

## UNIVERSITÀ POLITECNICA DELLE MARCHE FACOLTÀ DI INGEGNERIA

## Corso di Laurea Triennale in: INGEGNERIA ELETTRONICA

Tesi di Laurea:

## Progettazione e realizzazione di sensori wireless per l'analisi della qualità dell'aria

Design and construction of wireless devices for air quality analysis

	0 1:1 (
Relatore:	Candidatos
Prof. Ennio Gambi	Pasquale La Serra
Correlatore:	
Dott Paolo Olivetti	

## **INDICE**

**INTRODUZIONE** 

	PITOLO 1: CARATTERISTICHE E FUNZIO POSITIVI HARDWARE	ONALITÀ DEI
1.1	ESP8266 - DevKitC	5
1.2	MQ7	7
1.3	DHT22	8
1.4	M1002	9
1.5	M1005	11
1.6	M1015	13
1.7	MD3005	15
	PITOLO 2: CONFIGURAZIONI DI DGETTATI	EI SENSORI
2.1	SENSORE 1: RILEVAMENTO DI	17
	TEMPERATURA, UMIDITÀ, AMMONIACA	E
	MONOSSIDO DI CARBONIO	
	(ESP8266 + MQ7 + DHT22 + M1002)	
2.2	SENSORE 2: RILEVAMENTO DI	19
	TEMPERATURA, UMIDITÀ, ETANOLO E	
	MONOSSIDO DI CARBONIO	
	(ESP8266 + MQ7 + DHT22 + M1005)	
2.3	SENSORE 3: RILEVAMENTO DI	21
	TEMPERATURA, UMIDITÀ, ACETONE E	
	MONOSSIDO DI CARBONIO	
	(ESP8266 + MQ7 + DHT22 + M1015)	

3

2.4	SENSORE 4: RILEVAMENTO DI	23
	TEMPERATURA, UMIDITÀ,	
	AMMONIACA, GAS TVOC,	
	ACIDO SOLFIDRICO E	
	MONOSSIDO DI CARBONIO	
	(ESP8266 + MQ7 + DHT22 + MD3005)	
CAF	PITOLO 3: FIRMWARE E RISULTATI DEI	SENSORI
PRC	GETTATI	
3.1	SENSORE 1: ESP8266 + MQ7 + DHT22 + M1002	25
3.2	SENSORE 2: ESP8266 + MQ7 + DHT22 + M1005	30
3.3	SENSORE 3: ESP8266 + MQ7 + DHT22 + M1015	35
3.4	SENSORE 4: ESP8266 + MQ7 + DHT22 + MD3005	40

CONCLUSIONI

#### **INTRODUZIONE**

Lo sviluppo tecnologico nel settore dell'Internet of Things (IoT) negli ultimi decenni è stato talmente ampio e rapido che ha cambiato la vita di milioni di persone senza che molte di loro nemmeno se ne accorgessero: i lampioni nelle città che regolano la luminosità in base alla visibilità, i dispositivi per la gestione automatica o da remoto degli impianti nelle abitazioni per efficientare i consumi e il monitoraggio di vari parametri micro-climatici a supporto dell'agricoltura rappresentano degli esempi di come le tecnologie "intelligenti" e "interconnesse" vengono applicate nel mondo odierno.

Termini come "smart home", "smart city", "smart car" e "smart agriculture" diventano sempre più comuni nel linguaggio comune a prova del fatto che i campi di applicazione delle tecnologie IoT sono potenzialmente infiniti e hanno stravolto, in meglio, i processi del mondo produttivo e non solo.

A prova di ciò basti pensare all'utilizzo dei sensori per il monitoraggio ambientale atmosferico in un qualsiasi impianto: fino a pochi anni fa questi erano dispositivi complessi e di dimensioni importanti, come ad esempio le stazioni di rilevamento della qualità dell'aria installate dalle istituzioni locali in vari punti del territorio.

La miniaturizzazione dei componenti, l'abbattimento dei costi e delle dimensioni senza intaccare più di tanto la capacità nell'analisi dell'aria, hanno portato ad una diffusione e ad un utilizzo sempre più crescente dei sensori per il monitoraggio ambientale nei processi produttivi e non solo.

L'Internet of Things ha giocato un ruolo fondamentale: grazie all'impiego di sensori intelligenti si possono misurare e monitorare in tempo reale le emissioni in atmosfera (come la CO2), la concentrazione di polveri, le quantità di gas naturale o di energia impiegate. E, dunque, si possono mettere in atto azioni per correggere, e dunque ridurre, l'impatto ambientale delle attività e ottimizzare i processi e le performance produttive.

Inoltre, questo campo applicativo dell'IoT rappresenta un perfetto esempio di come la tecnologia possa rappresentare un grande aiuto per l'uomo: negli ultimi anni la qualità dell'aria e gli strumenti di misura connessi sono diventati sempre più importanti nell'opinione pubblica e in quella scientifica.

Infatti, secondo l'Agenzia Europea per l'Ambiente un decesso su otto è legato all'inquinamento¹ e non di rado, all'interno delle nostre abitazioni, nei luoghi di

<sup>&</sup>lt;sup>1</sup> https://www.repubblica.it/cronaca/2020/09/08/news/unione\_europea\_morti\_smog-266535310/

lavoro o nelle scuole, si possono misurare livelli di inquinanti oltre la soglia di attenzione stabilita dall'Organizzazione Mondiale della Sanità.

Il monitoraggio della qualità ambientale ha quindi assunto un'importanza sempre maggiore non solo in ambito scientifico ma anche medico e sociale, per consentire alle persone di incrementare le proprie conoscenze sulle cause e gli effetti dell'inquinamento sulla propria salute.

L'utilizzo e i vantaggi di tali tecnologie non sono limitati alle attività industriali o produttive: esistono sensori che possono essere utilizzati anche nelle abitazioni private per il monitoraggio di diversi parametri (tra cui umidità e temperatura) per regolare il riscaldamento/raffrescamento e di conseguenza efficientare i consumi energetici e risparmiare denaro.

In questo lavoro di tesi andrò quindi ad illustrare e ad analizzare diverse configurazioni di sensori che possono essere impiegati proprio per l'analisi della qualità dell'aria, descrivendo le caratteristiche dei componenti utilizzati, il firmware utilizzato per il loro funzionamento e l'analisi dei risultati derivanti dal monitoraggio e dal rilevamento di vari gas e parametri.

## CAPITOLO 1: CARATTERISTICHE E FUNZIONALITA' DEI DISPOSITIVI HARDWARE

#### 1.1 ESP8266 – DevKitC

L'ESP8266, in questa tesi nella versione "DevKitC", è un System on Chip (SoC) prodotto dall'azienda cinese Espressif: il suo basso costo ha permesso e permette a tutti di iniziare a produrre esperimenti nel campo dell'IoT.

L'ESP8266 è dotato di connettività wireless che consente sia di ricevere dati da sensori sia di controllare attuatori tramite i pin di input/output in grado di collegarsi alla rete internet, sfruttando appunto il collegamento Wi-Fi. L'ESP8266 può collegarsi ad una rete preesistente (modalità client) o crearne una propria (modalità server) alla quale possiamo poi collegarci con un pc, con uno smartphone o con un tablet.



L'ESP8266 lavora ad un'alimentazione di 3.3V e tale valore non può essere superato (pena bruciare il chip) mentre consuma circa 200-250 mA in fase di trasmissione. Di conseguenza anche i segnali in uscita dai pin RX/TX di comunicazione devono rispettare tale vincolo.

Sono presenti diversi pin per collegamenti digitali, uno per collegamento di tipo analogico (ADC) e anche un pin per l'alimentazione a 5 V.

Oltre al firmware di fabbrica di Espressif che utilizza i comandi AT, il modulo può essere sfruttato utilizzando altri firmware alternativi: infatti una caratteristica molto interessante di questo modulo è che può essere programmato

utilizzando anche l'ambiente di sviluppo (IDE) di Arduino, aspetto che apre la possibilità a diversi scenari per la programmazione del componente stesso. Ulteriori caratteristiche hardware e funzionalità proprie del microcontrollore possono essere trovate nella documentazione messa a disposizione dalla casa produttrice stessa<sup>2</sup>.

Quindi le dimensioni contenute, il basso costo di mercato, la capacità di connettersi WiFi, la possibilità di poter utilizzare diversi firmware di programmazione sono tutti punti a favore di questo modulo e ne favoriscono l'utilizzo nel campo dell'IoT.

<sup>&</sup>lt;sup>2</sup> https://www.espressif.com/sites/default/files/documentation/ESP8266-DevKitC getting started guide EN.pdf

#### 1.2 MQ7

L MQ7 è un sensore prodotto dalla società americana SparkFun Electronics atto a rilevare le concentrazioni di monossido di carbonio (CO) nell'aria: il range di rilevamento della CO nell'aria va da circa 10 a circa 500 ppm (parti per milione). Il sensore lavora a 5V con un consumo di corrente pari a 150 mA (900 mW), per poter funzionare correttamente necessita di un tempo di preriscaldamento superiore alle 48 ore ed è fornito di un'uscita analogica composta da una resistenza.



La buona sensibilità al monossido di carbonio, la lunga durata del prodotto, il basso costo e la semplicità di utilizzo fanno di questo sensore uno dei principali attori nell'analisi della qualità dell'aria.

Infatti 1 MQ7 può essere utilizzato sia in ambito domestico sia in ambito industriale per il monitoraggio della concentrazione di CO nell'aria.

#### 1.3 DHT22

Il DHT22 è un sensore di temperatura e umidità basilare e low-cost, prodotto dall'azienda americana Adafruit Industries. Utilizza un sensore di umidità capacitivo e un termistore per misurare l'aria circostante e comunica le sue misurazioni tramite un segnale digitale sul pin dei dati (non richiede un input pin analogico). I valori di umidità e di temperatura vanno dal 0 al 100% e dai -40 agli 80°C, con una accuratezza del 2-5% e una risoluzione del 0.1% per la prima e di ± 0.5°C e 0.1°C per la seconda.

Il sensore lavora a 3.3-5V, consuma al massimo 2.5 mA ed è in grado di fornire i valori di temperatura e umidità presenti nell'aria circostante ogni due secondi: questo è l'unico aspetto negativo del componente dato che ogni volta che andiamo a leggere i dati, questi potrebbero essere "vecchi" di 2 secondi.

Sono poi presenti 4 pin, spaziati di 2,54 mm che consentono una facile connessione ad una breadboard: il primo pin a sinistra è per l'alimentazione, il secondo pin è connesso al pin input dei dati del microcontrollore e infine il pin più a destra alla massa (ground). Con il sensore è inclusa una resistenza da 4,7k $\Omega$ -10k $\Omega$ , che può essere utilizzata come resistenza di pull-up dal pin dati verso l'alimentazione.



È particolarmente adatto per prodotti di consumo, stazioni meteo, applicazioni HVAC (Heating, Ventilation and Air Conditioning).

Tutte le informazioni sul DHT22 possono essere recuperate nella documentazione messa a disposizione dal produttore sul proprio sito internet<sup>3</sup>.

<sup>&</sup>lt;sup>3</sup> https://cdn-shop.adafruit.com/datasheets/DHT22.pdf

#### 1.4 M1002

L M1002 è un sensore prodotto dalla compagnia cinese Raimbow Technology Company secondo la tecnologia MEMS (Micro Electro-Mechanical Systems) che possiede una elevata sensibilità all'ammoniaca (NH<sub>3</sub>) e dunque può essere utilizzato per la rilevazione di quest'ultima nell'aria: il range di misurazione è 0-300 ppm (parti per milione), con una risoluzione di 2 ppm.

Le caratteristiche principali sono dunque l'alta sensibilità e la buona risoluzione nonostante il basso consumo di energia, l'alta stabilità e la lunga durata di funzionamento (secondo la Rainbow maggiore a 5 anni) e la possibilità di selezionare diverse modalità di output, tra cui la comunicazione seriale (UART), il segnale analogico o l'onda PWM.

Il modulo lavora a 3.3V con un consumo di corrente inferiore o pari a 20 mA, ha un tempo di preriscaldamento di circa 3-5 minuti e può operare ad una temperatura compresa tra i -10 e i 55°C.



Nella progettazione del sensore completo contenente l M1002 si è scelto di utilizzare la comunicazione seriale per il trasferimento dei dati in uscita tra quest'ultimo e il microcontrollore.

In questa modalità l M1002 comunica i dati rilevati ogni 3 secondi, attraverso un unico dato di 24 bytes in formato esadecimale, secondo il seguente schema:

1 byte	4 byte	4 byte	4 byte	4 byte	1	4 byte	1 byte	1 byte
					byte			
Frame	Temperature	Channel	Reserved	Reserved	Gas	gas	Reserved	Check
header		1 voltage			sort	concentration		Digit
(0xAA)								

Nel caso di dati formati da 4 bytes (come proprio per la concentrazione dell'ammoniaca nell'aria), quando questi vengono convertiti dal formato esadecimale al formato decimale, il valore di ordine maggiore viene posto alla fine mentre quello di ordine minore all'inizio e il valore risultante viene diviso per 1000: in questo modo si ottiene il valore decimale riferito alla grandezza rilevata.

Tutte le altre informazioni sul M1002 possono essere recuperate nella documentazione messa a disposizione dal produttore sul proprio sito internet<sup>4</sup>.

<sup>&</sup>lt;sup>4</sup> http://rainbowsensor.cn/en/mozu/364.html

#### 1.5 M1005

L M1005 è un sensore prodotto dalla compagnia cinese Raimbow Technology Company secondo la tecnologia MEMS (Micro Electro-Mechanical Systems) che possiede una elevata sensibilità all'alcol e dunque può essere utilizzato per la rilevazione dell'etanolo (CH<sub>3</sub>CH<sub>2</sub>OH) nell'aria: il range di misurazione è 0-300 ppm (parti per milione), con una risoluzione di 3 ppm.

Il modulo ha una elevata sensibilità, un'ottima stabilità nel lungo periodo, un basso consumo di energia e conserva queste caratteristiche sia col passare del tempo sia in seguito ad un importante utilizzo.

Lavora a 3.3V con un consumo di corrente inferiore o pari a 25 mA, ha un tempo di preriscaldamento di 3-5 minuti e può operare ad una temperatura compresa tra i -10 e i 55°C.



Nella progettazione del sensore completo contenente l M1005 si è scelto di utilizzare la comunicazione seriale per il trasferimento dei dati in uscita tra quest'ultimo e il microcontrollore.

In questa modalità l M1005 comunica i dati rilevati ogni 3 secondi, attraverso un unico dato di 24 bytes in formato esadecimale, secondo il seguente schema:

1 byte	4 bytes	4 bytes	4 bytes	4 bytes	1 byte	4 bytes	1 byte	1 byte
Frame	temperature	Channel 1	Keep	Keep	Gas type	Gas	reserve	Check
header		voltage				concentration		Digit
(0xAA)								

Nel caso di dati formati da 4 bytes (come proprio per la concentrazione dell'etanolo nell'aria), quando questi vengono convertiti dal formato esadecimale al formato decimale, il valore di ordine maggiore viene posto alla fine mentre quello di ordine minore all'inizio e il valore risultante viene diviso per 1000: in questo modo si ottiene il valore decimale riferito alla grandezza rilevata.

Tutte le altre informazioni sul M1005 possono essere recuperate nella documentazione messa a disposizione dal produttore sul proprio sito internet<sup>5</sup>.

-

<sup>&</sup>lt;sup>5</sup> http://rainbowsensor.cn/en/mozu/366.html

#### 1.6 M1015

L M1015 è un sensore prodotto dalla compagnia cinese Raimbow Technology Company secondo la tecnologia MEMS (Micro Electro-Mechanical Systems) che può essere utilizzato per la rilevazione dell'acetone (C<sub>3</sub>H<sub>6</sub>O) nell'aria: il range di misurazione è 0-300 ppm (parti per milione), con una risoluzione di 0.1 ppm. Le funzionalità di questo modulo prevedono un'alta sensibilità e un'alta

Le funzionalità di questo modulo prevedono un'alta sensibilità e un'alta risoluzione; garantisce un lungo periodo di funzionamento ed un basso consumo di energia. Le modalità di comunicazione verso l'esterno possono essere diverse (UART, analogica, PWM) ma tutte con un'ottima stabilità e linearità.



Il modulo lavora a 3.3V con un consumo di corrente inferiore o pari a 25 mA, ha un tempo di preriscaldamento di 3-5 minuti e può operare ad una temperatura compresa tra i -10 e i 55°C.

Nella progettazione del sensore completo contenente l M1015 si è scelto di utilizzare la comunicazione seriale per il trasferimento dei dati in uscita tra quest'ultimo e il microcontrollore.

In questa modalità l M1015 comunica i dati rilevati ogni 3 secondi, attraverso un unico dato di 24 bytes in formato esadecimale, secondo il seguente schema:

1 byte	4 byte	4 byte	4 byte	4 byte	1	4 byte	1 byte	1 byte
					byte			
frame	temperature	humidity	Channel	Channel	Gas	Gas	Alarm/level	Check
header			1	2 voltage	type	concentration		Digit
(0xAA)			voltage	(reserved)				

Nel caso di dati formati da 4 bytes (come proprio per la concentrazione dell'acetone nell'aria), quando questi vengono convertiti dal formato esadecimale al formato decimale, il valore di ordine maggiore viene posto alla fine mentre quello di ordine minore all'inizio e il valore risultante viene diviso per 1000: in questo modo si ottiene il valore decimale riferito alla concentrazione rilevata.

Tutte le altre informazioni sul M1015 possono essere recuperate nella documentazione messa a disposizione dal produttore sul proprio sito internet<sup>6</sup>.

\_

<sup>&</sup>lt;sup>6</sup> http://rainbowsensor.cn/en/mozu/370.html

#### 1.7 MD3005

L'MD3005 è un sensore prodotto dalla compagnia cinese Rainbow Technology Company, nonché il più completo utilizzato in questo lavoro di testi, che può essere usato per determinare il livello della qualità dell'aria ed in particolar modo è capace di effettuare il rilevamento di più gas come l'ammoniaca (NH<sub>3</sub>), composti organici volatili (TVOC) e l'acido solfidrico (H<sub>2</sub>S).

Nella tabella sottostante sono state riportate i diversi parametri di rilevamento, risoluzione e concentrazione massima associati ad ogni gas.

Sensor Model	MD3005		
Detected Gas	NH3	TVOC	H2S
Detection Range/ppm	0 ~ 300 ppm	0 ~ 50 ppm	0 ~ 5 ppm
Resolution/ppm	2 ppm	0.2 ppm	0.02 ppm
Maximum Permissible Concentration	1000ppm	500ppm	10ppm

Il modulo oltre a poter identificare e rilevare diversi gas, può essere usato anche per il monitoraggio di temperatura e umidità: queste caratteristiche lo rendono perfetto per determinare la qualità ambientale.

Altre funzionalità sono l'alta risoluzione e sensibilità, il basso consumo di energia e la costanza di utilizzo nel tempo. Le modalità di comunicazione verso l'esterno possono essere diverse (UART, analogica, PWM) ma tutte con un'ottima stabilità e linearità.

L MD3005 lavora a 3.3V, ha un consumo di energia inferiore a 200mW e un tempo di preriscaldamento di 5 minuti.



Nella progettazione del sensore completo contenente l MD3005 si è scelto di utilizzare la comunicazione seriale per il trasferimento dei dati in uscita tra quest'ultimo e il microcontrollore.

In questa modalità l MD3005 comunica i dati rilevati ogni 3 secondi, attraverso un unico dato di 40 bytes in formato esadecimale, secondo il seguente schema:

1 byte	1 byte	4 byte	4 byte	4 byte	4 byte	2	4 byte	4 byte	4 byte	4 byte	4 byte	1	1	1 byte
						byte						byte	byte	
frame	address	tem	Channel	Channel	Channel	Gas	Channel 1	Alarm	Channel	Channel	humidity	keep	frame	Check
head			1	2	3	type	concentra	/grade	2	3			tail	bit
			Voltage	Voltage	Voltage		tion		concent	concent				
									ration	ration				

Nel caso di dati formati da 4 bytes (come proprio per la concentrazione dei vari gas che il modulo può rilevare nell'aria), quando questi vengono convertiti dal formato esadecimale al formato decimale, il valore di ordine maggiore viene posto alla fine mentre quello di ordine minore all'inizio e il valore risultante viene diviso per 1000: in questo modo si ottiene i valori decimali delle concentrazioni rilevate.

Tutte le altre informazioni sul MD3005 possono essere recuperate nella documentazione messa a disposizione dal produttore sul proprio sito internet<sup>7</sup>.

-

<sup>&</sup>lt;sup>7</sup> http://rainbowsensor.cn/en/mozu/371.html

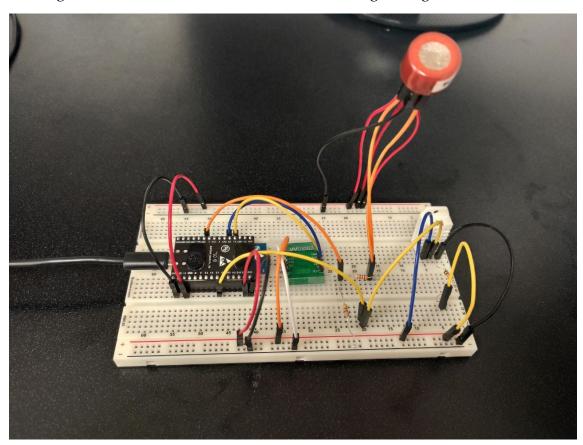
#### CAPITOLO 2: CONFIGURAZIONI DEI SENSORI PROGETTATI

### 2.1 SENSORE 1: RILEVAMENTO DI TEMPERATURA, UMIDITÀ, AMMONIACA E MONOSSIDO DI CARBONIO (ESP8266 + MQ7 + DHT22 + M1002)

Il primo sensore analizzato è composto dai seguenti moduli:

- ESP8266 in qualità di microcontrollore (MCU);
- MQ7 per il rilevamento del monossido di carbonio (CO) nell'aria;
- DHT22 per il monitoraggio di temperatura e umidità;
- M1002 per il rilevamento dell'ammoniaca (NH3) nell'aria.

Il collegamento tra i vari moduli è illustrato nella figura seguente



La comunicazione tra il microcontrollore e il DHT22 avviene attraverso un singlebus data in formato digitale; una resistenza di pull-up da  $10~\text{k}\Omega$  è posta tra il pin data del DHT22 e la tensione di riferimento.

L'ESP8266 e l MQ7 comunicano sfruttando l'uscita analogica del sensore, che va appunto collegata al pin analogico del microcontrollore (pin ADC).

Poiché 1 MQ7 lavora a 5V mentre l'ESP8266 a 3.3V, è necessario inserire un semplice partitore di tensione (costituito da una resistenza da 2.2 k $\Omega$  e una da 4.7 k $\Omega$ ) all'uscita del primo per poter collegare i pin analogici del modulo MQ7 a quello del controllore.

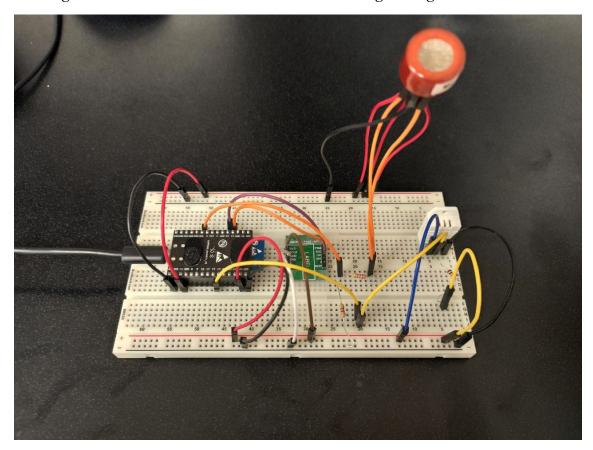
Infine, l M1002 e l'ESP8266 sono collegati tra loro utilizzando i pin relativi alla comunicazione seriale dell'uno e dell'altro.

### 2.2 SENSORE 2: RILEVAMENTO DI TEMPERATURA, UMIDITÀ, ETANOLO E MONOSSIDO DI CARBONIO (ESP8266 + MQ7 + DHT22 + M1005)

Il secondo sensore analizzato è composto dai seguenti moduli:

- ESP8266 in qualità di microcontrollore (MCU);
- MQ7 per il rilevamento del monossido di carbonio (CO) nell'aria;
- DHT22 per il monitoraggio di temperatura e umidità;
- M1005 per il rilevamento dell'etanolo (CH3CH2OH) nell'aria.

Il collegamento tra i vari moduli è illustrato nella figura seguente



La comunicazione tra il microcontrollore e il DHT22 avviene attraverso un singlebus data in formato digitale; una resistenza di pull-up da  $10~\text{k}\Omega$  è posta tra il pin data del DHT22 e la tensione di riferimento.

L'ESP8266 e l MQ7 comunicano sfruttando l'uscita analogica del sensore, che va appunto collegata al pin analogico del microcontrollore (pin ADC).

Poiché 1 MQ7 lavora a 5V mentre l'ESP8266 a 3.3V, è necessario inserire un semplice partitore di tensione (costituito da una resistenza da  $2.2~\rm k\Omega$  e una da  $4.7~\rm k\Omega$ 

 $k\Omega)$  all'uscita del primo per poter collegare i pin analogici del modulo MQ7 a quello del controllore.

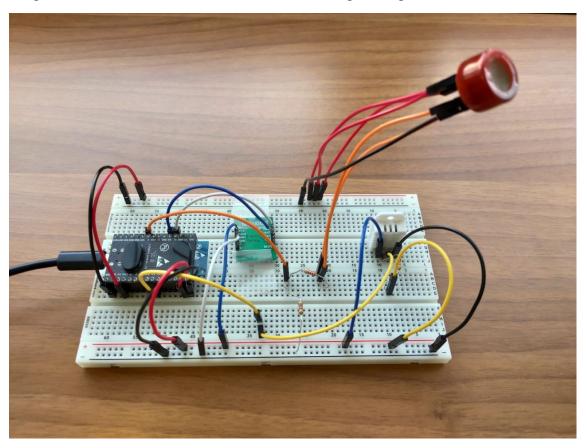
Infine, l M1005 e l'ESP8266 sono collegati tra loro utilizzando i pin relativi alla comunicazione seriale dell'uno e dell'altro.

# 2.3 SENSORE 3: RILEVAMENTO DI TEMPERATURA, UMIDITÀ, ACETONE E MONOSSIDO DI CARBONIO (ESP8266 + MQ7 + DHT22 + M1015)

Il terzo sensore analizzato è composto dai seguenti moduli:

- ESP8266 in qualità di microcontrollore (MCU);
- MQ7 per il rilevamento del monossido di carbonio (CO) nell'aria;
- DHT22 per il monitoraggio di temperatura e umidità;
- M1015 per il rilevamento dell'acetone (C<sub>3</sub>H<sub>6</sub>O) nell'aria.

Il collegamento tra i vari moduli è illustrato nella figura seguente



La comunicazione tra il microcontrollore e il DHT22 avviene attraverso un singlebus data in formato digitale; una resistenza di pull-up da  $10~\text{k}\Omega$  è posta tra il pin data del DHT22 e la tensione di riferimento.

L'ESP8266 e l MQ7 comunicano sfruttando l'uscita analogica del sensore, che va appunto collegata al pin analogico del microcontrollore (pin ADC).

Poiché 1 MQ7 lavora a 5V mentre l'ESP8266 a 3.3V, è necessario inserire un semplice partitore di tensione (costituito da una resistenza da 2.2 k $\Omega$  e una da 4.7 k $\Omega$ ) all'uscita del primo per poter collegare i pin analogici del modulo MQ7 a quello del controllore.

Infine, l M1015 e l'ESP8266 sono collegati tra loro utilizzando i pin relativi alla comunicazione seriale dell'uno e dell'altro.

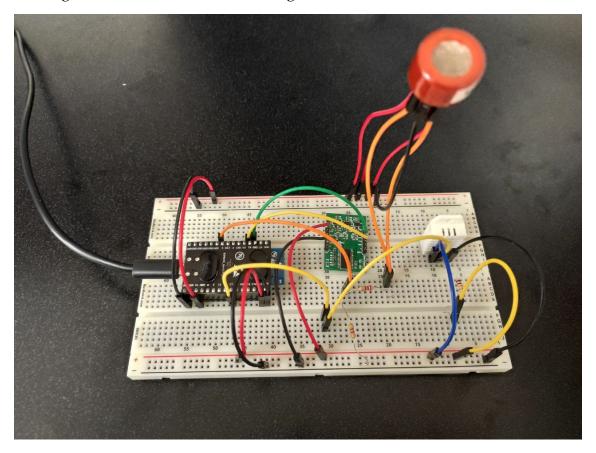
## 2.4 SENSORE 4: RILEVAMENTO DI TEMPERATURA, UMIDITÀ, AMMONIACA, GAS TVOC, ACIDO SOLFIDRICO E MONOSSIDO DI CARBONIO

(ESP8266 + MQ7 + DHT22 + MD3005)

Il quarto ed ultimo sensore analizzato è composto da:

- ESP8266 in qualità di microcontrollore (MCU);
- MQ7 per il rilevamento del monossido di carbonio (CO) nell'aria;
- DHT22 per il monitoraggio di temperatura e umidità;
- MD3005 per il rilevamento di ammoniaca (NH3), composti organici volatili (TVOC) e acido solfidrico (H2S).

Il collegamento tra i vari moduli è il seguente:



La comunicazione tra il controllore e il DHT22 avviene attraverso un single-bus data in formato digitale; una resistenza di pull-up da  $10~\text{k}\Omega$  è posta tra il pin data del DHT22 e la tensione di riferimento.

L'ESP8266 e l MQ7 comunicano sfruttando l'uscita analogica del sensore, che va appunto collegata al pin analogico del controllore (pin ADC).

Poiché 1 MQ7 lavora a 5V mentre l'ESP8266 a 3.3V, è necessario inserire un semplice partitore di tensione (costituito da una resistenza da 2.2 k $\Omega$  e una da 4.7 k $\Omega$ ) all'uscita del primo per poter collegare i pin analogici del sensore a quello del controllore.

Infine, l'MD3005 e l'ESP8266 sono collegati tra loro utilizzando i pin relativi alla comunicazione seriale dell'uno e dell'altro.

#### CAPITOLO 3: FIRMWARE E RISULTATI DEI SENSORI PROGETTATI

#### 3.1 SENSORE 1: ESP8266 + MQ7 + DHT22 + M1002

Per analizzare i dati monitorati e rilevati dal sensore è stato utilizzato l'ambiente di sviluppo integrato (IDE) di Arduino, settato appositamente per sfruttare le caratteristiche dell'ESP8266.

Lo sketch di Arduino creato per questo lavoro prevede la connessione del controllore alla rete WiFi in quanto i dati rilevati dal DHT22, dal M1002 e dal MQ7 vengono caricati in rete per poter essere monitorati anche da remoto. Successivamente avviene la lettura dei dati dei 3 sensori e il caricamento dei risultati ottenuti nel server.

Si ricordi come l MQ7 comunichi i dati in modalità analogica (e infatti è collegato al pin ADC dell'ESP8266 come illustrato nel paragrafo 2.1) mentre il DHT22 comunica i dati in output attraverso un pin digitale.

Per quanto riguarda la comunicazione dei dati rilevati dal M1002 è stata utilizzata la comunicazione seriale, come già mostrato nel paragrafo 2.1. Il modulo invia i dati al microcontrollore ogni 3 secondi attraverso un unico dato di 24 bytes in formato esadecimale, secondo il seguente schema:

1 byte	4 byte	4 byte	4 byte	4 byte	1	4 byte	1 byte	1 byte
					byte			
Frame	Temperature	Channel	Reserved	Reserved	Gas	gas	Reserved	Check
header		1 voltage			sort	concentration		Digit
(0xAA)								

Nel caso di dati formati da 4 bytes (come proprio per la concentrazione dell'ammoniaca nell'aria), quando questi vengono convertiti dal formato esadecimale al formato decimale, il valore di ordine maggiore viene posto alla fine mentre quello di ordine minore all'inizio e il valore risultante viene diviso per 1000: in questo modo si ottiene il valore decimale riferito alla concentrazione rilevata.

Lo sketch Arduino utilizzato per il rilevamento e la comunicazione dei parametri cercati è il seguente

```
//Includo le librerie necessarie per la connessione WiFi e il DHT22
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
```

```
#include <DHT.h>
//Definisco le costanti relative al DHT22
#define DHTPIN 12 //Pin a cui è connesso il DHT
#define DHTTYPE DHT22
                           //Tipo di DHT che stiamo utilizzando (DHT22)
//Inizializzo il DHT22
DHT dht(DHTPIN, DHTTYPE);
//Definisco le credenziali della rete WiFi a cui voglio collegarmi
const char* ssid = "xxxxx";
const char* password = "xxxxx";
//Creo il server alla porta 80 (http port)
ESP8266WebServer server(80);
//Definisco le variabili
byte byte1, byte2, byte3, byte4, byte5, byte6, byte7, byte8, byte9,
byte10, byte11, byte12, byte13, byte14, byte15, byte16, byte17, byte18,
byte19, byte20, byte21, byte22, byte23, byte24;
String page = "", frame header="aa", myString1, myString2, myString3,
myString4, myString5, myString6, myString7, myString8, myString9,
myString10, myString11, myString12, myString13, myString14, myString15,
myString16, myString17, myString18, myString19, myString20, myString21,
myString22, myString23, myString24, concatenated string, sensor string,
humidity, temperature, ammonia_ppm, co_ppm;
int hum, temp, analog data, Sensor Voltage, co data, ammonia byte data,
ammonia_data;
void setup(){
  //Inizializzo la porta seriale e il dht22
 Serial.begin(9600);
 dht.begin();
 //Attivo connessione WiFI
 WiFi.begin(ssid, password);
   while (WiFi.status() != WL CONNECTED) {
   delay(500);
   }
  server.on("/", [](){
      page = ""
                       +humidity+ "<br/>" +temperature+ "<br/>"
+sensor string+ "<br/>" +ammonia ppm+ "<br/>" +co ppm;
        server.send(200, "text/html", page);
        });
 server.begin();
void dht22(){
  //Leggo e stampo i valori dell'umidità e della temperatura
 hum = dht.readHumidity();
 humidity = "L'umidita' rilevata dal DHT22 e' pari al " + String(hum)
+ " %";
 temp = dht.readTemperature();
 temperature = "La temperatura rilevata dal DHT22 e' di " + String(temp)
+ " gradi Celsius";
```

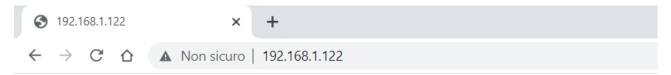
```
byte1 = (byte)Serial.read();
      myString1 = String(byte1, HEX);
      byte2 = (byte) Serial.read();
      myString2 = String(byte2, HEX);
      byte3 = (byte)Serial.read();
      myString3 = String(byte3, HEX);
      byte4 = (byte) Serial.read();
      myString4 = String(byte4, HEX);
      byte5 = (byte) Serial.read();
      myString5 = String(byte5, HEX);
      byte6 = (byte) Serial.read();
      myString6 = String(byte6, HEX);
      byte7 = (byte) Serial.read();
      myString7 = String(byte7, HEX);
      byte8 = (byte) Serial.read();
      myString8 = String(byte8, HEX);
      byte9 = (byte) Serial.read();
      myString9 = String(byte9, HEX);
      byte10 = (byte) Serial.read();
      myString10 = String(byte10, HEX);
      byte11 = (byte) Serial.read();
      myString11 = String(byte11, HEX);
      byte12 = (byte) Serial.read();
      myString12 = String(byte12, HEX);
      byte13 = (byte) Serial.read();
      myString13 = String(byte13, HEX);
      byte14 = (byte) Serial.read();
      myString14 = String(byte14, HEX);
      byte15 = (byte) Serial.read();
      myString15 = String(byte15, HEX);
      byte16 = (byte) Serial.read();
      myString16 = String(byte16, HEX);
      byte17 = (byte) Serial.read();
      myString17 = String(byte17, HEX);
      byte18 = (byte) Serial.read();
      myString18 = String(byte18, HEX);
      byte19 = (byte)Serial.read();
      myString19 = String(byte19, HEX);
      byte20 = (byte)Serial.read();
      myString20 = String(byte20, HEX);
      byte21 = (byte) Serial.read();
      myString21 = String(byte21, HEX);
      byte22 = (byte) Serial.read();
      myString22 = String(byte22, HEX);
      byte23 = (byte)Serial.read();
      myString23 = String(byte23, HEX);
      byte24 = (byte)Serial.read();
      myString24 = String(byte24, HEX);
//Concateno le 24 stringhe ottenute
\verb|concatenated_string=myString1+myString2+myString3+myString4+myString5+myString3+myString4+myString5+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+myString6+my
\verb|myString6+myString7+myString8+myString9+myString10+myString11+myString9+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+myString10+m
12+myString13+myString14+myString15+myString16+myString17+myString18+m
yString19+myString20+myString21+myString22+myString23+myString24;
```

void m1002(){

//Leggo i valori da seriale

```
//Controllo il frame header della stringa esadecimale che dovrei ottenere
if (concatenated string.startsWith(frame header)){
//Creo la stringa esadecimale da printare
sensor string = "La stringa del m1002 in formato esadecimale e': "
+myString1+" "+myString2+" "+myString3+" "+myString4+" "+myString5+"
"+myString6+" "+myString7+" "+myString8+" "+myString9+" "+myString10+"
"+myString11+" "+myString12+"
"+myString15+" "+myString16+"
"+myString19+" "+myString20+"
                                      "+myString13+"
"+myString17+"
                                                              "+myString14+"
                                                              "+myString18+"
                                         "+myString21+"
                                                              "+myString22+"
"+myString23+" "+myString24;
  //Converto il valore della concentrazione ottenuto da esadecimale a
decimale
  ammonia byte data = 0;
  ammonia byte data = (byte22 << 32) | (byte21 << 16) | (byte20 << 8) |
  ammonia data = ammonia byte data/1000;
  //Stampo il risultato ottenuto
  ammonia ppm = "La concentrazione di mmoniaca rilevata dal m1002 e'
pari a " + String(ammonia data) + " ppm";
  }
void MQ7() {
  //Imposto il pin analogico dell'ESP8266 come input
  pinMode(A0, INPUT);
  //Leggo il valore analogico dal pin
  analog data = analogRead(A0);
  //Ricavo il voltaggio dal valore analogico
  Sensor Voltage = analog data * 3.3 / 4095;
  //Calcolo le PPM del CO rilevato
  co data = 3.027*pow(2.7182, (1.0698*Sensor Voltage));
  //Stampo il risultato ottenuto
  co ppm = "La concentrazione di Monossido di Carbonio (CO) rilevata dal
MQ7 e' pari a " + String(co data) + " ppm";
void loop() {
  dht22();
  m1002();
  MQ7();
  server.handleClient(); //Carico i risultati ottenuti dai 3 sensori
  delay(3000);
```

I valori di umidità, temperatura, concentrazione di ammoniaca e monossido di carbonio nell'aria, in condizioni ambientali normali, sono stati:



L'umidita' rilevata dal DHT22 e' pari al 39 %

La temperatura rilevata dal DHT22 e' di 25 gradi Celsius

La stringa del m1002 in formato esadecimale e': aa a8 61 0 0 f6 9 0 0 1c 0 0 0 56 0 0 0 0 0 0 0 0 24

La concentrazione di ammoniaca rilevata dal m1002 e' pari a 0 ppm

La concentrazione di Monossido di Carbonio (CO) rilevata dal MQ7 e' pari a 3 ppm

Andando a stimolare la presenza di ammoniaca nell'aria si è invece ottenuto:



L'umidita' rilevata dal DHT22 e' pari al 41 %

La temperatura rilevata dal DHT22 e' di 25 gradi Celsius

La stringa del m1002 in formato esadecimale e': aa a8 61 0 0 79 9 0 0 1a 0 0 0 51 0 0 0 3 68 f b 0 0 a0

La concentrazione di ammoniaca rilevata dal m1002 e' pari a 49 ppm

La concentrazione di Monossido di Carbonio (CO) rilevata dal MQ7 e' pari a 3 ppm

Analogamente stimolando la presenza di CO, è variata la concentrazione rilevata dal MQ7:



L'umidita' rilevata dal DHT22 e' pari al 41 %

La temperatura rilevata dal DHT22 e' di 25 gradi Celsius

La stringa del m1002 in formato esadecimale e': aa a8 61 0 0 b1 8 0 0 16 0 0 0 61 0 0 0 0 0 0 0 0 0 e3

La concentrazione di ammoniaca rilevata dal m1002 e' pari a 0 ppm

La concentrazione di Monossido di Carbonio (CO) rilevata dal MQ7 e' pari a 5 ppm

Si noti quindi come il sensore progettato sia effettivamente ricettivo a variazioni della concentrazione di ammoniaca e monossido di carbonio nell'aria e risponda correttamente a tali situazioni.

#### 3.2 SENSORE 2: ESP8266 + MQ7 + DHT22 + M1005

Per analizzare i dati monitorati e rilevati dal sensore è stato utilizzato l'ambiente di sviluppo integrato (IDE) di Arduino, settato appositamente per sfruttare le caratteristiche dell'ESP8266.

Lo sketch di Arduino creato per questo lavoro prevede la connessione del controllore alla rete WiFi in quanto i dati rilevati dal DHT22, dal M1005 e dal MQ7 vengono caricati in rete per poter essere monitorati anche da remoto. Successivamente avviene la lettura dei dati dei 3 sensori e il caricamento dei risultati ottenuti nel server.

Si ricordi come l MQ7 comunichi i dati in modalità analogica (e infatti è collegato al pin ADC dell'ESP8266 come illustrato nel paragrafo 2.2) mentre il DHT22 comunica i dati in output attraverso un pin digitale.

Per quanto riguarda la comunicazione dei dati rilevati dal M1005 è stata utilizzata la comunicazione seriale, come già mostrato nel paragrafo 2.2. Il modulo invia i dati al microcontrollore ogni 3 secondi attraverso un unico dato di 24 bytes in formato esadecimale, secondo il seguente schema:

1 byte	4 bytes	4 bytes	4 bytes	4 bytes	1 byte	4 bytes	1 byte	1 byte
Frame	temperature	Channel 1	Keep	Keep	Gas type	Gas	reserve	Check
header		voltage				concentration		Digit
(0xAA)								

Nel caso di dati formati da 4 bytes (come proprio per la concentrazione dell'etanolo nell'aria), quando questi vengono convertiti dal formato esadecimale al formato decimale, il valore di ordine maggiore viene posto alla fine mentre quello di ordine minore all'inizio e il valore risultante viene diviso per 1000: in questo modo si ottiene il valore decimale riferito alla grandezza rilevata.

Lo sketch Arduino utilizzato per il rilevamento e la comunicazione dei parametri cercati è il seguente

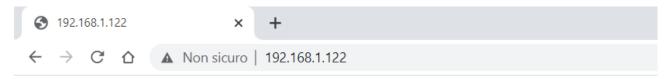
```
//Includo le librerie necessarie per la connessione WiFi e il DHT22
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <DHT.h>
```

```
//Definisco le costanti relative al DHT22
#define DHTPIN 12 //Pin a cui è connesso il DHT
#define DHTTYPE DHT22
                           //Tipo di DHT che stiamo utilizzando (DHT22)
//Inizializzo il DHT22
DHT dht(DHTPIN, DHTTYPE);
//Definisco le credenziali della rete WiFi a cui voglio collegarmi
const char* ssid = "xxxx";
const char* password = "xxxx";
//Creo il server alla porta 80 (http port)
ESP8266WebServer server(80);
//Definisco le variabili
byte byte1, byte2, byte3, byte4, byte5, byte6, byte7, byte8, byte9,
byte10, byte11, byte12, byte13, byte14, byte15, byte16, byte17, byte18,
byte19, byte20, byte21, byte22, byte23, byte24;
String page = "", frame header="aa", myString1, myString2, myString3,
myString4, myString5, myString6, myString7, myString8, myString9,
myString10, myString11, myString12, myString13, myString14, myString15,
myString16, myString17, myString18, myString19, myString20, myString21,
myString22, myString23, myString24, concatenated string, sensor string,
humidity, temperature, ethanol ppm, co ppm;
int hum, temp, analog data, Sensor Voltage, co data, ethanol byte data,
ethanol data;
void setup(){
  //Inizializzo la porta seriale e il dht22
  Serial.begin(9600);
  dht.begin();
  //Attivo connessione WiFI
  WiFi.begin(ssid, password);
   while (WiFi.status() != WL CONNECTED) {
   delay(500);
   }
  server.on("/", [](){
                                   "<br/>" +temperature+ "<br/>"
                       +humidity+
+sensor string+ "<br/>" +ethanol_ppm+ "<br/>" +co_ppm;
        server.send(200, "text/html", page);
  server.begin();
void dht22(){
  //Leggo e stampo i valori dell'umidità e della temperatura
  hum = dht.readHumidity();
  humidity = "L'umidita' rilevata dal DHT22 e' pari al " + String(hum)
+ " %";
  temp = dht.readTemperature();
  temperature = "La temperatura rilevata dal DHT22 e' di " + String(temp)
+ " gradi Celsius";
```

```
void M1005(){
  //Leggo i valori da seriale
  byte1 = (byte)Serial.read();
  myString1 = String(byte1, HEX);
  byte2 = (byte) Serial.read();
  myString2 = String(byte2, HEX);
  byte3 = (byte)Serial.read();
  myString3 = String(byte3, HEX);
  byte4 = (byte) Serial.read();
  myString4 = String(byte4, HEX);
  byte5 = (byte) Serial.read();
  myString5 = String(byte5, HEX);
  byte6 = (byte) Serial.read();
  myString6 = String(byte6, HEX);
  byte7 = (byte) Serial.read();
  myString7 = String(byte7, HEX);
  byte8 = (byte) Serial.read();
  myString8 = String(byte8, HEX);
  byte9 = (byte) Serial.read();
  myString9 = String(byte9, HEX);
  byte10 = (byte) Serial.read();
  myString10 = String(byte10, HEX);
  byte11 = (byte) Serial.read();
  myString11 = String(byte11, HEX);
  byte12 = (byte) Serial.read();
  myString12 = String(byte12, HEX);
  byte13 = (byte) Serial.read();
  myString13 = String(byte13, HEX);
  byte14 = (byte) Serial.read();
  myString14 = String(byte14, HEX);
  byte15 = (byte) Serial.read();
  myString15 = String(byte15, HEX);
  byte16 = (byte) Serial.read();
  myString16 = String(byte16, HEX);
  byte17 = (byte)Serial.read();
  myString17 = String(byte17, HEX);
  byte18 = (byte) Serial.read();
  myString18 = String(byte18, HEX);
  byte19 = (byte) Serial.read();
  myString19 = String(byte19, HEX);
  byte20 = (byte)Serial.read();
  myString20 = String(byte20, HEX);
  byte21 = (byte) Serial.read();
  myString21 = String(byte21, HEX);
  byte22 = (byte)Serial.read();
  myString22 = String(byte22, HEX);
  byte23 = (byte)Serial.read();
  myString23 = String(byte23, HEX);
  byte24 = (byte) Serial.read();
  myString24 = String(byte24, HEX);
//Concateno le 24 stringhe ottenute
  concatenated string
myString1+myString2+myString3+myString4+myString5+myString6+myString7+
myString8+myString9+myString10+myString11+myString12+myString13+myStri
ng14+myString15+myString16+myString17+myString18+myString19+myString20
+myString21+myString22+myString23+myString24;
```

```
//Controllo il frame header della stringa esadecimale che dovrei ottenere
 if (concatenated string.startsWith(frame header)){
//Creo la stringa esadecimale da printare
  sensor_string = "La stringa del M1005 in formato esadecimale e': "
+myString1+" "+myString2+" "+myString3+" "+myString4+" "+myString5+"
"+myString6+" "+myString7+" "+myString8+" "+myString9+" "+myString10+"
               _ ____+myString12+"
"+myString11+"
                                  "+myString13+"
                                                         "+myString14+"
"+myString15+"
                  "+myString16+"
                                      "+myString17+"
                                                         "+myString18+"
"+myString19+" "+myString20+"
                                     "+myString21+"
                                                         "+myString22+"
"+myString23+" "+myString24;
  //Converto il valore della concentrazione ottenuto da esadecimale a
decimale
 ethanol byte data = 0;
  \tt ethanol\_byte\_data = (byte22<<32) \mid (byte21<<16) \mid (byte20<<8) \mid
  ethanol data = ethanol byte data/1000;
 //Stampo i risultati ottenuti
 ethanol ppm = "La concentrazione di etanolo rilevata dal M1005 e' pari
a " + String(ethanol data) + " ppm";
}
void MQ7() {
  //Imposto il pin analogico dell'ESP8266 come input
  pinMode(A0, INPUT);
  //Leggo il valore analogico dal pin
  analog data = analogRead(A0);
  //Ricavo il voltaggio dal valore analogico
  Sensor Voltage = analog data * 3.3 / 4095;
  //Calcolo le PPM del CO rilevato
  co data = 3.027*pow(2.7182, (1.0698*Sensor Voltage));
  //Stampo il risultato ottenuto
  co ppm = "La concentrazione di Monossido di Carbonio (CO) rilevata dal
MQ7 e' pari a " + String(co data) + " ppm";
  server.handleClient();
void loop(){
  dht22();
 M1005();
 MQ7();
  server.handleClient(); //Carico i risultati ottenuti dai 3 sensori
 delay(3000);
```

I risultati ottenuti in condizioni ambientali normali sono stati:



L'umidita' rilevata dal DHT22 e' pari al 40 %

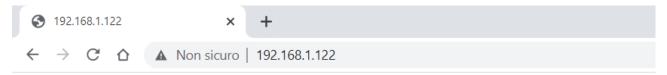
La temperatura rilevata dal DHT22 e' di 25 gradi Celsius

La stringa del M1005 in formato esadecimale e': aa a8 61 0 0 99 2 0 0 71 a 0 0 5e 0 0 0 0 0 0 0 0 27

La concentrazione di etanolo rilevata dal M1005 e' pari a 0 ppm

La concentrazione di Monossido di Carbonio (CO) rilevata dal MQ7 e' pari a 3 ppm

Stimolando la presenza di etanolo nell'aria invece si è ottenuto:



L'umidita' rilevata dal DHT22 e' pari al 40 %

La temperatura rilevata dal DHT22 e' di 25 gradi Celsius

La stringa del M1005 in formato esadecimale e' aa a8 61 0 0 10 8 0 0 1c 0 0 0 56 0 0 0 0 b d0 7 0 0 3d

La concentrazione di etanolo rilevata dal M1005 e' pari a 2 ppm

La concentrazione di Monossido di Carbonio (CO) rilevata dal MQ7 e' pari a 3 ppm

#### Analogamente per il monossido di carbonio



L'umidita' rilevata dal DHT22 e' pari al 40 %

La temperatura rilevata dal DHT22 e' di 25 gradi Celsius

La stringa del M1005 in formato esadecimale e': aa a8 61 0 0 2f 1 0 0 71 a 0 0 5e 0 0 0 0 0 0 0 0 bc

La concentrazione di etanolo rilevata dal M1005 e' pari a 0 ppm

La concentrazione di Monossido di Carbonio (CO) rilevata dal MQ7 e' pari a 6 ppm

Si noti quindi come il sensore progettato sia effettivamente ricettivo a variazioni della concentrazione di etanolo e monossido di carbonio nell'aria e risponda correttamente a tali situazioni.

#### 3.3 SENSORE 3: ESP8266 + MQ7 + DHT22 + M1015

Per analizzare i dati monitorati e rilevati dal sensore è stato utilizzato l'ambiente di sviluppo integrato (IDE) di Arduino, settato appositamente per sfruttare le caratteristiche dell'ESP8266.

Lo sketch di Arduino creato per questo lavoro prevede la connessione del controllore alla rete WiFi in quanto i dati rilevati dal DHT22, dal M1015 e dal MQ7 vengono caricati in rete per poter essere monitorati anche da remoto. Successivamente avviene la lettura dei dati dei 3 sensori e il caricamento dei risultati ottenuti nel server.

Si ricordi come l MQ7 comunichi i dati in modalità analogica (e infatti è collegato al pin ADC dell'ESP8266 come illustrato nel paragrafo 2.3) mentre il DHT22 comunica i dati in output attraverso un pin digitale.

Per quanto riguarda la comunicazione dei dati rilevati dal M1015 è stata utilizzata la comunicazione seriale, come già mostrato nel paragrafo 2.3. Il modulo invia i dati al microcontrollore ogni 3 secondi attraverso un unico dato di 24 bytes in formato esadecimale, secondo il seguente schema:

1 byte	4 byte	4 byte	4 byte	4 byte	1	4 byte	1 byte	1 byte
					byte			
frame	temperature	humidity	Channel	Channel	Gas	Gas	Alarm/level	Check
header			1	2 voltage	type	concentration		Digit
(0xAA)			voltage	(reserved)				

Nel caso di dati formati da 4 bytes (come proprio per la concentrazione dell'acetone nell'aria), quando questi vengono convertiti dal formato esadecimale al formato decimale, il valore di ordine maggiore viene posto alla fine mentre quello di ordine minore all'inizio e il valore risultante viene diviso per 1000: in questo modo si ottiene il valore decimale riferito alla grandezza rilevata.

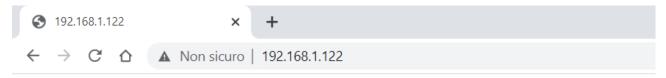
Lo sketch Arduino utilizzato per il rilevamento e la comunicazione dei parametri cercati è il seguente

```
//Inizializzo il DHT22
DHT dht(DHTPIN, DHTTYPE);
//Definisco le credenziali della rete WiFi a cui voglio collegarmi
const char* ssid = "xxxxx";
const char* password = "xxxxx";
//Creo il server alla porta 80 (http port)
ESP8266WebServer server(80);
//Definisco le variabili
byte byte1, byte2, byte3, byte4, byte5, byte6, byte7, byte8, byte9,
byte10, byte11, byte12, byte13, byte14, byte15, byte16, byte17, byte18,
byte19, byte20, byte21, byte22, byte23, byte24;
String page = "", frame header="aa", myString1, myString2, myString3,
myString4, myString5, myString6, myString7, myString8, myString9,
myString10, myString11, myString12, myString13, myString14, myString15,
myString16, myString17, myString18, myString19, myString20, myString21,
myString22, myString23, myString24, concatenated string, sensor string,
humidity, temperature, acetone ppm, co ppm;
int hum, temp, analog data, Sensor Voltage, co data;
float acetone byte data, acetone data;
void setup(){
  //Inizializzo la porta seriale e il dht22
  Serial.begin (9600);
  dht.begin();
  //Attivo connessione WiFI
  WiFi.begin(ssid, password);
   while (WiFi.status() != WL CONNECTED) {
   delay(500);
   }
  server.on("/", [](){
       page = ""
                        +humidity+
                                    "<br/>" +temperature+ "<br/>"
+sensor string+ "<br/>" +acetone ppm+ "<br/>" +co ppm;
        server.send(200, "text/html", page);
        });
  server.begin();
void dht22(){
  //Leggo e stampo i valori dell'umidità e della temperatura
  hum = dht.readHumidity();
  humidity = "L'umidita' rilevata dal DHT22 e' pari al " + String(hum)
+ " %";
  temp = dht.readTemperature();
  temperature = "La temperatura rilevata dal DHT22 e' di " + String(temp)
+ " gradi Celsius";
}
void m1015() {
  //Leggo i valori da seriale
  byte1 = (byte)Serial.read();
 myString1 = String(byte1, HEX);
 byte2 = (byte)Serial.read();
 myString2 = String(byte2, HEX);
```

```
byte3 = (byte)Serial.read();
  myString3 = String(byte3, HEX);
  byte4 = (byte)Serial.read();
  myString4 = String(byte4, HEX);
  byte5 = (byte) Serial.read();
  myString5 = String(byte5, HEX);
  byte6 = (byte) Serial.read();
  myString6 = String(byte6, HEX);
  byte7 = (byte) Serial.read();
  myString7 = String(byte7, HEX);
  byte8 = (byte) Serial.read();
  myString8 = String(byte8, HEX);
  byte9 = (byte) Serial.read();
  myString9 = String(byte9, HEX);
  byte10 = (byte) Serial.read();
  myString10 = String(byte10, HEX);
  byte11 = (byte) Serial.read();
  myString11 = String(byte11, HEX);
  byte12 = (byte) Serial.read();
  myString12 = String(byte12, HEX);
  byte13 = (byte) Serial.read();
  myString13 = String(byte13, HEX);
  byte14 = (byte) Serial.read();
  myString14 = String(byte14, HEX);
  byte15 = (byte) Serial.read();
  myString15 = String(byte15, HEX);
  byte16 = (byte) Serial.read();
  myString16 = String(byte16,HEX);
  byte17 = (byte) Serial.read();
  myString17 = String(byte17, HEX);
  byte18 = (byte) Serial.read();
  myString18 = String(byte18, HEX);
  byte19 = (byte) Serial.read();
  myString19 = String(byte19, HEX);
  byte20 = (byte) Serial.read();
  myString20 = String(byte20, HEX);
  byte21 = (byte)Serial.read();
  myString21 = String(byte21, HEX);
  byte22 = (byte)Serial.read();
  myString22 = String(byte22, HEX);
  byte23 = (byte)Serial.read();
  myString23 = String(byte23, HEX);
  byte24 = (byte) Serial.read();
  myString24 = String(byte24, HEX);
//Concateno le 24 stringhe ottenute
  concatenated string
myString1+myString2+myString3+myString4+myString5+myString6+myString7+
myString8+myString9+myString10+myString11+myString12+myString13+myStri
ng14+myString15+myString16+myString17+myString18+myString19+myString20
+myString21+myString22+myString23+myString24;
  //Controllo il frame header della stringa esadecimale che dovrei
ottenere
  if (concatenated string.startsWith(frame header)){
```

```
//Creo la stringa esadecimale da printare
  sensor string = "La stringa del M1015 in formato esadecimale e': "
+myString1+" "+myString2+" "+myString3+" "+myString4+" "+myString5+" "+myString6+" "+myString7+" "+myString8+" "+myString9+" "+myString10+"
                - +myString12+"
"+myString11+"
                                       "+myString13+"
                                                            "+myString14+"
"+myString15+"
                   "+myString16+"
                                        "+myString17+"
                                                            "+myString18+"
                "+myString20+"
"+myString19+"
                                       "+myString21+"
                                                            "+myString22+"
"+myString23+" "+myString24;
  //Converto il valore della concentrazione ottenuto da esadecimale a
decimale
 acetone byte data = 0;
  acetone byte data = (byte22 << 32) | (byte21 << 16) | (byte20 << 8) |
(byte19);
 acetone data = acetone byte data/1000;
 //Stampo e carico i risultati ottenuti
 acetone ppm = "La concentrazione di acetone rilevata dal M1015 e' pari
a " + String(acetone data,1) + " ppm";
}
void MQ7() {
  //Imposto il pin analogico dell'ESP8266 come input
  pinMode(A0, INPUT);
  //Leggo il valore analogico dal pin
  analog data = analogRead(A0);
  //Ricavo il voltaggio dal valore analogico
  Sensor Voltage = analog data * 3.3 / 4095;
  //Calcolo le PPM del CO rilevato
  co data = 3.027*pow(2.7182, (1.0698*Sensor Voltage));
  //Stampo e carico il risultato ottenuto
  co ppm = "La concentrazione di Monossido di Carbonio (CO) rilevata dal
MQ7 e' pari a " + String(co data) + " ppm";
  server.handleClient();
void loop() {
  dht22();
  m1015();
 MQ7();
  server.handleClient(); //Carico i risultati ottenuti dai 3 sensori
 delay(3000);
}
```

I valori di umidità, temperatura, concentrazione di acetone e monossido di carbonio nell'aria, in condizioni ambientali normali, sono stati



L'umidita' rilevata dal DHT22 e' pari al 46 %

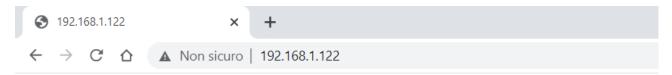
La temperatura rilevata dal DHT22 e' di 26 gradi Celsius

La stringa del M1015 in formato esadecimale e': aa a8 61 0 0 69 1 0 0 1b 0 0 0 63 0 0 0 0 0 0 0 0 9b

La concentrazione di acetone rilevata dal M1015 e' pari a 0.0 ppm

La concentrazione di Monossido di Carbonio (CO) rilevata dal MQ7 e' pari a 3 ppm

Andando a stimolare la presenza di acetone nell'aria si è invece ottenuto



L'umidita' rilevata dal DHT22 e' pari al 38 %

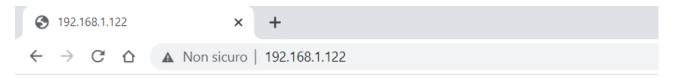
La temperatura rilevata dal DHT22 e' di 25 gradi Celsius

La stringa del M1015 in formato esadecimale e': aa a8 61 0 0 10 8 0 0 1c 0 0 0 56 0 0 0 0 b 84 3 0 0 3d

La concentrazione di acetone rilevata dal M1015 e' pari a 0.9 ppm

La concentrazione di Monossido di Carbonio (CO) rilevata dal MQ7 e' pari a 3 ppm

Analogamente stimolando la presenza di CO, è variata la concentrazione rilevata dal MQ7



L'umidita' rilevata dal DHT22 e' pari al 41 %

La temperatura rilevata dal DHT22 e' di 25 gradi Celsius

La stringa del M1015 in formato esadecimale e': aa a8 61 0 0 67 1 0 0 1c 0 0 0 69 0 0 0 0 0 0 0 0 0 0 0 a0 La concentrazione di acetone rilevata dal M1015 e' pari a 0.0 ppm

La concentrazione di Monossido di Carbonio (CO) rilevata dal MQ7 e' pari a 11 ppm

Si noti quindi come il sensore progettato sia effettivamente ricettivo a variazioni della concentrazione di acetone e monossido di carbonio nell'aria e risponda correttamente a tali situazioni.

#### 3.4 SENSORE 4: ESP8266 + MQ7 + DHT22 + MD3005

Per analizzare i dati monitorati e rilevati dal sensore è stato utilizzato l'ambiente di sviluppo integrato (IDE) di Arduino, settato appositamente per sfruttare le caratteristiche dell'ESP8266.

Lo sketch di Arduino creato per questo lavoro prevede la connessione del controllore alla rete WiFi in quanto i dati rilevati dal DHT22, dal M1015 e dal MQ7 vengono caricati in rete per poter essere monitorati anche da remoto. Successivamente avviene la lettura dei dati dei 3 sensori e il caricamento dei risultati ottenuti nel server.

Si ricordi come l MQ7 comunichi i dati in modalità analogica (e infatti è collegato al pin ADC dell'ESP8266 come illustrato nel paragrafo 2.4) mentre il DHT22 comunica i dati in output attraverso un pin digitale.

Per quanto riguarda la comunicazione dei dati rilevati dal MD3005 è stata utilizzata la comunicazione seriale, come già mostrato nel paragrafo 2.4.

In questa modalità l MD3005 comunica i dati rilevati ogni 3 secondi, attraverso un unico dato di 40 bytes in formato esadecimale, secondo il seguente schema:

1 byte	1 byte	4 byte	4 byte	4 byte	4 byte	2	4 byte	4 byte	4 byte	4 byte	4 byte	1	1	1 byte
						byte						byte	byte	
frame	address	tem	Channel	Channel	Channel	Gas	Channel 1	Alarm	Channel	Channel	humidity	keep	frame	Check
head			1	2	3	type	concentra	/grade	2	3			tail	bit
			Voltage	Voltage	Voltage		tion		concent	concent				
									ration	ration				

Nel caso di dati formati da 4 bytes (come proprio per la concentrazione dei vari gas che il modulo può rilevare nell'aria), quando questi vengono convertiti dal formato esadecimale al formato decimale, il valore di ordine maggiore viene posto alla fine mentre quello di ordine minore all'inizio e il valore risultante viene diviso per 1000: in questo modo si ottiene i valori decimali delle concentrazioni rilevate.

Lo sketch Arduino utilizzato per il rilevamento e la comunicazione dei parametri cercati è il seguente

```
//Inizializzo il DHT22
DHT dht(DHTPIN, DHTTYPE);
//Definisco le credenziali della rete WiFi a cui voglio collegarmi
const char* ssid = "xxxx";
const char* password = "xxxx";
//Creo il server alla porta 80 (http port)
ESP8266WebServer server(80);
//Definisco le variabili
byte byte1, byte2, byte3, byte4, byte5, byte6, byte7, byte8, byte9,
byte10, byte11, byte12, byte13, byte14, byte15, byte16, byte17, byte18,
byte19, byte20, byte21, byte22, byte23, byte24, byte25, byte26, byte27,
byte28, byte29, byte30, byte31, byte32, byte33, byte34, byte35, byte36,
byte37, byte38, byte39, byte40;
String page = "", frame header="aa", myString1, myString2, myString3,
myString4, myString5, myString6, myString7, myString8, myString9,
myString10, myString11, myString12, myString13, myString14, myString15,
myString16, myString17, myString18, myString19, myString20, myString21,
myString22, myString23, myString24, myString25, myString26, myString27,
myString28, myString29, myString30, myString31, myString32, myString33,
myString34, myString35, myString36, myString37, myString38, myString39,
myString40, concatenated string, sensor string, humidity, temperature,
ammonia ppm, nh3 ppm, tvoc ppm, h2s ppm, co ppm;
int hum, temp, nh3_byte_data, nh3_data, analog_data, Sensor_Voltage,
co data;
double tvoc_byte_data, h2s_byte_data, tvoc_data, h2s data;
void setup(){
  //Inizializzo la porta seriale e il dht22
  Serial.begin(38400);
  dht.begin();
  //Attivo connessione WiFI
  WiFi.begin(ssid, password);
    while (WiFi.status() != WL CONNECTED) {
   delay(500);
    }
  server.on("/", [](){
                       +humidity+ "<br/>" +temperature+ "<br/>"
+sensor string+ "<br/>" +nh3 ppm+ "<br/>" +tvoc ppm+ "<br/>" +h2s ppm+
"<br/>"-+co ppm;
        server.send(200, "text/html", page);
  server.begin();
void dht22(){
  //Leggo e stampo i valori dell'umidità e della temperatura
 hum = dht.readHumidity();
 humidity = "L'umidita' rilevata dal DHT22 e' pari al " + String(hum)
+ " %";
  temp = dht.readTemperature();
  temperature = "La temperatura rilevata dal DHT22 e' di " + String(temp)
+ " gradi Celsius";
}
```

```
void md3005() {
      //Leggo i valori da seriale
      byte1 = (byte) Serial.read();
      myString1 = String(byte1, HEX);
      byte2 = (byte)Serial.read();
      myString2=String(byte2, HEX);
      byte3 = (byte)Serial.read();
      myString3=String(byte3, HEX);
      byte4 = (byte)Serial.read();
      myString4=String(byte4, HEX);
      byte5 = (byte) Serial.read();
      myString5=String(byte5, HEX);
      byte6 = (byte)Serial.read();
      myString6=String(byte6, HEX);
      byte7 = (byte)Serial.read();
      myString7=String(byte7, HEX);
      byte8 = (byte)Serial.read();
      myString8=String(byte8, HEX);
      byte9 = (byte) Serial.read();
      myString9=String(byte9, HEX);
      byte10 = (byte) Serial.read();
      myString10=String(byte10,HEX);
      byte11 = (byte) Serial.read();
      myString11=String(byte11, HEX);
      byte12 = (byte)Serial.read();
      myString12=String(byte12, HEX);
      byte13 = (byte) Serial.read();
      myString13=String(byte13, HEX);
      byte14 = (byte)Serial.read();
      myString14=String(byte14, HEX);
      byte15 = (byte) Serial.read();
      myString15=String(byte15,HEX);
      byte16 = (byte) Serial.read();
      myString16=String(byte16, HEX);
      byte17 = (byte) Serial.read();
      myString17=String(byte17, HEX);
      byte18 = (byte) Serial.read();
      myString18=String(byte18, HEX);
      byte19 = (byte) Serial.read();
      myString19=String(byte19, HEX);
      byte20 = (byte) Serial.read();
      myString20=String(byte20, HEX);
      byte21 = (byte) Serial.read();
      myString21=String(byte21, HEX);
      byte22 = (byte) Serial.read();
      myString22=String(byte22, HEX);
      byte23 = (byte)Serial.read();
      myString23=String(byte23, HEX);
      byte24 = (byte)Serial.read();
      myString24=String(byte24, HEX);
      byte25 = (byte) Serial.read();
      myString25=String(byte25, HEX);
      byte26 = (byte)Serial.read();
      myString26=String(byte26, HEX);
```

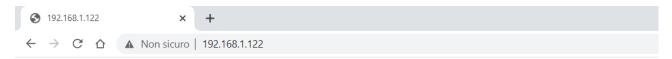
byte27 = (byte) Serial.read();
myString27=String(byte27, HEX);
byte28 = (byte) Serial.read();
myString28=String(byte28, HEX);

```
myString29=String(byte29, HEX);
           byte30 = (byte)Serial.read();
           myString30=String(byte30,HEX);
           byte31 = (byte)Serial.read();
           myString31=String(byte31,HEX);
           byte32 = (byte) Serial.read();
           myString32=String(byte32, HEX);
           byte33 = (byte) Serial.read();
           myString33=String(byte33, HEX);
           byte34 = (byte) Serial.read();
           myString34=String(byte34,HEX);
           byte35 = (byte)Serial.read();
           myString35=String(byte35,HEX);
           byte36 = (byte)Serial.read();
           myString36=String(byte36, HEX);
           byte37 = (byte) Serial.read();
           myString37=String(byte37, HEX);
           byte38 = (byte) Serial.read();
           myString38=String(byte38, HEX);
           byte39 = (byte)Serial.read();
           myString39=String(byte39, HEX);
           byte40 = (byte) Serial.read();
           myString40=String(byte40, HEX);
 //Concateno le 24 stringhe ottenute
concatenated string=myString1+myString2+myString3+myString4+myString5+
\verb|myString6+myString7+myString8+myString9+myString10+myString11+myString9|\\
12 + myString13 + myString14 + myString15 + myString16 + myString17 + myString18 
yString19+myString20+myString21+myString22+myString23+myString24+myStr
ing25+myString26+myString27+myString28+myString29+myString30+myString3
1+myString32+myString33+myString34+myString35+myString36+myString37+my
String38+myString39+myString40;
    //Controllo il frame header della stringa esadecimale che dovrei
ottenere
   if (concatenated string.startsWith(frame header)){
    //Creo la stringa esadecimale da printare
    sensor string = "La stringa del MD3005 in formato esadecimale e': "
+myString1+" "+myString2+" "+myString3+" "+myString4+" "+myString5+"
"+myString6+" "+myString7+" "+myString8+" "+myString9+" "+myString10+"
"+myString11+" "+myString12+" "+myString13+" "+myString14+"
"+myString15+" "+myString16+" "+myString17+" "+myString18+"
"+myString19+" "+myString20+" "+myString21+" "+myString22+"
"+myString23+" "+myString24+" "+myString25+" "+myString26+"
"+myString27+" "+myString28+" "+myString29+" "+myString30+"
"+myString31+" "+myString32+" "+myString33+" "+myString34+"
"+myString35+" "+myString36+" "+myString37+" "+myString38+"
"+myString39+" "+myString40;
    //Converto i valori delle concentrazioni rilevate da esadecimale a
decimale
   nh3\_byte\_data = 0;
    nh3_byte_data = (byte24<<32) | (byte23<<16) | (byte22<<8) | (byte21);
    nh3_data = nh3_byte_data/1000;
    tvoc byte data = 0;
```

byte29 = (byte) Serial.read();

```
tvoc byte data = (byte29<<32) | (byte28<<16) | (byte27<<8) | (byte26);
  tvoc_data = tvoc_byte_data/1000;
 h2s byte data = 0;
 h2s byte data = (byte33 << 32) | (byte32 << 16) | (byte31 << 8) | (byte30);
 h2s data = h2s byte data/1000;
 //Stampo e carico i risultati ottenuti
 nh3 ppm = "La concentrazione di ammoniaca rilevata dal MD3005 e' pari
a " + String(nh3 data) + " ppm";
 tvoc ppm = "La concentrazione di gas TVOC rilevata dal MD3005 e' pari
a " + String(tvoc data,1) + " ppm";
 h2s ppm = "La concentrazione di acido solfidrico rilevata dal MD3005
e' pari a " + String(h2s data,2) + " ppm";
 }
}
void MQ7() {
  //Imposto il pin analogico dell'ESP8266 come input
  pinMode(A0, INPUT);
  //Leggo il valore analogico dal pin
  analog data = analogRead(A0);
  //Ricavo il voltaggio dal valore analogico
  Sensor_Voltage = analog_data * 3.3 / 4095;
  //Calcolo le PPM del CO rilevato
  co_data = 3.027*pow(2.7182, (1.0698*Sensor_Voltage));
  //Stampo e carico il risultato ottenuto
 co ppm = "La concentrazione di Monossido di Carbonio (CO) rilevata dal
MQ7 e' pari a " + String(co data) + " ppm";
void loop() {
  dht22();
 md3005();
 MQ7();
  server.handleClient(); //Carico i risultati ottenuti dai 3 sensori
 delay(3000);
```

I valori di umidità, temperatura, concentrazione di ammoniaca, gas tvoc, acido solfidrico e monossido di carbonio nell'aria, in condizioni ambientali normali, sono stati



L'umidita' rilevata dal DHT22 e' pari al 40 %

La temperatura rilevata dal DHT22 e' di 25 gradi Celsius

La stringa del MD3005 in formato esadecimale e': aa 0 da 74 0 0 be 6 0 0 26 5 0 0 5 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7a 99 0 0 0 bb bd

La concentrazione di ammoniaca rilevata dal MD3005 e' pari a 0 ppm

La concentrazione di gas TVOC rilevata dal MD3005 e' pari a 0.0 ppm

La concentrazione di acido solfidrico rilevata dal MD3005 e' pari a 0.00 ppm

La concentrazione di Monossido di Carbonio (CO) rilevata dal MQ7 e' pari a 3 ppm

Si noti quindi come il sensore progettato sia effettivamente ricettivo a variazioni della concentrazione di vari gas nell'aria e risponda correttamente a tali situazioni.

#### CONCLUSIONI

Il lavoro di tesi descritto rappresenta dunque un percorso di progettazione, realizzazione e analisi di diversi sensori adibiti al rilevamento nell'aria di concentrazioni di particolari gas, oltre che al monitoraggio di temperatura e umidità.

Il primo step in assoluto è stata la descrizione dei vari moduli impiegati, dal microcontrollore ai diversi sensori scelti per il monitoraggio.

Si è poi quindi proceduto alla realizzazione hardware del sensore completo dei diversi moduli, opportunamente collegati tra loro, nelle varie configurazioni precedentemente descritte.

Nondimeno importante è stata la fase di programmazione, step indispensabile per il settaggio dei vari componenti: gli sketch realizzati sull'IDE di Arduino sono infatti indispensabili non solo per il corretto funzionamento dei singoli moduli e del sensore in configurazione completa, ma anche per raggiungere lo scopo di questa trattazione ossia il rilevamento di vari parametri della qualità dell'aria e la loro disponibilità in rete per consentire all'utente l'accesso da remoto proprio a tali dati.

L'ultima parte del lavoro è rappresentata dall'analisi dei parametri rilevati dai diversi sensori, effettuata dapprima in condizioni ambientali normali e poi in condizioni ambientali che hanno visto la sollecitazione della presenza dei vari gas nell'aria, passaggio fondamentale per vedere il corretto funzionamento dei sensori progettati.

Questa progettazione potrebbe essere anche ampliata: la scelta del microcontrollore con un'unica porta seriale per la comunicazione con i diversi moduli ha infatti rappresentato un limite per la realizzazione di un unico sensore, contenente tutti i diversi moduli, capace quindi di rilevare la concentrazione di diversi gas nell'aria nel medesimo istante. L'utilizzo di un componente come un espansore di porte seriali consentirebbe di realizzare tutto ciò, a patto di collegare, impostare e programmare i vari moduli utilizzati in maniera corretta. In conclusione, questo lavoro di progettazione, realizzazione e analisi di diverse configurazioni di sensori per il monitoraggio della qualità ambientale, non è altro che un perfetto esempio di semplici ma efficaci applicazioni delle tecnologie proprie dell'Internet of Things, qui applicate nel campo del monitoraggio della qualità dell'aria: come già detto nell'introduzione ciò è anche un esempio lampante in cui la tecnologia, la ricerca e l'innovazione vengono messe a servizio dell'uomo per aiutarlo a contrastare e a risolvere fenomeni che minano la salute di tutti e del pianeta intero.