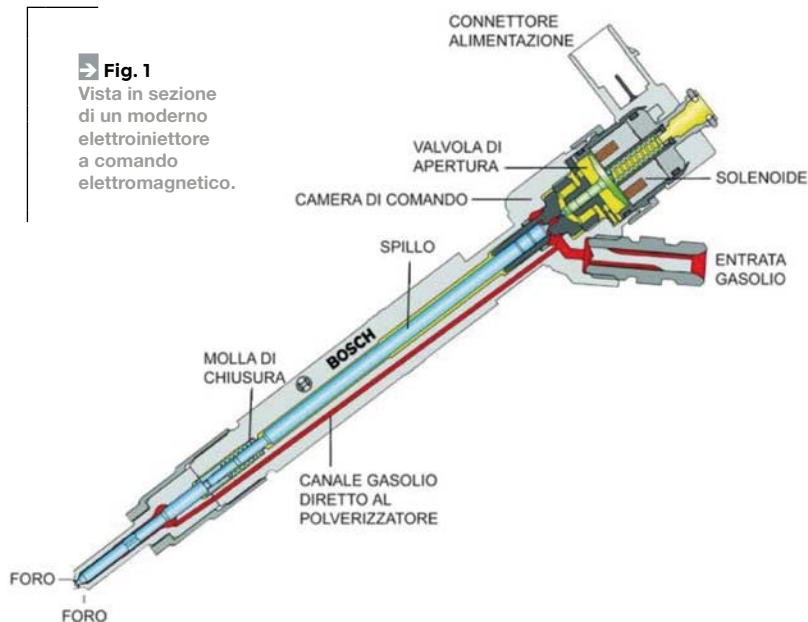
DAL MAGNETICO AL PIEZOELETTRICO

Questo ha imposto agli iniettori tempi di risposta che con quelli ad attuatore elettromagnetico (dove un solenoide, quando riceve l'impulso di comando dalla centralina, crea un campo elettromagnetico che attrae e solleva lo spillo che apre al gasolio la via verso il pulverizzatore) non sono ottenibili, almeno nei motori diesel automobilistici (i cosiddetti "diesel veloci"); la tecnologia motoristica ha quindi assistito al passaggio dall'iniettore elettromagnetico a quello piezoelettrico, il cui tempo di risposta è mediamente 5 volte inferiore. L'attuatore piezoelettrico dell'iniettore diesel è sicuramente più veloce di quello elettromagnetico, però presenta due limiti, che in alcuni sistemi di iniezione diesel, soprattutto nei Siemens-VDO e Continental Automotive (che poi sono ormai la stessa cosa) si sono rivelati tutt'altro che superabili: il primo è la limitata estensione dell'attuatore che comanda l'apertura della valvola a differenza di pressione, mentre il secondo è la durata dell'elettroattuatore, causata sia dall'insorgere di cortocircuiti nello stack piezoelettrico, sia da una sorta di effetto memoria dello stesso, che si adegua lentamente all'estensione imposta dagli impulsi di comando caratteristici del motore, fino a diventare, con il tempo e con l'usura della parte meccanica dell'iniettore, inadeguato a rispondere prontamente agli impulsi e quindi a rientrare nei parametri di calibrazione.

Per capire questo problema -che peraltro si può aggirare con una tecnica di riestensione dello stack piezoelettrico- bisogna spiegare come è

Fig. 1

Vista in sezione di un moderno elettroiniettore a comando elettromagnetico.



fatto l'iniettore ad attuatore piezo.

Si può iniziare col precisare che gli iniettori dei sistemi common-rail sono di due tipi: ad attuatore magnetico e piezoelettrico.

Nel primo, la valvola che lascia defluire il combustibile dalla camera superiore e che quindi permette l'iniezione è comandata da un solenoide che, percorso da corrente elettrica genera un campo elettromagnetico in grado di sollevare un tappo di metallo che scopre la luce di recupero, attraverso l'attrazione e il sollevamento di un cilindretto di azionamento (**Fig. 1**). L'alimentazione è ad impulsi dell'ordine di 12÷14V con correnti impulsive di decine di ampere.

Nel secondo, la valvola di uscita del gasolio viene gestita (aperta) da un dispositivo che sfrutta l'effetto piezoelettrico inverso, che consiste nella proprietà di alcuni materiali di deformarsi se sottoposti a una differenza di potenziale; la tipica struttura di questo elettroiniettore è proposta nella **Fig. 2**. L'iniettore a comando piezoelettrico consuma meno elettricità di quello magnetico ed è inoltre più veloce: si aziona cinque volte più velocemente di una elettrovalvola elettromagnetica. Il tipico attuatore per iniettore piezo è composto da un attuatore formato da più elementi piezoelettrici (anche più di 200) impilati, alimentati in serie-parallelo; la struttura siffatta (**Fig. 3**) serve perché un solo elemento non si dilaterrebbe abbastanza da spostare la valvola dell'iniettore fino a far alzare lo spillo.

Per fare un esempio pratico prendiamo in esame un attuatore piezo per iniettore del sistema Siemens VDO Piezo Common Rail (PCR): è costituito da una pila (stack) di circa 350 lamine di ceramica piezoelettrica, ognuna delle quali ha uno spessore di circa 80 μm . Sottoposto alla tensione di controllo, l'attuatore si espande fino a 40 μm . Poiché questa è una dimensione ridottissima, in sede di fabbricazione le superfici delle lamine devono essere realizzate con una tolleranza di 1 μm .

Nella **Fig. 4** è possibile vedere la tipica composizione dell'elettroattuatore piezoelettrico di un moderno iniettore Siemens-VDO: il dado serve ad avvitarlo al corpo meccanico dell'iniettore e la membrana (elastica) a sigillare lo stack ad evitare che il gasolio entri in contatto con la parte elettrica. La membrana flette quanto basta a consentire allo stack di dilatarsi in modo da permettere al pistone di muoversi e spingere la valvola ad aprirsi. Il problema della perdita di calibrazione dell'iniettore piezo al passare del tempo si comprende considerando che alle velocità di funzionamento

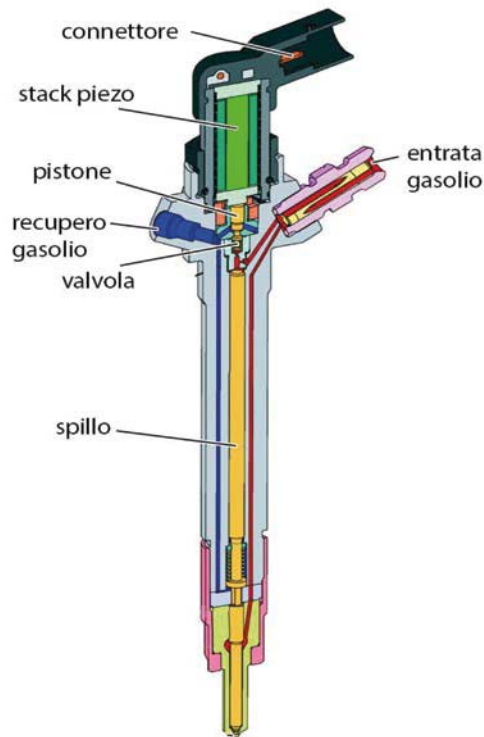


Fig. 2
Sezione dell'elettroiniettore a comando piezoelettrico.

dei moderni Common-Rail, la ECU deve conoscere esattamente il ritardo di apertura dello spillo e quindi di iniezione rispetto all'invio dell'impulso di comando all'elettroattuatore; questo ritardo viene rilevato in fabbrica con un apparato di test che come esito esprime il ritardo sotto forma di codice numerico (per esempio l'IMA, composto da 1 o 2 caratteri e/o colori per gli iniettori la cui risposta è stata misurata in un solo punto della curva caratteristica di funzionamento oppure 8/9 caratteri per gli iniettori codificati su 4 punti)

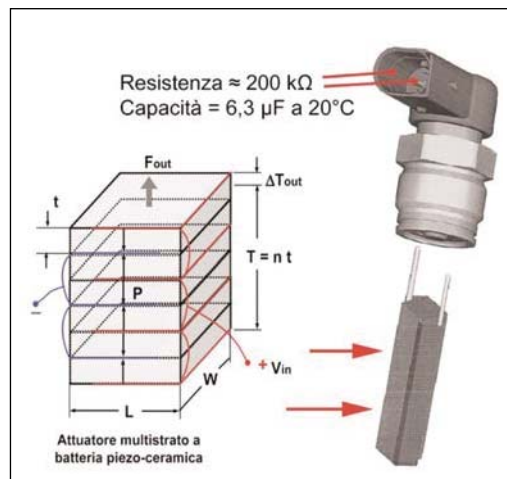
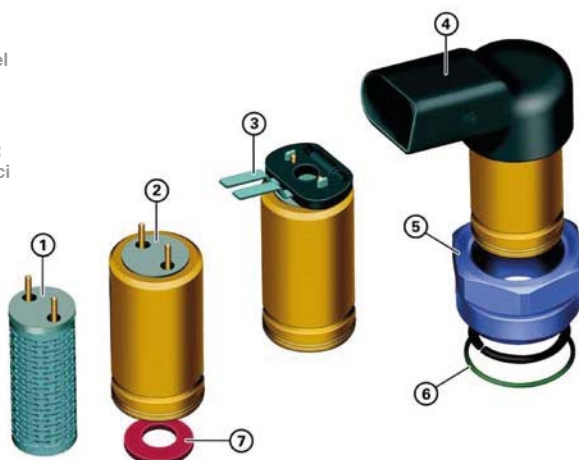


Fig. 3
Struttura dello stack piezoelettrico (a sinistra) e sua collocazione all'interno dell'elettroattuatore in un iniettore Siemens (ECU SID805).

Fig. 4
Composizione del blocco attuatore piezoelettrico:
1) stack;
2) alloggiamento;
3) contatti elettrici a lamina;
4) connettore;
5) dado;
6) guarnizione;
7) membrana separatrice.



stampato sul corpo dell'iniettore stesso. Il codice va caricato nella ECU, almeno nelle vetture con motore diesel dall'EURO V in poi. Nel funzionamento del motore, il suddetto tempo di risposta dell'iniettore può essere alterato da vari fattori: la densità del gasolio (dipendente tra l'altro dalla temperatura) e l'usura meccanica dello spillo e della valvola di apertura a differenza di pressione. Per questo la ECU ammette un range di tolleranza rispetto al tempo (ossia al codice) inserito per ciascun iniettore e soprattutto tenta periodicamente e in determinate condizioni operative del motore la ricalibrazione del tempo di attivazione. Il problema è che a questi scostamenti nella risposta si aggiunge l'usura dell'elettroattuatore piezoelettrico e della "spina" (pistone) che lo fa agire, in estensione sotto tensione elettrica, sulla valvola di comando dello spillo. Finché riesce, la ECU eseguendo un apprendimento dinamico

va a compensare l'aumento del tempo di risposta, ma ad un certo punto l'iniettore non può essere più ricalibrato.

Tale condizione si verifica anche perché lo stack di pastiglie piezoelettriche a un certo punto si abitua al regime di funzionamento (tanto più se al motore vengono chieste di rado elevate prestazioni) e quindi a estendersi per brevi periodi quali quelli richiesti dagli impulsi di attivazione forniti dalla ECU, i quali sono molto brevi soprattutto nei common-rail a più iniezioni per ciclo di combustione; questa ridotta capacità di estensione fa sì che a un certo punto la valvola non si apra più bene e che quindi l'iniettore non inietti più la quantità di gasolio richiesta, specie quando la ECU richiede il massimo delle prestazioni. Peraltro la risposta incorretta agli impulsi di comando non consente la corretta calibrazione degli iniettori da parte della ECU stessa e di conseguenza non è possibile far aprire gli iniettori nei momenti ottimali per il funzionamento del motore.

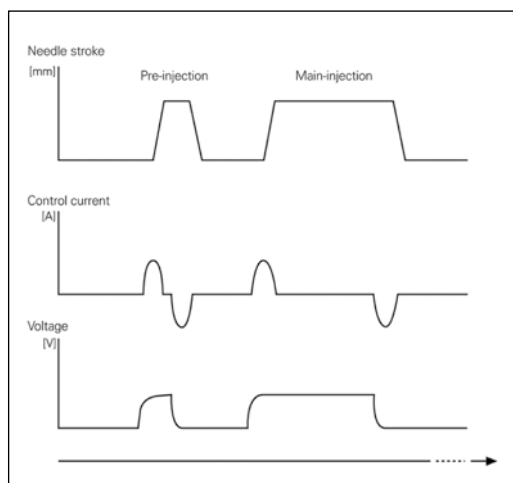
Quindi se in condizioni normali lo stack potrebbe recuperare il ritardo di apertura della valvola di comando dello spillo dell'iniettore, quando non riesce più a estendersi correttamente l'iniettore funziona male.

Tale concetto si spiega meglio comprendendo come vengono pilotati gli iniettori.

ATTIVAZIONE DELL'INIETTORE PIEZO

Il comando degli iniettori piezoelettrici è più complesso di quello degli iniettori elettromagnetici, perché mentre in quest'ultimi togliendo tensione il campo magnetico smette quasi subito (di norma basta applicare in antiparallelo un diodo di blocco) e lo spillo torna in posizione, nel piezoelettrico -che ha natura capacitiva- lo stack rimane dilatato a lungo perché la capacità parassita resta carica. Quindi il comando prevede quattro fasi: per aprire l'iniettore si applica un impulso di tensione dell'ordine dei 60-70V e poi lo si toglie; per mantenerlo aperto basta togliere tensione (perché l'elettroattuatore ha natura capacitiva e quindi rimane carico), mentre per chiuderlo occorre forzarne la scarica mediante un impulso a tensione inversa. Con riferimento alla **Fig. 5**, in apertura si applica un impulso di tensione positiva che determina un impulso di corrente entrante nello stack; poi si sospende la tensione per il tempo in cui l'iniettore deve rimanere aperto e quando arriva il momento della chiusura si cortocircuita il piezo e a questo punto si verifica un impulso di corrente in senso opposto dovuto alla scarica

Fig. 5
Andamento nel tempo degli impulsi di tensione e della corrente negli attuatori piezo di un sistema Siemens: Needle stroke è la quantità di combustibile iniettato.



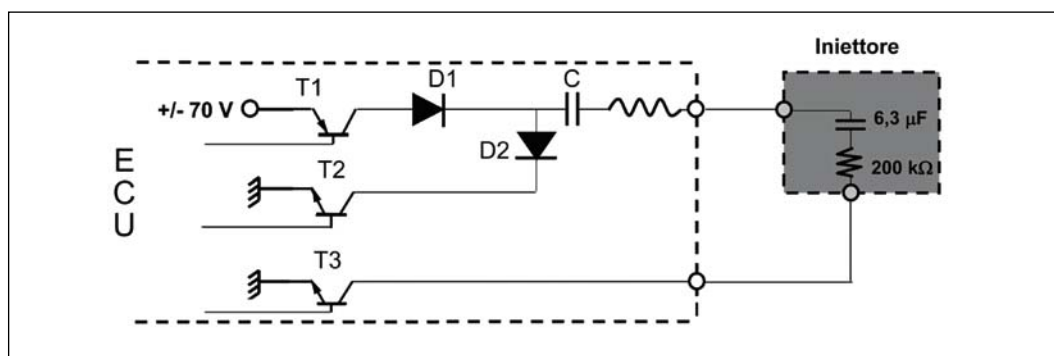
Nei motori più recenti gli iniettori vengono codificati nella ECU, inserendo mediante il software di diagnostica e manutenzione elettronica i codici riportati sul corpo; i codici sono stampati sull'elettroattuatore e forniscono un'informazione sulla tolleranza nel tempo di risposta dell'iniettore rispetto al valore teorico di riferimento. Esistono vari standard di codifica, uno dei quali è l'IMA, composto da 1 o 2 caratteri e/o colori per gli iniettori classificati (cioè la cui risposta è stata misurata in un solo punto della curva caratteristica di funzionamento) oppure 8/9 caratteri per gli iniettori codificati (risposta rilevata su 4 fasi del ciclo di iniezioni, ossia su pre-iniezione e iniezioni principali). La codifica serve quindi a calibrare il comando della centralina sulla base dell'effettiva risposta dell'iniettore, secondo le sue caratteristiche. La calibrazione (apprendimento dinamico degli iniettori) avviene periodicamente e in determinate condizioni, vale a dire in decelerazione entro un determinato range di giri (nei motori 1.6 Hdi PSA si svolge tra 2.600 e 1.600 giri/minuto) a una certa temperatura del liquido refrigerante e sopra una certa velocità. Se la ricalibrazione non riesce più entro il margine scritto nel firmware della ECU, si è nell'impossibilità di ottenere il dosaggio corretto del gasolio e una risposta adeguata dell'iniettore; tale anomalia viene riscontrata eseguendo la diagnosi con un tester OBD, anche uno generico, quindi segnalata con codici errore che sono P0263 (Iniettore cilindro 1 quantità carburante) P0266 (Iniettore cilindro 2 quantità carburante) P0269 (Iniettore cilindro 3 quantità carburante) P0272 (Iniettore cilindro 4 quantità carburante); per i motori a 6 cilindri, i codici di errore relativi dall'anomalia nel dosaggio della quantità di carburante sono P0275 (cilindro 5) e P0278 (cilindro 6). Per quelli a maggior numero di cilindri la numerazione dei codici di errore è simile, avanzando di 2 per volta.

RIGENERAZIONE DEL PIEZO

Tipicamente quando si verifica il problema descritto sugli iniettori piezoelettrici si va a sostituire l'attuatore piezo; purtroppo i produttori, da alcuni anni, per scelte commerciali, non forniscono più i ricambi della parte elettrica, ma solo dei polverizzatori e relative valvole, cioè non consentono alle officine di riparazione pompe e iniezione (i cosiddetti "pompisti"), come si faceva una volta, di revisionare totalmente l'iniettore sostituendo l'attuatore piezoelettrico. Le motivazioni verosimilmente nascono nel fatto che le normative dall'EURO 5 in poi impongono margini molto ristretti alla calibrazione e di conseguenza su un attuatore piezo non è possibile fornire un codice di calibrazione, perché dopo l'assemblaggio ci potrebbero essere delle discrepanze sull'iniettore ricalibrato; o meglio, sostituendo l'attuatore piezoelettrico, la ricalibrazione poi deve essere eseguita al banco con un macchinario che deve restituire i codici da inserire in centralina. Ora il problema è che teoricamente l'iniettore dovrebbe essere calibrato nel suo complesso e questo non è possibile vendendo e fornendo al pompista il semplice attuatore; per questo alcuni costruttori in un primo momento offrivano il servizio di revisione iniettori: un esempio era il programma DRS della Siemens-VDO che contemplava la rigenerazione del complesso dell'iniettore, che veniva inviato a dei centri autorizzati in cui i tecnici sostituivano l'attuatore e scrivevano loro un codice sull'attuatore stesso. Quindi l'iniettore veniva fornito codificato nel complesso, considerata la risposta elettrica dell'attuatore e la risposta della parte meccanica (lo spillo, la molla eccetera). Di fatto questo non avviene più, quindi la tendenza comune ai meccanici ma anche ai pompisti è quella di prendere l'iniettore guasto e sostituirlo con uno nuovo, con l'esborso che ne deriva per il cliente.

Fig. 7

Altro schema di comando per iniettori piezo: per aprire l'iniettore T1 conduce e porta corrente attraverso D1 e C; T3 conduce e T2 è interdetto. In chiusura T1 si apre e T2 conduce per scaricare la capacità interna.



Per consentire comunque il recupero degli elettroiniettori piezo non ricalibrabili dalla ECU motore, sono state sviluppate negli anni e testate, qualcuna con successo e qualcuna no, delle tecniche per riestendere lo stack piezo; proprio per fare la riestensione del piezo le macchine di test degli iniettori sono state aggiornate nel firmware da diversi costruttori (D.I.M., Checkstar Magneti Marelli, per esempio) per eseguire la rigenerazione. Ma in che cosa consiste sostanzialmente la rigenerazione? Consiste nel fornire ai contatti dell'elettroattuatore impulsi di tensione di corrente differenti più lunghi di quelli che riceve normalmente l'iniettore dalla sua ECU nel tipico funzionamento sul motore, nel tentativo di riportare fisicamente il piezo alla sua dimensione originale, o meglio alla sua estensione originale (l'estensione nominale del singolo elemento piezoelettrico dello stack è di 40 micron); chiaramente questo funziona soltanto quando non ci siano danni rilevanti nel reticolo cristallino delle singole lamine piezoelettriche, ovvero se anche si sia verificato il cortocircuito di una delle tante lamine nello stack, la quantità coinvolta sia minima: per esempio se in uno stack ci sono 100 lamine, il cortocircuito in 2, 3 o 4 non cambia molto l'esito, mentre se c'è un diffuso cortocircuito o il numero di lamine interessate è consistente, non è possibile la rigenerazione dello stack e dell'attuatore piezo, lo stesso è irrecuperabile.

Per questi motivi, prima di tentare la rigenerazione si verifica la resistenza ed anche la capacità del dell'elettroattuatore: la resistenza, nel caso degli iniettori Siemens non deve scendere sotto i 180 kohm (quella tipica è intorno ai 200 kohm) mentre per quanto riguarda la capacità, quella tipica è 5° 6 microfarad; negli iniettori piezoelettrici Bosch ci sono altri valori, però in linea di massima i range sono questi.

Il test si esegue con uno strumento specifico che va a misurare la resistenza e la capacità come farebbe un multimetro, ma con la differenza che generalmente gli strumenti specifici per questi test lavorano con tensioni sui 70-80 ed anche 120 volt in continua, mentre un tester comune che usiamo noi elettronici e gli elettricisti ha una batteria da 9 volt.

L'applicazione di una tensione tanto elevata è necessaria perché magari la risposta che si verifica a causa del cedimento della struttura cristallina del piezo (per esempio per breakdown elettrico) a una tensione di 9 volt non è la stessa che si verificherebbe con i 100 volt 80 volt e comunque



Fig. 8
Codici iniettori piezo scritti sull'attuatore: si tratta di iniettori Siemens VDO A2C59513556.

con quelle che normalmente sono le tensioni che la ECU invia all'iniettore durante il funzionamento nel motore.

In breve, un piezo che a 9V non è in cortocircuito può invece andarci a 70 volt, per ragioni legate alla rigidità dielettrica dei materiali costruttivi. Per la capacità vale lo stesso discorso: si usa una sorta di capacimetro che misura la frequenza o costante di tempo inserendo lo stack piezo elettricamente in un circuito alimentato a tensione comparabile con quella di normali pilotaggio fornita dalla ECU quando l'iniettore è montato nel motore.

Se la rigenerazione ha effetto, ossia se l'estensione del piezo ha successo, lo si nota perché misurando i parametri dell'iniettore dopo il procedimento, la resistenza si riporta praticamente ai valori di fabbrica dichiarati e si assiste anche a un aumento della capacità di questo condensatore virtuale che è formato dalle lamine piezoelettriche di tutto lo stack piezoelettrico.

Grazie alla procedura di rigenerazione dello stack piezoelettrico è possibile ripristinare la corretta funzionalità dell'elettroattuatore. Tensioni e correnti degli impulsi necessari alla rigenerazione dipendono dal tipo di iniettore, come pure la durata del trattamento elettrico richiesto. Su questo, tuttavia, ogni produttore ha sue filosofie che tende a non condividere.

Nei banchi di test Magneti Marelli Common Rail injectors testing serie DS1R/DS2R/CRU2R, per esempio, la procedura di rigenerazione ha una durata di 10-20 minuti in funzione del tipo di iniettore e del produttore; il costruttore del banco dichiara tensioni impulsive di rigenerazione e cicli di corrente che presumibilmente sono con valori al di sopra di quelli del normale funzionamento dell'iniettore. □

PER I TUOI PROGETTI UTILIZZA I SET DI COMPONENTI

Vasta scelta e grande risparmio!

Euro 14,90
cod. YM397



SET 1000 PEZZI DI LED
3 MM - VARI COLORI

Euro 5,90
cod. SETCONDELE



SET CONDENSATORI ELETROLITICI
12 VALORI

Euro 8,00
cod. YM403



SET 100 CONDENSATORI
POLIESTERE - VARIE CAPACITÀ

Euro 7,50
cod. KITFUS5X30



KIT FUSIBILI 82 PEZZI 6X30
ASSORTITI

Euro 7,50
cod. YM406



SET 300 PEZZI LED 3-5 MM
VARI COLORI

Euro 9,00
cod. YM374



SET 50 TRIMMER DI PRECISIONE
MULTIGIRI - MONTAGGIO VERTICALE

Euro 9,00
YM398



SET 500 PEZZI LED 5 MM
VARI COLORI

Euro 6,00
cod. YM379



SET COMPONENTI ELETTRONICI
ASSORTITI

Euro 12,00
cod. YM377



SET 300 TRANSISTOR ASSORTITI
TO-92

Euro 21,90
cod. SETELEC



SET COMPONENTI ELETTRONICI
VARI CON PIASTRA E ALIMENTATORE

Euro 8,00
cod. SET24INDU



SET INDUTTANZE 24 VALORI
10 PEZZI PER VALORE

Euro 10,90
cod. SMDRES3K610M



SET RESISTENZE SMD 0805
DA 3,6 KOHM A 10 MOHM

Euro 6,00
cod. 140ZENER



SET 140 DIODI ZENER
10 PEZZI PER VALORE

Euro 14,00
cod. SMDRES03K6



SET RESISTENZE SMD 0805
DA 0 A 3,6 KOHM

Euro 3,90
cod. SETCERAMICI



SET CONDENSATORI CERAMICI
22 VALORI - 10 PEZZI PER VALORE

Euro 6,50
cod. SETDIODI



SET DIODI RADDRIZZATORI
100 PEZZI

Euro 5,50
cod. SETTRIMMERMO



SET TRIMMER MONTAGGIO
ORIZZONTALE - ASSORTITI

Euro 7,00
cod. SETMULTIST



SET CONDENSATORI MULTISTRATO
VARI VALORI - 10 PEZZI PER VALORE

Euro 16,00
cod. KRES-0603



SET 1700 RESISTENZE SMD
(SERIE 0603) - ASSORTITE

Euro 6,50
cod. HS2133



SET 66 PZ ZOCOLI A LAMELLA
ASSORTITI



In questa puntata spieghiamo come inviare dati seriali ricevuti sull'UART del nostro dispositivo Azure Sphere reale al dispositivo virtuale creato in Azure IoT Central. Quarta puntata.

CONOSCERE

E USARE

Azure Sphere

4

di PIER CALDERAN

Continuiamo il nostro corso sul sistema Azure Sphere di Microsoft con un esempio che permette di leggere i dati della porta UART integrata e di inviarli a IoT Central.

L'esempio a cui faremo riferimento si chiama *Futura RX UART IoT Central* e si trova nella cartella *Futura_samples/Futura_RX_UART_IoT_Central* scaricabile, come anche tutti gli altri esempi di questo corso su GitHub all'indirizzo https://github.com/piercalderan/futura_MT3620.

UART

Il termine UART oggi viene spesso associato a una comunicazione seriale, ma forse non tutti sanno che UART è l'acronimo di Universal Asynchronous Receiver Transmitter ed è riferito a un dispositivo digitale fisico in grado di convertire un flusso di bit da un bus I/O parallelo a un bus I/O seriale.

La trasmissione è asincrona, per cui il formato dei dati e la velocità di trasmissione sono configurabili. I livelli e i metodi di segnalazione elettrica sono gestiti da un circuito di pilotaggio esterno all'UART. Un chip UART è solitamente un circuito integrato a se stante o parte di un microcontrollore utilizzato per le comunicazioni seriali, per esempio, su un computer o una periferica. Una o più periferiche UART sono comunemente integrate nei chip dei microcontrollori. Il protocollo di trasmissione hardware prevede che il trasmettitore (master) in un chip UART trasmetta i singoli bit in modo sequenziale. Nel ricevitore di destinazione (slave), un chip UART riassume i singoli bit in byte completi.

La velocità di trasmissione/ricezione UART si imposta in baud, un'unità di misura a "simboli" correlata al bit rate, ma che si differenzia dal bit rate perché a un simbolo corrisponde un numero di bit differente

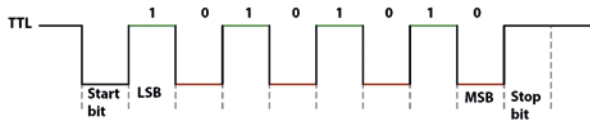


Fig. 1 - Trasmissione UART con configurazione 8-N-1.

variano a seconda del protocollo seriale usato. Per esempio, un pacchetto di dati tipico, chiamato anche 8-N-1, è formato dai bit seguenti (**Fig. 1**):

- 1 bit di start
- 8 bit di dati (8 bit)
- N bit di parità (nessuno)
- 1 bit di stop

LA LIBRERIA UART

Il progetto include la libreria di sistema UART opportunamente progettata per MT3620. Va inclusa all'inizio del codice con l'istruzione:

```
#include <aplibs/uart.h>
```

La libreria gestisce tutti i bus UART, ma per ottimizzare le risorse hardware, nella nostra scheda Futura MT3620 sono stati esposti il pin fisico 28 (GPIO38 condiviso con SCL di SAMPLE_ISU2_I2C) per RX UART e il pin fisico 27 (GPIO37 condiviso con SDA di SAMPLE_ISU2_I2C) per UART TX. In questo modo si possono gestire le due porte escludendo il bus I2C ISU2, che nella scheda è previsto per il sensore opzionale RTC DS3231. Per usare il nostro esempio con RX UART, bisogna escludere comunque il bus I2C ISU2. Per fare questo, basta impostare il DIP switch 2 su ON e usare il pin SDA del connettore C3 della scheda. Per chiudere il circuito, basta un qualsiasi pin GND della scheda (presente anche nel connettore C3). Nella **Fig. 2** è illustrato il collegamento fisico al connettore C3 e la posizione del DIP switch 2 su ON.

APRIRE E AGGIORNARE L'ESEMPIO IN VISUAL STUDIO

Dal menu file, selezionate la voce Apri e quindi CMake. Dalla finestra di dialogo che si aprirà andate nella cartella

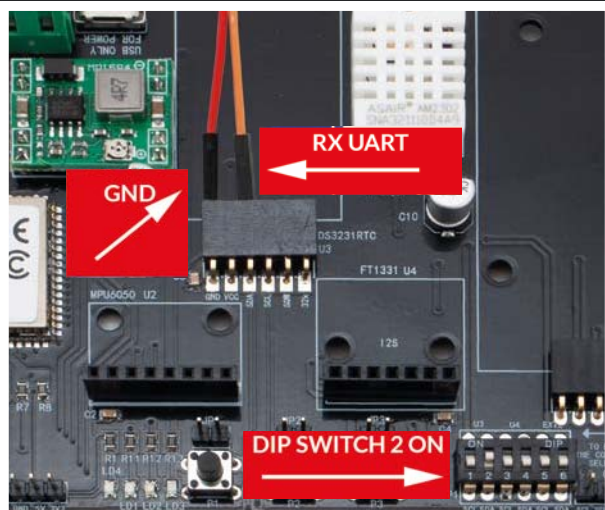


Fig. 2 - Il collegamento RX UART di CN3 e il DIP switch 2 su ON.

a seconda delle tecniche di modulazione usate. Per fare un esempio, negli standard per i modem telefonici a una velocità di 2400 baud, un simbolo di 6 bit corrisponde a una velocità espressa in bit al secondo 14,4 kilobit/s, ovvero il prodotto di 2400 per 6. Di solito, i dati vengono trasmessi in pacchetti di bit che

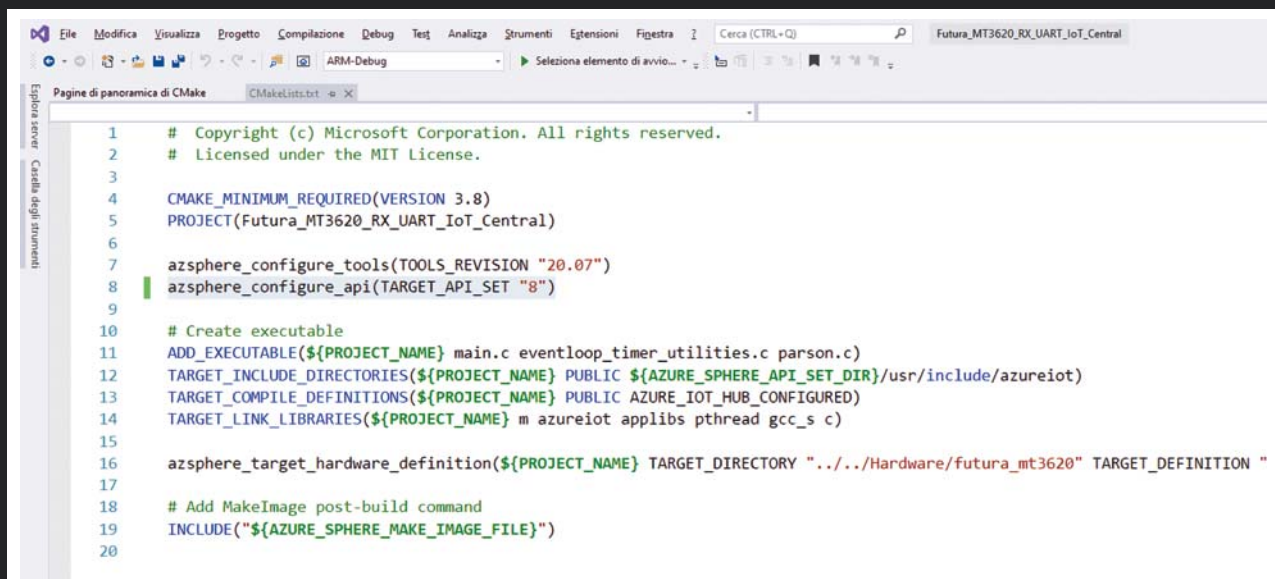


Fig. 3 - La versione API da aggiornare nel file CMakeList.txt.

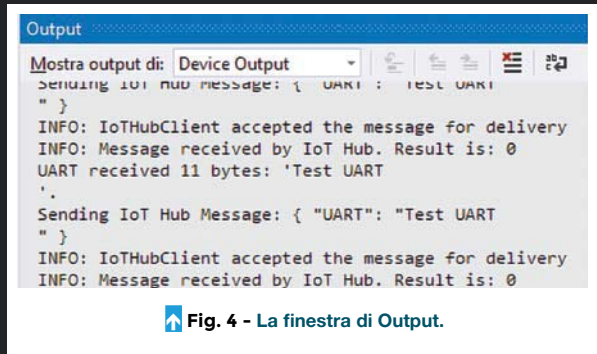


Fig. 4 - La finestra di Output.

Futura_samples e aprite la cartella *Futura_MT3620_RX_UART_IoT_Central*. Selezionate il file *CMakeLists.txt*.

Una volta caricato il progetto, se avete installato l'ultima versione dell'SDK apparirà sicuramente nella finestra Messaggi un avviso di errore di questo tipo:

*CMake Error at C:/Program Files (x86)/Microsoft Azure Sphere SDK/CMakeFiles/AzureSphere.cmake:104 (message):
This app targets API set '6', but this build targets the latest API set '8'.*

To configure the app to target API set '8', update CMakeLists.txt with `azsphere_configure_api(TARGET_API_SET "8")`.

To configure the build to target API set '6', set the `AZURE_SPHERE_TARGET_API_SET` parameter to "6" in `CMakeSettings.json` (if using Visual Studio), `.vscode/settings.json` (if using Visual Studio Code), or via `-DAZURE_SPHERE_TARGET_API_SET="6"` (if using command-line build).

Fra le righe del messaggio di errore si legge chiaramente che: *Questa app è destinata al set di API "6", ma questa build è destinata al set di API più recente "8". Per configurare l'app per il set di API di destinazione "8", aggiornare CMakeLists.txt con `azsphere_configure_api(TARGET_API_SET "8")`.*

Questo succede perché all'epoca in cui sono stati fatti gli esempi la versione API dell'SDK era precedente a quella attuale, pertanto, per aggiornare il progetto alla versione attuale, basta solamente cambiare il numero della versione API corrente. Nel nostro caso (Fig. 3), bisogna cambiare la riga:

```
azsphere_configure_api(TARGET_API_SET "6")
```

con

```
azsphere_configure_api(TARGET_API_SET "8")
```

Una volta salvato il file *CMakeLists.txt* con la versione API aggiornata, apparirà il messaggio confortante che tutto è a posto:

1> Variabili CMake estratte.

1> Intestazioni e file di origine estratti.

1> Modello di codice estratto.

1> Percorsi includes estratti.

1> La generazione CMake è terminata.

Listato 1

```
// Crea un oggetto UART_Config, apre l'UART e configura il gestore di eventi UART
UART_Config uartConfig;
UART_InitConfig(&uartConfig);
uartConfig.baudRate = 115200;
uartConfig.flowControl = UART_FlowControl_None;
uartFd = UART_Open(SAMPLE_ISU2_UART, &uartConfig);
if (uartFd < 0) {
    Log_Debug("ERROR: Could not open UART: %s (%d).\n", strerror(errno), errno);
    return ExitCode_Init_UartOpen;
}
uartEventReg = EventLoop_RegisterIo(eventLoop, uartFd, EventLoop_Input,
UartEventHandler, NULL);
if (uartEventReg == NULL) {
    return ExitCode_Init_RegisterIo;
}
```

Tenete presente che anche in futuro potrà capitare di dover aggiornare le versioni API in base all'SDK Azure Sphere più recente. Vi ricordiamo che potete scaricare l'SDK Azure Sphere aggiornato alla versione più recente direttamente da questo link: <https://aka.ms/AzureSphereSDKDownload/Windows>.

CONFIGURAZIONE UART

Per prima cosa, nel codice di esempio abbiamo creato una configurazione UART (vedi Listato 1) per poter comunicare con un dispositivo seriale con uscita TTL e dati a 115200 baud. In altre parole dovete collegare l'uscita TX di un qualsiasi dispositivo seriale all'ingresso RX della nostra scheda. Ponete attenzione al file descriptor "uartFd" che apre la porta *SAMPLE_ISU2_UART* che viene configurata in questo modo:

```
UART_Config uartConfig;
UART_InitConfig(&uartConfig);
uartConfig.baudRate = 115200;
uartConfig.flowControl = UART_FlowControl_None;
uartFd = UART_Open(SAMPLE_ISU2_UART, &uartConfig);
```

NOTA: tutti i riferimenti delle assegnazioni hardware della nostra scheda sono contenuti nel file *sample_hardware.json* nella cartella *Futura_samples\Hardware\futura_mt3620*.

DISPOSITIVO TX UART

Per la trasmissione seriale potete usare un qualsiasi dispositivo dotato di una porta UART. La soluzione più semplice potrebbe essere una scheda Arduino UNO o simili. Per il collegamento fisico, basta collegare il pin TX di Arduino con il pin RX della scheda e un pin GND di entrambe le schede, come descritto in precedenza. Per il test, è sufficiente scrivere poche righe di codice su uno sketch di Arduino:

```
void setup() {
    Serial.begin(115200);
}

void loop() {
    char * bytesSent = "Test UART";
    Serial.println(bytesSent);
    delay(10000);
}
```

↓ Listato 2

```
static void UartEventHandler(EventLoop* el, int fd, EventLoop_IoEvents events, void* context)
{
    const size_t receiveBufferSize = 256;
    uint8_t receiveBuffer[receiveBufferSize + 1];
    ssize_t bytesRead;

    // Legge i dati UART in arrivo
    bytesRead = read(uartFd, receiveBuffer, receiveBufferSize);
    if (bytesRead < 0) {
        Log_Debug("ERROR: Could not read UART: %s (%d).\n", strerror(errno), errno);
        exitCode = ExitCode_UartEvent_Read;
        return;
    }

    if (bytesRead > 0) {
        // Stampa il log sull'Output e invia telemetria
        receiveBuffer[bytesRead] = 0;
        Log_Debug("UART received %d bytes: '%s'.\n", bytesRead, (char*)receiveBuffer);
        SendTelemetry("UART", (char*)receiveBuffer);
    }
}
```

↓ Listato 3

```
static void SendTelemetry(const unsigned char *key, const unsigned char *value)
{
    static char eventBuffer[100] = {0};
    static const char *EventMsgTemplate = "{ \"%s\": \"%s\" }";
    int len = sprintf(eventBuffer, sizeof(eventBuffer), EventMsgTemplate, key, value);
    if (len < 0)
        return;

    Log_Debug("Sending IoT Hub Message: %s\n", eventBuffer);

    bool isNetworkingReady = false;

    if ((Networking_IsNetworkingReady(&isNetworkingReady) == -1) || !isNetworkingReady) {
        Log_Debug("WARNING: Cannot send IoT Hub Message because network is not up.\n");
        return;
    }

    IOTHUB_MESSAGE_HANDLE messageHandle = IoHubMessage_CreateFromString(eventBuffer);

    if (messageHandle == 0) {
        Log_Debug("WARNING: unable to create a new IoT Hub Message\n");
        return;
    }

    if (IoHubDeviceClient_LL_SendEventAsync(iohubClientHandle, messageHandle, SendMessageCallback,
                                           /*&callback_param*/ 0) != IOTHUB_CLIENT_OK) {
        Log_Debug("WARNING: failed to hand over the message to IoT Hub Client\n");
    } else {
        Log_Debug("INFO: IoT Hub Client accepted the message for delivery\n");
    }

    IoHubMessage_Destroy(messageHandle);
}
```

Se volete fare qualcosa di più utile, potete collegare un qualsiasi sensore alla scheda Arduino e mandare i dati a IoT Central.

In questo modo, doterete Arduino di un'interfaccia di monitoraggio di altissimo livello, nonché una totale sicurezza dei dati.

IL CODICE DI ESEMPIO

Una volta scelta l'opzione GDB Debugger (HL Core) dal menu "Seleziona elemento di avvio", potete mandare in esecuzione il nostro esempio.

Dopo la connessione con IoT Central, appariranno nella fine-

stra di Output a intervalli di 10 secondi i seguenti messaggi (Fig. 4):

UART received 11 bytes: 'Test UART'.

Sending IoT Hub Message: { "UART": "Test UART" }

INFO: IoT Hub Client accepted the message for delivery

INFO: Message received by IoT Hub. Result is: 0

Il messaggio Test UART è quello inviato da Arduino e il messaggio { "UART": "Test UART" } è la coppia *chiave:valore* in formato JSON mandata a IoT Central.

I messaggi INFO indicano che il messaggio è stato accettato e che il risultato di ritorno è 0, quindi valido.

LA FUNZIONE UARTEVENTHANDLER

Questo esempio invia i messaggi a IoT Central solo quando riceve i dati dalla porta UART e non quando si preme un pulsante, come nell'esempio della scorsa puntata.

Tutto il lavoro viene svolto all'interno della funzione "UartEventHandler" che legge i dati in arrivo alla porta seriale (vedi [Listato 2](#)).

Se nella funzione la variabile `bytesRead` riceve un `byte > 0`, allora somma i dati nel buffer `receiveBuffer` e, al termine della ricezione, invia il buffer alla funzione `SendTelemetry`. Come vedremo fra poco, questa funzione serve a inviare la coppia in formato `JSON {key:value}` con l'argomento `key` impostato come "UART" e l'argomento `value` impostato con il contenuto del buffer.

LA FUNZIONE SENDTELEMETRY

Come avete già avuto modo di vedere nelle scorse puntate, la funzione `SendTelemetry` è responsabile dell'invio dei dati di telemetria di ogni dispositivo collegato alla scheda.

In questo caso i dati da inviare a IoT Central sono quelli provenienti dalla porta UART, alla quale potete collegare, come spiegato prima, anche una scheda Arduino.

Nel [Listato 3](#) potete vedere come il buffer identificato con l'array `eventBuffer[100]` possa leggere un messaggio di 100 byte alla volta.

La variabile `EventMsgTemplate = "{ \"%s\\\": \"%s\\\" }"` è proprio quella che converte il buffer in dati JSON secondo il template `{key:value}`.

La funzione `snprintf(eventBuffer, sizeof(eventBuffer), EventMsgTemplate, key, value)` trasferisce il buffer formattandolo secondo il template `{key:value}`. Dopodiché la stringa JSON viene inviata a IoT Central con il gestore `messageHandle = IoTHubMessage_CreateFromString(eventBuffer)`.

IL FILE APP_MANIFEST.JSON

Per l'utilizzo della porta UART bisogna impostare nelle "Capabilities" del file `app_manifest.json` la definizione hardware `SAMPLE_ISU2_UART`. Gli argomenti di `CmdArgs`, `AllowedConnections` e `DeviceAuthentication` sono gli stessi ottenuti dalla procedura spiegata nella scorsa puntata.

Ricordate che dovrete riportare sempre questi argomenti in tutti gli esempi del corso.

```
{
  "SchemaVersion": 1,
  "Name": "AzureIoT",
  "ComponentId": "819255ff-8640-41fd-aea7-f85d34c491d5",
  "EntryPoint": "/bin/app",
  "CmdArgs": [ "0ne001C85E4" ],
  "Capabilities": {
    "AllowedConnections": [
      "global.azure-devices-provisioning.net",
```

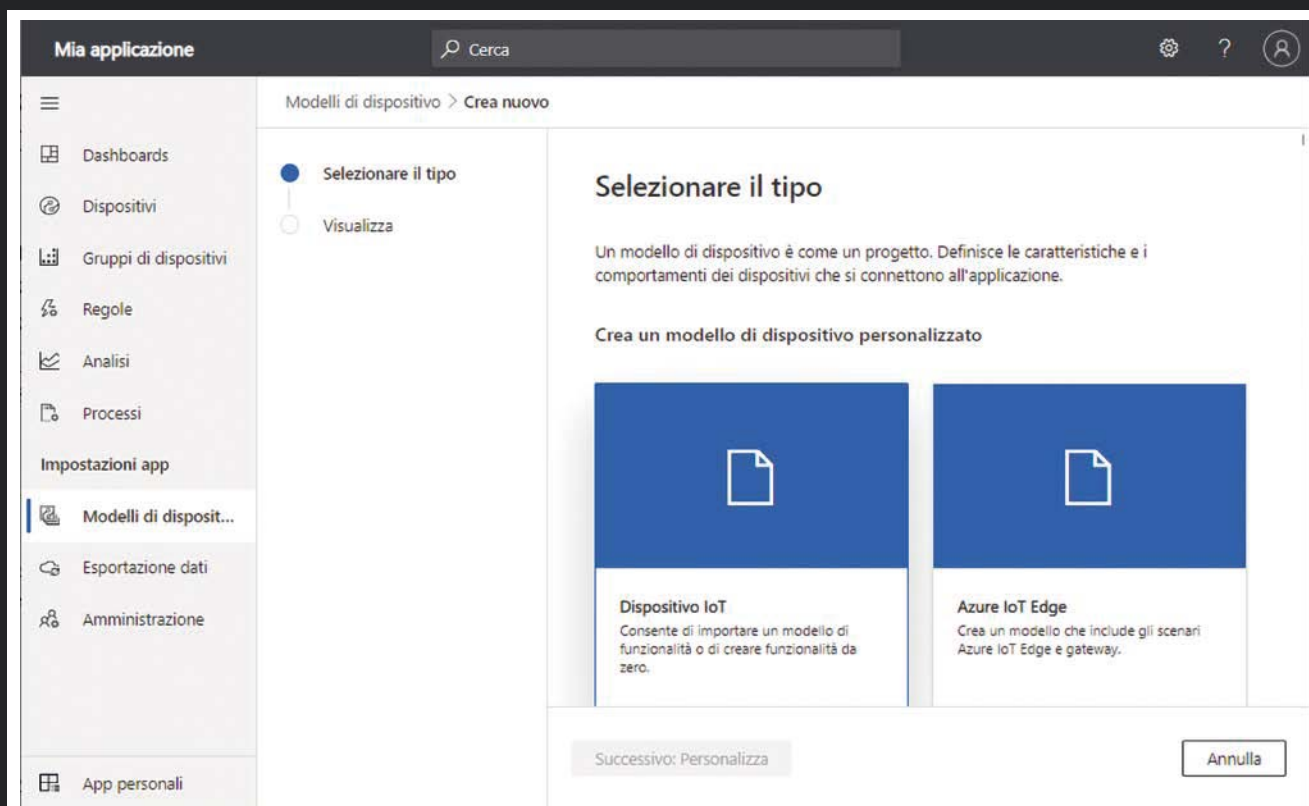


Fig. 5 - La pagina Selezionare il tipo.

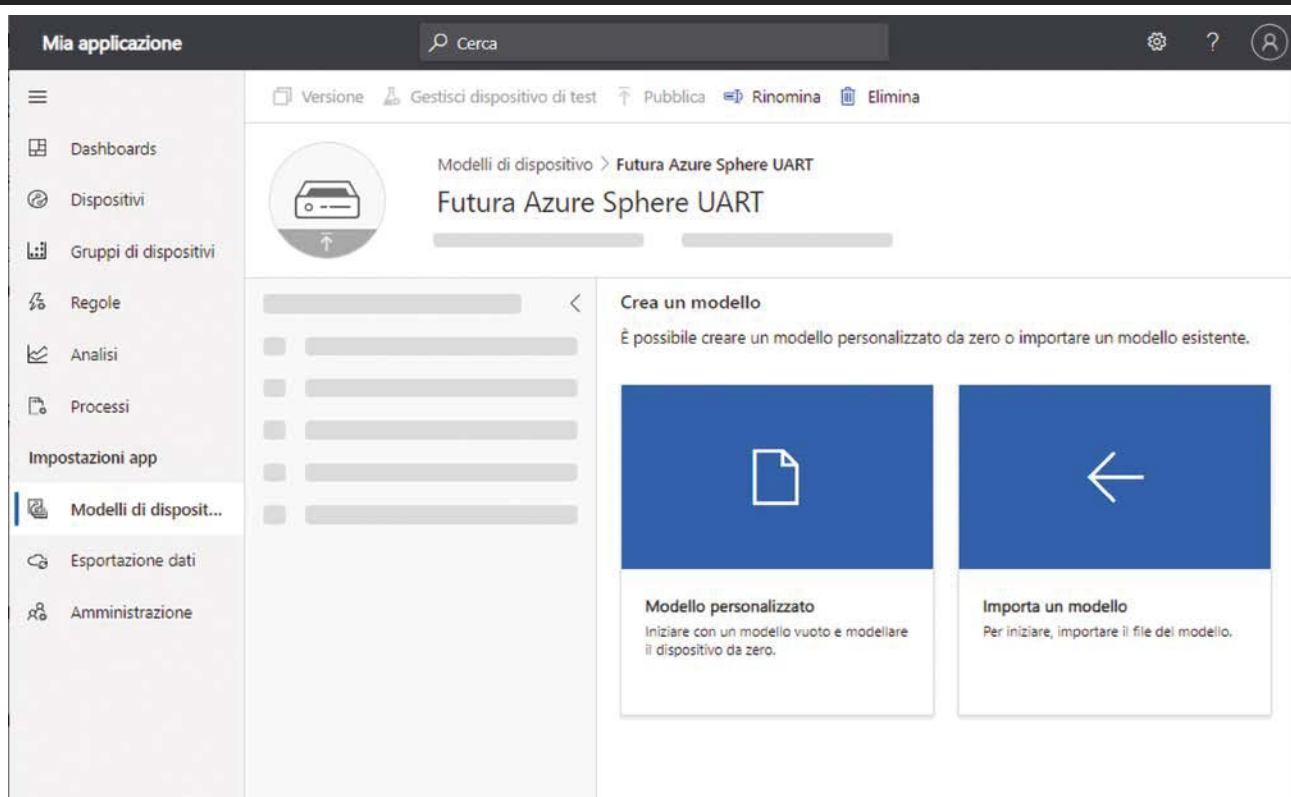


Fig. 6 - La pagina con il modello del dispositivo.

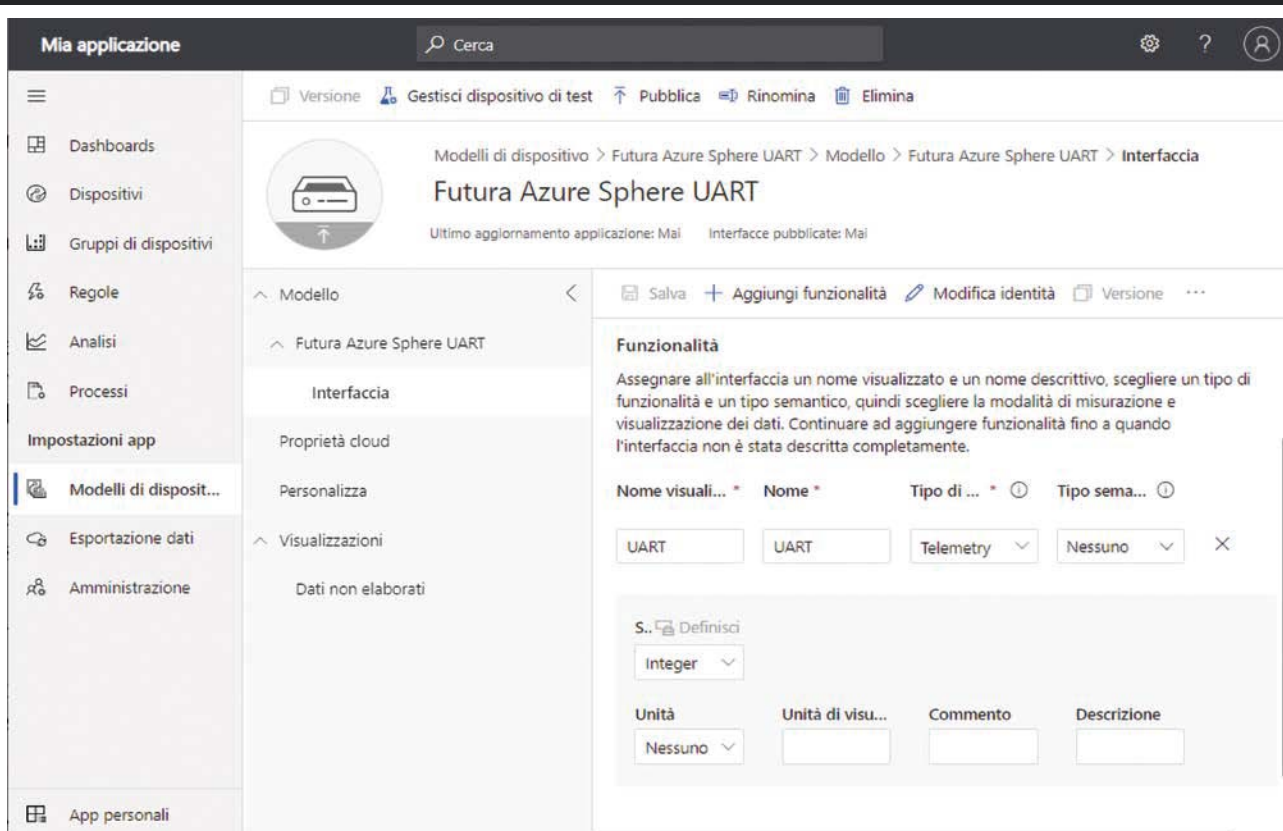


Fig. 7 - La finestra con Interfaccia e Funzionalità.

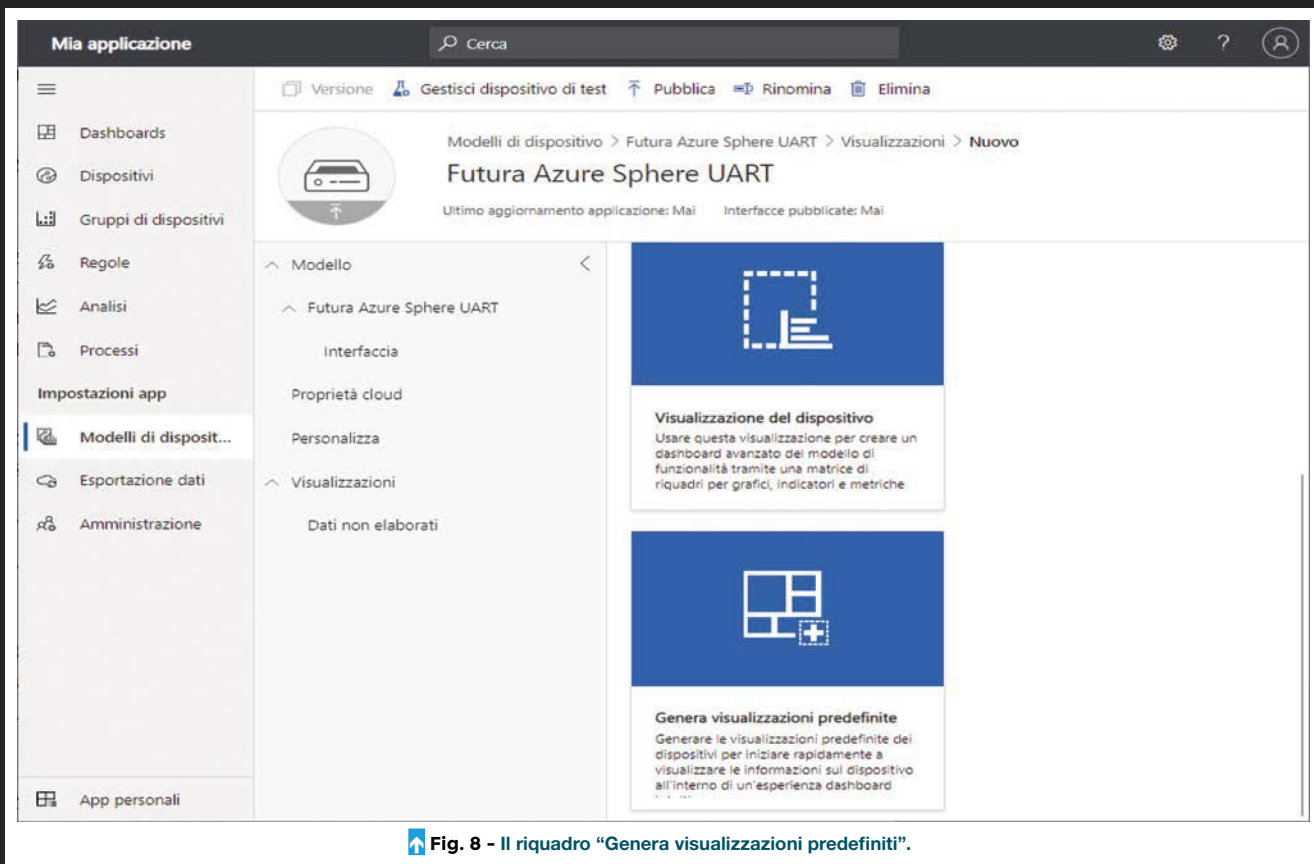


Fig. 8 - Il riquadro "Genera visualizzazioni predefiniti".

```

    "iotc-f18c6479-0fd2-4197-80f0-337ba9198b77.azure-devices.net"
  ],
  "Uart": [ "$SAMPLE_ISU2_UART" ],
  "DeviceAuthentication": "df7ba4db-4a32-4bfd-b401-59f2846cc320"
},
"ApplicationType": "Default"
}

```

VISUALIZZARE GLI EVENTI UART SU IOT CENTRAL

Per semplificare le cose abbiamo preparato un modello di dispositivo già pronto all'uso. Ovviamente nessuno vieta di seguire la procedura descritta nella precedente puntata, ma in questo modo, imparate una cosa nuova, ovvero come importare modelli di dispositivi preesistenti. Potete trovare il modello già pronto all'uso nella directory degli esempi *Futura_samples*. Una volta aperta la vostra applicazione su IoT Central, ecco come fare.

1. Selezionate "Modelli di dispositivo"
2. Create un nuovo modello facendo clic su "+ Nuovo".
3. Apparirà la finestra "Selezionare il tipo".
4. Selezionate "Dispositivo "IoT" (Fig. 5).
5. Fate clic su "Successivo: Personalizza".
6. Apparirà la finestra "Personalizza".
7. Compilate il campo Nome, per esempio, con il testo "Futura Azure Sphere UART".
8. Fate clic su "Avanti: Rivedi".

9. Apparirà la finestra "Visualizza".

10. Fate clic su "Crea".

11. Verrà creato il modello "Futura Azure Sphere UART" (Fig. 6).

12. Fate clic su "Importa un modello".

13. Si aprirà una finestra di dialogo.

14. Cercate nella cartella Futura_samples il file: "Futura Azure Sphere UART.json" e apritelo.

15. Si aprirà la pagina "Interfaccia".

16. Fate clic su "Interfaccia" del pannello a sinistra.

17. Si aprirà la funzionalità di telemetria UART importata dal file e pronta all'uso (Fig. 7). Notare che lo schema è impostato su "Integer" per visualizzare dati con numeri interi, ma potete cambiarlo con qualsiasi altro schema come, per esempio, string, date, double e così via.

AGGIUNGERE LA VISUALIZZAZIONE DEI DATI UART

Nell'ottica di semplificare le cose, abbiamo scelto una comoda modalità di visualizzazione predefinita, anche se, ovviamente, nessuno vieta di seguire la procedura che abbiamo spiegato nella scorsa puntata. Rimanendo nella finestra del modello del dispositivo, selezionate "Visualizzazioni" e quindi "Visualizzazione del dispositivo". Si aprirà la pagina "Selezionare per aggiungere una nuova visualizzazione". A questo punto, selezionate "Genera visualizzazioni predefinite" (Fig. 8).

Si aprirà la pagina "Selezionare le visualizzazioni applicabili

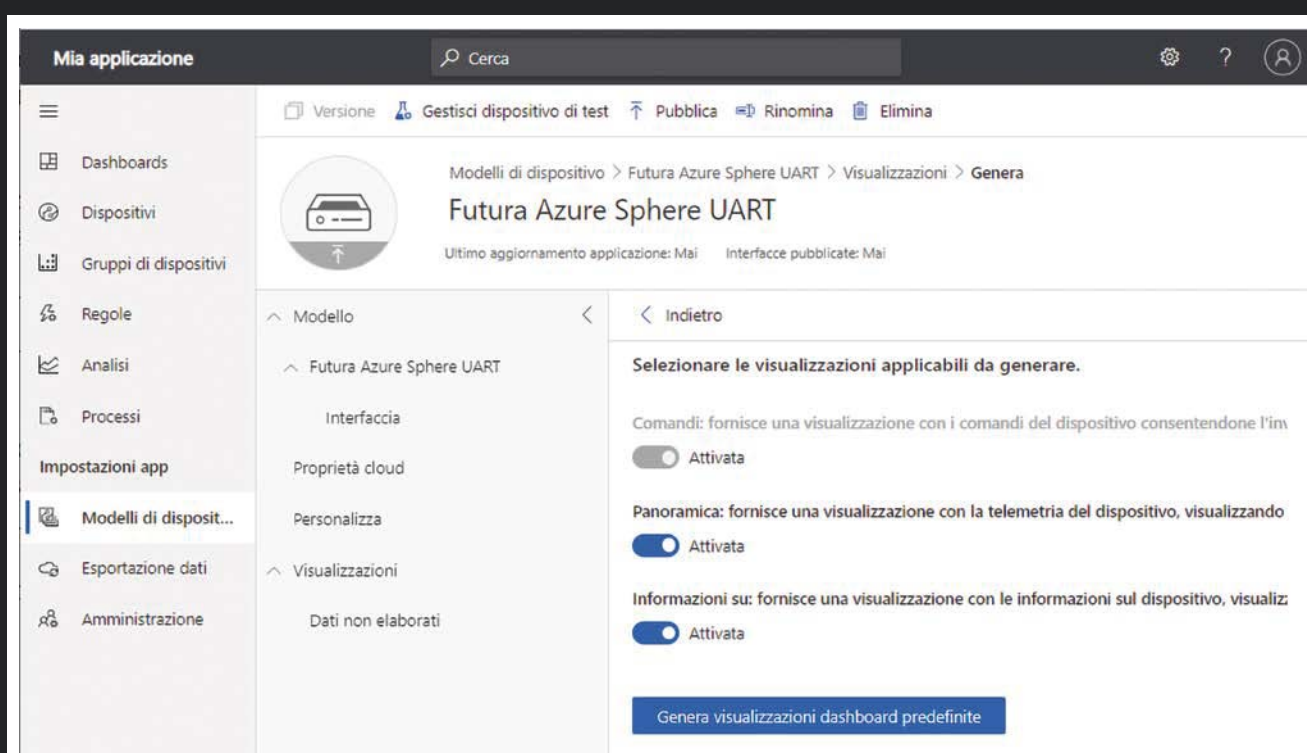


Fig. 9 - La pagina "Selezionare le visualizzazioni applicabili da generare".

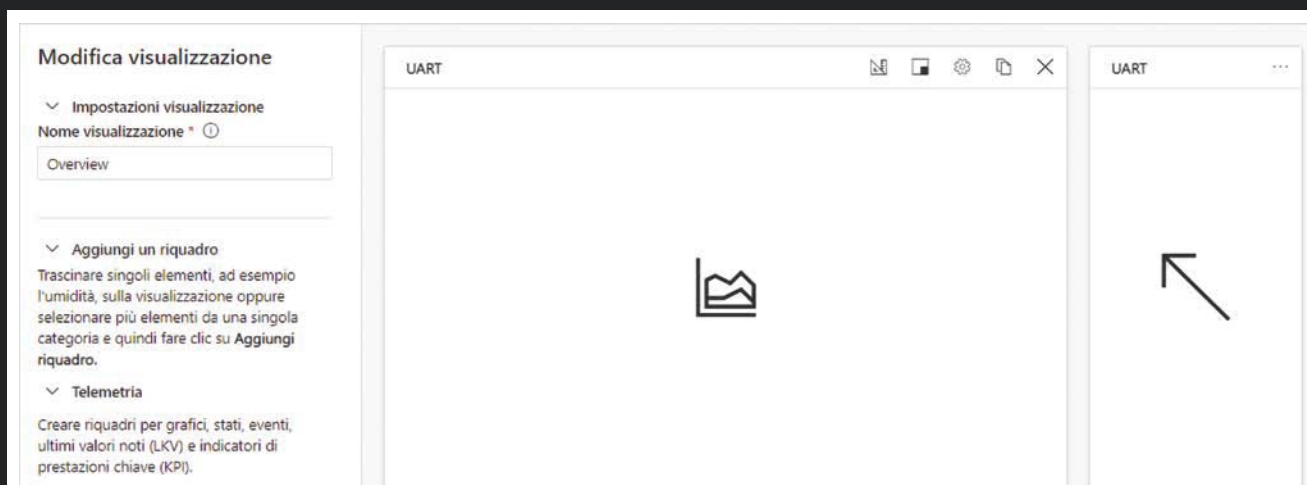


Fig. 10 - La pagina del modello "Modifica visualizzazione".



Fig. 11 - La finestra "Tutti i dispositivi" con "Esegui migrazione".



Fig. 12- La finestra "Esegui migrazione".

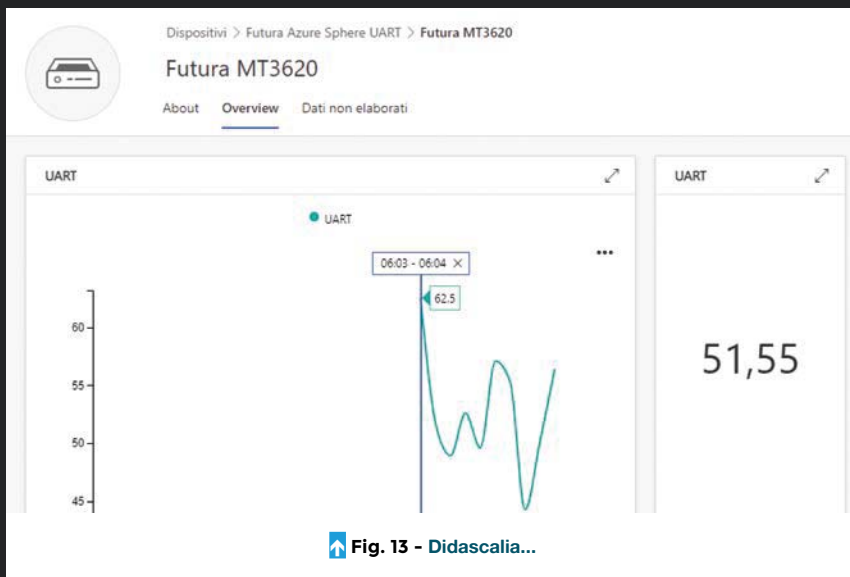


Fig. 13 - Didascalia...

da generare". Le opzioni da attivare sono:

- **Panoramica:** fornisce una visualizzazione con la telemetria del dispositivo, visualizzando grafici e metriche.
- **Informazioni su:** fornisce una visualizzazione con le informazioni sul dispositivo, visualizzandone le proprietà.

Lasciate selezionate entrambe e fate clic su "Genera visualizzazioni dashboard predefinite" (Fig. 9). In questo modo verranno create le visualizzazioni standard per tutte le funzionalità. Fate clic su "Overview" sotto "Visualizzazioni". Si aprirà una pagina con il riquadro UART in un grafico a linee e un riquadro UART di tipo KPI (Fig. 10). Siccome le visualizzazioni sono in bozza le potete modificare a piacere. Al termine delle eventuali modifiche o se accettate la visualizzazione standard proposta, premete su "Salva" e pubblicate il modello facendo clic su "Pubblica".

MIGRAZIONE

Se avete già creato un modello precedente o ne avete creato uno nuovo e lo volete utilizzare con il vostro dispositivo, potete eseguire la cosiddetta migrazione del dispositivo al nuovo modello. Questo vale anche per il modello che avete appena creato e pubblicato come Futura Azure Sphere UART. Dalla pagina "Tutti i dispositivi", selezionare il dispositivo che volete usare e fate clic su "Esegui migrazione" (Fig. 11). Si aprirà una finestra simile alla Fig. 12 in cui scegliere il nome del modello di dispositivo. Selezionate "Futura Azure Sphere UART" e fate clic su Esegui la migrazione. A questo punto, il dispositivo userà le funzionalità e le visualizzazioni del nuovo modello.

Ad ogni ricezione di un messaggio UART i dati di telemetria verranno inviati a IoT Central e nella finestra "Overview" del dispositivo si vedrà qualcosa di simile alla Fig. 13. Per comodità, abbiamo creato un semplice sketch in Arduino che invia a intervalli di 5 secondi dati casuali da 10 a 100 per simulare un possibile dispositivo collegato alla scheda Arduino.

```
void setup() {
  Serial.begin(115200);
}

void loop() {
  int bytesSent = random(10,100);
  Serial.println(bytesSent);
  delay(5000);
}
```

CONCLUSIONI

Per oggi ci fermiamo qui. Con la speranza di essere stati sufficientemente chiari, vi diamo appuntamento alla prossima puntata, nella quale approfondiremo altri esempi di applicazioni. □



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>



Siemens supporta i bus zero-emission in Repubblica Ceca

Siemens Smart Infrastructure fornirà una soluzione di ricarica per la flotta di 24 autobus elettrici Solaris ad Ostrava, nella Repubblica Ceca. Basata su quattro stazioni di ricarica Sicharge UC, caricatori ad alta potenza da 400 kW, l'infrastruttura di trasporto urbano conterà su 28 caricabatterie mobili, infrastrutture elettriche e software di automazione energetica. I bus elettrici verranno ricaricati dal caricabatterie rapido Sicharge UC grazie a un pantografo che si eleverà in postazione, potendo contare su un'uscita CC massima di 500 A fino a 1000 V; le unità Simatic S7 ne controlleranno e ottimizzeranno il funzionamento. Tutti i punti di ricarica saranno dotati di un sistema di telecamere di sorveglianza e funzioneranno in modalità self-service. Verrà inoltre implementata un'applicazione di controllo della microgrid Siemens per misurare e trasmettere i dati di ricarica ai sistemi dell'operatore di trasporto per la gestione dell'energia e la pianificazione delle operazioni. Ciò consentirà il controllo remoto e automatico del processo di ricarica in base agli orari degli autobus e sarà possibile modificare la potenza di ricarica in base all'energia disponibile. Siemens fornirà anche un trasformatore di media tensione a secco da 800 kVA e due da 1.250 kVA per collegare i punti di ricarica alla rete elettrica urbana, nonché quadri di media tensione, incluso un GIS Siemens 8DJH. 28 caricatori mobili, ciascuno con una

potenza di 22,5 kW, verranno utilizzati per la ricarica notturna o in un garage di servizio, bilanciando così la potenza e mantenendo il funzionamento della flotta di eBus. Ogni stazione di ricarica può alimentare in sequenza fino a cinque veicoli tramite distributori di ricarica; oltre ai cavi di ricarica (sistema plug-in) possono essere utilizzati pantografi top-down (OppCharge) o calotte di collegamento per pantografi bottom-up. Una combinazione flessibile di ricarica tramite collegamento via cavo e pantografo, ad esempio, ottimizza i costi e lo spazio di installazione. Autobus elettrici per Ostrava Solaris Bus & Coach fornirà una flotta di 24 autobus elettrici all'autorità di transito di Ostrava. I modelli Solaris Urbino 12 selezionati avranno batterie Solaris High Power con una capacità totale di 91,4 kWh. Questa capacità rappresenta una scelta ottimale per frequenti ricariche brevi, tipiche dei percorsi in cui verranno impiegati i veicoli. Gli autobus elettrici verranno forniti nella normale lunghezza di 12 metri; ciascuno può trasportare fino a 80 passeggeri (seduti e in piedi). Grazie alla ricarica graduale presso le stazioni terminali, l'autonomia giornaliera dei nuovi autobus elettrici arriva fino a 400 chilometri. Con la tecnologia di ricarica ad alte prestazioni utilizzata, la ricarica durante il giorno non richiederà più di 10 minuti per autobus.

www.siemens.com



Pronto il sistema di lancio della navicella Orion

Boeing sta consegnando alla NASA lo stadio del nucleo criogenico del razzo Space Launch System (SLS) in preparazione del lancio della missione Artemis I, la prima missione lunare dopo quasi 50 anni per l'ente spaziale statunitense, finalizzata a testare la capsula Orion.

Il blocco verrà portato al Kennedy Space Center in Florida, per l'integrazione con la navicella spaziale Orion, lo stadio superiore e propulsori a razzo solido in preparazione per il lancio. SLS lancerà la missione Artemis I della NASA, che invierà in un primo momento una capsula Orion per astronauti ma senza equipaggio: la missione prevede un giro intorno alla luna e relativo ritorno. Se questa missione di prova avrà l'esito atteso, sarà seguita dal lancio della Artemis II, ossia dal primo fly-by lunare con equipaggio del programma Artemis.

Boeing è il principale appaltatore della NASA per il nucleo di SLS, gli stadi superiori e l'avionica. L'azienda sta unendo gli elementi principali per il palco principale di Artemis II ora presso la Michoud Assembly Facility della NASA a New Orleans. Boeing sta anche lavorando su capacità evolutive per il sistema missilistico come l'Exploration Upper Stage (EUS), che sta entrando in produzione da Michoud. Il più potente SLS con EUS sarà in grado di portare tre volte più massa nello spazio profondo rispetto a qualsiasi altro razzo, consentendo sostenibilità per missioni di esplorazione, scienza e sicurezza con e senza equipaggio.

La capacità dello Space Launch System di trasportare carichi utili consistenti e multipli in un unico lancio riduce la complessità e il rischio della missione aumentando la sicurezza, l'affidabilità e la probabilità di successo della missione stessa.

www.boeing.com

Elettroliti selettivi per batterie dalla tecnica farmacologica

Le membrane che consentono a determinate molecole di passare sono alla base delle batterie e delle celle a combustibile, dove consentono il trasporto degli ioni, necessari per mantenere il flusso di elettricità. Le membrane più selettive soffrono di una bassa permeabilità nei confronti degli ioni, il che limita la potenza e l'efficienza energetica della batteria. Per superare i compromessi tra selettività della membrana e permeabilità, bisogna aumentare la solubilità e la mobilità degli ioni all'interno della membrana, consentendo a un numero maggiore di essi di transitare nell'elettrolito. In quest'ottica i ricercatori hanno progettato una membrana polimerica con gabbie molecolari incorporate nei suoi pori che trattengono gli ioni caricati positivamente da un sale di litio. Queste "gabbie di solvatazione", comprendono molecole che insieme agiscono come un solvente che circonda ogni ione di litio. Il team, guidato dai ricercatori del Lawrence Berkeley National Laboratory (Berkeley Lab) del DOE ha scoperto che le gabbie di solvatazione aumentano il flusso di ioni di litio attraverso la membrana di un ordine di grandezza rispetto alle membrane standard. La tecnica deriva dal collaudato processo di scoperta dei farmaci, dove si costruiscono molecole con strutture diverse per individuare una che si lega a una molecola biologica di interesse. La solvatazione che si verifica quando gli ioni di litio si associano alle gabbie nei pori della membrana fa sì che essi si concentrino più che nelle membrane standard senza gabbie di solvatazione, con l'effetto che il litio si deposita in modo uniforme sull'elettrodo, tanto che la batteria si carica e si scarica più rapidamente ed efficacemente.

newscenter.lbl.gov



Il motore a riluttanza rivoluzionerà la trazione elettrica?

Turntide Technologies, startup nota per la sua rivisitazione del motore elettrico utilizzato nei sistemi HVAC, punta ora alla trazione dei veicoli elettrici commerciali. Turntide afferma che l'efficienza intrinseca del suo motore, SRM fornisce un risparmio energetico medio del 64 per cento rispetto alla trazione con motori tradizionali.

Il motore a riluttanza commutata (SRM=Switched Reluctance Motor) è stato inventato nel lontano 1838. Mentre un motore convenzionale utilizza la forza del campo magnetico per mettere una parte in movimento, qui i poli del rotore tendono ad allinearsi con



i poli dello statore. Commutando la corrente in ciascuna serie di avvolgimenti, i poli del rotore vengono attratti verso i poli dello statore, cui si avvicinano. La riluttanza è la resistenza che il flusso magnetico incontra: serve ma bisogna superarla. Per ora il motore a riluttanza ha trovato un utilizzo marginale perché si è rivelato difficile da controllare e affetto da rumore e vibrazioni. Ma aumentare il numero di poli del rotore può risolvere i problemi perché dà più spazio sullo statore per gli avvolgimenti in rame.

turntide.com/technology

Il chip per AI grande quanto un wafer

Cerebras Systems annuncia il Wafer Scale Engine 2 (WSE-2), il chip che occupa un intero wafer, basato sul processo TSMC a 7 nm. Nel calcolo AI, l'addestramento delle reti neurali richiede molto tempo perché i dati devono spostarsi dal processore alla DRAM esterna e viceversa. La soluzione è integrare in un chip i core del processore AI e la RAM. Il WSE-2 è grande come un intero wafer di silicio (46.225 mm²); rispetto a un processore AI come l'A100 Nvidia, è 56.000 volte più grande e integra 2,6 trilioni di transistor contro 54,2 miliardi, 850mila core contro 7.344, una memoria on-chip di 40 gigabyte contro 40 megabyte, una memory bandwidth di 20 petabyte/s contro 155 GB/s e una Fabric bandwidth di 220 petabit/s contro 600 gigabyte/s. Nel wafer, il WSE-2

è diviso in una griglia di rettangoli 7x12. Il sistema informatico CS-2 che ospita WSE 2 occupa un terzo di un rack standard 19", consuma circa 20 kW, ha un sistema di raffreddamento a liquido a circuito chiuso e ventole di raffreddamento. Il chip assorbe circa 20 kA alimentato da un milione di connessioni in rame fornite da un circuito stampato in fibra di vetro in cima al wafer. Il software consente di scrivere i modelli di apprendimento automatico utilizzando framework standard come PyTorch e TensorFlow. Quindi il suo compilatore dedica regioni a diversi livelli della rete neurale specificata, ottimizzando luogo e percorso affinché tutti i livelli finiscano il loro lavoro all'incirca alla stessa velocità.

cerebras.net



Virgin Galactic svela i dettagli di SpaceShip III

Dopo il rinvio del test di volo sub-orbitale con passeggeri annuncia- to nello scorso mese di febbraio, Virgin Galactic si prepara ai nuovi voli di prova di VSS Unity, puntando a risolvere i problemi di interferenze radioelettriche che causarono lo spegnimento del razzo. Nel frattempo annuncia la sua terza serie di navicelle: la VSS Imagine. L'astronave è l'emblema dell'innovazione di Virgin Galactic nel design e nell'esperienza di volo degli astronauti, ma anche nell'efficienza, nell'ottica di un'attività a lungo termine. VSS Imagine inizierà i test a terra, con voli in pianata pianificati per questa estate dallo Spaceport America nel New Mexico. L'innovativo design della livrea, rifinita con un materiale simile a uno specchio, riflette la luce dell'ambiente circostante, cambiando continuamente colore e aspetto mentre viaggia dalla terra allo Spazio. Oltre a fornire protezione termica, questo materiale attrae naturalmente l'occhio, riflettendo la nostra innata attrazione per il volo spaziale. Sfruttando un design modulare, i veicoli della classe SpaceShip III sono costruiti per consentire prestazioni migliorate in termini di accesso per la manutenzione e velocità di volo. Questa terza generazione di astronavi getterà le basi per la progettazione e la produzione di veicoli futuri. Quando VSS Imagine inizierà i test a terra, la produzione proseguirà con la gemella VSS Inspire, il secondo veicolo della classe SpaceShip III della flotta Virgin Galactic, pietra miliare nello sforzo di Virgin Galactic per arrivare a breve a effettuare 400 voli turistici all'anno, per spaziorporto.

www.virgingalactic.com

OSCILLOSCOPIO 2 CANALI 70 MHZ

- ✓ Frequenza di campionamento fino a 1 GS/s
- ✓ Funzione FFT integrata
- ✓ Software per analisi in tempo reale con il PC
- ✓ Porta USB Host

cod. DS072

€ 319,00

Disponibile anche
nella versione a 200MHz

cod. DS0202

€ 459,00



OSCILLOSCOPIO 2 CANALI 100 MHZ CON GENERATORE DI FORME D'ONDA



cod. DS0FG102

€ 429,00

- ✓ Frequenza di campionamento fino a 1 GS/s
- ✓ Funzione FFT integrata
- ✓ Software per analisi e controllo in tempo reale con il PC
- ✓ Porta USB Host / USB Device

Disponibile anche
nella versione a 200MHz

cod. DS0FG202

€ 549,00

GENERATORE DI FUNZIONI E FORME D'ONDA ARBITRARIE 2 CANALI 30 MHZ

- ✓ Frequenza onda DC 30 MHz
- ✓ 2 canali
- ✓ Frequenza di campionamento 250 MSa/s
- ✓ Risoluzione verticale 16 Bit

cod. HDG32B

€ 439,00



Disponibile anche
nella versione a 60MHz

cod. HDG62B

€ 510,00

OSCILLOSCOPIO PORTATILE 2 CANALI - 40 MHZ CON MULTIMETRO

Dispone di ampio display LCD TFT a colori da 2,8. La larghezza di banda è di 40 MHz, 250 MSa/s di velocità di campionamento in tempo reale a canale singolo. Il multimetro fornisce misurazioni di tensione continua e alternata, corrente continua e alternata, resistenze, capacità, test diodi e continuità. È interfacciabile a PC tramite software scaricabile gratuitamente. Supporta Windows 7 / 8 / 10 a 32 e 64 bit.



cod. MH02C42

€ 178,00

Disponibile anche nella versione
a 2 canali 70 MHz con generatore
di forme d'onda e multimetro

cod. MH02D72GEN

€ 268,00

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)

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

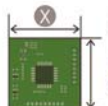
1 Layer

2 Layers

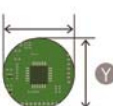
4 Layers

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.

www.futurashop.it

Futura Group srl
 Via Adige, 11 • 21013 Gallarate (VA)
 Tel. 0331/799775

**FUTURA
 ELETTRONICA®**