UNIVERSITA' POLITECNICA DELLE MARCHE

FACOLTA' DI INGEGNERIA

Corso di Laurea in Ingegneria Elettronica

SVILUPPO E REALIZZAZIONE HW E SW DI UN SENSORE WIRELESS PER IL MONITORAGGIO DELLA QUALITÀ DELL'ARIA

DESIGN HW AND SW OF A WIRELESS SENSOR TO CONTROL INDOOR AIR QUALITY

Relatore:

Prof. **Ennio Gambi**

Tesi di Laurea di:

Alessandro Fermanelli

Correlatore:

Ing. **Adelmo De Santis**

A.A. 2020/2021

INDICE

1. INTRODUZIONE

La qualità dell'aria al chiuso , o **IAQ** (Indoor Air Quality), è un problema di cui si è a conoscenza da circa gli anni '70. La correlazione tra una scarsa qualità dell'aria e la salute dell'uomo è ormai una certezza scientifica, affermata da studi nei campi dell'epidemiologia, della tossicologia e della fisiologia clinica. Una valutazione condotta dall'**Organizzazione Mondiale della Sanità** (OMS o WHO) ha accertato che le morti premature dovute all'inquinamento dell'aria indoor e outdoor, ammontano a circa due milioni l'anno (quelle legate solo alla combustione dei combustibili solidi). In aggiunta c'è da dire che il clima indoor differisce da quello outdoor sotto diversi punti di vista:

1. Origine degli inquinanti; gli inquinanti presenti nel clima indoor in parte derivano dalle concentrazioni outdoor (ovviamente questo avviene per infiltrazione e aerazione), ma diversi sono i punti emissivi all'interno dell'abitacolo. Tali sorgenti possono essere definite continue o intermittenti. Le prime generano emissioni uniformi nel tempo (ad es. elementi di arredo che emettono formaldeide), mentre le altre variano nell'arco della giornata i livelli di concentrazione (ad es. deodoranti, candele, bastoncini profumati, prodotti per la pulizia oppure una parete verniciata che varia l'emissione in base a temperatura e umidità)
2. "Permanenza" degli inquinanti; è chiaro che un ambiente outdoor è uno scenario altamente stocastico in cui le concentrazioni, ma in particolare l'esposizione scende al dettaglio, nel senso che ogni individuo è soggetto a una condizione mutevole dei contributi inquinanti. Diversamente per l'indoor a essere colpiti sono gli occupanti nel loro insieme e senza discrimine; ovviamente esistono soggetti più sensibili di altri perciò non è sempre vero. In generale l'IAQ è una causa comune, manifesto di una scarsa cura per il microclima all'interno della struttura. A riguardo si fa riferimento alla cosiddetta sindrome dell'edificio malato, una condizione di malessere che sta ad indicare una situazione di cura superficiale della struttura, che si manifesta negli occupanti in una particolare insofferenza, nonché nello sviluppo di alcune patologie.

Molte organizzazioni nonché ricercatori hanno partecipato all'approfondimento delle cause, delle conseguenze e delle possibili soluzioni proprio per far fronte a un problema tanto evidente. Ciò ha condotto ad avere una vasta documentazione al riguardo, considerando che solo nell'ultimo ventennio ben 7287 sono state le pubblicazioni all'interno dell'Unione Europea, includendo i paesi extracomunitari Norvegia, Turchia e Svizzera (grande partecipazione da parte dell'Italia con ben

il 10.4%). Questa crescente e continua attenzione per l'IAQ ha dimostrato la necessità di un cambio socio-culturale volto alla prevenzione della salute e di movimenti promotori di una maggiore sensibilizzazione al riguardo.

Un dato importante, dedotto dagli studi condotti, riporta che **la maggior parte della popolazione spende il 90% del proprio tempo in locali al chiuso**, sottolineando ancor più l'importanza di una politica per garantire una buona IAQ. Questo dato può essere ben compreso se si considera che indoor è un termine che sta ad indicare non solo case, ma anche scuole, università, banche, uffici, ospedali, punti di ristoro (ristoranti e bar), locali ricreativi e trasporti pubblici, questo giustifica quindi la percentuale vista precedentemente. In questi ambienti l'aria può essere significativamente più inquinata rispetto all'esterno, è perciò necessario elaborare dei certificati/bolli di qualità che accertino una cura del microclima interno. A livello europeo le organizzazioni che si sono mosse per fornire una valida legislazione e uno standard, sono state il **CEN** (European commitee for standardization) , l'**ISO** (International organization for standardization), l'**ECA** (European collaborative action) e la già citata **WHO**. In particolare, quest'ultima ha stilato una serie di linee guida riguardo l'IAQ, facendo riferimento a un numero di inquinanti presenti negli interni, concentrando l'attenzione sugli elementi più pericolosi a cui è esposto l'uomo. Le sostanze oggetto di studio sono state: benzene, diossido di azoto(NO_2), plyclicic aromatic hydrocarbons (PAHs), naftalina ($C_{10}H_8$), monossido di carbonio (CO), radon, trichlorethylene (C_2HCl_3) e tetrachloroethylene (C_2Cl_4). La lista di inquinanti è poi aumentata con gli anni, i risultati sono riassunti nella [Tabella 1].

TABELLA 1. loor air contaminants: reference values used in some European countries, guide values, and unitary risk of the World Health Organization (WHO).

Contaminant (Unit of Measurement)	WHO Guidelines (Outdoor [a])	WHO Guidelines (Indoor [a])	France	Germany	Netherlands	United Kingdom	Belgium (Flanders)	Finland [c]	Austria	Portugal	Norway	Lithuania	Poland (Residential)	Poland (Public Offices)
Reference	[32,33]	[34]	[21–25]	[18]	[26]	[20]	[28]	[27]	[29]	[30]	[35]	[36]	[37]	[37]
Benzene [b] ($\mu g\ m^{-3}$)	0.17 (UR/lt) 10^{-6} 1.7 (UR/lt) 10^{-5}	0.17 (UR/lt) 10^{-6} 1.7 (UR/lt) 10^{-5}	30 (24 h) 10 (1 y) RA: 10 LP: 2 0.2 (UR/lt) 10^{-6} 2 (UR/lt) 10^{-5}	-	20	5 (1 y)	GV ≤ 2 IV 10	-	-	5 (8 h)	-	-	10 (24 h)	20 (8 h)
Formaldehyde ($\mu g\ m^{-3}$)	100 (30 m)	100 (30 m)	50 (2 h) 10 (1 y) 30 (10 from 2023) RA: 100 LP: 10	120	120 (30 m) 10 (1 y) 1.2 (LP)	100 (30 m)	GV10 (30 m) IV100 (30 m)	50	100 (30 m) 60 (24 h)	100 (8 h)	100 (30 m)	100	50 (24 h)	100 (8 h)
CO ($mg\ m^{-3}$)	100 (15 m) 60 (30 m) 30 (1 h) 10 (8 h)	100 (15 m) 35 (1 h) 10 (8 h) 7 (24 h)	100 (15 m) 60 (30 m) 30 (1 h) 10 (8 h)	1.5 (8 h) RWI 6 (30 m) RWI 60 (30 m) RWII 15 (8 h) RWII	100 (15 m) 60 (30 m) 30 (1 h) 10 (8 h)	100 (15 m) 60 (30 m) 30 (1 h) 10 (8 h)	GV 5.7 (24 h) IV 30 (1 h)	8	–	10 (8 h)	25 (1 h) 10 (8 h)	10	25 (1 h)	10 (8 h)
NO_2 ($\mu g\ m^{-3}$)	200 (1 h) 40 (1 y)	200 (1 h) 40 (1 y)	200 (1 h) 40 (1 y)	350 (30 m) RWII 60 (7 d) RWII	200 (1 h) 40 (1 y)	300 (1 h) 40 (1 y)	GV 135 (1 h) IV 200 (1 h)	-	-	-	200 (1 h) 100 (24 h)	-	-	-
Naphthalene ($\mu g\ m^{-3}$)	-	10 (1 y)	10 (1 y)	20 (7 d) RWI 200 (7 d) RWII	25	-	-	-	-	-	-	-	100 (24 h)	150 (8 h)
Styrene ($\mu g\ m^{-3}$)	260 (7 d)70 (30 m)	-	-	30 (7 d) RWI 300 (7 d) RWII	900	-	-	1	40 (7 d) 10 (1 h)	-	-	-	20 (24 h)	30 (8 h)
PAHs (as BaP) [b] ($ng\ m^{-3}$)	0.012 ng m^{-3} (UR/lt) 10^{-6} 0.12 ng m^{-3} (UR/lt) 10^{-5}	0.012 ng m^{-3} (UR/lt) 10^{-6} 0.12 ng m^{-3} (UR/lt) 10^{-5}	-	-	1.2	0.25 (1 y)	-	-	-	-	-	-	-	-
Tetrachloroethylene ($\mu g\ m^{-3}$)	250 (1 y) 8000 (30 m)	250 (1 y)	1380 (1–14 d) 250 (1 y) RV: 250 LP: 250	1 (7 d)	250	-	100	-	250 (7 d)	250 (8 h)	-	-	-	-
Trichloroethylene [b] ($\mu g\ m^{-3}$)	2.3 $\mu g\ m^{-3}$ (UR/lf) 10^{-6} 23 $\mu g\ m^{-3}$ (UR/lf) 10^{-5}	2.3 $\mu g\ m^{-3}$ (UR/lf) 10^{-6} 23 $\mu g\ m^{-3}$ (UR/lf) 10^{-5}	800 (14 d^{-1} y) RA: 10 RV: 2 LP: 2.0 (UR/lt) 10^{-6} 20 (UR/lt) 10^{-5}	1 (7 d)	-	-	200	-	–	25 (8 h)	-	-	150 (24 h)	200 (8 h)
Dichloromethane ($\mu g\ m^{-3}$)	3000 (24 h) 450 (7 d)	-	-	200 (24 h) RWI 2000 (24 h) RWII	200 (1 y)	-	-	-	-	-	-	-	-	-

TABELLA 1. *Cont.*

Contaminant (Unit of Measurement)	WHO Guidelines (Outdoor [a])	WHO Guidelines (Indoor [a])	France	Germany	Netherlands	United Kingdom	Belgium (Flanders)	Finland [c]	Austria	Portugal	Norway	Lithuania	Poland (Residential)	Poland (Public Offices)
Toluene ($\mu g\ m^{-3}$)	260 (7 d) 1000 (30 m)	-	-	300 (1–14 d) RWI 3000 (1–14 d) RWII	200 (1 y)	-	260	-	75 (1 h)	250 (8 h)	-	-	200 (24 h)	250 (8 h)
Total VOCs ($\mu g\ m^{-3}$)	-	-	-	-	200 (1 y)	-	200	-	-	600 (8 h)	400	600	400	-
PM_{10} ($\mu g\ m^{-3}$)	50 (24 h) 20 (1 y)	-	50 (24 h) 20 (1 y) RA: 75 LP: 15	-	50 (24 h) 20 (1 y)	-	40 (24 h)	50	-	50 (8 h)	90 (8 h)	100	90 (8 h)	-
PM_{25} ($\mu g\ m^{-3}$)	25 (24 h) 10 (1 y)	-	25 (24 h) 10 (1 y) RA: 50 LP: 10	25 (24 h)	25 (24 h) 10 (1 y)	-	15 (1 y)	-	-	25 (8 h)	40 (8 h)	-	40 (8 h)	-

[a] the indoor air quality guide values indicate the concentration levels in the air of the pollutants, associated with the exposure times, to which adverse health effects are not expected, in regards to non-carcinogenic pollutants; [b] the upper-bound excess lifetime cancer risk estimated to result from continuous exposure to an agent at a concentration of 1 $\mu g\ m^{-3}$ in air (or 1 $\mu g\ L^{-1}$ in water); [c] the guide values for indoor environments apply to buildings that are occupied for at least six months and where the ventilation system is kept constantly on. Abbreviations: UR unit risk; lt lifetime; RA rapid action; LP long period; RW I (all-day use) and RW II (danger threshold) German guide values (Richtwert); GV guideline value; IV intervention value; RV reference value; PAHs Polycyclic Aromatic Hydrocarbons; BaP Benzo[a]pyrene; VOCs Volatile Organic Compounds; y year; d day; h hour; m minute.

A determinare l'IAQ non è soltanto la concentrazione degli inquinanti, ma anche la temperatura e l'umidità raggiunta all'interno della struttura Parlando di umidità, questa influisce sulla formazione e proliferazione di agenti biologici (funghi, muffe, …) in aria, sulle superfici e all'interno dei materiali presenti. Agisce fortemente sull'aggregazione di particelle solide, sulla formazione e sulla dimensione degli aerosol, favorisce inoltre la presentazione di irritazione agli occhi. Per la temperatura bisogna evidenziare che influenza sia lo sviluppo di agenti microbiologici che di emissioni di COV. Eventuali superfici calde possono generare dei moti convettivi locali tali da risollevare e disperdere sia particelle sia agenti microbiologici. Le stesse superfici calde possono favorire reazioni chimiche dando origine a prodotti secondari. L'innalzamento della temperatura è legato in maniera direttamente proporzionale alla velocità di emissione di alcuni inquinanti. L'attenzione per questi parametri deve essere ben intesa e integrata nella definizione di IAQ.

Queste linee guida hanno fornito un'indicazione e uno stimolo a stilare un riferimento legislativo nazionale, e così è stato per vari paesi europei. In Italia, tuttavia, non esiste ancora un vero e proprio riferimento legale all'IAQ. A muovere i primi passi verso il riconoscimento del problema è stato l'**ISS** (Istituto Superiore di Sanità) che nel 2010 ha formato il Gruppo di Studio composto da esperti del settore. Tale organo dalla fondazione a oggi ha fornito dettagliate documentazioni (chiamati **Rapporti ISTISAN**) su varie tematiche sensibili all'IAQ, aiutando la diffusione della consapevolezza pubblica.

Un termine fondamentale per determinare l'IAQ è la concentrazione dei gas nell'aria, più precisamente quello che interessa è l'esposizione. È evidente che in un ambiente indoor non è tanto la concentrazione di un gas nell'aria a essere dannosa, ma il tempo di esposizione ad essa; soprattutto se si parla di un luogo confinato in cui si trascorre gran parte della giornata. Ricordando che non esiste una normativa italiana al riguardo, verranno usati come appunto i valori-limite forniti dalla documentazione medica e scientifica (facendo dove possibile riferimento alle linee guida della WHO). La misura della concentrazione viene resa possibile comunicando con il set di sensori disponibili per il progetto, i quali via trasmissione seriale forniscono i dati espressi in ppm o in g/m3, a seconda delle specifiche delle periferiche. La concentrazione può essere espressa in ppm o g/m3 , la conversione dall'una all'altra è immediata se si conosce la massa molecolare del gas target:

$$\text{concentrazione (mg/m3)} = 0.0409 \text{ x concentrazione (ppm) x massa molecolare}$$

Un' ulteriore problematica legata all'IAQ, oltre alla salute pubblica, è la protezione dei siti culturali, in particolare delle opere contenute nei musei o nelle librerie. I beni culturali sono sensibili al microclima a cui sono soggetti: parametri come le

sorgenti di luce, l'umidità e la temperatura sono decisivi per il corretto mantenimento e conservazione. Dunque, anche centri di questo genere hanno interesse nel monitoraggio dell'IAQ, tant'è che anche in tale ambito esiste un vasto riferimento normativo sviluppato da organizzazioni come UNESCO e WHO, ma anche il Ministero della Cultura italiano.

, lo scopo del progetto è quello di fornire uno strumento sensibile alla qualità dell'aria indoor, riuscire a monitorarne gli sviluppi e accedere via Wi-Fi ai dati in maniera diretta ed esaustiva. Confrontando i dati con i valori-limite si possono prevedere eventuali complicazioni alla salute dovute all'inalazione di gas presenti indoor e adottare strategie di risanamento. Risulta quindi un buon approccio per la prevenzione e per la raccolta dati, fornendo a un possibile utente un prospetto di IAQ. Possibili applicazioni vedono scenari come: residenza, la classe di un edificio scolastico, un ufficio, una stanza ospedaliera e simili.

Di seguito sono riportati dei valori di concentrazione ammissibile e non ammissibile, per i vari gas che il sensore IoT riesce a stimare, con una breve panoramica dei sintomi più comuni (**N.B.** : i livelli descritti sono il risultato di esperimenti molto eterogenei, perciò ci si può aspettare una reazione diversa e soggettiva, per es. nel caso dei soggetti più vulnerabili):

Acido solfidrico (H2S)[4] : le linee guida WHO riportano un valore di 1800 mg/m3 (8h) (-> 687.012 ppm) . Effetti sulla salute: organi bersaglio dell'inalazione di H2S sono il sistema nervoso centrale e il tratto respiratorio. Principali sintomi percepiti: vertigini, mal di testa, nausea, tosse.

Acetone[6] : il limite per essere percepito a livello olfattivo (odour threshold) è posto a 13.9 mg/m3 . I limiti all'esposizione sono 200 ppm (60 minuti) (-> 475.094 mg/m3) , 200 ppm (8-h). Danni permanenti alla salute sono accertati per una concentrazione di 3200ppm (1h) , 950 (8h) (-> 2256.698 mg/m3) . Effetti sulla salute: breve esposizioni sono causa di irritazione agli occhi e al tratto respiratorio. Lunghe esposizioni portano a secchezza della pelle e spossatezza.

TVOCs , total volatile organic compounds : basandosi sulle linee guida di Portogallo e Olanda il limite alle concentrazioni risulta 600 µg/m3 (8h) (-> 200ppm) , 200 µg/m3 (1y) (-> 0.06ppm) . Effetti sulla salute: livelli di esposizione inaccettabili superiori ai 3 mg/m3 sono causa di irritazione agli occhi, naso, gola, infezioni alle vie respiratorie e reazioni allergiche.

Ammoniaca (NH3)[5] : una concentrazione pari a 30 ppm (8h) (-> 21 mg/m3) può essere riferita come al limite poiché a questi livelli si può incorrere in una leggera irritazione cutanea, valori superiori ad es. 100ppm (8h) / 220 ppm (10m) (-> 70 mg/m3 / 153 mg/m3) è causa di irritazione agli occhi e alla gola, nonché

alle vie respiratorie. Effetti sulla salute: principali cause dovute a esposizioni elevate sono irritazione agli occhi, alle vie respiratorie e in casi estremi ulcerazioni.

Etanolo (C2H6O)[7] : odour threshold 80 ppm. Limite accettabile per la salute risulta essere 1000ppm(8h) (-> 1884.263 mg/m3). Effetti sulla salute: irritazione a occhi e pelle, spossatezza e mal di testa.

Anidride carbonica (CO2)[8] : la soglia di esposizione è posta a 5.000 ppm (-> 9000 mg/m3) per un'esposizione prolungata (8h) e a 30.000 ppm (-> 54000 mg/m3) per esposizione breve (15m) . Effetti sulla salute : incoscienza, alterazione del metabolismo e in casi estremi asfissia.

PM2.5 : per il particolato ci si affida alle linee guida WHO relativamente ai limiti outdoor, questo perché le concentrazioni indoor risultano essere sempre minori o uguali ai valori dell'esterno (in genere non è vero, ma per il particolato è diverso perché l'origine di tale è prevalentemente outdoor). Il limite è posto a 25 ug/m3 in un arco di 24h .

PM10 : per il particolato ci si affida alle linee guida WHO relativamente ai limiti outdoor, questo perché le concentrazioni indoor risultano essere sempre minori o uguali ai valori dell'esterno (in genere non è vero, ma per il particolato è diverso perché l'origine di tale è prevalentemente outdoor). Il limite è posto a 50 ug/m3 in un arco di 24h .

2. ARCHITETTURA E HARDWARE

La realizzazione del sensore IoT ha richiesto lo studio dei diversi sensori e relativo interfacciamento, per l'ottimizzazione dello spazio e delle periferiche delle boards. L'idea implementata prevede l'utilizzo della Teensy4.0 come board di controllo della sensoristica e della comunicazione e la HUZZAH8266 come board di supporto wireless. Nella realizzazione del prototipo è stato necessario effettuare saldature a stagno ed attingere alle conoscenze di elettronica ed elettrotecnica.

Nella figura sottostante il prototipo completato.

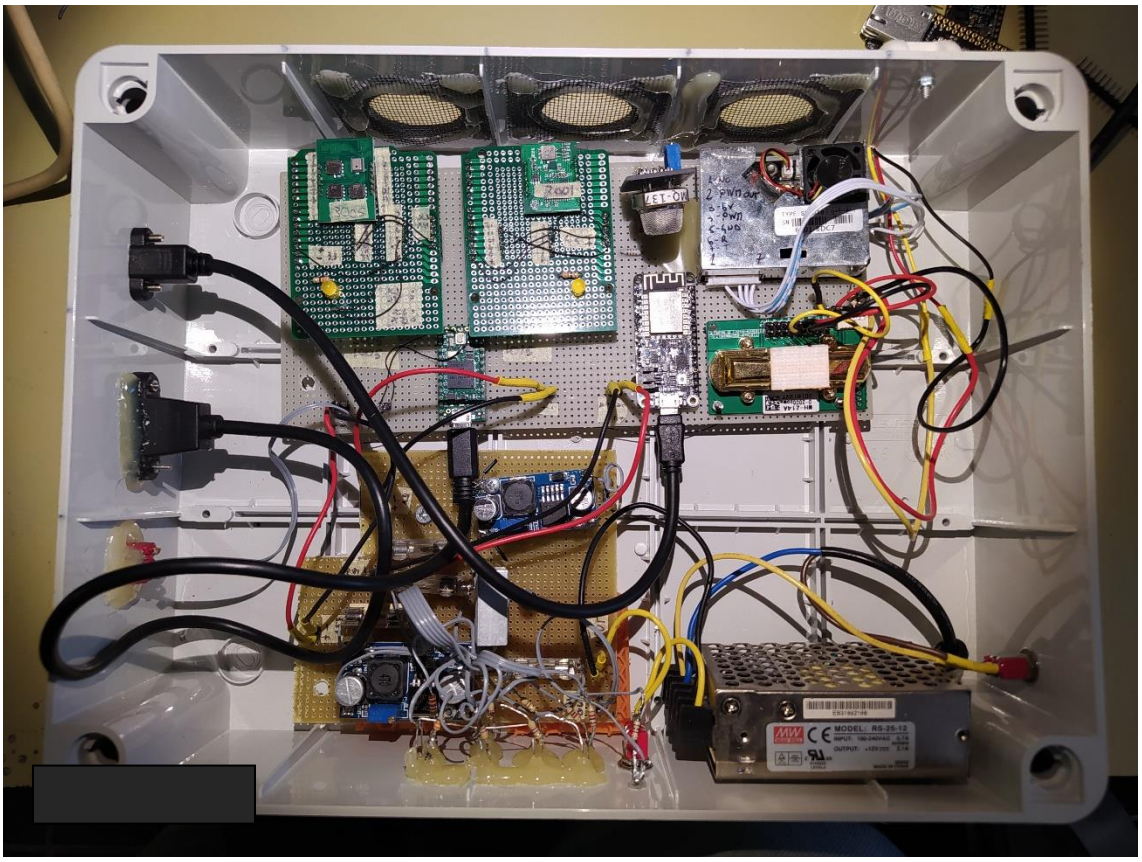


Figura 2.1. Interno del sensore IoT.

2.1. BOARDS

Il progetto prevede l'utilizzo di due board, la **Teensy4.0** e la **HUZZAH32**, sviluppate rispettivamente da PJRC e da Adafruit. La prima viene utilizzata come master device, dal momento che vigila sullo scambio dati tra sensoristica e modulo WiFi. Ciascun sensore via interfaccia seriale o analogica

comunica con la Teensy4.0 fornendo i corrispettivi valori di concentrazione, queste informazioni vengono poi incapsulate in un pacchetto dati (sostanzialmente un array di double types) che contiene valori espressi in ppm, mg/m3 o Celsius o RH%, poi sarà precisato nell'interfaccia di canale ThingSpeak.

sensore	gas	indice array
M1005	Etanolo	[0]
M1002	NH3	[1]
M1015Y	Acetone	[2]
MD3005	NH3	[3]
MD3005	TVOC	[4]
MD3005	H2S	[5]
SDS018	PM2.5	[6]
SDS018	PM10	[7]
MHZ14	CO2	[8]
MQ137	NH3	[9]
DHT22	Temperatura	[10]
DHT22	Umidità Relativa	[11]

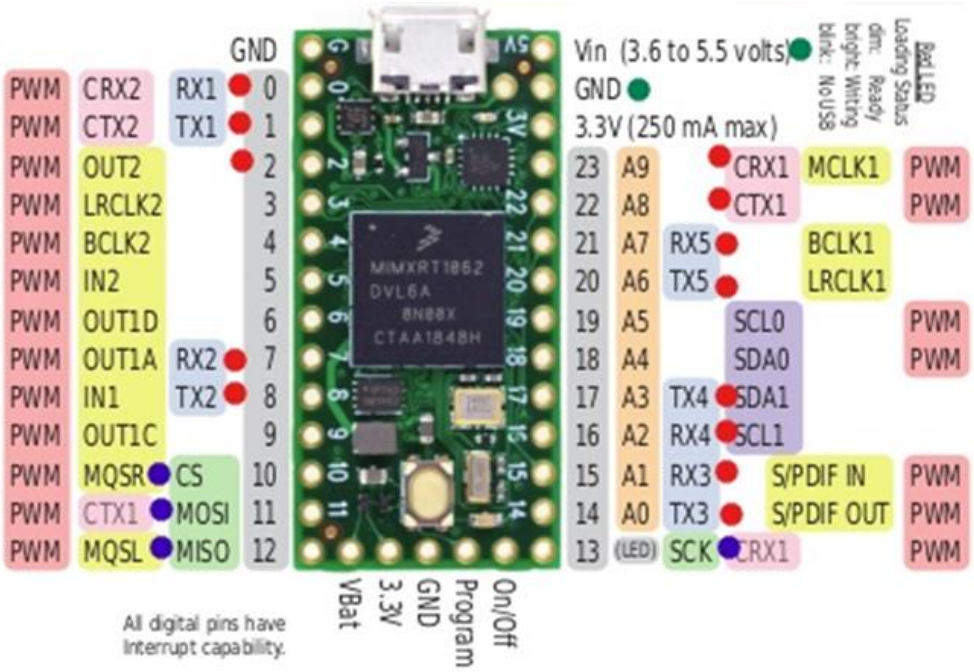
Il frame viene poi inviato via SPI alla HUZZAH, il cui scopo è quello di fornire feedback sulla ricezione e immettere i dati nel canale WiFi.

2.1.1 TEENSY4.0

La board TEENSY mette a disposizione caratteristiche uniche nel suo genere che sono state sfruttate per il progetto. Fa affidamento su un processore ARM CortexM7 che lavora a 600 MHz, dispone di 40 digital IO pins e 14 analog pins, può interfacciarsi via seriale tramite 7 porte UART, 3 SPI e 3 I2C. In particolare, l'alta densità di porte seriali è risultata ottimale per interfacciare i sensori, evitando di attrezzarsi di un UART Expander, il quale avrebbe complicato l'architettura.

Nello specifico cinque delle sette porte UART sono state utilizzate (sono state ignorate quelle sul retro), i pin SPI sono stati utilizzati per comunicare con la HUZZAH32 e tre pin analogici sono usati per il l'acquisizione di segnali analoci. Riguardo l'alimentazione la board necessita di 5V in ingresso, che tipicamente vengono forniti dall'USB (via connettore microUSB), a meno che non si intervenga fisicamente sulla scheda, separando la linea di alimentazione VIN dalla USB. Nel fare ciò è stato necessario l'utilizzo del multimetro per accertarsi di aver tagliato correttamente le connessioni evitando dunque un flow back di potenza sul calcolatore. Questo lavoro ha portato ad avere l'alimentazione esclusivamente da VIN, mentre la porta USB è dedicata solo al debug del codice. All'interno della board è installato

un regolatore di tensione a 3.3V, tensione a cui lavora il processore e da cui si può decidere di alimentare altri dispositivi esterni. Il fatto che la logica interna della board sia a 3.3V suggerisce un'intolleranza dei pin ai 5V, confermata dalla documentazione, è stato dunque necessario accertarsi di non danneggiare la TEENSY4.0 per sovratensioni. Per quanto concerne la programmazione, la TEENSY4.0 è pensata per lavorare in ambiente Arduino che mette a disposizione numerose librerie gratuite; inoltre è necessario disporre del bootloader, che prende il nome di Teensy Loader, il quale viene automaticamente avviato da Arduino in fase di upload.



Il pinout viene riportato nella pagina successiva per maggior chiarezza. (in rosso sono state evidenziate le porte per comunicare con i sensori, in blu le porte per il collegamento alla HUZZAH, in verde alimentazione e ground).

Figura 2.2. Pinout Teensy4.0 Board.

2.1.2 HUZZAH8266

Mentre l'abbondanza di interfacce e GPIO rendono la Teensy4.0 idea per l'acquisizione di datiresta di fatto l'assenza della componente IoT, rendendo necessario l'acquisto e l'integrazione di un device ulteriore che possa colmare questa mancanza. Per questo ruolo è stata utilizzata la HUZZAH8266, che integra la compattezza della famiglia Feather con un modulo WiFi targato Espressif ben noto nella comunità IoT, ovvero ESP8266. La board dispone di interfaccia seriale per il debug con il calcolatore, che viene condivisa anche per alimentare via microUSB. Visto che il sensore IoT deve essere portatile, l'alimentazione viene fornita dal regolatore a 3.3V presente nella sezione di alimentazione. Sono stati usati i pin SPI proprio per la comunicazione già citata in stile master-slave con la

Teensy4.0 . Anche in questo caso la programmazione è completamente supportata da Arduino e dalle varie librerie gratuite.

Per completezza viene riportato il pinout della board.

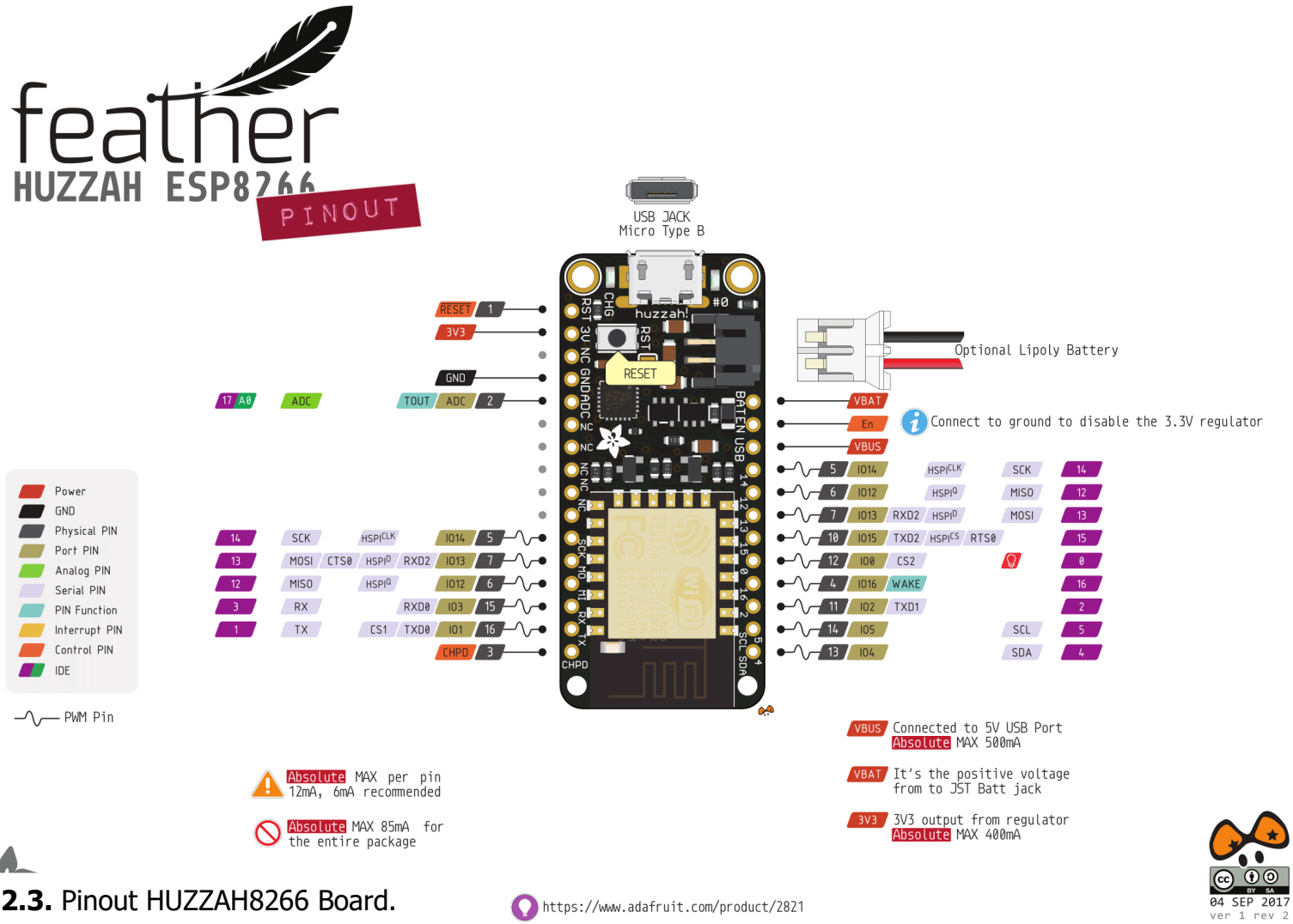


Figura 2.3. Pinout HUZZAH8266 Board.

2.2 SENSORI

Alla base del progetto ci sono i sensori. Sono interfacciati al master device (Teensy 4.0) attraverso diversi protocolli, con lo scopo di collezionare i parametri per la valutazione IAQ e inviarli - - al modulo WiFi per arrivare al destinatario finale, il canale ThingSpeak.

Sono stati utilizzati otto sensori differenti, di produttori diversi e con periferiche diverse, per cui è bene descrivere nello specifico features e capacità dei dispositivi.

2.2.1 DHT22[9] (o AM2302)

Sensore usato per rilevare la temperatura (°C) e l'umidità (RH) nell'indoor. Adotta una comunicazione Single Wire che richiede un solo filo per dialogare con il master. Oltre ai pin di alimentazione e ground, ce n'è un quarto che però non ha alcuna funzione e viene lasciato floating. Questi quattro pin rendono il dispositivo molto compatto, con un piccolo "costo" nell'implementazione dello stack protocollare. Per ambiente Arduino, ciò è semplificato dalla presenza di librerie custom pensate proprio per questo genere di dispositivi. Il consumo di corrente è moderato considerando che a regime, cioè mentre trasmette e riceve dati, può richiedere max. 1.5mA . Va inteso però che gran parte del tempo di esecuzione il dispositivo lo passa in standbymode che è una modalità basso consumo, con assorbimento dell'ordine delle decine di microAmpere. La tensione di alimentazione tipica è di 5V.

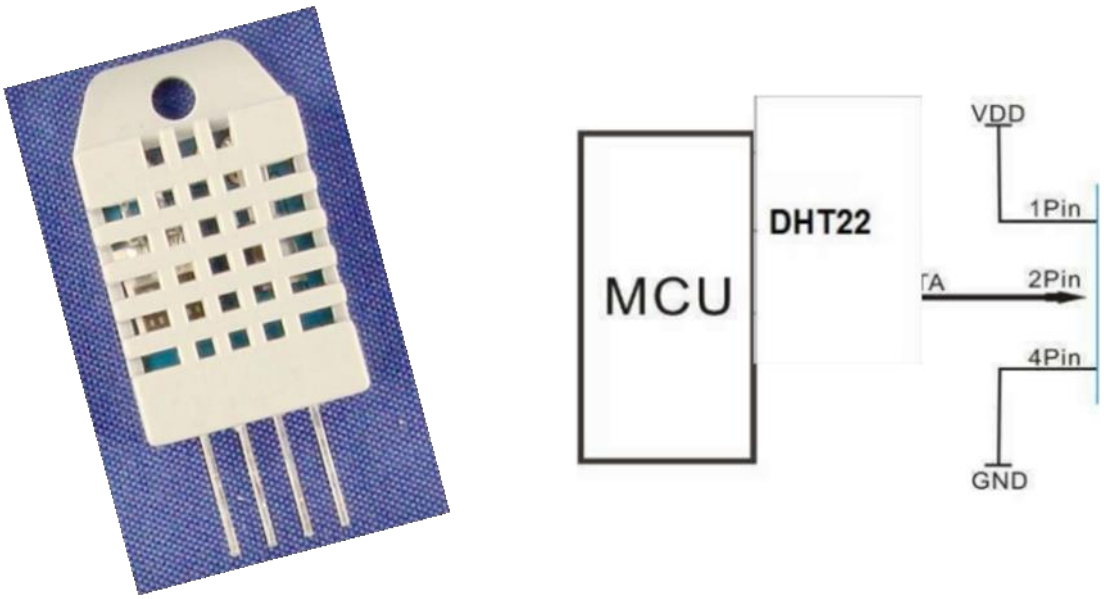


Figura 2.4. DHT22 foto e pinout.

2.2.2 MD3005[14]

Sensore capace di rilevare le concentrazioni di 3 diversi inquinanti gassosi: ammoniaca, componente organico volatile (TVOC) e acido solfidrico. Il dispositivo è alimentato a 3.3V sul pin VCC_3.3V e collegato a massa via il pin GND. I dati sono trasmessi in active upload (ogni 3 secondi viene inviato un pacchetto dati) tramite l'interfaccia UART disposta sui pin UART_RXD e UART_TXD. Questi vengono collegati rispettivamente ai pin 17 e 16, corrispondenti alla porta seriale 4. Sul datasheet vengono fornite istruzioni ulteriori riguardo alle opzioni di collegamento (baudrate e polarizzazione) e al formato dati. L'algoritmo per estrapolare i dati delle concentrazioni (in ppm) è descritto anch'esso nel datasheet e viene implementato in codice nell'ambiente Arduino.

Figura 2.5. MD3005 foto.

2.2.3 M1005[11]

Il dispositivo 1005 è sensibile alla presenza dell'etanolo nell'aria ed è in grado di fornire la sua concentrazione (in ppm). Il sensore è alimentato a 3.3V tramite il pin VCC_3.3V e collegato a massa tramite il pin GND. I dati vengono ricevuti via seriale utilizzando i pin UART_RX e UART_TX, i quali vengono corrispettivamente connessi ai pin della Teensy n.0 e n.1 (ovvero, UART1). Nel datasheet vengono fornite indicazioni riguardo i parametri di collegamento (baudrate e polarizzazione) e al formato dati. L'algoritmo per estrapolare i dati viene anch'esso specificato nel datasheet e viene implementato a sua volta in linguaggio Arduino (per riferimenti all'algoritmo vedere pagina 30).

Figura 2.6. M1005 foto.

2.2.4 M1015Y[12]

Per rilevare la presenza di acetone viene utilizzato il sensore M1015Y. È un dispositivo miniaturizzato basato su una combinazione di tecnologie (MEMS, thin film, thick film e metal packaging technologies) che gli permettono di fornire output digitale. Tra le varie proprietà del dispositivo si sottolinea la lunga durata prevista (~ 5 anni), la compensazione della temperatura integrata e una stabilità a lungo termine. I pin utilizzati nel progetto riguardano l'alimentazione (3V), la massa (G) e il protocollo UART (RX e TX). Nel datasheet vengono riportate le caratteristiche principali riguardo la comunicazione seriale : baudrate a 9600, configurazione 8N1 e dettagli sul frame. Per estrarre i dati non è sufficiente leggere i byte in successione, ma occorre applicare un semplice algoritmo che ordini e sommi adeguatamente i valori (per riferimenti all'algoritmo vedere pagina 33). Per quanto riguarda l'alimentazione, il dispositivo è alimentato a 3.3V, senza particolari specifiche sul ripple.

Figura 2.7. M1015Y foto.

2.2.5 M1002[10]

Il sensore M1002 è richiesto per valutare la presenza di ammoniaca nell'aria. Esso si basa su tecnologia MEMS. Proprietà da evidenziare sono: l' "aspettativa di vita" (~ 5 anni), la stabilità e le varie uscite che è possibile implementare; in questo caso si è scelto di usare il protocollo UART. I pin che vengono usati sono: l'alimentazione (VCC_3.3V), la massa (GND) e i pin di ricezione e trasmissione UART (UART_RX e UART_TX). Sul datasheet viene descritta la configurazione per la connessione seriale: baudrate a 9600, formato 8N1 e sul formato del frame. Per estrarre i dati non è sufficiente leggere i byte in successione, ma occorre applicare un semplice algoritmo che ordini e sommi adeguatamente i valori (per riferimenti all'algoritmo vedere pagina 33). Per quanto riguarda l'alimentazione, il dispositivo è alimentato a 3.3V, senza particolari specifiche sul ripple.

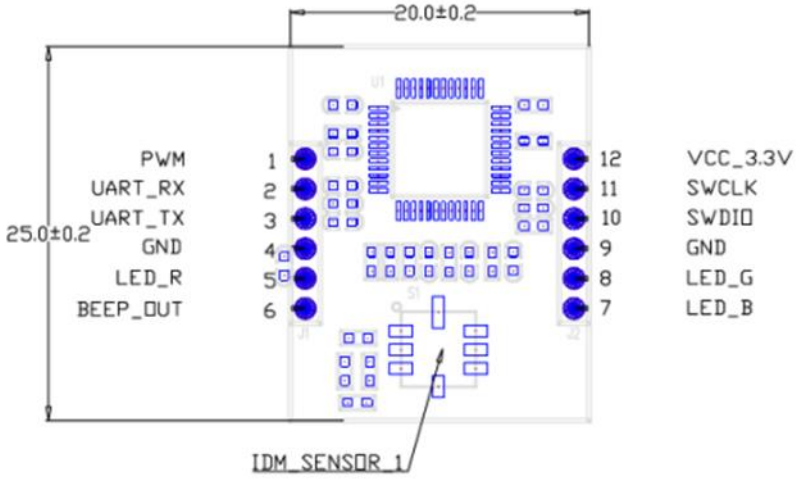


Figura 2.8. M1002 pinout.

2.2.6 LASER SDS018[13]

Per misurare le concentrazioni di particolato nell'aria viene integrato il sensore a laser SDS018. Questo sensore si basa sul principio della diffusione laser: sostanzialmente le particelle che si trovano nel raggio d'azione del laser diffondono la luce a seconda della dimensione, queste onde riflesse vengono convertite in segnali elettrici, amplificati ed elaborati. E' possibile identificare il diametro di una particella in base alla luce riflessa, permettendo la misura del particolato PM10 (a diametro di 10 μm) e PM2.5 (a diametro di 2.5 μm). Caratteristiche salienti: ventola integrata, alta risoluzione e risposta rapida (il tempo di adattamento a un cambio di scenario è inferiore a 10 secondi). Per comunicare con la Teensy4.0 vengono usati i pin UART (R e T) e la configurazione prevista: baudrate a 9600, impostazione 8N1 e packet frequency di 1Hz. Il frame inviato ha una struttura documentata nel datasheet, perciò deve essere previsto

un dettagliato framing nel codice Arduino per non corrompere i risultati, va inoltre applicato un banale algoritmo di ordinamento e somma dei byte. Per quanto concerne l'alimentazione, il dispositivo lavora a 5V e con un ripple che deve essere inferiore ai 20 mV (questa specifica ha richiesto l'utilizzo di un alimentatore lineare).

2.Interface specification

No	Name	Comment
1	NC	Not Connect
2	1μm	PWM output, see for more details
3	5V	5V Input
4	2.5μm	PWM output, see for more details
5	GND	Ground
6	R	RX of UART（TTL）@3.3V
7	T	TX of UART（TTL）@3.3V

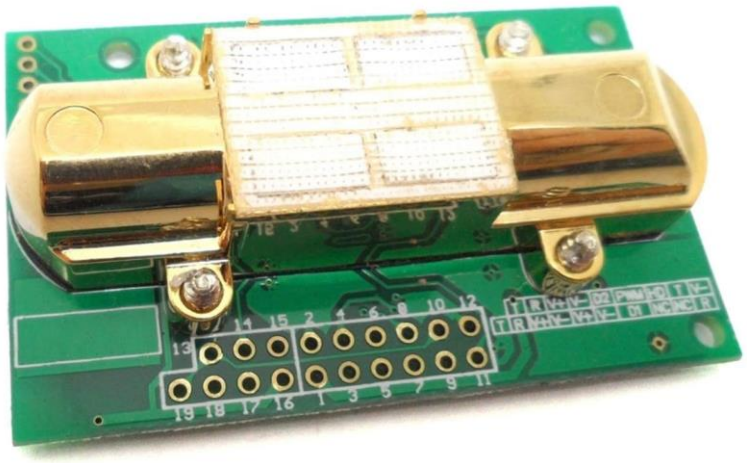
PS：The distance between each pin is 2.54mm.

Figura 2.9. SDS018 foto e interface specification.

2.2.7 MHZ14[16]

Il sensore MHZ14 è capace di rilevare la concentrazione nell'aria dell'anidride carbonica (CO2). Questo dispositivo si basa su tecnologia a infrarossi non dispersiva che sfrutta l'assorbimento della radiazione infrarossa da parte dei gas. Tra le varie features di cui dispone, si nota: la compensazione della temperatura integrata, le varie modalità di output (analogica, pwm e digitale) e l'alta risoluzione. E' prevista anche un'immunità alle concentrazioni di vapore acqueo, che spesso si presenta in concomitanza della CO2 (*sistema respiratorio animale). Per comunicare con la board, viene associata l'uscita analogica ad uno dei pin su cui lavora l'ADC interno alla Teensy4.0 (pin A9). La procedura di valutazione della concentrazione prevede la misura della tensione analogica disposta dal pin associato via media aritmetica, e successivamente il valore viene convertito in ppm come descritto dal datasheet (viene indicata una dipendenza

lineare tra tensione e concentrazione). Da notare che l'uscita massima disponibile sul pin analogico corrisponde a 2V, ciò permette quindi di evitare l'implementazione di partitori, dal momento che i pin della Teensy4.0 sono intolleranti a tensioni superiori ai 3.3V. Per quanto concerne l'alimentazione, questo sensore richiede una tensione che si aggira sui 5V, senza particolari specifiche sul ripple.



PIN	Description
Pad1/Pad15/Pad17	Vin（input voltage 4.5V~5.5V）
Pad2/Pad3/ Pad12/Pad16	GND
Pad4	Vout2 (0.4~2V)
Pad5	Vout1 (0~2.5V)
Pad6	PWM
Pad8	HD
Pad7/Pad9	NC
Pad11/Pad14/Pad18	UART（RXD） 0~3.3V input digital
Pad10/Pad13/Pad19	UART（TXD） 0~3.3V output digital

Figura 2.10. MHZ14 foto e interface specification.

2.2.8 MQ137[15]

E' previsto un altro dispositivo per rilevare le concentrazioni di NH3 e questo è il MQ137. Un sensore prettamente analogico, di cui esiste una versione montata su una board minimale (usata nel progetto) con semplici operazioni digitali implementate; mi riferisco in particolare all'uscita digitale (non usata) che permette di stabilire una concentrazione di threshold, la quale viene confrontata con il valore attuale presente nell'aria, e in caso quest'ultima fosse superiore al limite, il pin fornisce un 1 logico. Il setup del dispositivo richiede la conoscenza di alcuni parametri elettrici tipo la resistenza di uscita sulla quale si basa il calcolo dei ppm. Sulla base di questa necessità è utile analizzare lo schema circuitale:

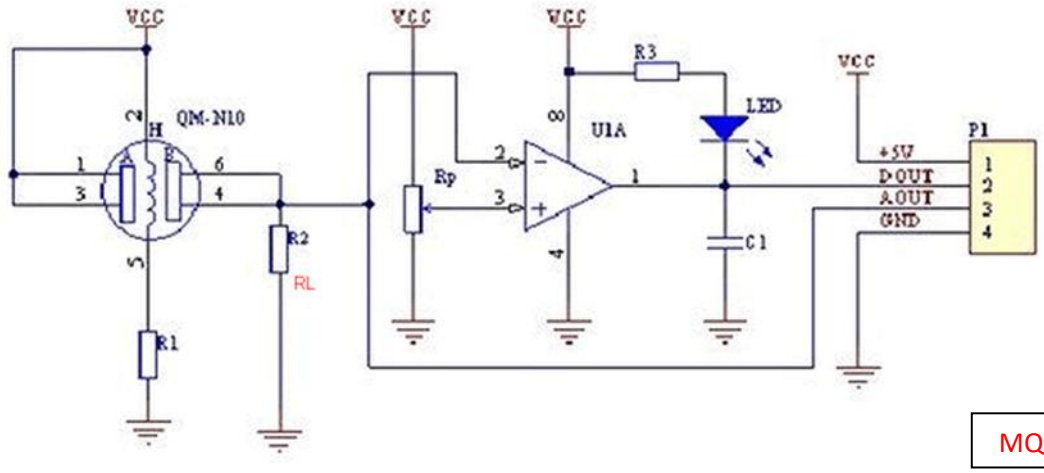


Figura 2.11. Schema circuitale MQ137

Il valore della resistenza, qui sopra denotata come RL, può esse ricavato in due modi: usando un multimetro con opzione di misura di resistenza tra i capi AOUT e GND, oppure andando a leggere il valore sul resistore SMD (surface-mount technology). Si deduce che il valore di RL è pari a 1kΩ. Per il calcolo della concentrazione di NH3 è necessario adoperare un algoritmo e alcune formule matematiche che mettono in relazione tensioni e resistenze. Innanzitutto, questo genere di dispositivi dimostra una caratteristica di sensibilità documentata nel datasheet con il grafico :

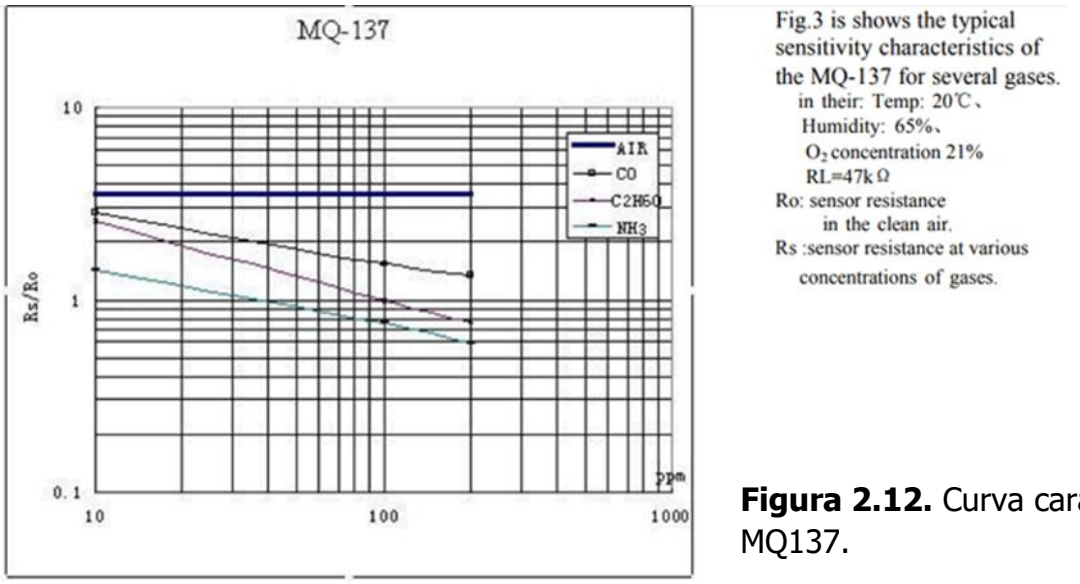


Fig.3 sensitivity characteristics of the MQ-137

Figura 2.12. Curva caratteristica MQ137.

La curva verde mette in relazione il rapporto Rs/Ro con la concentrazione di NH3, mentre quella blu dimostra che il rapporto è costante in condizioni di aria pulita (all'incirca vale 3.6). Per il calcolo di Rs esiste una formula:

$$Rs = \left(\left(\frac{Vcc}{Vrl}\right) - 1\right) * RL$$

Dove Vcc sono i 5V di alimentazione, Vrl è la tensione ai capi di RL, RL è la resistenza in ohm. Quindi per calibrare il dispositivo va posto alla prima misurazione in un ambiente che si può presumere pulito, poi viene calcolato il valore di Rs in ohm tramite la formula, dopodiché può essere determinato Ro pari a Rs/3.6 . Questi passaggi sono stati svolti e si è riscontrato un valore di Ro pari a 10 kΩ . Conoscendo Ro e misurando via ADC la Rs è possibile determinare volta per volta il rapporto e quindi la concentrazione. È stato introdotto anche un partitore di tensione all'uscita di RL per evitare picchi di tensione di 5V che potrebbero danneggiare la Teensy4.0 .

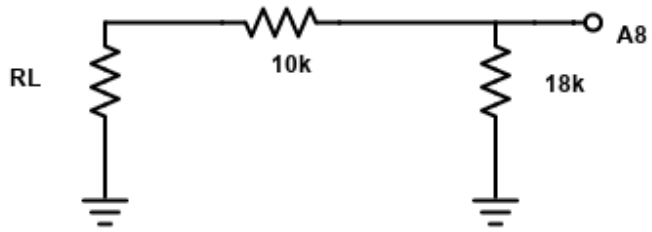


Figura 2.13. Partitore su uscita MQ137.

2.3 ALIMENTAZIONE

Oltre alla sezione di sensoristica e la parte dedicata alle board, si trova il modulo di alimentazione. All'interno del progetto sono richieste diverse specifiche di alimentazione: molti sensori necessitano di una tensione in ingresso a 5V, altri a 3.3V, il laser richiede 5V ma con un ripple massimo di 20mV. Non solo i sensori vanno alimentati, ma anche le board, che nativamente supportano i 5V da USB, e che internamente dispongono di regolatori di tensione per poter operare con logica a 3.3V. Ne risulta, però, che la potenza proveniente dalla porta USB e fornita dalla board tramite pin appositi, non è sufficiente a soddisfare il consumo di tutte le periferiche. La soluzione proposta prevede l'utilizzo di un alimentatore a commutazione (**RS-25-12**) che viene connesso alla rete elettrica e fornisce in uscita 12V e una corrente massima di 2.1A, che soddisfa pienamente il fabbisogno energetico del prototipo. Essendo l'alimentatore realizzato in tecnologia switching, garantisce delle misure e un peso ridotti, rispetto all'equivalente lineare, favorendo compattezza, portabilità e buon rapporto potenza/peso. Principalmente si basa su un transistor di potenza per generare una tensione ad alta frequenza che viene poi trattata da un trasformatore e da un filtro per rimuovere la componente AC e il rumore. Mentre un alimentatore lineare fornisce DC filtrando la tensione alternata e usando un trasformatore. A questo punto si dispone di una tensione a 12V "sporcata" del rumore tipico degli alimentatori switching, questa va portata ai livelli dei 5V e dei 3.3V. A questo scopo sono

disposti due regolatori di tensione, uno da 5V e l'altro da 3.3V. Questi regolatori sono anch'essi realizzati in switching pertanto possono presentare un ripple massimo di 100mV, che potrebbe danneggiare il laser SDS018. Perciò è previsto un ulteriore dispositivo, il regolatore lineare **LM7805**. Si tratta di un elemento che genera 5V a basso ripple come desiderato, ma che genera molto calore e quindi necessita di un dissipatore dedicato e di una buona circolazione d'aria per il raffreddamento. Dunque, viene installata anche una ventola con la doppia funzione di limitare l'innalzamento di temperatura e favorire l'afflusso dell'aria locale per consentire una rilevazione ottimale. Questa serie di implicazioni hanno portato anche alla conclusione di localizzare esternamente il dispositivo per l'ottenimento della temperatura e dell'umidità, il DHT22, per non corrompere i dati con i fenomeni interni al sensore IoT. E' stata disposta inoltre un'area dedicata ai fusibili e alla segnalazione LED. Quest'ultimi sono posti a valle (indicati come POST) o a monte (PRE) dei fusibili e permettono di individuare il punto di failure dell'alimentazione, garantendo continuità del sistema e permettendo un'azione mirata.

Nell'immagine qui sotto il sensore IoT con evidenziata in rosso la sezione di alimentazione sopra descritta.

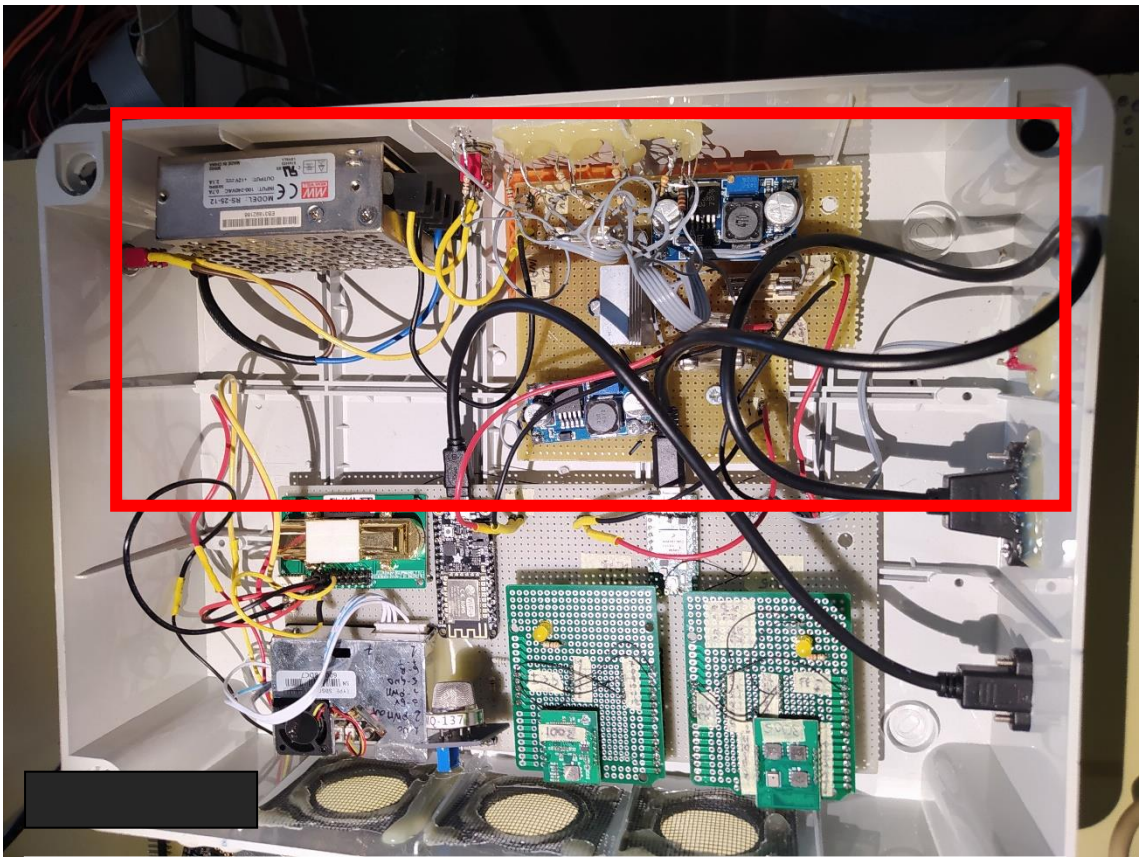


Figura 2.14. Prototipo con evidenziata la sezione di alimentazione.

3. CODICE

Entrambe le board sono compatibili con Arduino, perciò i firmware sono stati scritti in tale ambiente e il debug è possibile tramite il Monitor Seriale direttamente integrato nella piattaforma. La Teensy4.0 fa affidamento su Teensyduino, un add-on pensato esclusivamente per Teensy, che mette a disposizione una vasta varietà di librerie ottimizzate proprio per la famiglia Teensy. Per quanto riguarda Huzzah8266 è necessario inizializzare il Board Manager di Arduino adeguatamente, mentre l'intera documentazione, su librerie e non solo, è disponibile al link https://arduino-esp8266.readthedocs.io/en/3.0.2/ , invece alla pagina github https://github.com/esp8266/Arduino è disponibile il codice sorgente degli elementi delle librerie (utile se si vuole scendere a livello più basso di astrazione) e vari esempi molto utili. Di seguito verranno riportati i due sketches implementati con alcuni commenti laddove sarà necessario chiarire dei passaggi.

Inclusione di librerie utili per operazioni sui byte, sui moduli ADC e SPI della Teensy4.0, sul sensore DHT22.

```
//#include
#include <math.h>
#include <ADC.h>
#include <DHT.h>
#include <SPI.h>
```

Le seriali vengono rinominate ad indicare il collegamento hardware; alcuni pin vengono etichettati a seconda della funzione; le costanti utilizzate nell'algoritmo di MQ137 vengono definite per agevolare i calcoli (le costanti sono deducibili dal grafico fornito dal datasheet).

```
//#define
#define DHT_PIN 2
#define DHT_TYPE DHT22
#define SSPIN 10
//rinomino le seriali
#define ethanolSerial Serial1
#define ammoniaSerial Serial2
#define acetoneSerial Serial3
#define TVOCSerial Serial4
#define LASERSerial Serial5
//elementi per calcolo MQ137
#define b0 0.42
#define m0 -0.263
```

Dichiarazione di funzioni e di variabili varie usate all'interno del codice.

```
//dichiarazioni di funzioni
void collectData(double b[], int length); //packet
formatting
byte transferAndWait(byte b); //famiglia di funzioni per
SPI
void _pulseSS();
void writeData(const char * data);
String readData();
void readData(uint8_t * data);
void writeData(uint8_t * data, size_t len);

//gp variables
DHT dht(DHT_PIN, DHT_TYPE);
float temperature;//from dht (*C)
float humidity; //from dht (RH)
double datacollect[12]; // [0]ethanol, [1]nh3, [2]acetone,
[3]NH3, [4]TVOC, [5]H2S, [6]PM2.5, [7]PM10, [8]CO2, [9]NH3,
[10] temperature, [11] humidity
double Ro = 10000;
//adc
ADC *adc = new ADC(); //10 bits-resolution ; vref3.3 ;
conversion speed + sampling speed

//variabili per il datacapture
byte incomingByte;

uint8_t eth[24];
unsigned long ETH_tot;
double eth_tot ;

uint8_t ace[24];
unsigned long ACE_total;
double ace_total;

uint8_t amm[24];
unsigned long AMM_tot;
double amm_tot;

uint8_t odor[40];
unsigned long ch1_tot ;
double CH1_tot ;
unsigned long ch2_tot;
double CH2_tot;
unsigned long ch3_tot ;
double CH3_tot ;

uint8_t laser[10];
double PM25tot;
double PM10tot;
```

```
double CO2PPM;
double analogValue = 0.00;
```

Funzione di setup della board in cui:

1. vengono inizializzate le seriali con il rispettivo baudrate indicato dalla documentazione;
2. vengono configurati la maggior parte dei sensori in Q&A mode (ovvero il dato viene fornito solo in seguito a una richiesta, e non spontaneamente); il comando in questione che permette lo switch tra le modalità è 0xAA00FA; di default il sensore è in modalità active uploading (comunica la rilevazione ogni 3 secondi), quindi è necessario fornire al setup un comando di switch;
3. vengono configurati i pin con funzioni di ADC prendendo in considerazione anche velocità di conversione, bit di risoluzione e velocità di campionamento; la libreria relativa all'ADC mette a disposizione varie velocità per quanto riguarda il sampling e la conversione, ad es. per quest'ultima possiamo scegliere tra

VERY_LOW_SPEED
LOW_SPEED
MED_SPEED
HIGH_SPEED_16BITS
HIGH_SPEED
VERY_HIGH_SPEED
ADACK_2_4
ADACK_4_0
ADACK_5_2

   Per cui non avendo condizioni stringenti sotto questo punto di vista, si è scelta LOW_SPEED come soluzione sicura, la quale, secondo la documentazione, risulta essere la velocità più bassa per qualsiasi tipo di risoluzione (comunque più alta di 2MHz);*
4. vengono inizializzate altre periferiche come SPI e DHT;
5. Un altro procedimento fondamentale è la stima della Ro di MQ137; il valore di questa resistenza è fondamentale poiché rientra nella formula per il calcolo della concentrazione. Per determinare tale valore si è fatta una

* la documentazione sulla libreria ADC è disponibile sul sito web:

prima misurazione in aria pulita, dal momento che -in tale circostanza- esiste una relazione lineare con la Rs (altra resistenza che rientra nella formula) e tale Rs si può calcolare conoscendo la tensione rilevata dall'ADC (per un riferimento accurato sul metodo di rilevazione del MQ137 si rimanda a pagina 22). Questo procedimento è sufficiente eseguirlo una volta sola così da determinare la Ro per l'applicazione dell'algoritmo, perciò è commentato nel codice qui sotto.

```
//inizializzazione
void setup() {

  dht.begin();

  Serial.begin(9600);
  //Q&A mode
  ethanolSerial.begin(9600);
  ethanolSerial.write(0xAA);
  ethanolSerial.write(0x00);
  ethanolSerial.write(0xFA);
  ammoniaSerial.begin(9600);
  ammoniaSerial.write(0xAA);
  ammoniaSerial.write(0x00);
  ammoniaSerial.write(0xFA);
  acetoneSerial.begin(9600);
  acetoneSerial.write(0xAA);
  acetoneSerial.write(0x00);
  acetoneSerial.write(0xFA);
  TVOCSerial.begin(38400);
  TVOCSerial.write(0xAA);
  TVOCSerial.write(0x00);
  TVOCSerial.write(0xFA);
  LASERSerial.begin(9600);
  pinMode(A9, INPUT);
  pinMode(A8, INPUT);

  pinMode(13, OUTPUT); //status LED
  digitalWrite(13, LOW);

  //adc configuration;
  adc->adc0-
>setSamplingSpeed(ADC_SAMPLING_SPEED::LOW_SPEED);
  adc->adc0-
>setConversionSpeed(ADC_CONVERSION_SPEED::LOW_SPEED);
  adc->adc0->setAveraging(1);
  adc->adc0->recalibrate();

  //  //establishing fresh air resistance !(Alla prima
esecuzione del dispositivo si deve trovare in aria pulita)!
  //  double RS_air;
  //  double analogValue2;
```

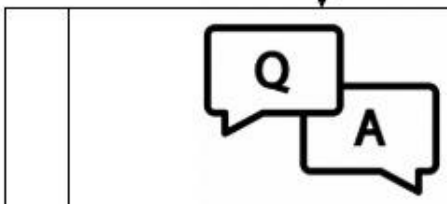
```
  //  for (int i = 0; i < 500; i++) {
  //    analogValue2 += adc->adc0->analogRead(A8);
  //  }
  //  analogValue2 = analogValue2 / 500.0; //averaged
  //  analogValue2 = (analogValue2 / 1024) * (3.3);
//analog voltage
  //  analogValue2 = analogValue2 * (14.0 / 9.0);
//considero il partitore
  //  RS_air = (((5.0) / (analogValue2)) - 1) * 1000;
  //  Ro = RS_air / 3.6; //Ro @ clean air is 10k
  //  Serial.print("Valore di Ro in aria pulita (mq137) in
ohm: ");
  //  Serial.println(Ro);

  Serial.println("Start communication....");
  delay(500);

  //setup SPI
  SPI.begin();
  pinMode(SSPIN, OUTPUT);

  delay(1000);
}
```

Successivamente al setup è presente il vero e proprio corpo del firmware, ovvero il loop. Nella funzione loop vengono messe in atto una serie di step basati su eventi che possono manifestarsi sulla Teensy4.0 o sulla HUZZAH8266. Di seguito uno schema che riproduce la macchina a stati finiti pensata per il progetto.

TEENSY4.0 FIRMWARE: gestione degli eventi

1. Si possono inviare dei comandi via seriale principalmente per il DEBUG. Comandi implementati:

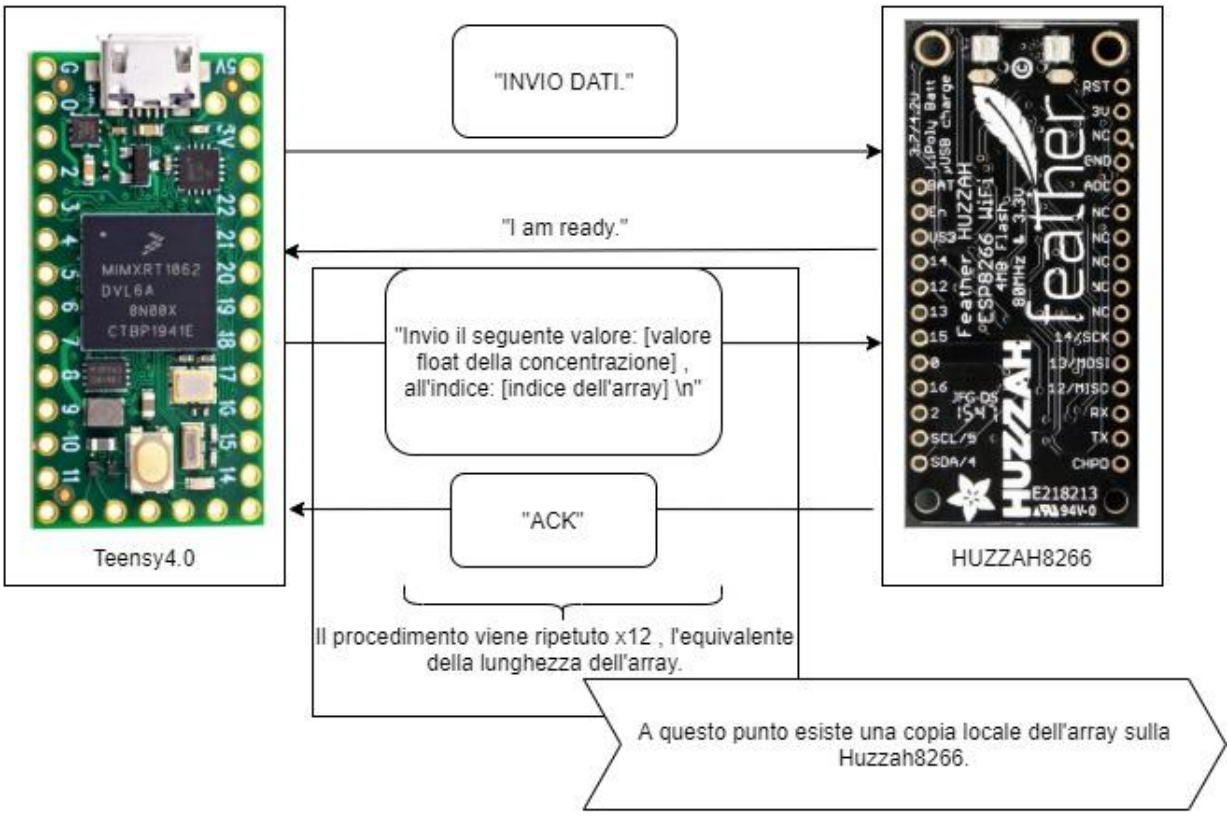
   * "huzzah reset\n"



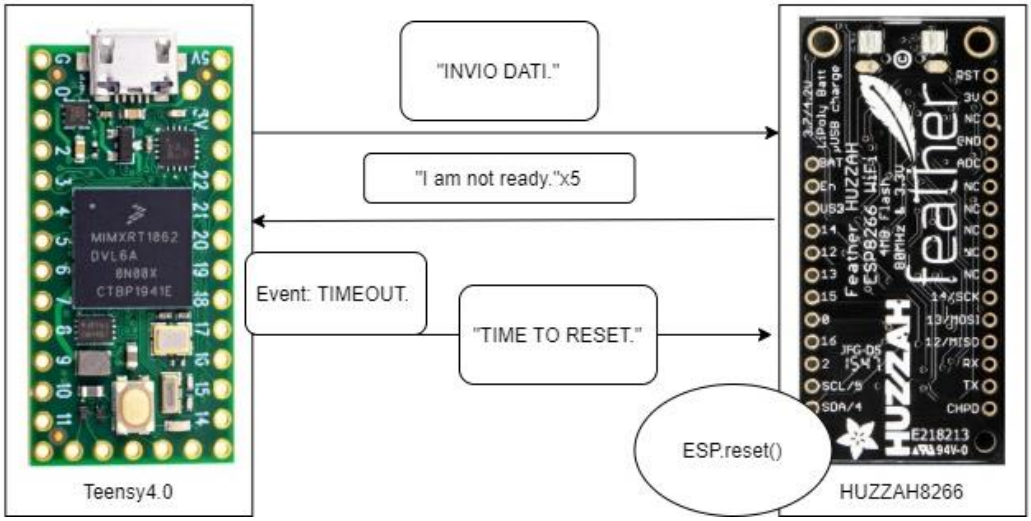
2. **DATA RETRIEVING**: tramite il meccanismo di Q&A, l'ADC e la periferica del DHT è possibile popolare l'array con i dati più recenti delle concentrazioni.

3. A questo punto l'array va condiviso con la HUZZAH8266 perciò interviene lo **stack SPI**.

CASO 1: comunicazione ideale.



CASO 2: comunicazione fallita. ⟶ *Causa principale*: errore di uploading dei dati sul channel Thingspeak e loop nella funzione di sendWiFi()



Durante il debug si è notata la possibilità di fail nel caricamento dati sul canale ThingSpeak, e ciò non è ammissibile dal momento che blocca in loop la HUZZAH8266, è pertanto necessario intervenire o manualmente via seriale fornendo il comando "huzzah reset\n" (in fase di debug) o attendere che la Teensy4.0 gestisca automaticamente l'evento.

Ripercorrendo con lo schema quindi prima si ha il data retrieving.

```
//retrieving data from sensors
 Serial.println("Collecting sensors data...");
 collectData(datacollect, 12);
 delay(180000);
```

La funzione che svolge l'attività di popolamento dell'array è **collectData()** .

```
//funzione chiamata per riempire il vettore dei dati
void collectData(double b[], int length) {

  Serial.println("Ethanol sensor:");
  ethanolSerial.write(0xAA);
  ethanolSerial.write(0x00);
  ethanolSerial.write(0xF9);
  if (ethanolSerial.available()) {
    for (int i = 0; i < 24; i++) {
      incomingByte = ethanolSerial.read();
      Serial.println(incomingByte, HEX);
      eth[i] = incomingByte;
    }
    uint8_t chk =  CheckSum(eth, sizeof(eth));
    Serial.printf("Checksum calculated: %x \n", chk);
```

```
    if (eth[0] == 0xAA && eth[23] == chk) {
      ETH_tot = (eth[21] * pow (16, 6) + eth[20] * pow(16,
4) + eth[19] * pow(16, 2) + eth[18]);
      eth_tot = ETH_tot / 1000;
      b[0] = eth_tot;
    }
    Serial.println("Concentration of ethanol is: ");
    Serial.println(b[0]);
  }
  delay(500);


  Serial.println("Ammonia sensor:");
  ammoniaSerial.write(0xAA);
  ammoniaSerial.write(0x00);
  ammoniaSerial.write(0xF9);
  if (ammoniaSerial.available()) {
    for (int i = 0; i < 24; i++) {
      incomingByte = ammoniaSerial.read();
      Serial.println(incomingByte, HEX);
      amm[i] = incomingByte;
    }
    uint8_t chk = CheckSum(amm, sizeof(amm));
    Serial.printf("Checksum calculated: %x \n", chk);
    if (amm[0] == 0xAA && amm[23] == chk) {
      AMM_tot = (amm[21] * pow (16, 6) + amm[20] * pow(16,
4) + amm[19] * pow(16, 2) + amm[18]);
      amm_tot = double(AMM_tot) / 1000;
      b[1] = amm_tot;
    }
    Serial.println("Concentration of Ammonia is: ");
    Serial.println(b[1]);
  }
  delay(500);


  Serial.println("Acetone sensor:");
  acetoneSerial.write(0xAA);
  acetoneSerial.write(0x00);
  acetoneSerial.write(0xF9);
  if (acetoneSerial.available()) {
    for (int i = 0; i < 24; i++) {
      incomingByte = acetoneSerial.read();
      Serial.println(incomingByte, HEX);
      ace[i] = incomingByte;
    }
    uint8_t chk = CheckSum(ace, sizeof(ace));
    Serial.printf("Checksum calculated: %x \n", chk);
    if (ace[0] == 0xAA && ace[23] == chk) {
      ACE_total = (ace[21] * pow (16, 6) + ace[20] *
pow(16, 4) + ace[19] * pow(16, 2) + ace[18]);
```

```
      ace_total = ACE_total / 1000;
      b[2] = ace_total;
    }
    Serial.println("Concentration of Acetone is: ");
    Serial.println(b[2]);
  }
  delay(500);


  Serial.println("Odor sensor:");
  //algorithm reset:
  //      TVOCSerial.write(0xAA);
  //      TVOCSerial.write(0x00);
  //      TVOCSerial.write(0xFE);
  //      TVOCSerial.write(0xBB);
  //  TVOCSerial.flush();
  TVOCSerial.write(0xAA);
  TVOCSerial.write(0x00);
  TVOCSerial.write(0xF9);
  if (TVOCSerial.available()) {
    for (int i = 0; i < 40 ; i++) {
      incomingByte = TVOCSerial.read();
      Serial.println(incomingByte, HEX);
      odor[i] = incomingByte;
    }
    uint8_t chk = CheckSum(odor, sizeof(odor));
    Serial.printf("Checksum calculated: %x \n", chk);
    if (odor[0] == 0xAA && chk == odor[39] && odor[38] ==
0xBB) {
      ch1_tot = (odor[23] * pow (16, 6) + odor[22] *
pow(16, 4) + odor[21] * pow(16, 2) + odor[20]);
      CH1_tot = ch1_tot / 1000;
      ch2_tot = (odor[28] * pow (16, 6) + odor[27] *
pow(16, 4) + odor[26] * pow(16, 2) + odor[25]);
      CH2_tot = ch2_tot / 1000;
      ch3_tot = (odor[32] * pow (16, 6) + odor[31] *
pow(16, 4) + odor[30] * pow(16, 2) + odor[29]);
      CH3_tot = ch3_tot / 1000;
      b[3] = CH1_tot;
      b[4] = CH2_tot;
      b[5] = CH3_tot;
    }
    Serial.println("Concentration of CHANNEL #1 (Ammonia)
is: "); //controllare i gas effettivi dei canali
    Serial.println(b[3]);

    Serial.println("Concentration of CHANNEL #2 (TVOC) is:
");
    Serial.println(b[4]);
```

```
    Serial.println("Concentration of CHANNEL #3 (H2S) is:
");
    Serial.println(b[5]);

  }
  delay(500);


  Serial.println("Laser sensor output:");
  if (LASERSerial.available()) {
    for (int i = 0; i < 10 ; i++) {
      incomingByte = LASERSerial.read();
      Serial.println(incomingByte, HEX);
      laser[i] = incomingByte;
    }
    if (laser[0] == 0xAA && laser[9] == 0xAB) {
      byte high25 = laser[3];
      byte low25 = laser[2];
      byte high10 = laser[5];
      byte low10 = laser[4];
      PM25tot = (high25 * 256 + low25) / 10;
      b[6] = PM25tot;
      PM10tot = (high10 * 256 + low10) / 10;
      b[7] = PM10tot;
    }
    Serial.println("PM2.5 concentration (ug/m3) is :");
    Serial.println(PM25tot);
    Serial.println("PM10 concentration (ug/m3) is :");
    Serial.println(PM10tot);
  }


  Serial.println("Calculating concentration of CO2:");
  for (int i = 0; i < 500; i++) {          // from 0 to
1023; serve un delay per far completare l'operazione di
conversione??
    analogValue += adc->adc0->analogRead(A9);
  }
  analogValue = analogValue / 500.0; //averaged
  analogValue = (analogValue / 1024) * (3.3);
  if (analogValue < 0.4 || analogValue > 2.1) {
    Serial.println(analogValue);
    Serial.println("Errore con il sensore CO2.");
  }
  else {
    CO2PPM = (analogValue - 0.4) / (0.0008);
    b[8] = CO2PPM;
    Serial.println("Concentration of CO2 is:");
    Serial.println(CO2PPM);
  }
```

```
  double RS_air;
  double analogValue2 = 0;
  for (int i = 0; i < 500; i++) {
    analogValue2 += adc->analogRead(A8);
  }
  analogValue2 = analogValue2 / 500.0; //averaged
  analogValue2 = (analogValue2 / 1024) * (3.3); //analog
voltage
  analogValue2 = analogValue2 * (180 / 380); //considero il
partitore
  Serial.println("Valore tensione NH3");
  Serial.println(analogValue2);
  RS_air = (((5.0) / (analogValue2)) - 1) * 1000;
  double MQtot = pow(10, ((log10(RS_air / Ro) - b0) / m0));
  b[9] = MQtot;
  Serial.println("Sensore NH3:");
  Serial.println(MQtot);


  double temperature = double(dht.readTemperature());
//celsius
  double humidity = double(dht.readHumidity()); //RH
  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Errore con DHT22.");
  }
  else {
    Serial.println("Sensore temperatura:");
    Serial.println(temperature);
    b[10] = temperature;
    Serial.println("Sensore umidità:");
    Serial.println(humidity);
    b[11] = humidity;
  }
}
```

Per i sensori in UART è stata verificata la conformità del frame tramite il valore checkup. In tal modo si è reso il codice resistente a eventuali fail di formato: se viene ricevuto un frame errato, allora verrà mantenuto l'ultimo dato valido della sessione (gli errori di questo tipo possono capitare, ma sono molto rari).

Una volta ottenuto l'array, va instaurata la comunicazione SPI con la HUZZAH8266.

```
void loop() {

  Serial.println("Write an instruction (or nothing) now via
serial, then wait 8 seconds and system will go on: ");
  delay(8000);
```

```
  //reset instruction
  String serialINSTR = "";
  if (Serial.available()) {
    while (Serial.available()) {
      char serialByte = (char)Serial.read();
      Serial.printf("Carattere in arrivo: %c \n",
serialByte);
      serialINSTR += serialByte;
    }
    if (serialINSTR == "huzzah reset\n") {
      Serial.println("Resetto la HUZZAH (possibile errore
nell'upload wifi).");
      writeData("TIME TO RESET.");
    }
  }

  //retrieving data from sensors
  Serial.println("Collecting sensors data...");
  collectData(datacollect, 12);
  delay(180000);

  int timeout = 0;

instruction:
  Serial.println("Invio istruzione.");
  writeData("INVIO DATI.");
  String receivedINSTR = readData();
  Serial.print("Risposta slave: ");
  Serial.println(receivedINSTR);
  if (!receivedINSTR.equals("I am ready.")) {
    if (timeout == 5) {
      Serial.println("Resetto la HUZZAH (possibile errore
nell'upload wifi).");
      writeData("TIME TO RESET.");
      timeout = 0;
      delay(8000);
      goto instruction;
    }
    else {
      timeout++;
      delay(5000);
      goto instruction;
    }
  }
  delay(1000);

  String ack;
  for (int m = 0; m < 12; m++) {
    Serial.printf("Invio il seguente valore: %f ,
all'indice: %d \n", datacollect[m], m);
    String dataTX = String(datacollect[m], 3);
```

```
    const char* temp = dataTX.c_str();
    writeData(temp);
    delay(500);
    ack = readData();
    delay(1000);
    while (!ack.equals("ACK")) {
      Serial.println("Devo reinviare valore all'indice:");
      Serial.println(String(m));
      writeData(temp);
      delay(500);
      ack = readData();
      delay(1000);
    }
  }


  delay(5000); //5 secondi tra una macro e l'altra
}
```

All'interno del codice sopra mostrato sono contenuti vari aspetti già presentati con il grafico della macchina a stati finiti (riferimento a pagina 29), e che trovano implementazione software, come ad es. il sistema di acknowledgment e il modulo per il reset di HUZZAH8266 in caso di fail.

Per completezza si inseriscono anche le funzioni che assistono la trasmissione e la ricezione via SPI.

```
//inviare dati di lunghezza(byte) len via SPI
void writeData(uint8_t * data, size_t len) {
  uint8_t i = 0;
  digitalWrite(10, LOW);
  SPI.transfer(0x02);
  SPI.transfer(0x00);
  while (len-- && i < 32) {
    SPI.transfer(data[i++]);
  }
  while (i++ < 32) {
    SPI.transfer(0);
  }
  digitalWrite(10, HIGH);
}

//leggere in SPI dei byte
void readData(uint8_t * data) {
  _pulseSS();
  SPI.transfer(0x03);
  SPI.transfer(0x00);
  for (uint8_t i = 0; i < 32; i++) {
    data[i] = SPI.transfer(0);
  }
```

```
  _pulseSS();
}

//leggi stringa da SPI
String readData() {
  char data[33];
  data[32] = 0;
  readData((uint8_t *)data);
  return String(data);
}

//inviare una stringa di caratteri via SPI
void writeData(const char * data) {
  writeData((uint8_t *)data, strlen(data));
}

//Slave Select pulse
void _pulseSS()
{
  digitalWrite(10, HIGH);
  delayMicroseconds(5);
  digitalWrite(10, LOW);
}

//correttezza stringa (sensor format)
uint8_t CheckSum(uint8_t *p, uint8_t len) {
  uint8_t i;
  uint32_t sum = 0;
  for (i = 0; i < len - 1; i++)
  {
    sum += p[i];
  }
  sum = sum & (0xFF);
  return sum;
}
```

Il firmware della HUZZAH8266 risulta essere molto più diretto e meno articolato, ciò accentua il ruolo di slave nell'architettura del progetto. Nella sostanza il dispositivo si deve accertare che una volta stabilita la comunicazione SPI sia in grado di capire il dato e popolare l'array locale, dopodiché via WiFi verrà recapitato il formato dati al router che permetterà l'upload sui canali ThingSpeak.

Alcune informazioni sulla piattaforma ThingSpeak. Traducendo dalla descrizione sulla relativa pagina web, "ThingSpeak è un servizio di analisi IoT che consente di **aggregare**, **visualizzare** e **analizzare** dati in real time nel cloud". Il sistema è integrato alla piattaforma MATLAB per cui i dati possono essere importati, esportati e trattati in maniera diretta e personalizzata.

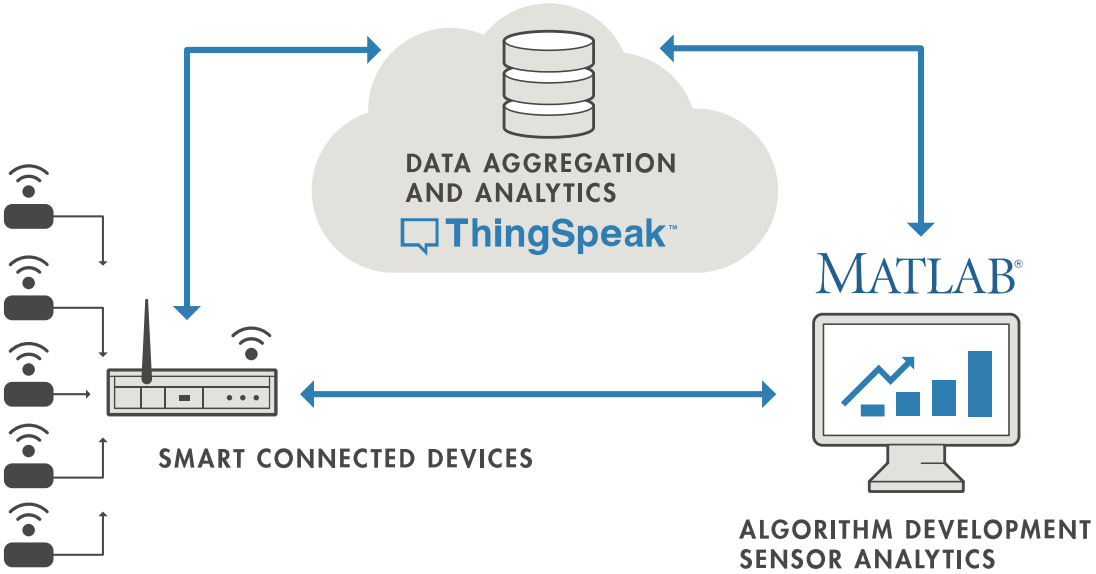


Figura 3.1. Schema di utilizzo di ThingSpeak in applicazioni IoT.

È possibile fare l'update dei dati in due diverse modalità: tramite REST API o MQTT API. Per il progetto è stata scelta questa seconda modalità. Il motivo principale che c'è dietro questa decisione riguarda il tempo di invio ai canali. Mentre l'uso del protocollo HTTP ,supportato dalla libreria "Thingspeak.h", richiede meno sforzo a livello implementativo, ha come contro il tempo di upload, ovvero il tempo consentito per passare dall'update su un canale all'altro, che deve essere necessariamente sopra ai 15 secondi. Questo apre una finestra temporale dilatata (tempo minimo = (15s) * (12canali) = 3 minuti)) in cui la HUZZAH8266 è bloccata in fase di upload. Viene da sé che un tempo così lungo è anche sinonimo di una maggiore possibilità di failure in questa fase.

L'approccio MQTT, invece, permette di eliminare tale delay, al costo di un codice più lungo. MQTT è anch'esso un protocollo per la comunicazione, ma specializzato per il settore IoT, a differenza di HTTP, le cui applicazioni sono molto eterogenee. E' basato su un'architettura publish/subscribe over TCP/IP che lo rende molto più maneggevole e "leggero" rispetto alla sua controparte. I due ruoli previsti dal protocollo sono il client e il broker. Per client si intende un qualsiasi dispositivo che desidera ricevere/inviare dati (sono client sia il sensore IoT che il canale ThingSpeak), mentre il broker è il software che riceve i messaggi dai client, li indirizza e gestisce le connessioni con i client. Il broker MQTT di ThingSpeak fa riferimento all'URL "mqtt3.thingspeak.com" e alla porta TCP 1883. L'update dei canali viene fatto pubblicando sui fields, il che avviene con un solo comando e in tempo pressoché zero.

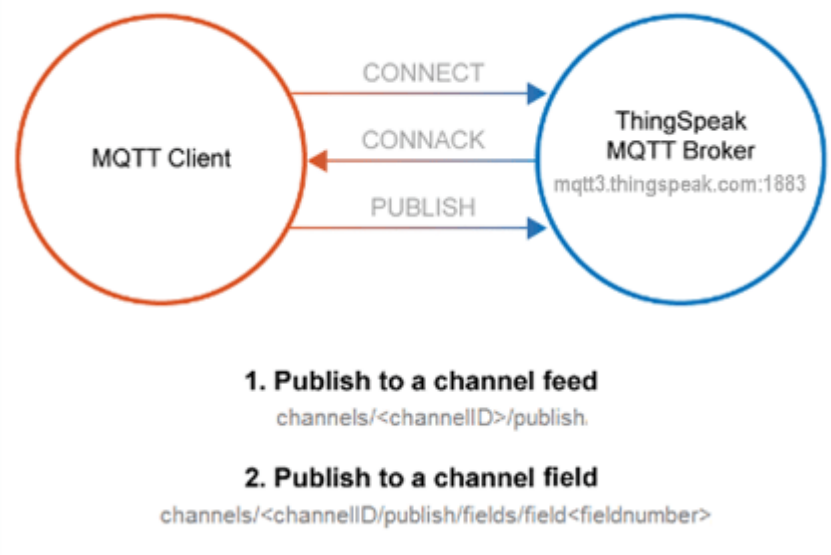


Figura 3.2. MQTT Publish.

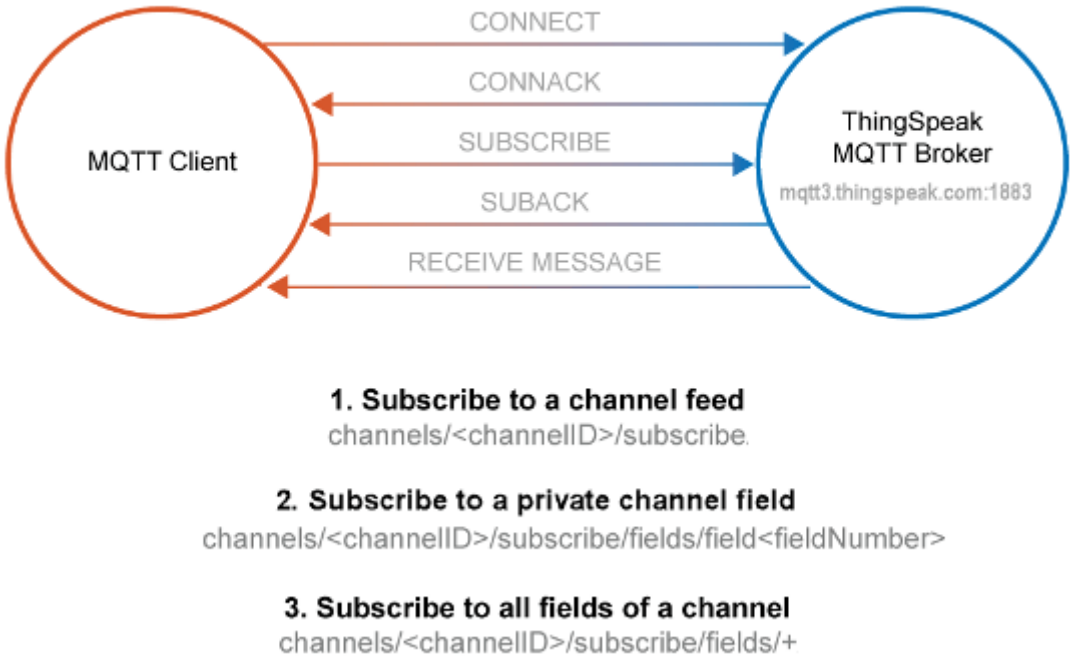


Figura 3.3. MQTT Subscribe.

La prima parte del codice prevede le dichiarazioni delle funzioni, l'inizializzazione delle variabili e la funzione di setup.

```
#include "SPISlave.h"
#include "ThingSpeak.h"
#include "secrets.h"
#include <ESP8266WiFi.h>
#include <ESP8266WiFiMulti.h>
#include <PubSubClient.h> //MQTT version

//#def e #undef
```

```
#undef DEBUG_ESP_WIFI
#undef DEBUG_ESP_PORT
#undef DEBUG_WIFI

//function declaration
void sendWiFi();
void mqttconnect();
void mqttPublish(long pubChannelID, String message);
void mqttSubscribe( long subChannelID );

//gpr variables
double datacollect[12]; //dati ricevuti da TEENSY
WiFiClient client;
char ssid[] = SECRET_SSID;
char pass[] = SECRET_PASS;
unsigned long myChannelNumber1 = SECRET_CH_ID1;
String myWriteAPIKey1 = SECRET_WRITE_APIKEY1;
char* WriteAPIKey1 = SECRET_WRITE_APIKEY1;
unsigned long myChannelNumber2 = SECRET_CH_ID2;
String myWriteAPIKey2 = SECRET_WRITE_APIKEY2;
char* WriteAPIKey2 = SECRET_WRITE_APIKEY2;
const char* ssidSA = "pttdns_Axx";
const char* passwordSA = "aa00.11T";
ESP8266WiFiMulti WiFiMulti;
//MQTT version
//char* topic="channels/<channelID/publish/<channelAPI>
char* topic1 = "channels/1448321/publish/";
char* topic2 = "channels/1451246/publish/";
const char* server = "mqtt3.thingspeak.com";
PubSubClient mqttclient(server, 1883, client);
const char mqttUserName[]= MQTT_USERNAME;
const char mqttPass[]= MQTT_PASS;
const char clientID[] = MQTT_USERNAME;

//inizializzazione
void setup() {

  Serial.begin(9600);
  Serial.setDebugOutput(true);

  //inizializzazione per SPI
  SPISlave.begin();//setup SPI Slave registers and pins

  //setup WiFi module
  WiFiMulti.addAP(ssid, pass);
  WiFi.softAP(ssidSA, passwordSA, 1, 1);
  ThingSpeak.begin(client);

  //controllo connessione WiFI
  if (WiFiMulti.run() != WL_CONNECTED) {
    Serial.print("Attempting to connect to SSID: ");
```

```
    Serial.println(SECRET_SSID);
    while (WiFiMulti.run() != WL_CONNECTED) {
      WiFi.begin(ssid, pass); // Connect to WPA/WPA2
network. Change this line if using open or WEP network
      Serial.print(".");
      Serial.println(WiFi.status());
      delay(3000);
    }
    Serial.println("\nConnected.");
    Serial.println(WiFi.localIP()); //show assigned IP
  }

  mqttconnect();

}
```

Diverse costanti sono contenute nel file header "secrets.h" presente nella directory del file .ino . Questi valori sono riservati perché contengono informazioni sulla rete WiFi e sulle chiavi dei canali ThingSpeak.

A questo punto la HUZZAH8266 è collegata ad Internet e in attesa di dati da fornire in upload, questi passaggi sono contenuti nella funzione **loop()** .

```
int pos = 0; //variabile di posizionamento all'interno del
vettore di ricezione
int prox = 0; //variabile di procedura
int count =0;

void loop() {


  SPISlave.onData([](uint8_t * data, size_t len) {
    String message = String((char *)data);
    if (message.equals("INVIO DATI.")) {
      Serial.println("Istruzione ricevuta:");
      Serial.println(message);
      (void) len;
      if (prox == 0) {
        SPISlave.setData("I am ready.");
        count =0;
        pos = 0;
      }
      else {
        SPISlave.setData("I am not ready.");
        count++;
        if (count == 7){
          count = 0;
          ESP.reset();
          }
```

```
      }
    }
    else if (message.equals("TIME TO RESET.")) {
      prox = 0;
      ESP.reset();
    }
    else {
      String datiRX = message;
      Serial.println("Ho ricevuto questi dati:");
      Serial.println(datiRX);
      datacollect[pos] = datiRX.toDouble();
      SPISlave.setData("ACK");
      pos++;
      if (pos == 12) {
        Serial.println("I valori nel vettore OLD sono: ");
        for (int xx = 0; xx < 12; xx++) {
          Serial.printf("%f ; " , datacollect[xx]);
        }
        Serial.println("");
        prox = 1;
      }
    }
  });

  if (prox == 1) {
    sendWiFi();
  }

}
```

Nella main function è a tutti gli effetti presente un interrupt generato da SPI: una volta ricevuto un messaggio sui pin dedicati, avviene la valutazione della richiesta e dello stato della macchina locale per poter determinare l'evento. Nel dettaglio, la variabile indicata come **prox** sta a rappresentare se la HUZZAH8266 è occupata ancora nell'invio sul canale WiFi (prox = 1) o se è disponibile per generare un nuovo array (prox = 0).La variaible **pos** tiene da conto l'indice dell'array che si sta generando. Una volta usciti dal modulo di gestione interrupt, se l'operazione è andata a buon fine, si chiama la funzione **sendWiFi()** .

Nella funzione sotto riportata avviene un riordinamento dell'array per favorire l'upload sui channel, dopodichè sfruttando la procedura MQTT vengono caricati i valori sui fields associati.

```
void sendWiFi() {
  int i = 0;
  String newdatacollect[12];
  newdatacollect[0] = String(datacollect[0], 3); //etanolo
  newdatacollect[1] = String(datacollect[1], 3);
//ammoniaca
```

```
  newdatacollect[2] = String(datacollect[2], 3); //acetone
  newdatacollect[3] = String(datacollect[8], 3); //co2
  newdatacollect[4] = String(datacollect[6], 3); //pm25
  newdatacollect[5] = String(datacollect[7], 3); //pm10
  newdatacollect[6] = String(datacollect[10], 3); //temp
  newdatacollect[7] = String(datacollect[11], 3);
//humidity
  newdatacollect[8] = String(datacollect[3], 3); //NH3
  newdatacollect[9] = String(datacollect[4], 3); //TVOC
  newdatacollect[10] = String(datacollect[5], 3); //H2S
  newdatacollect[11] = String(datacollect[9], 3);
//nh3(MQ137)
  //MQTT VERSION
  if (!mqttclient.connected()) {
    mqttconnect();
    mqttSubscribe(myChannelNumber1);
    mqttSubscribe(myChannelNumber2);
  }
  mqttclient.loop();
  String payload1 = "field1=" + newdatacollect[0] +
"&field2=" + newdatacollect[1];
  payload1 += "&field3=" + newdatacollect[2] + "&field4=" +
newdatacollect[3];
  payload1 += "&field5=" + newdatacollect[4] + "&field6=" +
newdatacollect[5];
  payload1 += "&field7=" + newdatacollect[6] + "&field8=" +
newdatacollect[7];
  payload1 += "&status=MQTTPUBLISH";
  mqttPublish(myChannelNumber1, payload1);

  String payload2 = "field1=" + newdatacollect[8] +
"&field2=" + newdatacollect[9];
  payload2 += "&field3=" + newdatacollect[10] + "&field4="
+ newdatacollect[11];
  payload2 += "&status=MQTTPUBLISH";
  mqttPublish(myChannelNumber2, payload2);
  prox = 0;
}
```

Con le funzioni ausiliarie così implementate:

```
void mqttconnect(){

  while(!mqttclient.connected()){
      if (mqttclient.connect(clientID, mqttUserName,
mqttPass)) {
      Serial.print( "MQTT to " );
      Serial.print( server );
      Serial.print (" at port ");
      Serial.print( 1883 );
      Serial.println( " successful." );
```

```
  }
  else {
     Serial.print( "MQTT connection failed, rc = " );
     Serial.print( mqttclient.state() );
      Serial.println( " Will try again in a few seconds" );
      delay( 1000 );
  }
    }
  }

  void mqttSubscribe( long subChannelID ){
  String myTopic = "channels/"+String( subChannelID
)+"/subscribe";
  mqttclient.subscribe(myTopic.c_str());
}

void mqttPublish(long pubChannelID, String message) {
  String topicString ="channels/" + String( pubChannelID )
+ "/publish";
  mqttclient.publish( topicString.c_str(), message.c_str()
);
}
```

Utilizzate rispettivamente per stabilire la connessione, per operazioni di subscribe e di publish.

4. CATTURE E CASI DI STUDIO

Di seguito vengono riportati prima alcuni screenshot della console seriale Arduino, tramite cui sarà possibile osservare fasi della macchina a stati finti e update su canale ThingSpeak, poi verranno trattati alcuni scenari per osservare il comportamento del sensore.

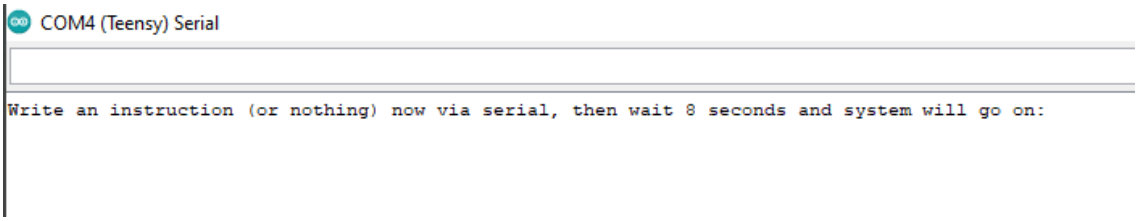


Figura 4.1. Comando seriale per DEBUG. Comando implementato : "huzzah reset\n"

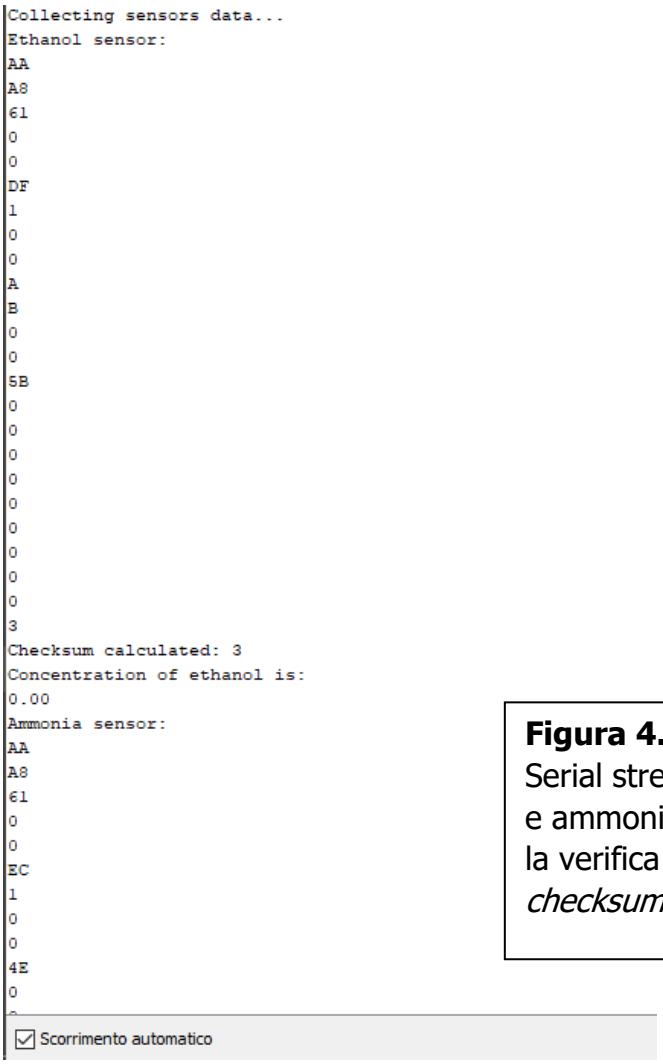


Figura 4.2. Il data retrieving ha inizio. Serial stream del sensore per etanolo M1005 e ammoniaca M1002 . Si può già osservare la verifica della validità del formato tramite *checksum equivalence*.

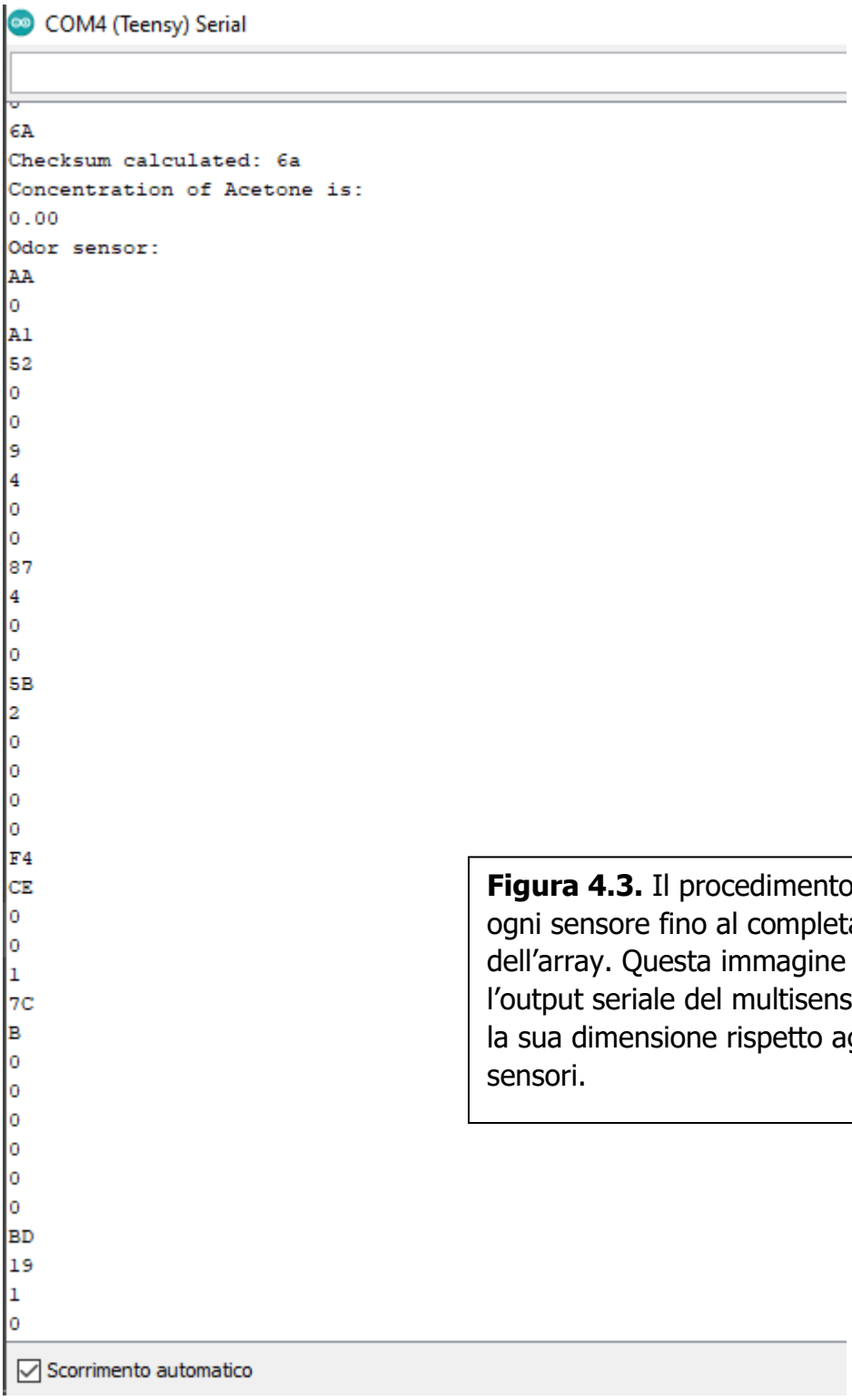


Figura 4.3. Il procedimento continua per ogni sensore fino al completamento dell'array. Questa immagine mostra l'output seriale del multisensor MD3005 e la sua dimensione rispetto agli altri sensori.

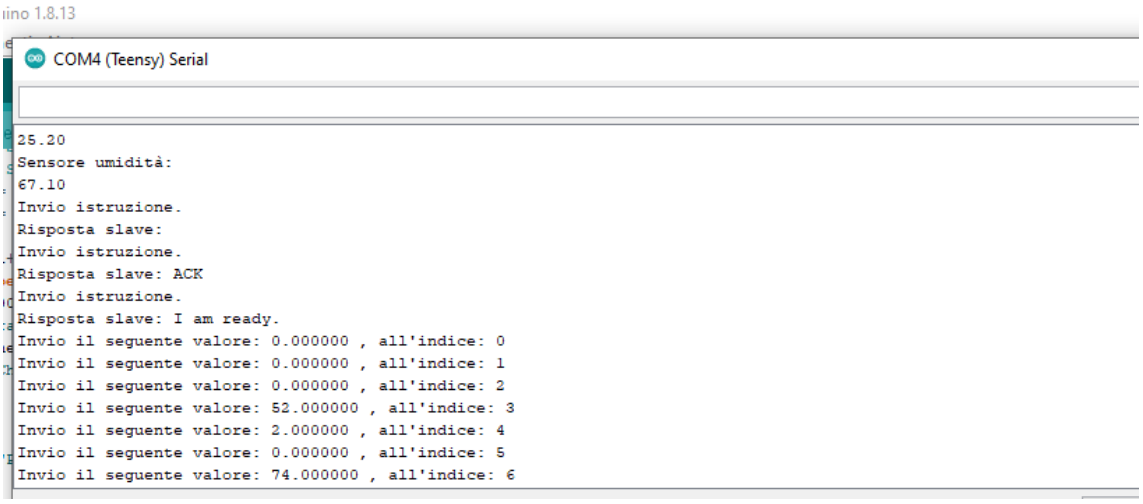


Figura 4.4. La Teensy4.0 inizia la comunicazione SPI. Alla risposta "I am ready." invia l'array indice per indice.

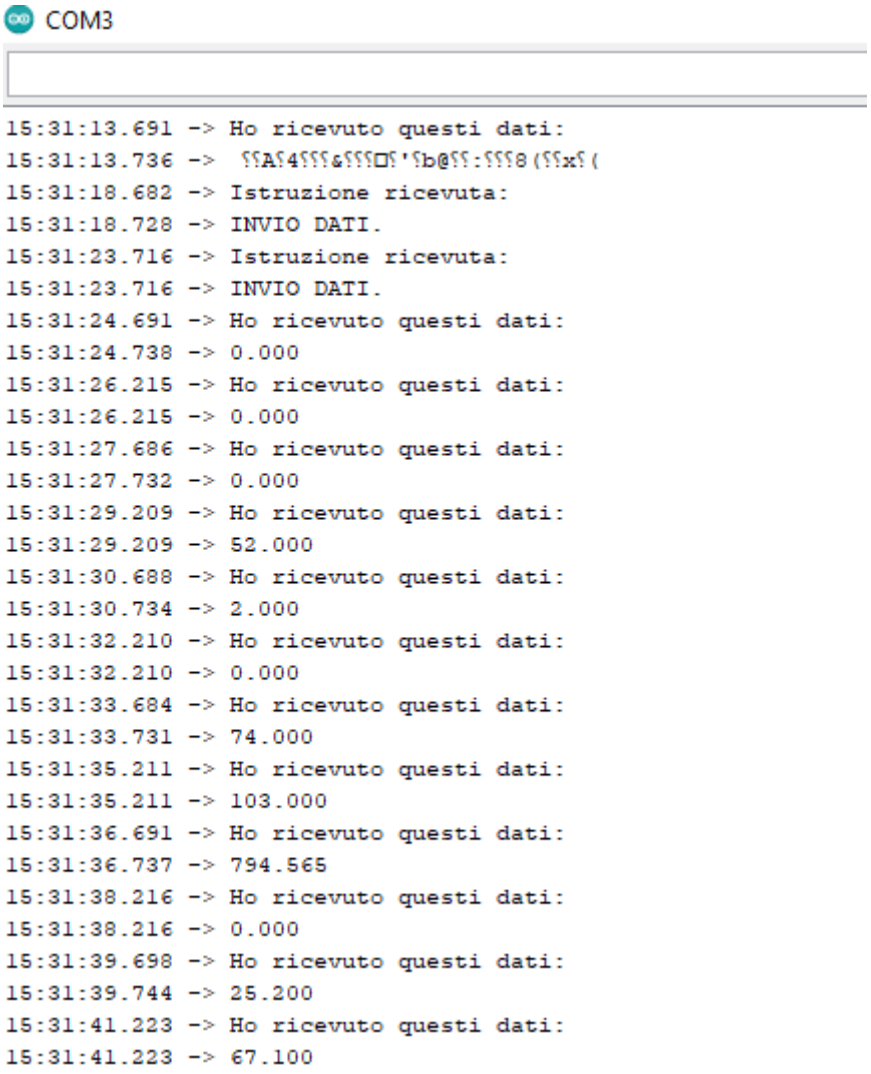


Figura 4.5. La HUZZAH8266 riconosce il comando "INVIO DATI." e popola l'array locale elemento per elemento.

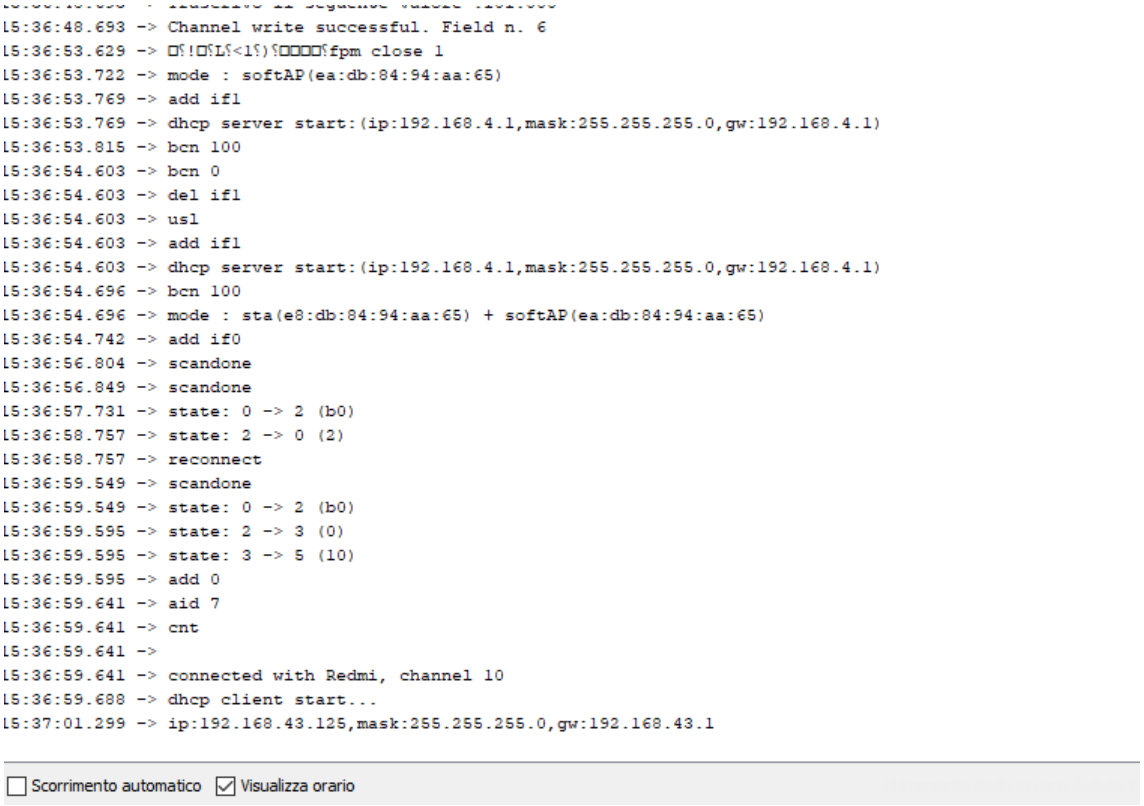


Figura 4.6. Qui si può notare il RESET forzato da TEENSY su HUZZAH. All'avvio del firmware la HUZZAH si connette al router, con tanto di acquisizione IP e creazione del softAP.

Passando agli scenari. In un locale indoor sono state predisposte delle condizioni per la rilevazione di alcuni gas compresi nella lista del sensore IoT: il banale utilizzo dell'acetone per rimuovere lo smalto dalle unghie o adoperare prodotti per la pulizia della casa, hanno permesso di rilevare concentrazioni di inquinanti.

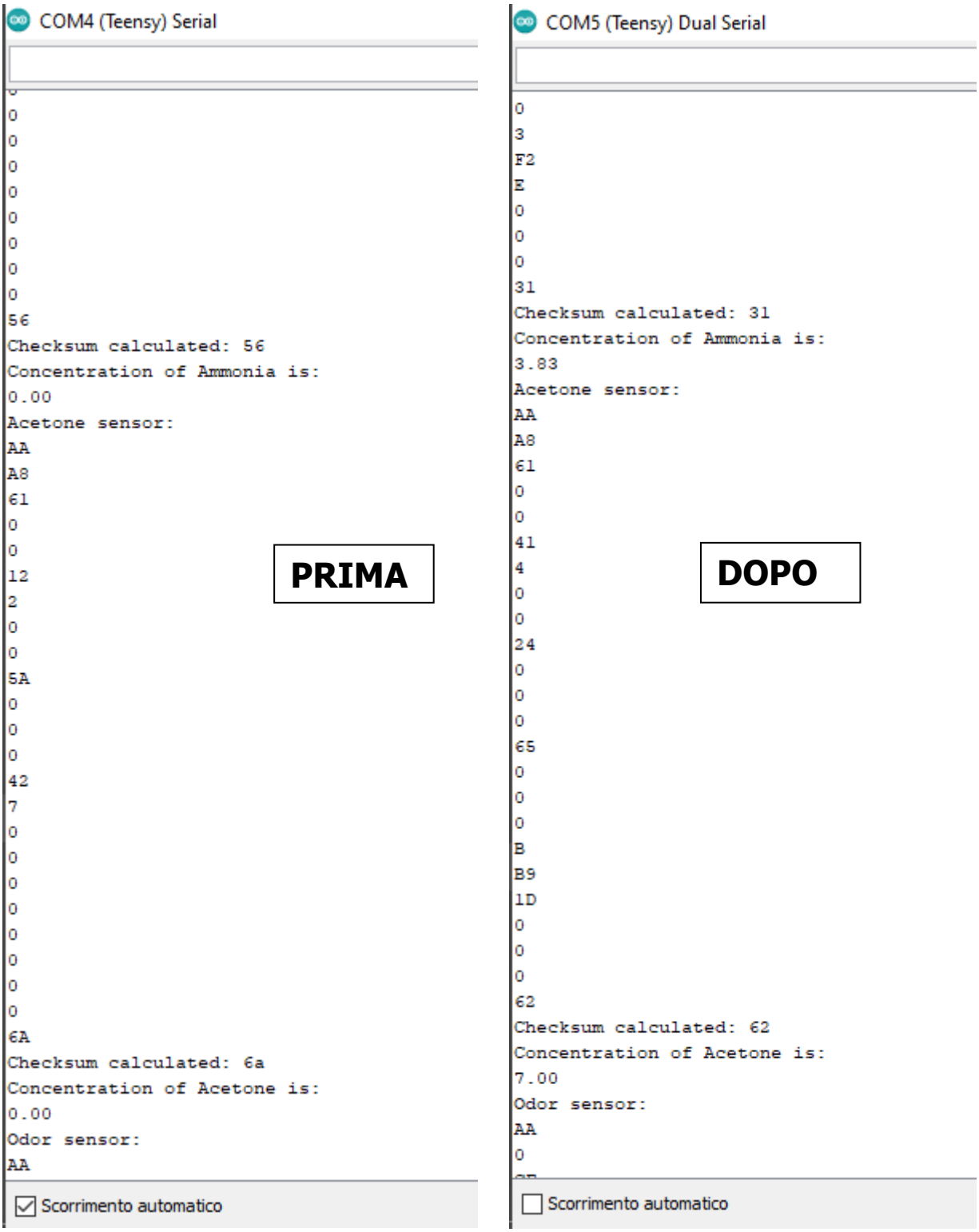


Figura 4.7. Le due immagini confrontate mostrano la rilevazione del sensore nello stesso punto in un istante prima dell'esperimento e in uno successivo.

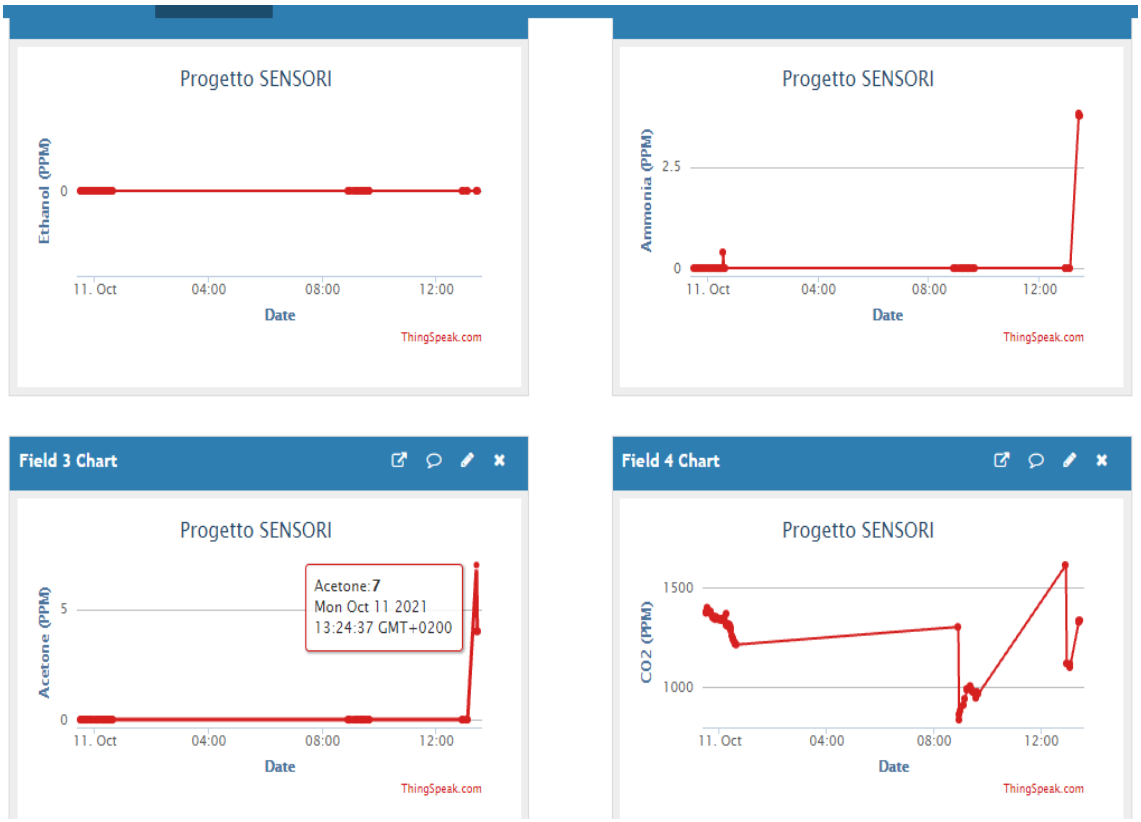


Figura 4.8. Ovviamente il risultato ottenuto via seriale verrà caricato sul canale nel rispettivo field. Questo è l'output dall'interfaccia ThingSpeak.

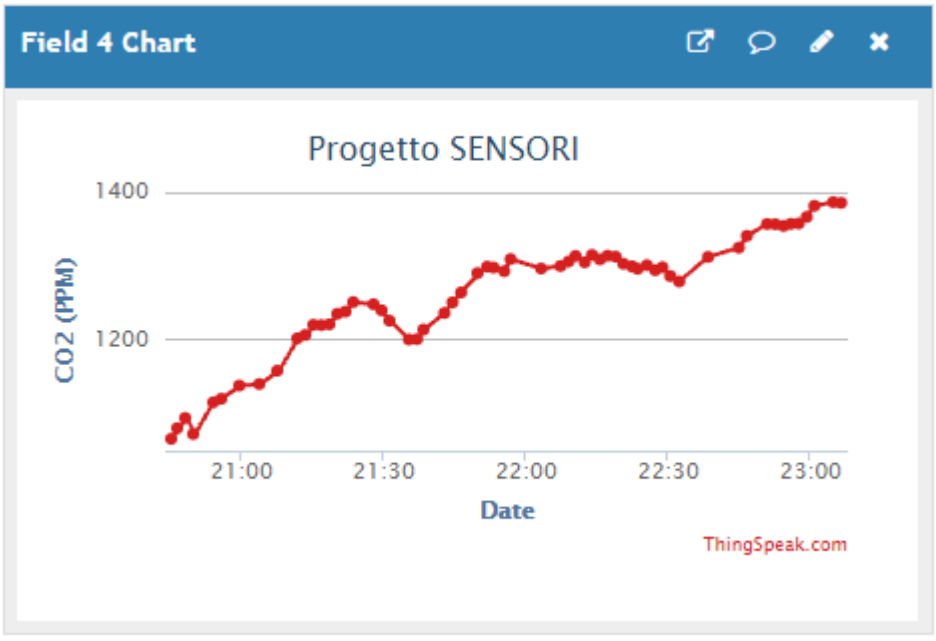


Figura 4.9. Questa immagine non riguarda il caso prima proposto, però è anch'essa molto interessante e mostra il tasso di crescita delle concentrazioni di CO2 nelle ore notturne in una stanza (finestre chiuse, porte chiuse, abitanti in un'unica stanza).

5. CONCLUSIONI

Il sensore IoT ,sviluppato durante l'attività di tirocinio, si è rivelato essere un progetto molto completo, ovvero che ha richiesto di comprendere e approfondire tematiche relative a più ambiti dell'elettronica: dal design hardware alla tecnica di montaggio, dall'ambiente Arduino alla piattaforma ThingSpeak. In particolare, si pone l'accento su un argomento del tutto attuale come quello dell' IoT, che offre vantaggi davvero rilevanti per quanto riguarda consumo, portabilità, ma soprattutto l'idea di interconnettere elementi nativamente "analogici", per sfruttarli al 100% e avere un interfacciamento praticamente immediato su ciò che ci circonda nella vita di tutti i giorni.

La tematica dell' Indoor Air Quality sta assumendo sempre più rilevanza nella società, richiedendo nuovi paradigmi della convivenza al chiuso. È bene quindi attrezzarsi e intraprendere di prima persona iniziative di monitoraggio e prevenzione, bastano piccoli accorgimenti e misure contenitive per poter migliorare il benessere delle decine di occupanti, il che può portare, in alcuni ambiti, anche a un miglioramento della produttività nel luogo di lavoro.

Bibliografia.

[1]. "Indoor Air Quality: A Focus on the European Legislation and State-of-the-Art Research in Italy"; G. Settimo, M. Manigrasso, P. Avino; 2020

[2]. "La qualità dell'aria *indoor*: attuale situazione nazionale e comunitaria. L'esperienza del Gruppo di Studio Nazionale Inquinamento *Indoor*."; Istituto Superiore di Sanità; Roma, 28 maggio 2014; Anna Santarsiero, Loredana Musmeci e Sergio Fuselli per il Gruppo di Studio Nazionale sull'Inquinamento *Indoor* 2015

[3]. "Parametri microclimatici e inquinamento *indoor*."; Anna Santarsiero, Loredana Musmeci, Annino Ricci, Sandra Corasaniti, Paolo Coppa, Gianluigi Bovesecchi, Riccardo Merluzzi, Sergio Fuselli per il Gruppo di Studio Nazionale sull'Inquinamento *Indoor* 2015

[4]. Riferimenti per H2S: "Hydrogen Sulfide: Human Health Aspects", C.-H. Selene J. Chou, World Health Organization, 2003 ; (Rapporto ISTISAN 16/15) "Presenza di CO2 e H2S in ambienti indoor: attuali conoscenze e letteratura scientifica" per il Gruppo di Studio Nazionale sull'Inquinamento Indoor 2016

[5]. Riferimenti per NH3: "Acute Exposure Guideline Levels for selected airborne chemicals", National Research Council (US) Committee on Acute Exposure Guideline Levels, Washington (DC): National Academies Press (US); 2008.

[6]. Riferimenti per Acetone: https://www.epa.gov/aegl/acetone-results-aegl-program

[7]. Riferimenti per Etanolo: https://pubchem.ncbi.nlm.nih.gov/source/hsdb/82.

[8]. Riferimenti per CO2: https://www.kane.co.uk/knowledge-centre/what-are-safe-levels-of-co-and-co2-in-rooms

[9]. DHT22_DATASHEET (https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf)

[10]. M1002_DATASHEET (http://rainbowsensor.cn/en/mozu/364.html)

[11]. M1005_DATASHEET (http://rainbowsensor.cn/en/mozu/366.html)

[12]. M1015Y_DATASHEET (http://rainbowsensor.cn/en/mozu/370.html)

[13]. LASER SDS018_DATASHEET

(https://ecksteinimg.de/Datasheet/SDS018%20Laser%20PM2.5%20Product%20Spec%20V1.5.pdf)

[14]. MD3005_DATASHEET
(http://rainbowsensor.cn/en/mozu/371.html)

[15]. MQ137_DATASHEET
(https://circuitdigest.com/microcontroller-projects/arduino-mq137-ammonia-sensor)

[16]. MHZ14_DATASHEET
(https://www.winsen-sensor.com/d/files/MH-Z14.pdf)

[17]. TEENSY_specifications
(https://www.pjrc.com/store/teensy40.html)

[18]. HUZZAH8266_specifications
(https://learn.adafruit.com/adafruit-feather-huzzah-esp8266)

[19]. THINGSPEAK
(https://it.mathworks.com/products/thingspeak.html)
(https://it.mathworks.com/help/thingspeak/rest-api.html)