Raspberry Pi
&
Arduino Create

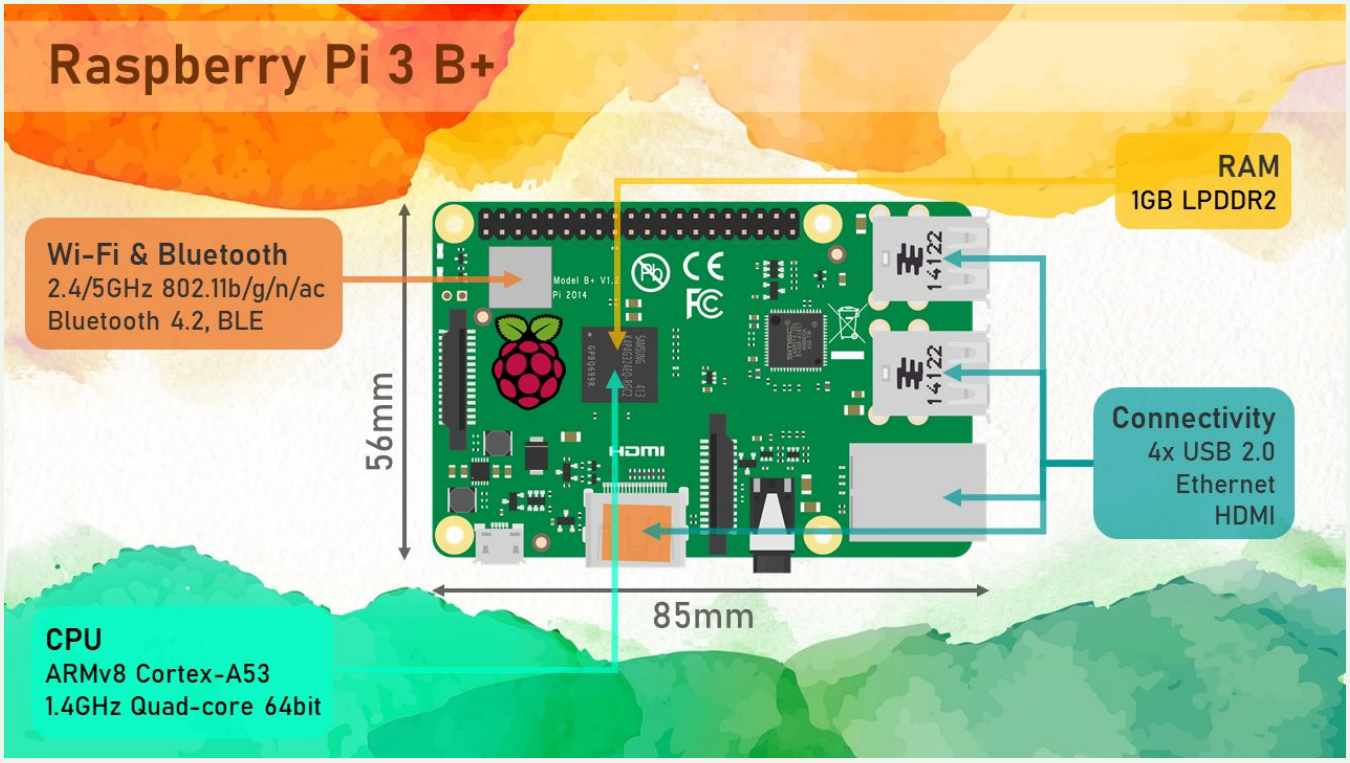
Andrea Scipioni
Davide D'Ascenzo

Indice

Introduzione al Raspberry

Il **Raspberry Pi** è una serie di SBCs (Single-Board Computer) dotati di processore Broadcom SoC (System-on-a-Chip). L'unione di questi due elementi ha creato schede ad alte prestazioni, con dimensioni minime e prezzo contenuto. È questo, infatti, il punto di forza di tutto il brand.
Quelle che analizzeremo qui, però, sono solo due delle boards attualmente disponibili sul mercato, per poi compararle con le più famose piattaforme utilizzate nell'ambito embedded. L'obiettivo è quello di offrire al lettore una panoramica sulle schede Pi che normalmente non sono una prima scelta nei sistemi embedded, andando a ricercare le ragioni, se esistono, di tale decisione ed esaminando i punti di forza e di debolezza di questi dispositivi.

Il primo modello preso in considerazione è il **Raspberry Pi 3 B+**, la cui struttura è visibile in basso:

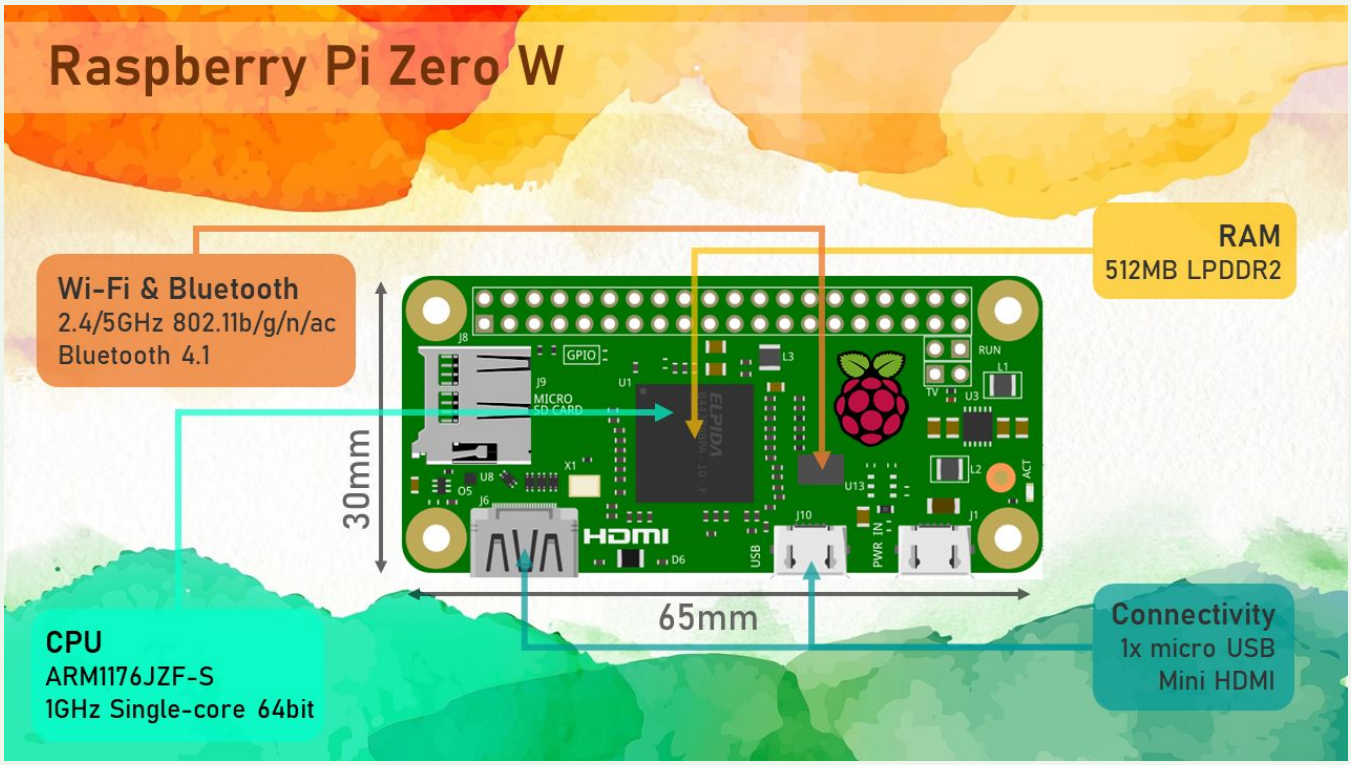


This file was derived from https://commons.m.wikimedia.org/wiki/File:Raspberry_Pi_B%2B_illustration.svg and it is licensed under the Creative Commons Attribution-Share Alike 3.0 Unported license

Il processore da 1.4GHz Quad-core (con GPU integrata) e la memoria RAM da 1GB rendono questo dispositivo un vero e proprio computer. È infatti possibile (ed è il modo classico in cui viene utilizzato) installare una distribuzione Linux adattata per questa board dagli sviluppatori stessi. Le performance non sono paragonabili, come è intuibile, a quelle di un personal computer; tuttavia la piattaforma non è stata

sviluppata con questo intento, e il prezzo (di cui discuteremo in seguito) giustifica pienamente le prestazioni.
La connettività è il secondo punto di forza della board, avendo la possibilità di gestire I/O "strutturati", via cavo (come possono essere l'USB, l'Ethernet, l'HDMI) e wireless (come il Wi-Fi e il Bluetooth), e più a basso livello (GPIO, General Purpose Input/Output). Questa combinazione rende la scheda perfetta per essere utilizzata come interfaccia tra un dispositivo embedded e un computer per l'analisi dei dati.

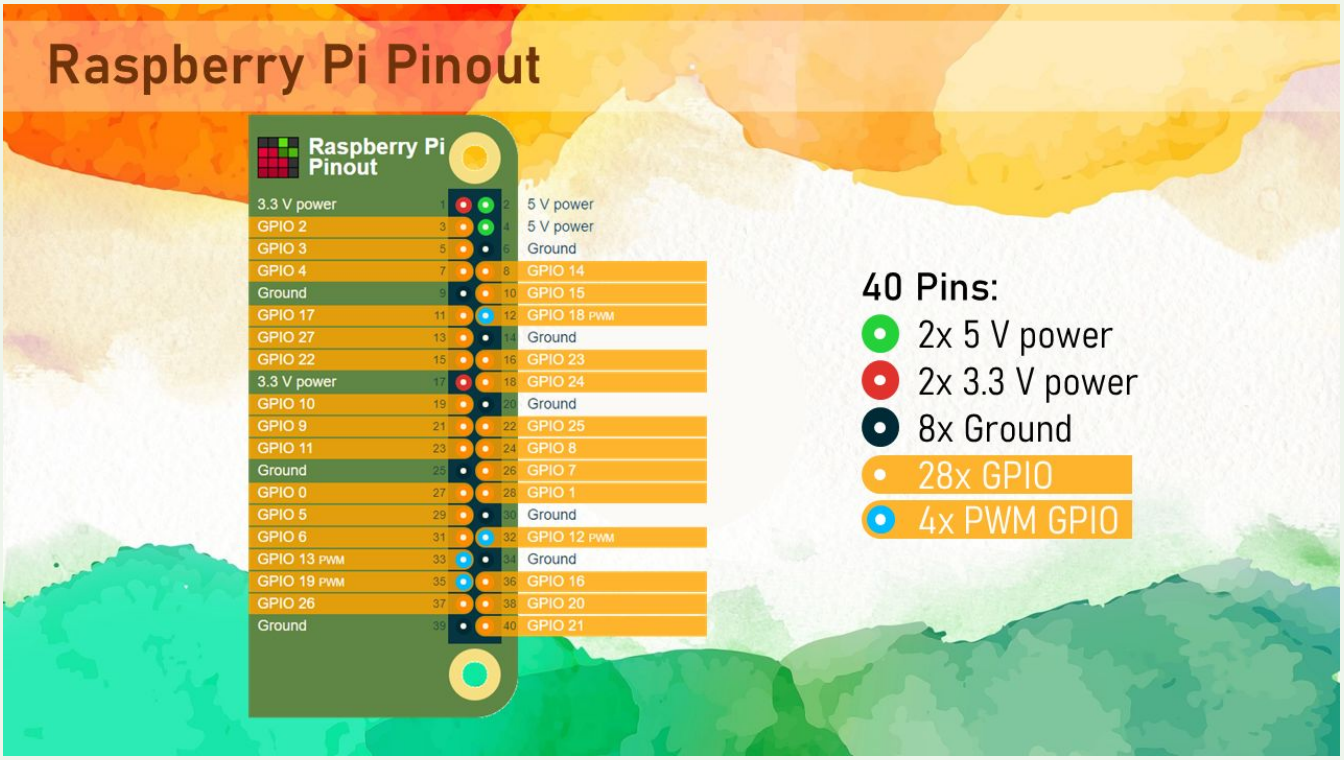
Una variante di questa scheda, dalle dimensioni e costo ancora più contenuti, è il **Raspberry Pi Zero W**, visibile sotto:



This image was derived from Fritzing Raspberry Pi Zero image

Le caratteristiche non differiscono poi molto dal modello precedente: è presente un processore da 1Ghz Single-core e una memoria RAM da 512MB. La connettività rimane pressoché la stessa, con il passaggio di alcune porte alla versione mini/micro (la dicitura "**W**" nel nome della scheda indica che anch'essa presenta un'interfaccia Wi-Fi). Il costo e le sue dimensioni estremamente ridotti la rendono perfetta per l'elaborazione di dati provenienti da sensori ed altri dispositivi embedded. È infatti possibile creare un cluster di queste mini-schede per adempiere a questo scopo.

Tuttavia ciò che si vuole esplorare da vicino è l'interfaccia che permette a queste due boards di comunicare con dispositivi a più basso livello: il GPIO.
Entrambe le schede (e in generale tutte o quasi le schede Pi), infatti, sono dotate di 40 identici pins di input/output, visibili di seguito:



This image was derived from pinout.xyz

Da notare i 4 pins PWM, che ci permettono di simulare delle tensioni quasi "analogiche" in una scheda con soli pins digitali. Tuttavia sulle boards sono presenti solo due canali PWM, questo significa che è possibile generare solo 2 segnali diversi alla volta.

Arduino Create

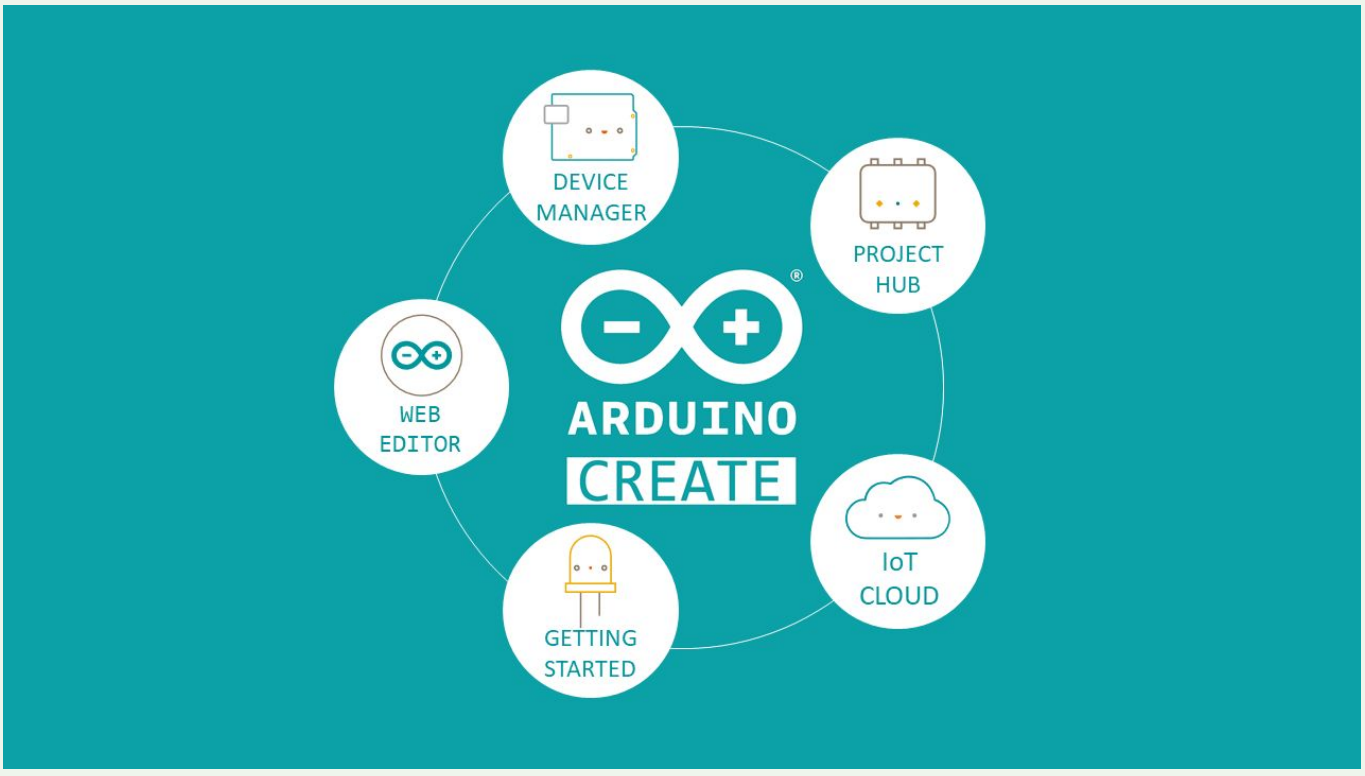
Un aspetto importante nell'analisi delle schede programmabili nel panorama embedded è l'ambiente di sviluppo. In generale, infatti, lo sviluppo di software per una scheda programmabile non avviene direttamente sulla scheda (perché tipicamente non è neanche possibile avviare la scheda senza software caricato), quanto più su un dispositivo ausiliario (un personal computer) che permette di configurarla e di interfacciarsi con essa.

Con il Raspberry Pi questo non è del tutto vero. Infatti, come abbiamo visto sopra, questa serie di dispositivi sono dei veri e propri computer con sistema operativo.

Tuttavia, dato che tale sistema non nasce con l'obiettivo principale di operare come sistema embedded, è opportuno utilizzare un ambiente già maturo che ci consenta di programmare efficacemente le più comuni operazioni.

Nasce quindi un'integrazione di **Arduino Create**[1] (piattaforma per lo sviluppo, principalmente IoT, su Arduino; da qui in poi chiamato anche Create) che permette di creare sketches (così vengono chiamati i programmi del mondo Arduino) su processori ARM e, più in particolare, su Raspberry Pi.

Le componenti fondamenti di **Arduino Create** sono cinque:

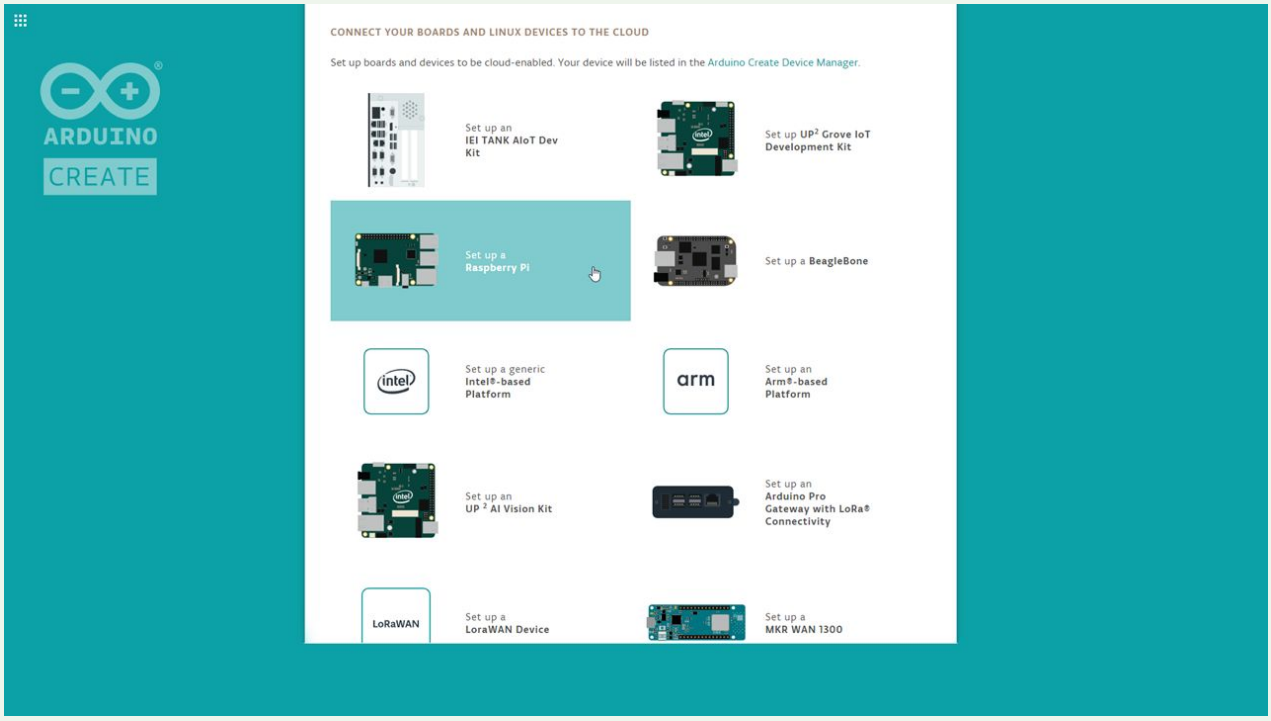


This image was derived from arduino.cc

[1] https://create.arduino.cc/

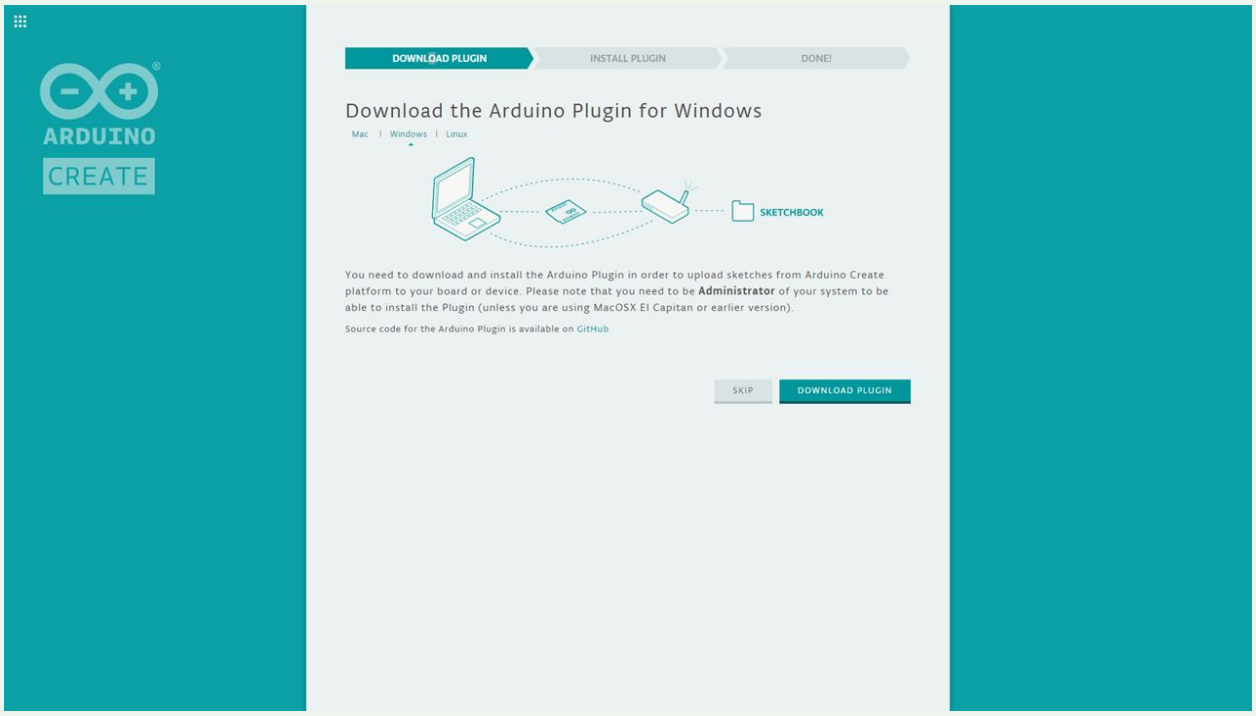
- **GETTING STARTED**

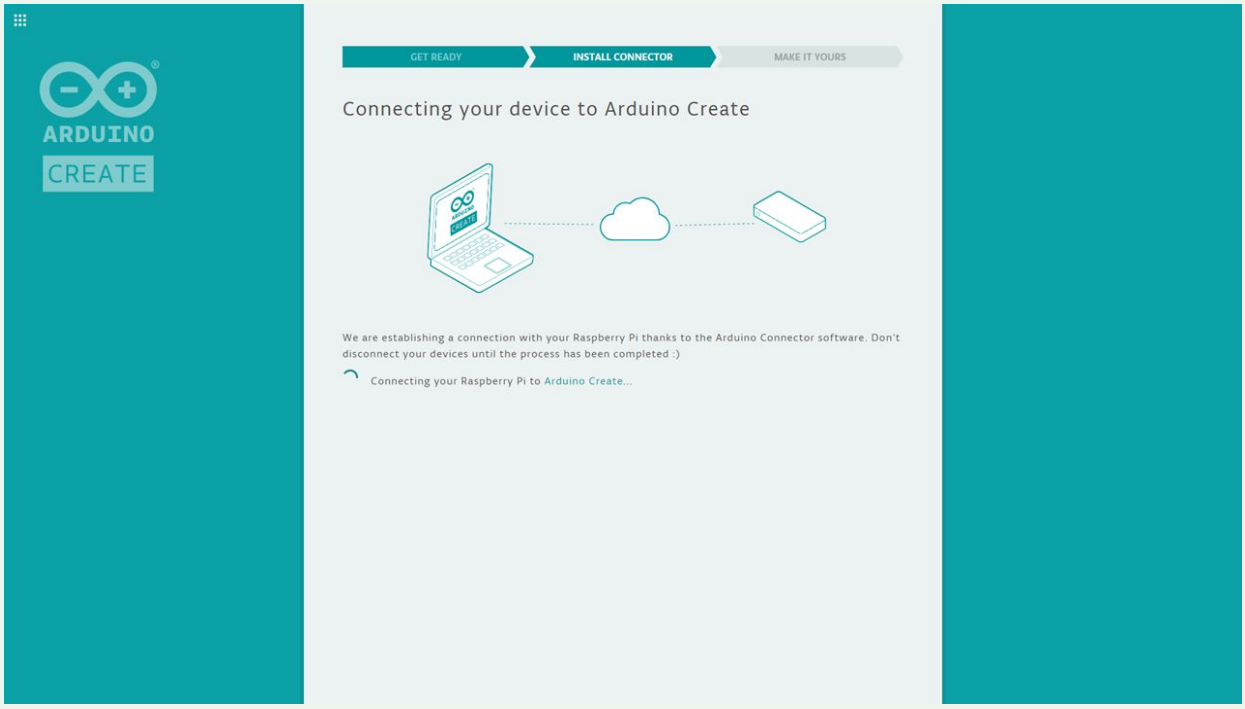
  Permette la configurazione guidata di una nuova scheda tra quelle supportate dalla piattaforma:



La caratteristica di questa procedura è l'installazione dell'**Arduino Plugin**[2] (anche chiamato Arduino Create Agent) e dell'**Arduino Connector**[3]. Entrambi servono per interfacciare e autenticare ad Arduino Create, rispettivamente, il proprio computer e la scheda di sviluppo.

[2] https://github.com/arduino/arduino-create-agent sotto licenza GNU GPL v2
[3] https://github.com/arduino/arduino-connector sotto licenza Apache-2.0





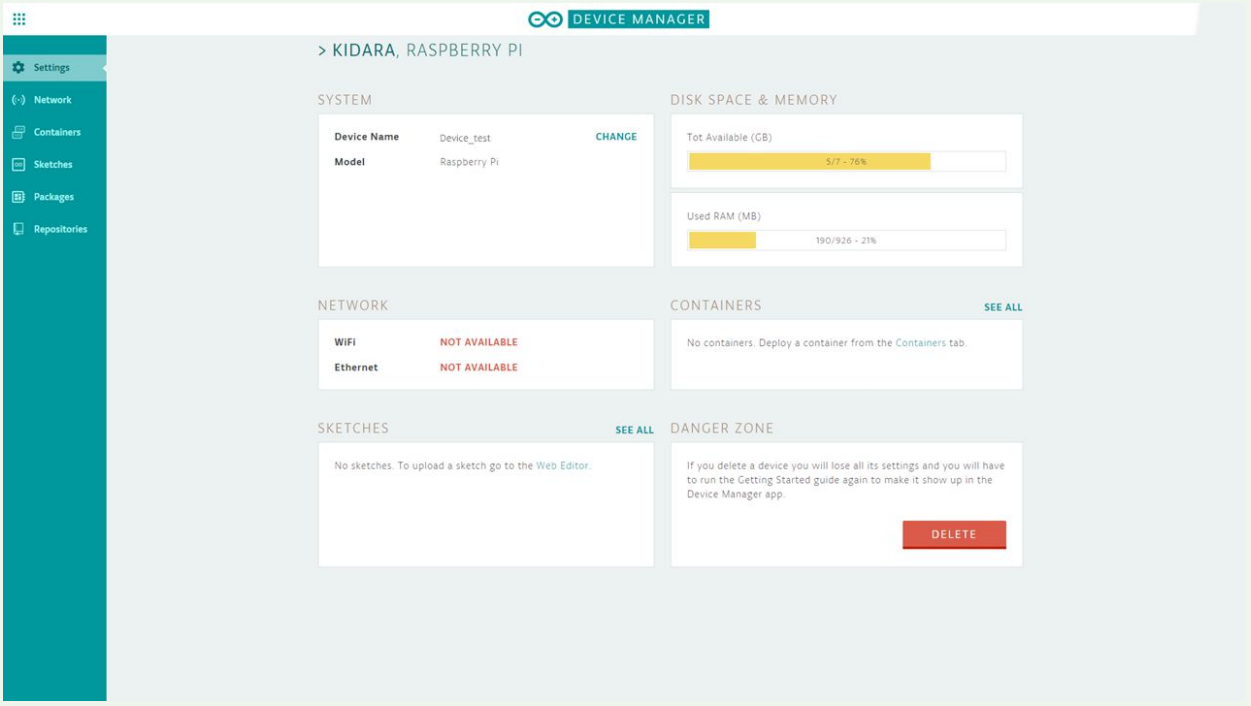
L'**Arduino Create Agent**, nello specifico, è un fork di un serial port JSON websocket[4], che consente di comunicare con i dispositivi embedded da applicazioni web attraverso la porta seriale. In esecuzione, ascolta richieste web nel range di porte 8990-9000. Attraverso una richiesta GET sulle suddette porte (e.g. http://127.0.0.1:8990/info), è possibile trovare quella che trasporta il file JSON contenente le informazioni di nostro interesse.

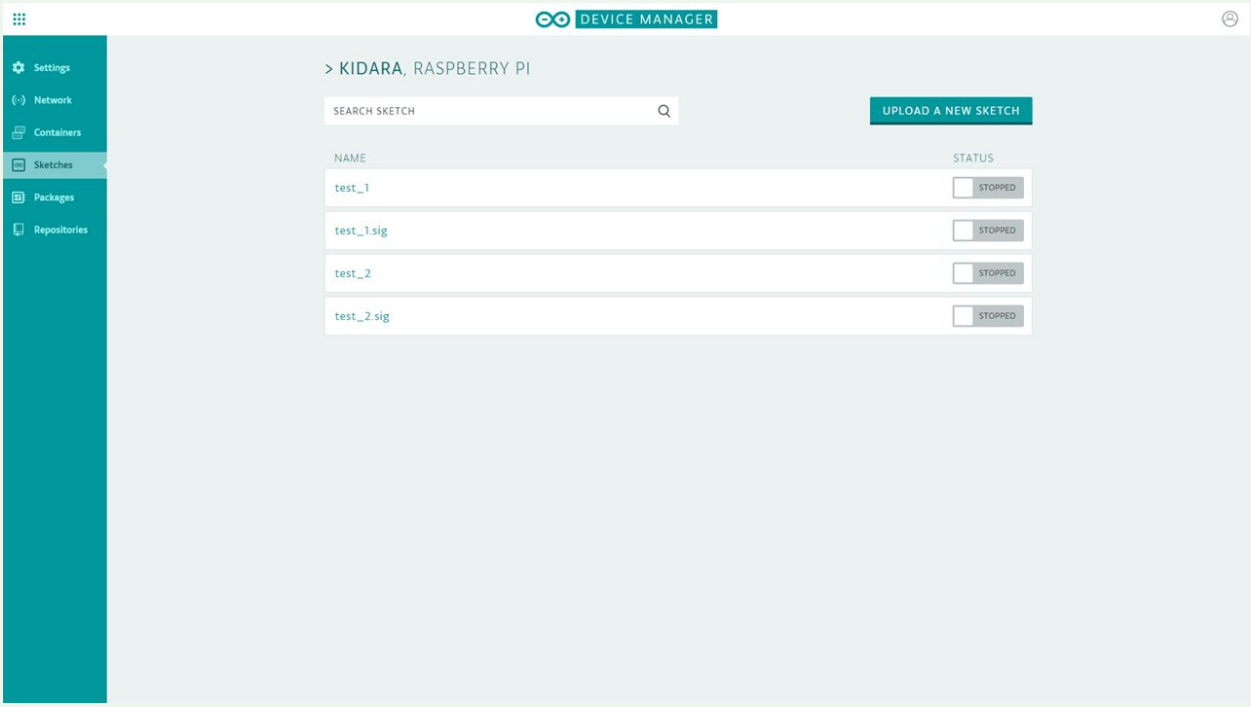
[4] https://github.com/johnlauer/serial-port-json-server

L'**Arduino Connector**, invece, crea una chiave PGP e un certificato da associare alla board per consentire connessioni protette attraverso **MQTT**.

- **DEVICE MANAGER**

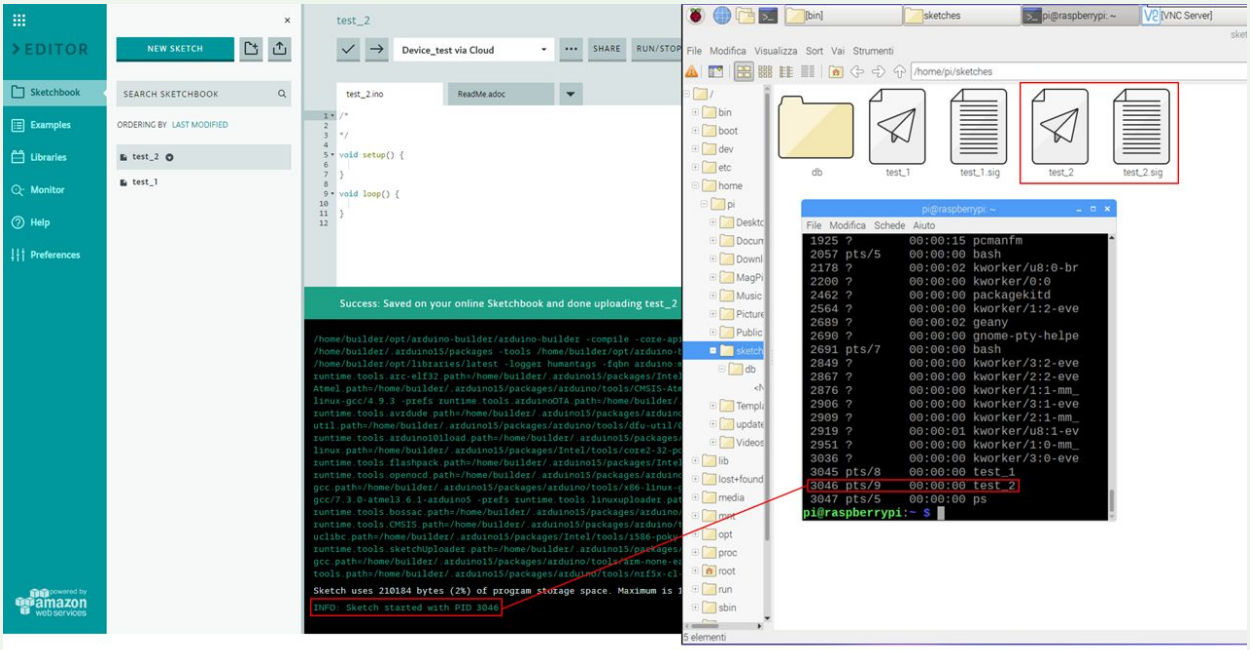
Consente di visualizzare un riepilogo delle informazioni più importanti della scheda collegata, oltre a offrire un'interfaccia user-friendly per la gestione degli sketches e l'aggiornamento dei repositories.





- **WEB EDITOR**

Permette di creare, organizzare e cancellare gli sketches da lanciare sulla scheda. Nel caso del Raspberry, fornisce anche il PID del processo associato allo sketch eseguito.



Una volta scritto, lo sketch è compilato in cloud in un formato eseguibile su processori ARM, viene inviato al Raspberry attraverso la connessione instaurata con l'Arduino Connector ed è salvato all'interno del Raspberry in un'apposita cartella. Ogni sketch ha un file di firma associato, per verificarne l'integrità e l'affidabilità. A questo punto lo sketch viene avviato e viene restituito il PID del processo in esecuzione sul terminale di Arduino Create.

È chiaro, quindi, come sia possibile automatizzare e gestire gli sketches senza utilizzare fisicamente il Raspberry, se non per la prima configurazione.

Questa separazione, ottenuta grazie ad Internet e ad Arduino Create, fornisce un potente strumento per la creazione di software embedded su schede che non sono effettivamente nate per questo scopo.

- **IOT CLOUD**

  Come dice la parola stessa, fornisce un'interfaccia per la gestione dell'IoT (Internet of Things), esula dall'obiettivo di questo testo.

- **PROJECT HUB**

  Una raccolta di tutti i progetti condivisi dalla community di Arduino, con tutorial, guide e consigli per neofiti e esperti del settore. Prevalemente contenuti DIY (Do It Yourself).

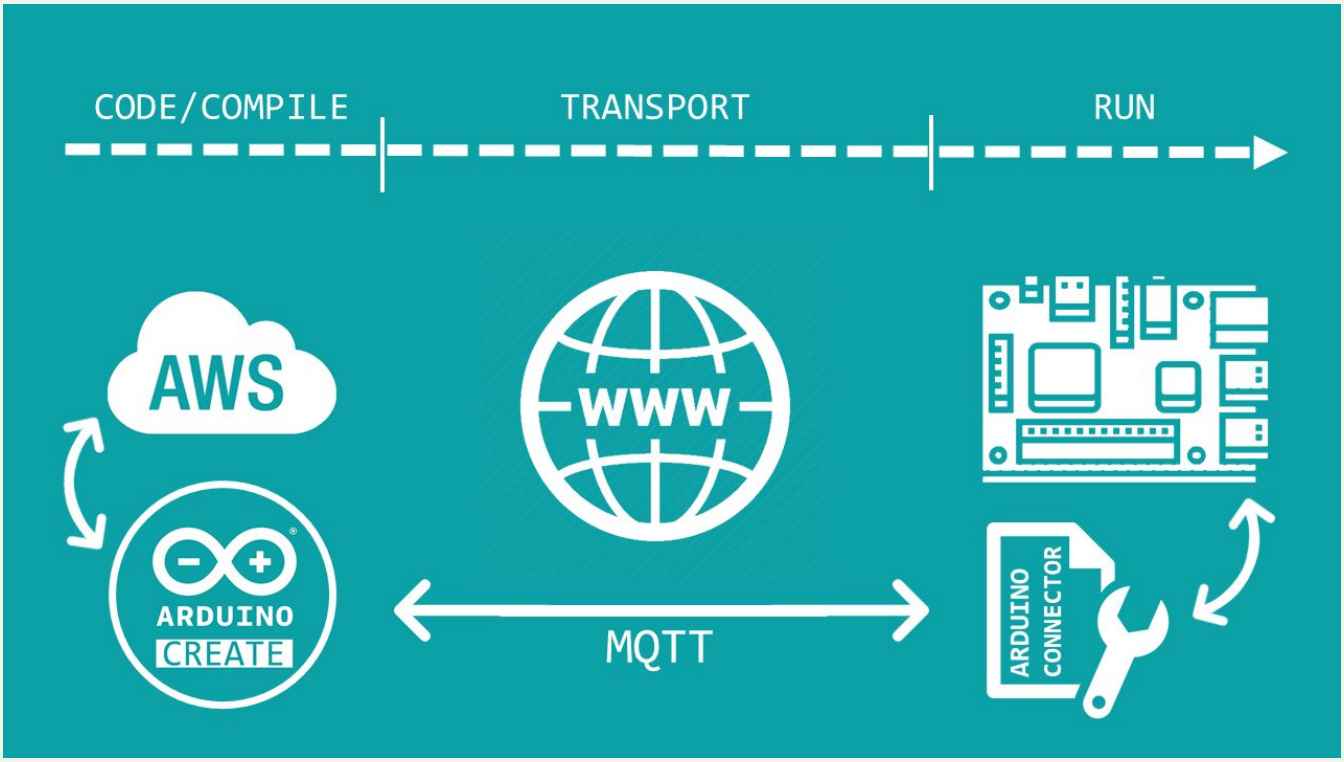
Ora che abbiamo analizzato la struttura di Arduino Create, è più chiaro come esso possa fungere da strumento per la realizzazione di sistemi embedded attraverso le schede Pi. Ciò che manca da capire è come funziona, nel dettaglio, la connessione tra Arduino Create e il nostro Raspberry; e soprattutto se è possibile dissociarsi dal puro ambiente web continuando a beneficiare dei vantaggi che questo strumento ci mette a disposizione.

Arduino Create (Under the hood)

Ciò che accade quando installiamo Arduino Connector sul nostro Raspberry, è la creazione di un canale di comunicazione protetto per scambiare informazioni con Create. Questa comunicazione avviene attraverso Internet, ed è quindi necessario un protocollo per veicolarla. Si è scelto di utilizzare MQTT (Message Queue Telemetry Transport), rappresentante uno standard nel panorama embedded per la sua efficienza nei costi energetici. Il principio di funzionamento è di tipo publish-subscribe (talvolta chiamato observable-observer dagli amanti della programmazione ad oggetti):

- un dispositivo funge da broker, gli altri rappresentano i client;
- ogni client può inviare un messaggio con un certo topic (argomento, anche più di uno) e può registrarsi per la ricezione di messaggi di altri argomenti;
- il broker si occupa del processo di smistamento dei messaggi ai client registrati.

Il modo in cui questo protocollo opera, tra Arduino Create e la nostra scheda, è tramite l'invio di messaggi JSON per il controllo (esecuzione e interruzione) e la raccolta dati degli sketches; tutto ciò utilizzando chiavi e certificati generati al momento dell'installazione del Connector.



This image was derived from arduino.cc

La particolarità di questo sistema è l'utilizzo di AWS (Amazon Web Services) da parte di Create. Il team di Arduino ha infatti deciso di creare la propria piattaforma web

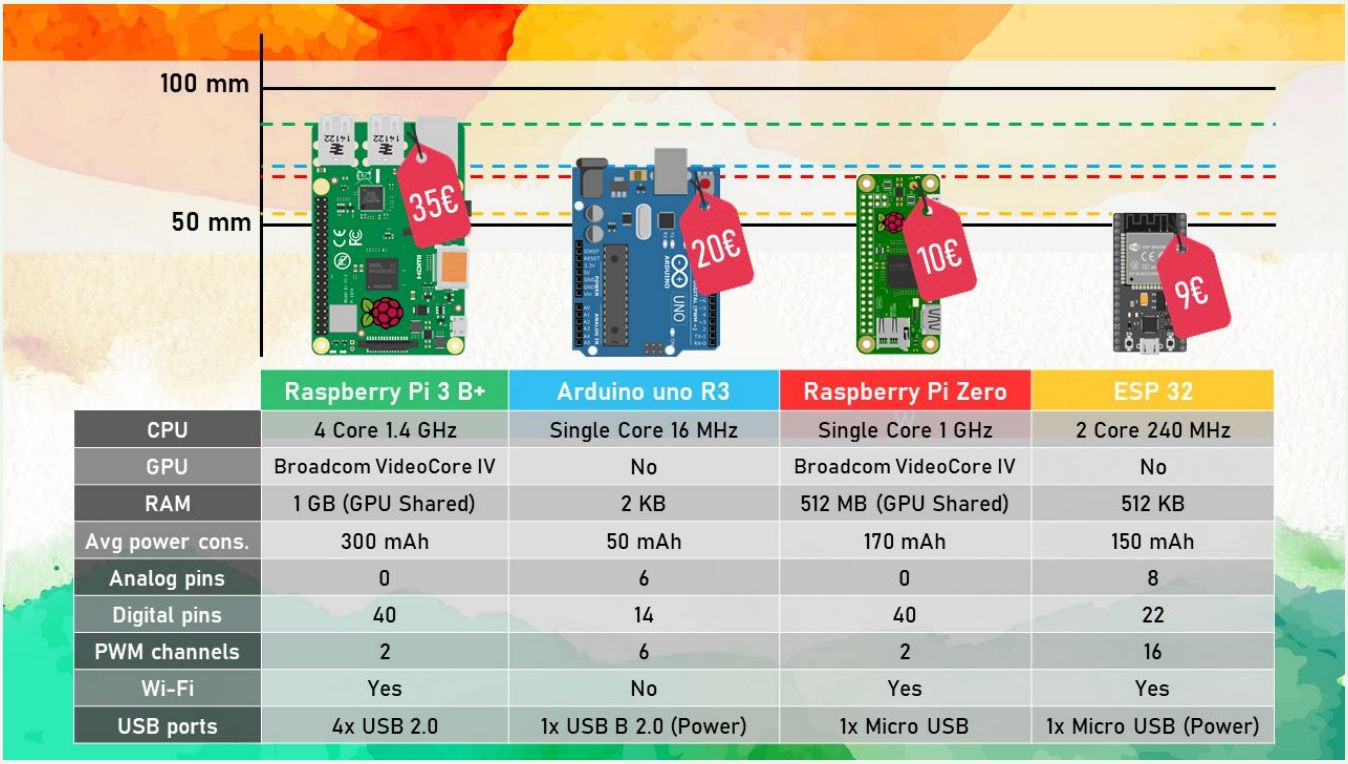
appoggiandosi ai servizi offerti da Amazon. A causa di ciò, nonostante siano disponibili i sorgenti del Connector su GitHub, non è possibile utilizzare tale sistema distaccandosi dalla piattaforma originaria (Create & AWS), almeno non con poche difficoltà.
Se si vuole creare, ad esempio, il proprio dispositivo ad uso commerciale, sarà quindi necessario fare affidamento ai servizi offerti da AWS, o necessariamente analizzare e riscrivere parte del codice sorgente del Connector.

In conclusione, Arduino Create rappresenta un ottimo strumento per la prototipazione e lo sviluppo di applicazioni embedded su schede Pi, dall'interfaccia chiara e dalla facilità di configurazione; ricordando, però, che i nostri dati verranno inviati, oltre che ai nostri dispositivi, ad Arduino ed Amazon. In questi tempi, dove la raccolta di Big Data rappresenta lo strumento più importante per le analisi di mercato, l'utilizzo di questa piattaforma potrebbe non essere sempre la scelta migliore.

Comparazione boards

Siamo arrivati, dunque, al capitolo finale dello sviluppo embedded su Raspberry, dove si vuole mostrare, più da vicino, quali sono i competitors sul mercato. Analizzeremo, oltre alle sopracitate schede Pi, due delle più famose boards usate in questo ambiente: Arduino UNO R3 ed ESP 32.

Ecco un estratto delle loro caratteristiche:



	Raspberry Pi 3 B+	Arduino uno R3	Raspberry Pi Zero	ESP 32
CPU	4 Core 1.4 GHz	Single Core 16 MHz	Single Core 1 GHz	2 Core 240 MHz
GPU	Broadcom VideoCore IV	No	Broadcom VideoCore IV	No
RAM	1 GB (GPU Shared)	2 KB	512 MB (GPU Shared)	512 KB
Avg power cons.	300 mAh	50 mAh	170 mAh	150 mAh
Analog pins	0	6	0	8
Digital pins	40	14	40	22
PWM channels	2	6	2	16
Wi-Fi	Yes	No	Yes	Yes
USB ports	4x USB 2.0	1x USB B 2.0 (Power)	1x Micro USB	1x Micro USB (Power)

This image was derived from Wikimedia and Fritzing

Si nota subito che la potenza di calcolo tra le schede Raspberry e i due competitors non è paragonabile. Questo perché l'Arduino e l'ESP nascono per l'ambiente embedded e sono stati progettati per far fronte a problemi tipici di questo mondo, come l'efficienza energetica. Di fatto, proprio per quanto riguarda il consumo energetico medio, l'Arduino si piazza al primo posto, con soli 50mAh, seguito dall'ESP 32 con 150mAh.

Perché, però, tra due schede progettate entrambe per l'embedded, c'è una così sostanziale differenza di consumi?

La ragione è da ricercarsi nella connettività di questi due dispositivi. Infatti, l'interfaccia Wi-Fi, mancante solo nell'Arduino, ha un consumo non irrilevante, arrivando anche a 150mAh nelle trasmissioni intensive.

Altro fattore importante riguarda i pins: le due schede progettate per l'embedded offrono infatti 6/8 pins analogici, che consentono di ricevere e inviare una tensione continua variabile nel tempo. Questo è molto importante quando si vogliono controllare determinati dispositivi. I Raspberry cercano di sopperire a questa mancanza con 4 pins e 2 canali PWM, ben poco per l'ambiente embedded. Ecco perché spesso una scheda Pi, dalla (relativamente) grande potenza computazionale, viene affiancata ad una scheda Arduino (o ESP 32, o altre schede programmabili sviluppate per l'embedded), utilizzata come hub per la raccolta di informazioni e segnali sensoristici, da inviare poi al Raspberry per essere meglio analizzati.

Effettivamente, sia Arduino che ESP (ma in particolare Arduino) possiedono una memoria RAM molto ridotta (2/512KB), che devono utilizzare per l'esecuzione del programma, il salvataggio di variabili e la creazione di stacks in caso di chiamate a funzione.

Infine, dando uno sguardo ai prezzi, le schede Pi si collocano perfettamente sul mercato, con costi paragonabili alle altre due schede. L'ESP 32 rimane un'ottima scelta per il puro mondo embedded, vantando caratteristiche migliori rispetto ad Arduino. D'altro canto, quest'ultimo, rimane una pietra miliare in questo campo, avendo creato una community ed un ecosistema che influenzano, tuttora, lo sviluppo delle altre schede che cercano di mantenere la compatibilità con questo gigante.

Le due boards Pi, come abbiamo visto, fanno difficoltà a sopperire alle mancanze per il puro mondo embedded, ma sono ottime se affiancate ad altre schede per l'interfacciamento sensoristico, offrendo un sistema operativo dotato di GUI e una potenza computazionale non indifferente.

Con questo si conclude l'analisi del Raspberry Pi nell'universo embedded, con i suoi pregi e difetti, posizionandosi come ottima interfaccia tra una scheda a più basso livello ed il proprio personal computer.