Programmazione per l'IoT - Laurea Magistrale in Informatica Applicata - Università degli Studi di Urbino Carlo Bo
Data di pubblicazione 27/09/2022 - DOI: 10.5281/zenodo.7115909

Monitoraggio ambientale e controllo degli accessi attraverso una semplice applicazione in ambito Internet of Things

Andrea Cogorno[1]*, Emanuele Lattanzi[2]

Sommario
L'internet delle cose o Internet of Things (IoT) è ormai una tecnologia affermata e il suo potenziale è enorme. Possiamo considerare l'IoT una sorta di "rete neurale universale" che col passare del tempo, permeerà ogni aspetto della nostra vita. L'IoT è composta da sensori, device e piccoli apparati intelligenti che interagiscono e comunicano con altre macchine, oggetti, ambienti e infrastrutture. Enormi volumi di dati vengono generati, elaborati e tradotti in azioni che possono gestire e controllare le cose (things) che ci circondano, per rendere la nostra vita molto più semplice e sicura; tuttavia, questo insieme di apparati e di tecnologie non è esente da criticità legate alla sicurezza e alla privacy dei dati e al consumo energetico necessario per la loro conservazione ed elaborazione. In questo articolo verrà illustrato un progetto per la realizzazione di una stazione di monitoraggio dei dati ambientali e di controllo degli accessi ad un'ipotetica sala server. Si tratta di un progetto molto accessibile sia dal punto di vista economico che sotto il profilo delle conoscenze informatiche necessarie per la sua implementazione. Allo stesso tempo, il progetto si presta a diversi possibili sviluppi e migliorie, sia dal punto di vista delle funzionalità che dell'ottimizzazione, anche grazie alla continua evoluzione dell'hardware e delle tecnologie che costituisco l'Internet of Things.

Keywords
Arduino IDE — DHT-11 — ESP8266 — InfluxDB — Python — Raspberry PI — Sense HAT

[1] *Studente della Laurea Magistrale in Informatica Applicata, Università degli Studi di Urbino Carlo Bo, Urbino, Italia*
[2] *Docente di Programmazione per l'Internet of Things, Università degli Studi di Urbino Carlo Bo, Urbino, Italia*
***Corresponding author**: email: a.cogorno@campus.uniurb.it

Introduzione

L'Internet delle cose, meglio conosciuto con l'accezione inglese di Internet of Things (o più brevemente IoT) negli ultimi anni ha avuto un incredibile e rapidissimo sviluppo e una massiccia diffusione in moltissimi campi di applicazione e ricerca. Volendo semplificare al massimo potremmo definire l'IoT come un insieme di tecnologie che consentono di collegare ad Internet qualsiasi tipo di apparato, dal frigorifero di casa, alla telecamera di sorveglianza di una banca, dalla stazione meteorologica in alta montagna, alla fit band al nostro polso. In realtà, dietro a questa visione un po' semplicistica, si nascondono architetture e tecnologie tutt'altro che banali con diverse potenzialità, ma anche con criticità importanti che coinvolgono diversi campi della scienza e della ricerca.

Una delle definizioni che meglio descrivono l'IoT è quella fornita da Oracle che la definisce come una rete di oggetti fisici dotati di sensori integrati, software ed altre tecnologie che consente la connessione e lo scambio di dati con altri sistemi e device connessi alla Rete. Inoltre, l'IoT ha aggiunto un nuovo, aspetto alla comunicazione tra macchine: oltre alle già esistenti dimensioni "any time" e "any place" ora si è aggiunta l'estensione "any things" che di fatto consente a questa tecnologia di controllare in maniera continuativa e immersiva l'ambiente in cui è implementata, trasferendo i dati monitorati in un punto di raccolta e di elaborazione con lo scopo di programmare una risposta "intelligente" sfruttando le capacità cognitive del sistema.

Dal punto di vista funzionale quindi un sistema Iot svolge quattro attività principali:

1. Raccolta dei dati (data collection)
2. Elaborazione dei dati (data processing)
3. Visualizzazione dei dati (data visualization)
4. Azione (acting)

Da un punto di vista architetturale invece, possiamo identificare tre livelli (layers) in cui vengono svolte queste attività. Il livello che sta alla base è il cosiddetto **edge layer** costituito da sensori e attuatori di ogni tipo che però solitamente hanno scarse capacità di elaborazione e di comunicazione: a questo livello avviene la raccolta dei dati. Il livello immediatamente sopra è il **fog layer** costituito da device con maggiori potenzialità di calcolo e di comunicazione che spesso fungono da gateway tra i sensori di basso livello cui sono connessi. L'ultimo livello, ovvero il **Cloud layer** è dove i dati vengono salvati, aggregati ed elaborati.

Appare subito evidente come i dati raccolti a livello di edge layer costituiscano una mole incredibilmente vasta di informazioni, meglio nota come "big data", che sono strettamente legati alle tecnologie di Machine Learning e di Artificial Intelligence ed acquisiscono un valore sempre più importante se possono essere acquisiti e conservati in formati inter-operabili, anche se allo stato attuale, siamo ancora lontani dalla definizione di uno standard in tal senso.

Questa importante e rapida espansione dell'Iot si è concretizzata grazie alla convergenza di tecnologie e infrastrutture centrate attorno alla diffusione di Internet:

- L'industria 4.0 e la digitalizzazione
- La diffusione di sensori e attuatori sempre più economici e di facile integrazione
- La diffusione di smartphone e altri device sempre connessi
- Il cloud computing
- Il machine learning e l'AI

I campi di applicazione dell'IoT sono molteplici e sempre in evoluzione, ma quelli che al momento stanno trainando lo sviluppo delle varie tecnologie coinvolte sono:

- L'Industrial Internet of Things (IIoT)
- La sorveglianza e la sicurezza
- La salute (smart health)
- La domotica (smart homes)
- Le Smart cities
- La Smart mobility

Anche a causa di questa inarrestabile diffusione, alcuni aspetti delle tecnologie che sostituisco l'universo IoT sono fonte di discussione. Una delle tematiche maggiormente dibattute, riguarda la sicurezza dei dati e della privacy dei soggetti coinvolti, infatti le tecnologie per la trasmissione e conservazione di questa grande mole di dati a livello cloud, allo stato attuale, non garantiscono un perfetto anonimato. A questo si aggiungono altri elementi di discussione tra loro correlati: la quantità di dati prodotti dai device nell'edge layer, che secondo uno studio (pubblicato da Springer Nature journal) hanno raggiunto i 40 ZB/annui nel 2019 (v. figura 1) produce un elevato consumo energetico necessario per la sua elaborazione e l'aumento esponenziale di sensori e attuatori collegati ad Internet, seppur estremamente efficienti e poco avidi di energia, necessitano di essere alimentati con una conseguente ricaduta sui fattori energetici.

Il Progetto

Per l'esame di Programmazione dell'Internet of Things del corso di Laurea Magistrale dell'Univeristò Carlo Bo di Urbino ho pensato di realizzare un sistema di monitoraggio e controllo di una sala server. Oltre alla supervisione in tempo reale dei parametri di temperatura ed umidità a cui lavorano i computer presenti nella sala, è previsto un sistema di vigilanza degli accessi per consentire l'ingresso esclusivamente a chi ne ha diritto.

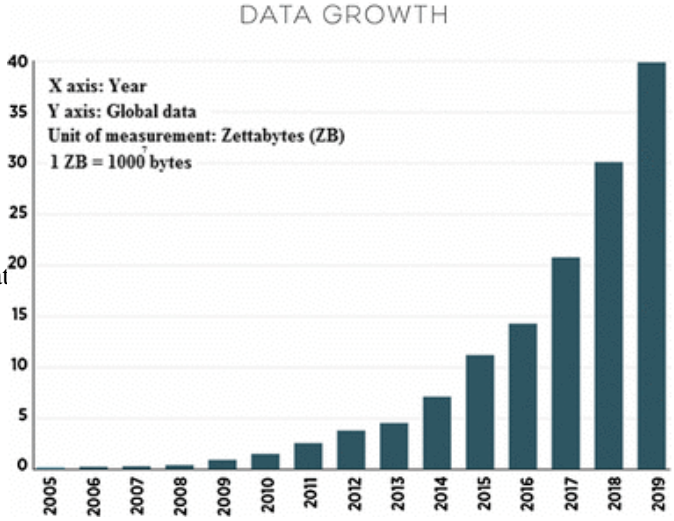

Figura 1. crescita dei dati prodotti da sistemi Iot

Il sistema è costituito da una scheda ESP8266 dotata di Development Kit che registra ad intervalli regolari la temperatura ambiente e l'umidità e li invia attraverso una rete wireless ad un database InfluxDB sotto forma di time-series. Gli strumenti di InfluxDB consentono di avere un grafico in tempo reale dei dati registrati e contestualmente offrono un repository per lo storico dei dati.

Per la gestione dell'accesso alla sala invece si è utilizzato un Raspberry Pi provvisto di una scheda Sense HAT che è dotata di una serie di sensori e di un piccolo pannello a matrice di led. Data la vicinanza dei sensori al processore, la temperatura di esercizio della CPU influenza l'acquisizione dei dati da parte dei sensori presenti sulla Sense HAT, per questo per monitorare i dati ambientali ho preferito utilizzare una scheda ESP che può essere collocata in un punto maggiormente funzionale e adatto allo scopo. La piattaforma Raspberry Pi è stata utilizzata per gestire il riconoscimento vocale sfruttando una API messa a disposizione da Google e che permette anche la sintesi vocale. In particolare, l'accesso sarà consentito solo alle persone autorizzate che conoscono la keyword. Ho ipotizzato che nella realtà questo potrebbe avvenire azionando un apri-porta automatico dotato di interfaccia Wi-Fi. A corredo di questa funzionalità ho utilizzato il pannello a matrice di led presente nella scheda Sense HAT per segnalare la possibilità di accedere o meno ai locali visualizzando un' apposita segnaletica.

Per programmare la ESP8266 ho utilizzato Arduino IDE, un sistema estremamente semplice (e per questo molto diffuso) e versatile, mentre per la programmazione di Raspberry Pi ho utilizzato il linguaggio Python 3. Come device di input per i comandi vocali ho utilizzato un comune microfono USB. Di seguito vediamo i dettagli delle soluzioni hardware e software adottati per la realizzazione del progetto.

ESP8266 + DevKit

Si tratta di un modulo comunemente identificato come NodeMCU DevKit che è appunto costituito da una scheda ESP8266 che

ha le seguenti caratteristiche:

- CPU Tensilica Xtensa LX106
- 64Kb di memoria RAM per le istruzioni
- 96Kb di memoria RAM per i dati
- Stack TCP/IP e connettività wifi b/g/n
- GPIO 16
- UART / I2C / I2S/ SPI /
- Modulo ADC a 10 bit

A tale scheda è collegato il modulo DHT-11 (vedi figura 2) che è appunto un sensore digitale di temperatura e umidità. Il modulo ESP8266 può essere programmato con il firmware NodeMCU (già presente sulle schede NodeMCU DevKit) che permette di interagire utilizzando il linguaggio LUA, tuttavia si è optato per utilizzare l'IDE Arduino a cui sono state aggiunte le librerie per gestire questo modulo (recuperabili a questo indirizzo)

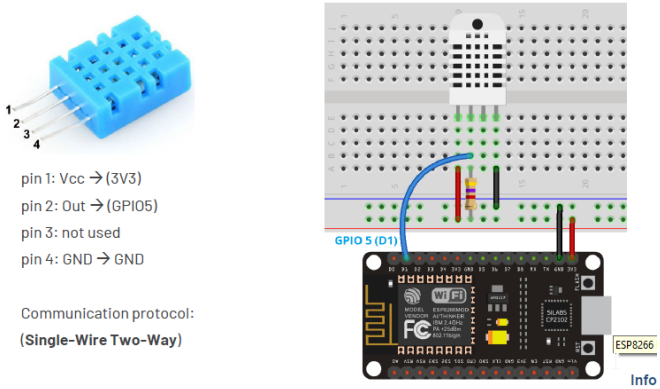

Figura 2. Schema collegamento DHT-11

Il codice utilizzato: Arduino IDE

Per lo sviluppo del codice necessario a programmare l'ESP8266 la scelta è ricaduta su Arduino IDE (basato sul linguiaggio C) per la sua facilità di utilizzo e per i numerosi esempi facilmente reperibili in rete. Per permettere la stesura del codice sorgente, l'IDE include un editor di testo dotato di alcune particolarità, come il syntax highlighting, il controllo delle parentesi e l'indentazione automatica. Ogni sketch di Arduino ha una struttura basata su due funzioni:

void setup() è la prima funzione ad essere invocata durante l'esecuzione dello sketch. ed è qui che vengono generalmente definite tutte le variabili necessarie al funzionamento del codice.

void loop() è la funzione che, come suggerisce il nome, sfrutta la tecnica di programmazione nota come super loop (o anche endless, o infinite loop) e che contiene il codice vero e proprio del programma che verrà ripetuto appunto all'infinito.

Nella prima parte del codice (vedi figura 3) ho importato le librerie necessarie per interfacciarsi con il sensore DHT-11, con il database InfluxDB e quelle per poter sfruttare la connessione wireless. Inoltre sono state dichiarate alcune costanti necessarie per la connessione al database e alla rete

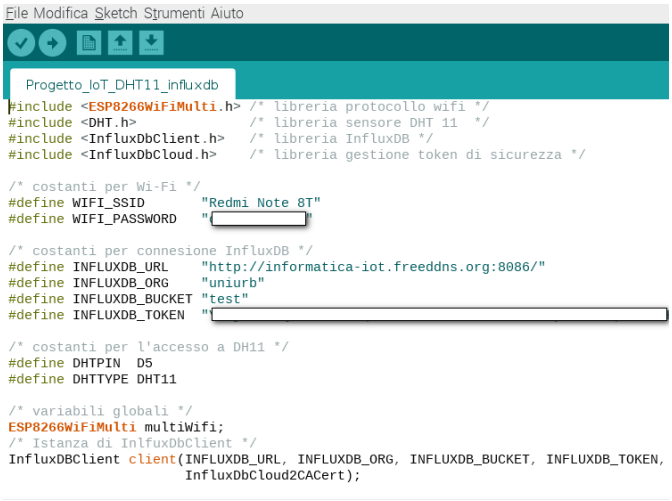

Figura 3. codice Arduino IDE - prima parte

Wi-Fi (offuscate nell'immagine) e per collegarsi al sensore DHT-11.

All'interno della funzione setup() ho inizializzato la comunicazione seriale impostando la velocità e il sensore DHT-11 oltre alla connessione Wi-Fi passando come parametri il codice SSID e la password; quindi ho testato la connessione al database.

All'interno della funzione loop() semplicemente vengono letti i dati relativi alla temperatura e alla pressione rilevati dal sensore DHT-11 e successivamente vengono inviati al database. É stato inserito un delay tra l'esecuzione di un loop e il successivo. I dati raccolti e inviati al database InfluxDB possono essere facilmente visualizzati utilizzando la funzionalità "Data Explorer" messa a disposizione dall'interfaccia web, impostando la query nel modo che si ritiene più opportuno (vedi figura 4).

RASPBERRY PI + Sense HAT

Raspberry Pi è un computer a scheda singola (single board computer) progettato per ospitare sistemi operativi basati sul kernel Linux o RISC OS, ma è dotato di un sistema operativo appositamente dedicato, chiamato Raspberry Pi OS sviluppato a partire da Debian GNU/Linux. Raspberry Pi è "un system-on-a-chip" di fabbricazione Broadcom (in particolare il BCM2837 per il Raspberry Pi 3 utilizzato in questo progetto) che incorpora un processore ARM, una GPU VideoCore IV, e 1 Gigabyte di memoria. La piattaforma non prevede hard disk, ma utilizza una scheda SD per il boot e per la memoria non volatile. Si alimenta attraverso un connettore micro USB con cc di ingresso da 5 V/2,5 A ed è di dimensioni davvero ridotte: 120 mm x 75 mm x 34 mm per 75 g di peso. Tra le altre caratteristiche annotiamo:

- LAN wireless 2,4/5 GHz IEEE 802.11.b/g/n/ac
- Bluetooth 4.2/BLE
- 4 porte USB 2.0
- Header con GPIO a 40 pin
- uscita video HDMI full size

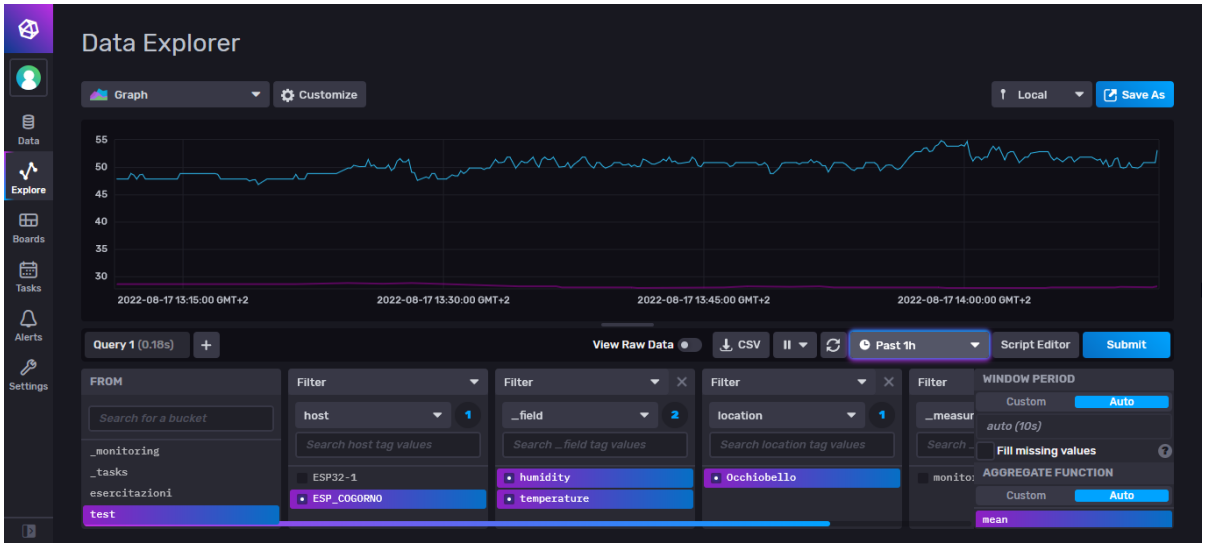

Figura 4. InfluxDB - DataExplorer

- uscita stereo a 4 poli

Il Sense HAT è una scheda dotata di diversi sensori, un dispositivo di output (display a matrice di LED) e un dispositivo di input (micro joystick a 5 posizioni). I sensori d'ambiente sono in grado di registrare temperatura, umidità e pressione atmosferica a cui si aggiungono anche i sensori di orientamento: giroscopio, accelerometro e un magnetometro. Per poter utilizzare la scheda Sense HAT è necessario installare da terminale il software che consentirà di sfruttare la relativa API. Per l'input vocale ho utilizzato un piccolo microfono USB a condensatore. Quando viene collegato al Raspberry Pi, il microfono viene riconosciuto dal sistema e gli viene assegnato un ID che ho utilizzato all'interno del codice per identificare la sorgente audio.

La libreria SpeechRecognition ("Library for performing speech recognition, with support for several engines and APIs, online and offline") permette di usare diversi motori per il riconoscimento vocale o di utilizzare una delle tante API di servizi online. L'utilizzo della libreria TTS (Text To Speech) di Google è un esempio di applicazione di Machine Learning ad un sistema IoT applicato al Cloud Layer. In questo caso il microfono agisce come sensore mentre il Raspberry Pi funge da gateway che invia i dati nel cloud Layer dove vengono elaborati e successivamente inviati al Raspberry Pi. Come accennato nell'introduzione questo potrebbe essere un potenziale pericolo per la sicurezza, infatti la keyword per entrare nella sala server potrebbe essere potenzialmente intercettata.

Il codice utilizzato: Python 3

Python è uno dei linguaggi di programmazione più versatili e ampiamente usati (come riportato dalle classifiche redatte da TIOBEe PYPL). Grazie all'enorme disponibilità di librerie e framework, si presta a un'ampia serie di utilizzi e consente di sviluppare applicativi di ogni genere. Inoltre è un linguaggio interpretato e non richiede di essere compilato prima dell'esecuzione.

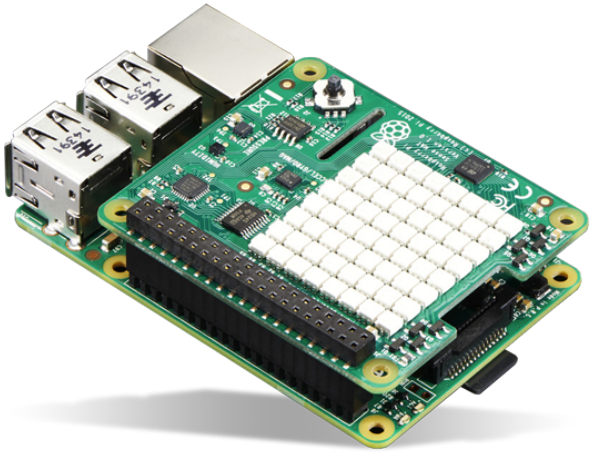

Figura 5. Raspberry PI con Sense HAT

Come ambiente di sviluppo ho utilizzato Thonny un IDE Open Source multi-piattaforma ideale anche per principianti.

Nelle prime righe di codice ho importato alcune librerie necessarie per interfacciarsi con l'API di Google Text To Speech e per interagire con il sistema operativo e con la scheda Sense HAT. Inoltre ho definito alcune costanti sotto forma di stringhe di testo che saranno utilizzate per la sintesi vocale. Tra queste c'è anche la keyword che consente l'accesso alla sala server. Inoltre ho dichiarato altre variabili per gestire la matrice di led in dotazione con la scheda Sense HAT.

Anche in questo caso ho utilizzato la tecnica di programmazione del super loop, ma per acquisire il parlato ed effettuare la sintesi vocale ho definito due funzioni:

get_audio() prende come input i dati provenienti dal microfono USB che ho settato come sorgente audio, quindi tramite l'API di Google cerco di interpretare l'input del microfono. Se questo non viene riconosciuto il codice torna "in ascolto".

accesso_vocale.py

```python
import speech_recognition as sp # libreria per il riconoscimento vocale che supporta diverse API
import os, subprocess           # os = libreria per interagire con il sistema operativo
                                # subprocess è un modulo che consente di gestire nuovi processi

from gtts import gTTS           # gTTS è la libreria per interfacciarsi con l'API Google Text To Speech
from time import sleep          # la funzione sleep() del modulotime consente di sospendere l'esecuzione
from sense_hat import SenseHat  # modulo per gestire Sense HAT

# istanza per SenseHat
sense = SenseHat()
# reset della matrice di Led del Sense HAT
sense.clear()

clearConsole = lambda: print('\n' * 150) #cancello la console di Python

#definisco alcune stringhe di testo
entra   = "andrea" # questa è l'unica che consente l'accesso alla sala
granted = "bentornato, buon lavoro!"
denied  = "mi dispiace, non puoi entrare"
esci    = "arrivederci"
```

Figura 6. codice Python - prima parte

speak(text) passa alla libreria gTTS di Google il testo da riprodurre, settando l'italiano come lingua e salvo l'output in un file mp3. Se il file esiste già viene cancellato e sostituito con quello nuovo. Quindi lancio il lettore mp3 mpg123 per inviarlo all'output audio predefinito.

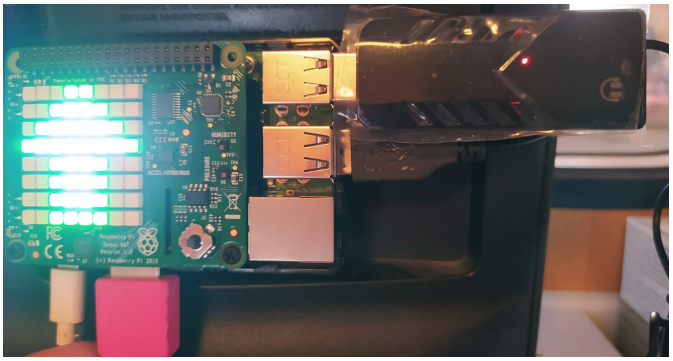

Figura 7. Matrice led: freccia verde

Nel ciclo while che gestisce il super loop il codice rimane in ascolto di un eventuale input proveniente dal microfono e attraverso la funzione get_audio() lo assegno ad una variabile. Se il testo corrisponde alla stringa che consente l'accesso alla sala, il programma dà il benvenuto e sulla matrice led viene visualizzata una freccia verde. In caso contrario l'accesso è negato e la matrice visualizza l'immagine di divieto di accesso. Quando l'utente esce dalla stanza sempre tramite comando vocale potrà chiudere nuovamente la porta, in quanto il loop rimane costantemente in ascolto.

Possibili sviluppi

I due codici utilizzati in questo progetto sono volutamente molto semplici e si prestano a ulteriori sviluppi e migliorie.

Tanto per cominciare si sarebbe potuto utilizzare il protocollo MQTT (Message Queue Telemetry Transport)un protocollo semplice e leggero per lo scambio di messaggi, per minimizzare il traffico sulla rete. MQTT è mirato a ridurre al minimo i requisiti dei dispositivi e per monitorare temperatura ed umidità attraverso il modulo ESP8266 si sarebbe potuto utilizzare Eclipse Mosquitto, uno dei più noti broker MQTT Open Source. In fondo la variazione di questi parametri difficilmente cambierà con tale rapidità da richiedere un aggiornamento real-time.

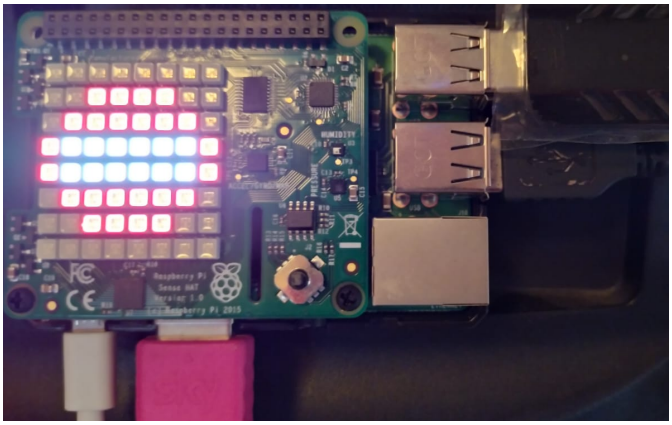

Figura 8. Matrice led: divieto di accesso

Sempre per quanto riguarda la programmazione della ESP32 con DevKit, si sarebbe potuto utilizzare FreeRTOS un sistema operativo in tempo reale distribuito gratuitamente con la licenza open source MIT, costruito con particolare attenzione all'affidabilità e alla facilità d'uso.

Un ulteriore viluppo riguarda l'utilizzo di ThingSpeak, una piattaforma open source per l'IoT che permette di collezionare e salvare nel cloud dati prodotti da sensori e dispositivi vari. Inoltre, la piattaforma consente di analizzare e visualizzare i dati tramite MATLAB e di agire su di essi.

Conclusioni

Il progetto è volto a dimostrare come sia possibile realizzare un sistema IoT a basso costo, altamente accessibile, sia da un punto di vista dell'hardware che delle necessarie competenze di programmazione. L'utilizzo di Arduino IDE rende la programmazione della scheda ESP8266 accessibile a chiunque. L'uso di Python 3 per il codice da utilizzare su Raspberry (dotato della scheda Sense HAT) è comunque facilitato da una community molto attiva e dalla possibilità di reperire prototipi di codice esemplificativi. Il database scelto per la memorizzazione dei dati è InfluxDB, particolarmente adatto per le time-series e grazie agli strumenti online è possibile avere in tempo reale grafici dei dati rivelati dal sensore DHT-11. L'utilizzo della API di Google per il riconoscimento vocale è un esempio di Machine Learning applicato al Cloud Layer dell'architettura IoT.

Nel corso degli ultimi anni, l'IoT è diventata una delle tecnologie più importanti del 21° secolo. Ora che possiamo collegare oggetti di uso quotidiano a Internet, è possibile realizzare la comunicazione tra persone, processi e cose. Mediante l'elaborazione a basso costo, il cloud, i Big Data, e le tecnologie "mobile", gli oggetti fisici possono condividere e raccogliere i dati con un minimo intervento umano e in tal modo il mondo fisico incontra e si fonde con il mondo digitale.