



UNIVERSITÀ DEGLI STUDI DI GENOVA

**DIPARTIMENTO DI INGEGNERIA NAVALE, ELETTRICA, ELETTRONICA E
DELLE TELECOMUNICAZIONI**

**CORSO DI STUDIO IN INGEGNERIA ELETTRONICA E TECNOLOGIE
DELL'INFORMAZIONE**

Tesi di Laurea Triennale

Febbraio 2020

**Progetto e realizzazione di un sistema embedded per il monitoraggio
della qualità dell'aria ambientale**

Candidato: Simone Benzi

Relatore: Prof. Riccardo Berta

Sommario

La presente tesi prevede la progettazione e la realizzazione di un sistema dedicato all'acquisizione e al monitoraggio di dati relativi a gas, utili a valutare la qualità dell'aria in un determinato ambiente. Nel sottosistema relativo all'acquisizione di dati è stata utilizzata una particolare versione di Arduino, il MKR WiFi 1010, oltre ad un sensore dedicato alla rilevazione di differenti gas. I gas presi in esame sono Monossido di Carbonio, Diossido di Azoto e Metano. Le misurazioni effettuate vengono inviate su un cloud attraverso specifiche API rilasciate da un framework detto Atmosphere, il quale funge da interfaccia tra il sottosistema di acquisizione dati e quello relativo al monitoraggio. Quest'ultimo si basa su una applicazione web, composta da più pagine, all'interno della quale è possibile visualizzare tabelle contenenti le rilevazioni effettuate dal sensore, grafici che mostrano l'andamento delle concentrazioni di gas nel tempo ed è possibile modificare l'intervallo di tempo che passa tra una misurazione e quella successiva.

1. Introduzione

Il progetto prevede la realizzazione di un sistema destinato all'acquisizione e al monitoraggio di dati relativi alla concentrazione di gas nocivi all'interno di un ambiente chiuso.

Il sistema complessivo si può suddividere in due diversi sottosistemi comunicanti fra loro grazie alle API fornite da un cloud chiamato "Atmosphere":

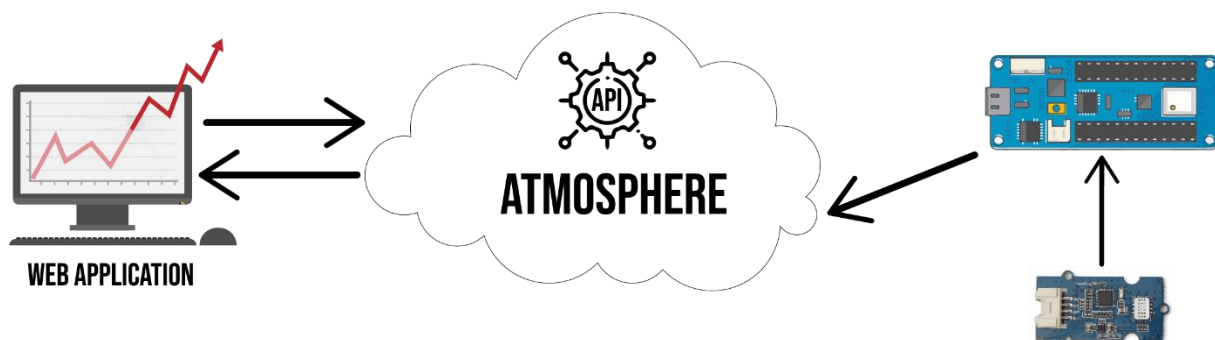
- Un primo sottosistema ha il compito di acquisire i dati attraverso un particolare sensore e di inviare questi ultimi al cloud
- Il secondo prevede, invece, una semplice applicazione web, la quale richiede i valori memorizzati all'interno del server per poterli inserire in tabelle e grafici utili all'interazione fra sensore ed utente umano.

Nel processo di acquisizione dei dati, le concentrazioni vengono rilevate attraverso un sensore multicanale, il quale permette di poter ottenere misurazioni relative a diversi gas contemporaneamente. Nel dettaglio, in questo progetto, le misure utili sono quelle relative a Monossido di Carbonio (CO), Biossido di Azoto (NO₂) e Metano (CH₄).

I valori misurati, utilizzando delle semplici richieste POST HTTP, vengono inviati da Arduino (collegato ad una rete WiFi) al cloud per mezzo di un'API, presentata nei successivi capitoli.

La Web Application, destinata al monitoraggio dei dati misurati nel tempo in un determinato ambiente, è basata su 4 pagine differenti:

- 1) Una homepage nella quale viene brevemente descritto il progetto.
- 2) Una pagina nella quale sono presenti tre differenti tabelle, una per ciascun gas trattato, contenenti le misurazioni inviate da Arduino all'API ed un riferimento temporale. I valori presenti all'interno delle celle risulteranno, ad ogni aggiornamento della pagina, sincronizzati con quelli memorizzati su Atmosphere; questo perché, ogni volta che la pagina viene aggiornata, una richiesta GET HTTP viene creata ed inviata da un semplice script all'API corrispondente.
- 3) Una pagina che presenterà un grafico per ciascun gas trattato, avente sulle ascisse la data e l'orario di misurazione e sulle ordinate il valore in ppb della concentrazione in un certo istante. Esattamente come le tabelle, i grafici risulteranno sincronizzati con il cloud ad ogni aggiornamento della pagina grazie ad uno script simile al precedente.
- 4) Un'ultima pagina, che implementa una funzionalità molto importante per il sistema progettato, è quella relativa alla visualizzazione e modifica dell'intervallo di tempo che deve trascorrere fra una misurazione del sensore e quella immediatamente successiva. Questa impostazione si basa su di una richiesta HTTP di tipo PUT ad un'URL, differente dalla precedente, facente riferimento sempre allo stesso server principale (Atmosphere) ed è necessaria onde evitare un numero eccessivo di letture e quindi di richieste all'API da parte di Arduino.
Dalla stessa pagina sarà possibile visualizzare, prima dell'eventuale modifica, l'ultimo intervallo di misurazione impostato dall'utente.



2. Metodi e strumenti utilizzati

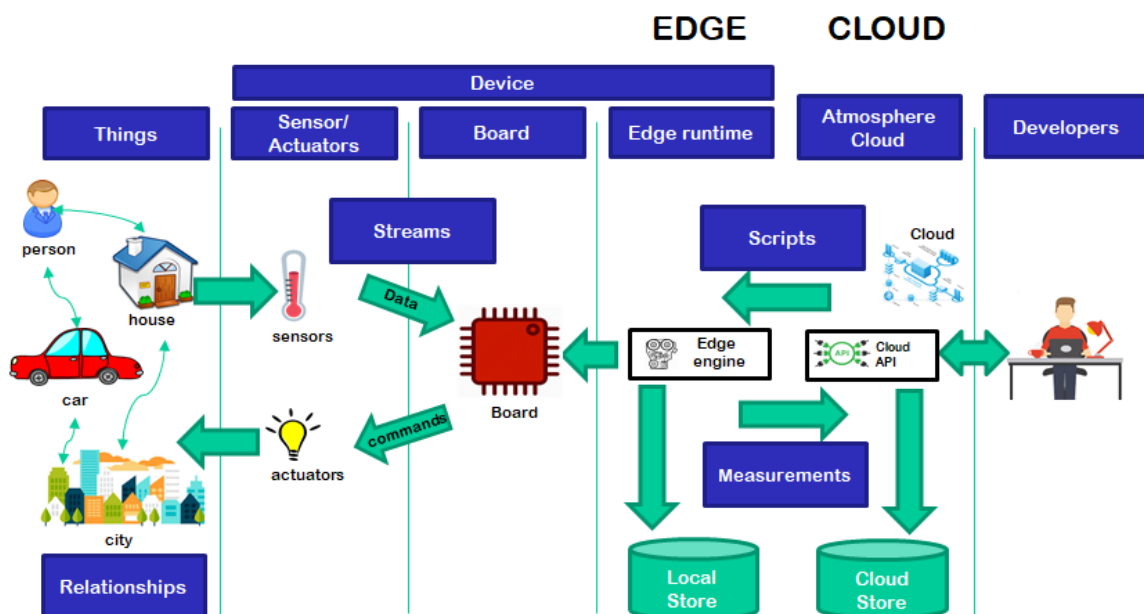
2.1 Atmosphere

Atmosphere è un framework che permette la comunicazione fra un'applicazione o una qualsiasi tipo di board, avente accesso alla rete, ed un cloud, al cui interno sono memorizzati diversi tipi di dato facenti riferimento a diverse "things".

L'architettura di Atmosphere si basa su diversi concetti, qui verranno trattati i soli fondamentali a capirne il suo funzionamento:

- **Developer:** è un'entità finalizzata allo sviluppo di applicazioni o alla programmazione di boards (Arduino, Raspberry Pie...) che sfruttano i servizi forniti dal framework.
- **Application:** è una applicazione software sviluppata da un developer utilizzando i servizi forniti da Atmosphere.
- **Thing:** è un qualsiasi "oggetto" che può essere soggetto di una misurazione (persona, casa...). Nel nostro particolare caso le misurazioni hanno come soggetto la stanza (room) a cui fa riferimento la concentrazione di gas misurata, la quale sarà allora la nostra "thing".
- **Feature:** fa riferimento ad una particolare dimensione fisica (temperatura in un certo ambiente, umidità, rumore...) misurata da un dispositivo opportuno. Nel caso del nostro sistema a questo parametro corrisponde uno dei tre gas presi in considerazione (CO, NO₂, CH₄).
- **Device:** con questa parola si fa riferimento ad un dispositivo generico che può misurare una certa dimensione fisica (Feature), che si riferisce ad una determinata cosa (Thing). Nel nostro caso il device sarà composto dal sensore multicanale, utile alla misurazione delle concentrazioni di diversi gas, collegato ad Arduino. Può essere possibile che piuttosto che ad un sensore, il device faccia riferimento ad un attuatore, il quale agisce su di una cosa modificandone il suo stato (LED, applicazioni domotiche...).
- **Measurement:** si tratta di una misurazione effettuata da un dispositivo (Device) per una determinata cosa (Thing), avente come riferimento una particolare dimensione fisica (Feature). Nel nostro caso indica una concentrazione di gas in ppb, misurata in uno specifico istante.

Quanto esposto fino ad ora può essere schematizzato come nella figura sottostante:



Qualsiasi tipo di accesso al cloud, necessario al device o alla Web Application per inviare o ricevere diversi tipi di dati, può essere effettuato attraverso delle specifiche RESTful API, le quali vengono chiamate da richieste HTTP differenti, a seconda del particolare caso.

Supponendo, ad esempio, di voler inviare una misurazione, effettuata dal nostro sensore, al cloud di Atmosphere, sarà necessario utilizzare la seguente chiamata:

POST http://students.atmosphere.tools/v1/measurements

Accompagnata da un body, codificato in formato JSON, del seguente tipo:

```
[
  {
    "thing": "room",
    "feature": "{{gas}}",
    "device": "air-quality-sensor",
    "samples": [{ "values": [{{num}}] } ]
  },
  {
    "thing": "room",
    "feature": "{{gas}}",
    "device": "air-quality-sensor",
    "samples": [ { "values": [{{num}}] } ]
  }
]
```

Tra gli header della richiesta, ne andrà inserito uno particolare per l'autenticazione del device:

"Authorization": "{{token}}"

Per ottenere il token di autenticazione sarà necessario effettuare un'altra POST al seguente indirizzo:

http://students.atmosphere.tools/v1/login

Specificando un body, che identifica il nostro device, di questo tipo:

```
{
  "username" : "env-sensor-user-username",
  "password" : "env-sensor-user-password"
}
```

In risposta a tale chiamata si ottiene, codificato anch'esso nel formato JSON, il token necessario all'autenticazione del sensore nelle richieste effettuate dal sistema. Questo token resterà valido per un tempo pari a 30 minuti, al termine dei quali dovrà essere inviata un'altra richiesta dello stesso tipo per ottenere una nuova stringa da utilizzare per l'autenticazione.

Questi sono solo due esempi di chiamate che possono essere effettuate al cloud, tuttavia ne esistono di molti altri tipi ed in particolare nel progetto descritto da questa tesi verranno utilizzate delle GET per ottenere, lato applicazione, i valori delle misurazioni oppure, lato device, il codice esposto dallo script che regola l'intervallo di rilevazione delle concentrazioni di gas e, infine, una PUT utile all'utente della Web Application per modificare lo script stesso.

2.2 Arduino MKR WiFi 1010



MKR WiFi 1010 è una particolare versione della celebre piattaforma Arduino. Questa scheda è composta fondamentalmente da tre blocchi principali, i quali verranno solo citati senza essere descritti nel dettaglio:

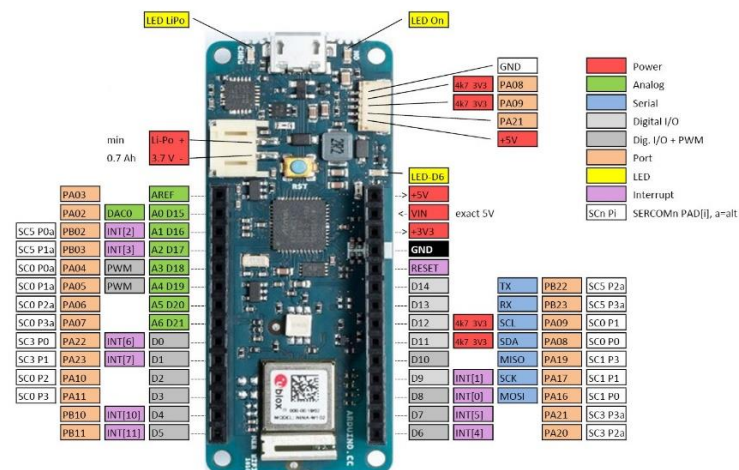
- SAMD21: si tratta di un microcontrollore a bassa potenza avente un'architettura ARM e che lavora a 32 bit.
- U-BLOX NINA-W10 Series: un modulo WiFi anch'esso a bassa potenza, fondamentale nel sistema trattato per consentire l'accesso alla rete alla scheda.
- ECC508 Crypto Authentication: un Cryptochip necessario per rendere sicure le comunicazioni, attraverso crittografia SHA-256.

Esattamente come il più conosciuto Arduino Uno, MKR WiFi 1010 mette a disposizione un vasto set di pin I/O ed inoltre anch'esso può essere programmato utilizzando l'IDE ufficiale di Arduino.

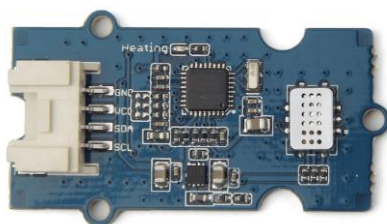
La porta Micro USB presente ha lo scopo principale di connettere la scheda al PC per caricare lo Sketch sviluppato dal progettista o per interagire con l'IDE in fase di debug; inoltre dalla stessa porta è possibile alimentare la scheda a 5V. Questa non è l'unica soluzione per alimentare MKR poiché quest'ultimo può essere alimentato anche con un'alimentazione esterna sempre a 5V oppure attraverso delle batterie Li-Po da 3.7V; le batterie inoltre possono essere ricaricate mentre la scheda è in funzione mediante un'alimentazione esterna (il passaggio da un'alimentazione all'altra viene effettuato automaticamente).

Risulta importante effettuare un'osservazione su questa particolare board: la massima tensione tollerabile dai pin I/O è pari a 3.3V, al contrario della maggior parte delle schede Arduino (compreso Arduino Uno) per cui si hanno 5V. Applicare una tensione superiore a quella massima potrebbe danneggiare la scheda.

Viene riportata un'illustrazione che specifica la posizione sulla scheda di tutti i pin I/O:



2.3 Grove Multichannel Gas Sensor (MiCS-6814)



MiCS-6814 è un sensore MOS di piccole dimensioni utile alla rilevazione di concentrazioni di gas differenti all'interno di un ambiente chiuso. Al suo interno si trovano tre circuiti, completamente indipendenti, sensibili alla variazione di diversi gas; purtroppo, come evidenziato di seguito, per alcuni si avrà una maggior precisione rispetto ad altri.

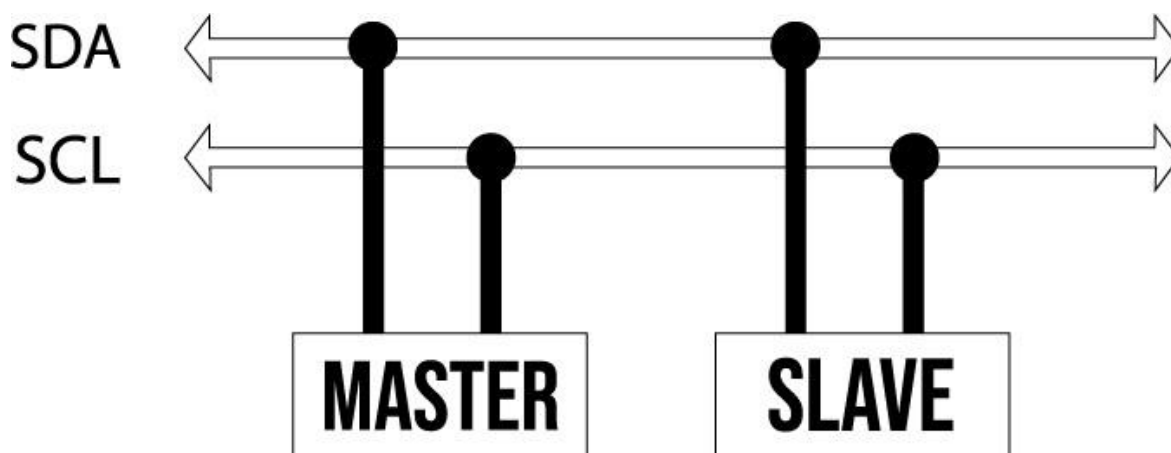
In tabella si trovano i gas che potranno essere rilevati ed il corrispondente intervallo di sensibilità:

GAS	INTERVALLO SENSIBILITÀ
Monossido di Carbonio (CO)	1 ÷ 1000 ppm
Diossido di Azoto (NO ₂)	0.05 ÷ 10 ppm
Etanolo (C ₂ H ₅ OH)	1 ÷ 1000 ppm
Idrogeno (H ₂)	10 ÷ 500 ppm
Ammoniaca (NH ₃)	1 ÷ 500 ppm
Metano (CH ₄)	> 1000 ppm
Propano (C ₃ H ₈)	> 1000 ppm
Isobutano (C ₄ H ₁₀)	> 1000 ppm

2.3.1 Protocollo I²C

La comunicazione fisica tra MiCS-6814 e Arduino è resa possibile dall'implementazione del protocollo I²C (Inter Integrated Circuit).

Questo protocollo prevede l'utilizzo di un bus a due fili, più uno per il riferimento comune di tensione. Le informazioni, secondo tale protocollo, devono essere inviate serialmente su di una linea utilizzata esclusivamente per i dati, detta SDA (Serial Data Line), mentre una seconda linea, detta SCL (Serial Clock Line), deve essere utilizzata per condividere il solo segnale di clock. In figura si può trovare una semplice schematizzazione di una comunicazione I²C.



Il dispositivo master è quello che ha il controllo del bus, condividendo il segnale di clock e generando i segnali di START e STOP, i quali, rispettivamente, aprono e chiudono la comunicazione con lo slave. Il dispositivo (possono essere più di uno) slave è quello che, semplicemente, attende l'arrivo di un segnale il quale indica allo slave stesso la presenza di un dato in lettura, inviato dal master, oppure la richiesta da parte di quest'ultimo di un particolare dato, il quale sarà reso disponibile dallo slave sulla linea SDA dedicata ai dati.

In base a quanto descritto sopra, è possibile prevedere 4 differenti possibilità di utilizzo del canale:

- Master in trasmissione: controlla il segnale di clock e invia i dati allo slave.
- Master in ricezione: controlla il segnale di clock e riceve dati dallo slave.
- Slave in trasmissione: invia i dati al master una volta ricevuta la richiesta da quest'ultimo.
- Slave in ricezione: legge i dati resi disponibili dal master sul canale.

Nel particolare caso del sistema descritto in questa tesi, il master corrisponderà alla scheda Arduino mentre lo slave troverà riscontro nel sensore multicanale: Arduino richiede i valori misurati dal sensore dopo un certo intervallo e quest'ultimo li renderà disponibili alla lettura condividendoli sul canale SDA.

2.3.2 Libreria MiCS6814-I2C.h

Al fine di rendere più semplice la scrittura del codice necessario alla comunicazione fra la scheda ed il sensore è stato utile ricercare una libreria compatibile con il Grove Multichannel Gas Sensor.

La possibilità, solo in apparenza, più semplice sarebbe stata quella di utilizzare la libreria, fornita sul sito ufficiale del dispositivo, chiamata "MutichannelGasSensor.h". Questa in fase di test, durante la quale è stato utilizzato un classico Arduino UNO, ha funzionato perfettamente restituendo le misurazioni corrette effettuate dal sensore. Purtroppo, andando a sostituire l'Arduino UNO con il MKR WiFi 1010 (il quale ci garantisce connessione alla rete), si sono riscontrati dei problemi di compatibilità con la nuova scheda utilizzata.

Al fine di risolvere questa problematica si è optato per l'utilizzo di una seconda libreria, questa volta non ufficiale, chiamata MiCS6814-I2C.h. Questa presenta caratteristiche del tutto simili alla prima, sia per quanto riguarda le classi definite al suo interno sia per quanto riguarda le funzionalità dei metodi implementati all'interno di queste ultime.

Di seguito vengono elencati e definiti gli elementi fondamentali della libreria, utilizzati nel progetto:

- **MiCS6814**: la dichiarazione di un oggetto appartenente a questa classe corrisponde alla dichiarazione di un sensore all'interno dello sketch ed ovviamente permette di avere accesso a tutti i metodi implementati all'interno della classe stessa.
- **begin()**: questo metodo è necessario per la connessione del sensore all'Arduino utilizzando l'indirizzo I²C di default, ovvero 0x04. In alternativa, è possibile passare come argomento a begin() l'indirizzo da utilizzare nel seguente modo: begin(address).
- **powerOn()**: questo ha il semplice scopo di attivare il circuito di riscaldamento, fondamentale per la corretta rilevazione dei gas.
- **measureCO()**, **measureNO2()**, **measureCH4()**: questi metodi hanno tutti il medesimo scopo, ovvero quello di restituire la misura in ppm della concentrazione rilevata, ma si riferiscono a gas differenti.

2.3.3 Preheat Loop

Un'ulteriore considerazione va effettuata sul seguente loop, il quale va inserito nel setup dello sketch Arduino che controlla l'acquisizione dei dati dal sensore:

```
while (preheatTime > 0) {  
    Serial.print(preheatTime);  
    Serial.print(" ... ");  
  
    // Wait a minute  
    delay (60000);  
  
    preheatTime = preheatTime - 1;  
}
```

Questo ha lo scopo di attendere un certo intervallo temporale, memorizzato, in minuti, all'interno di una variabile chiamata *preheatTime*. Questo intervallo, che si aggira fra i 15 e i 30 minuti ha lo scopo di lasciare al sensore il tempo necessario per scaldarsi e, di conseguenza, settarsi su misurazioni coerenti con i valori reali presenti in un certo ambiente.

Per fare questo, come si può intuire dal codice, è sufficiente inserire all'interno del ciclo while un ritardo (delay) pari a 60000 millisecondi, corrispondenti ad un minuto. Dopo aver atteso il tempo specificato dal comando delay, la variabile *preheatTime* viene decrementata di un'unità, essendo trascorso un minuto.

Queste istruzioni vengono iterate fino a che la variabile non raggiunge un valore pari a 0; a questo punto lo sketch può continuare la sua esecuzione ed i valori letti dal sensore saranno corretti.

2.4 HTML & JavaScript

Per la progettazione della web application si è sfruttata una tecnica basata sullo sviluppo di diversi file in formato HTML, JavaScript e CSS, interconnessi fra loro.

HTML è l'acronimo di HyperText Markup Language ed è, come esplicitato dal nome, un linguaggio di markup ("contrassegno") ovvero utile alla disposizione di elementi di diverso tipo all'interno di una pagina web. È fondamentale non confondere questo linguaggio con un linguaggio di programmazione che, per definizione, presenta dei costrutti caratteristici tra cui i loop e le istruzioni condizionate (eseguite al verificarsi di una condizione).

Una pagina HTML fa sempre riferimento ad un semplice file di testo avente estensione ".html", scritto e modificato attraverso un semplice editor testuale come il classico notepad.

Al fine di disporre gli elementi con un ordine ed una logica ben precisa vengono sfruttati i TAG, che si trovano all'interno di parentesi angolari, con alcuni attributi associati ad essi che descrivono in maniera più specifica la funzione o il tipo di elemento.

I TAG fondamentali all'interno di un documento HTML sono:

- `<!DOCTYPE html>`: non ha una particolare funzione nella struttura della pagina ma ha il solo scopo di specificare che il file si riferisce ad una pagina HTML.
- `<html>`: racchiude l'intera pagina e permette di specificare, attraverso qualche attributo, alcune caratteristiche della pagina stessa tra cui, ad esempio, la lingua del documento (lang = "it");
- `<head>`: fa riferimento all'header della pagina e racchiude al suo interno un certo numero di altri TAG utili alla gestione della stessa. Ad esempio, con il TAG `<title>` sarà possibile definire ciò che apparirà nel browser come titolo della scheda all'interno della quale è aperta la nostra pagina. Come descritto più avanti, qui sarà possibile associare la pagina al file, scritto in CSS, che ne definisce lo stile.
- `<body>`: si riferisce al corpo della pagina web, qui, attraverso altri TAG, vengono definiti tutti gli elementi che verranno visualizzati dall'utente attraverso l'utilizzo di un browser.

Nel caso di questa tesi il solo linguaggio HTML non è sufficiente allo sviluppo dell'intera applicazione, la quale necessita di interazione con l'utente e di caricare dinamicamente alcuni dati da visualizzare a schermo in modo tale che siano sempre aggiornati ogniqualvolta l'utente decida di ricaricare la pagina. Per questo motivo i file HTML del sito in questione contengono semplicemente un'ossatura delle diverse pagine, mentre il caricamento delle tabelle, delle caselle di testo e di tutto ciò che necessita di dinamicità o interazione con l'utente viene delegato a dei file JavaScript, associati alle varie pagine, attraverso il seguente TAG, contenuto all'interno del body:

```
<script src="./file.js"></script>
```

L'attributo *src* è di fondamentale importanza poiché è quello che specifica il percorso del file contenente lo script da eseguire.

JavaScript, che non ha nulla a che vedere con Java, è un linguaggio di programmazione orientato agli oggetti, adottato da tutti i principali browser per l'esecuzione di script nelle pagine web. Questa non è comunque la sua unica applicazione poiché viene sfruttato ampiamente anche al di fuori dei browser, in ambienti differenti come ad esempio node.js.

JavaScript si basa sugli stessi costrutti degli altri linguaggi di programmazione tra cui istruzioni condizionate e loop.

Come accennato in precedenza, questo linguaggio offre allo sviluppatore la possibilità di implementare degli script utili alla costruzione dinamica della pagina web. Per fare ciò, viene sfruttata la sua capacità di accedere, in lettura e modifica, al DOM tramite alcuni metodi dedicati.

Il DOM (Document Object Model) è un modello, indipendente dal tipo di browser utilizzato, dalla versione e dal sistema operativo, che descrive come i diversi elementi della pagina sono fra loro collegati.

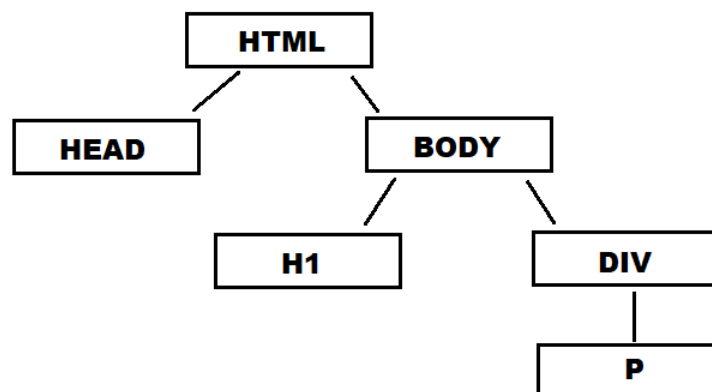
Importante è capire che il DOM non dipende in modo diretto da JavaScript ma quest'ultimo è semplicemente uno dei possibili mezzi per accedervi.

Per comprenderlo al meglio, lo si può immaginare come una descrizione gerarchica della pagina riconducibile ad una struttura ad albero. Ogni elemento di questa struttura viene detto "nodo" e rappresenta un semplice elemento della pagina. Ogni nodo, a sua volta, tolto il caso della radice, è associato ad un altro nodo detto "genitore" ed inoltre da ciascuno possono partire dei collegamenti ai cosiddetti nodi "figli".

Supponendo una semplice pagina HTML così strutturata:

```
<html>
  <head></head>
  <body>
    <h1>This is a title</h1>
    <div>
      <p>This text is inside the div</p>
    </div>
  </body>
</html>
```

Il DOM corrispondente sarà riconducibile ad una struttura di questo tipo:



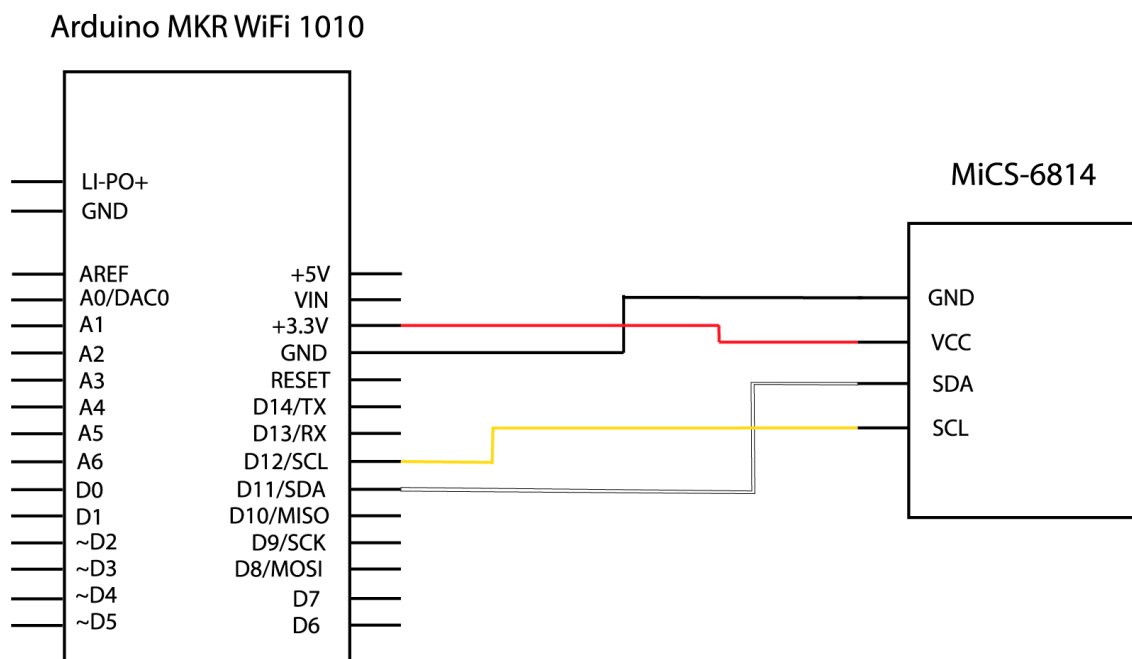
3. Sperimentazione e risultati

3.1 Acquisizione dati

Il sottosistema necessario all'acquisizione dei dati ed al loro invio su Atmosphere è stato realizzato sfruttando un Arduino MKR WiFi 1010, al quale è stato collegato un semplice sensore, descritto nel capitolo precedente, utile alla rilevazione di diversi gas.

Come evidenziato in precedenza, la trasmissione di dati tra questi due dispositivi è resa possibile dall'implementazione del protocollo I²C, fisicamente realizzato sfruttando i pin di Arduino dedicati a questo tipo di comunicazione (11 per SDA e 12 per SCL) collegati con i reciproci pin presenti sul sensore.

Inoltre, al fine di assicurare un corretto funzionamento, è stato necessario alimentare il sensore; a tale scopo è stato sufficiente sfruttare le uscite VCC e GND di Arduino, le quali si riferiscono rispettivamente a 3.3 V e a massa.



Il comportamento dell'intero sistema di acquisizione dati è regolato da uno sketch, caricato su Arduino, che implementa due diverse fasi di funzionamento: la fase di setup e quella del loop (dove si fa realmente riferimento alla rilevazione dei dati).

Nella fase di setup vengono svolte due importanti operazioni:

- Viene aperta una connessione con la rete Internet controllando il modulo WiFi integrato su MKR; a tale scopo sono state utilizzate alcune funzioni specifiche contenute nella libreria "WiFinINA.h".
- Si verifica che il sensore sia effettivamente collegato; avuto un riscontro positivo si procede con il "Preheat Loop", già trattato nel precedente capitolo, necessario al riscaldamento del dispositivo.

All'interno del loop viene implementato l'algoritmo raffigurato a pagina 13, nel quale si possono distinguere i seguenti step:

- Viene calcolato, come primo passo, il tempo trascorso dall'ultima misurazione effettuata. A tal fine viene sfruttata la funzione *millis()*; questa restituisce, in millisecondi, il tempo che è passato da quando Arduino ha iniziato ad eseguire lo sketch.

```
unsigned long currentMillis = millis();  
unsigned long passedTime = currentMillis - previousMillis;
```

In *previousMillis* è memorizzato l'istante in cui sono stati rilevati gli ultimi valori di concentrazione dei gas.

- Se il tempo trascorso dall'ultima misurazione è maggiore di un certo intervallo fissato, si procede con una nuova rilevazione di dati e con l'invio di quest'ultimi ad Atmosphere tramite una richiesta POST avente un body del tutto simile a quello illustrato nel capitolo precedente. La costruzione della richiesta è semplificata da alcune funzioni contenute nella libreria "ArduinoHttpClient.h":

- *beginRequest()*: inizializza la richiesta.
- *post(String path)*: specifica che si tratti di una POST e riceve come argomento il percorso dell'API a cui verrà inviata la richiesta.
- *sendHeader(String name, String value)*: aggiunge un header, avente nome e valore coincidenti a quelli ricevuti come argomento, alla richiesta HTTP.
- *endRequest()*: termina la richiesta e la invia al percorso specificato in *post(String path)*.
- *HttpClient.print(String postBody)*: associa alla richiesta una stringa contenente il body desiderato.

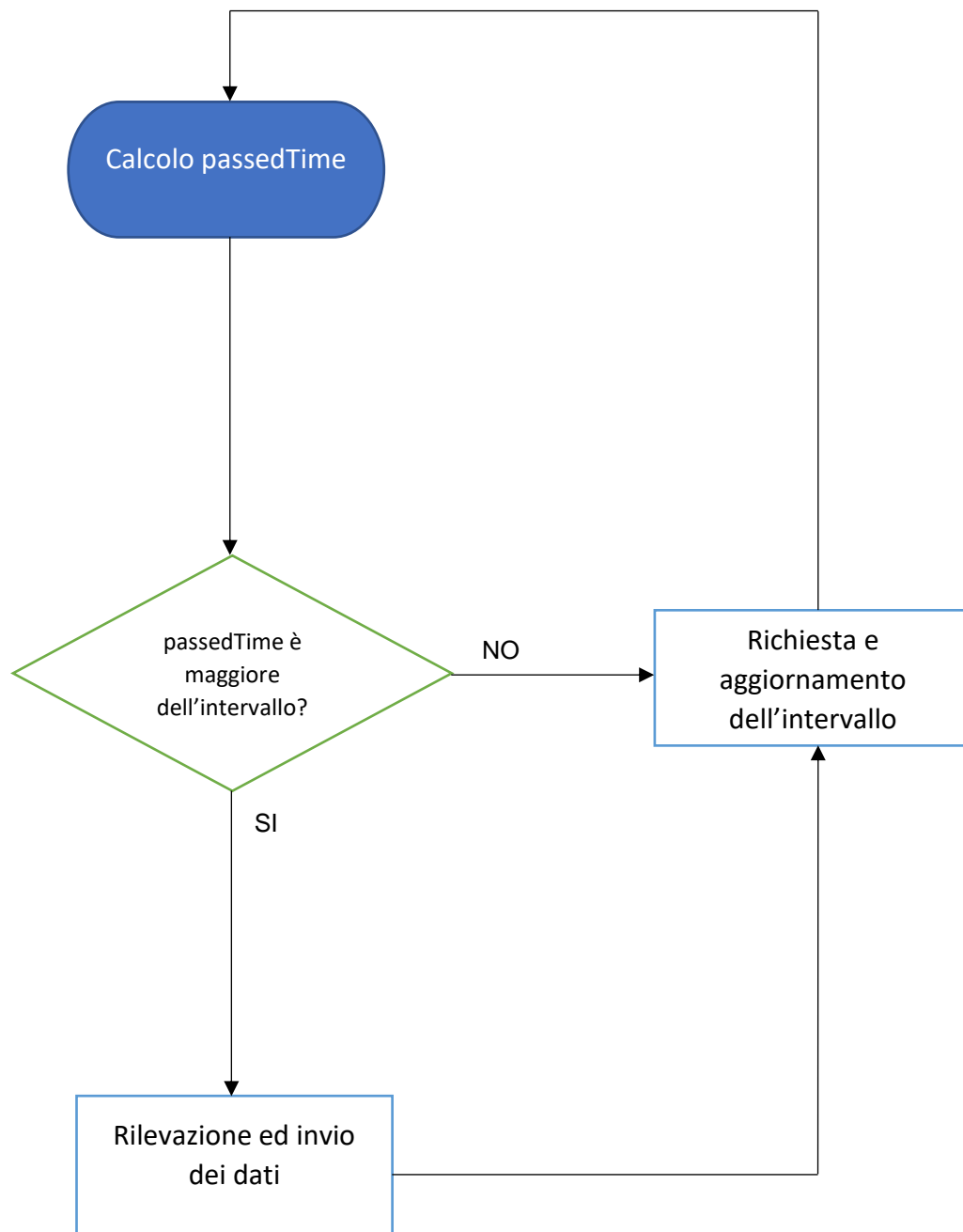
- Al termine del passaggio precedente oppure nel caso in cui, terminato il primo step, il tempo trascorso sia inferiore all'intervallo memorizzato, si procede con una richiesta GET alla seguente API: *http://students.atmosphere.tools/v1/scripts/air-quality-script*. Questa viene costruita in un modo del tutto analogo alla POST descritta in precedenza ma anziché utilizzare la funzione *post(String path)*, si utilizza *get(String path)*. Inoltre, una richiesta di tipo GET non prevede alcun body ad essa associata; per questo motivo non si ha *HttpClient.print(String postBody)*.

La risposta a questa GET sarà codificata in JSON e conterrà un campo "code", anch'esso codificato in JSON, nel quale sarà presente il valore, in millisecondi, dell'intervallo di tempo che deve passare tra una rilevazione e quella successiva:

```
{  
  "visibility": "private",  
  "tags": [],  
  "_id": "air-quality-script",  
  "code": "{\"interval\":1800000}",  
  "owner": {  
    "_id": "5dcec779c67ed54963bc865e",  
    "username": "env-sensor-user-username",  
    "type": "provider"  
  }  
}
```

Tale valore verrà estrapolato dalla risposta attraverso alcune funzioni contenute nella libreria "ArduinoJson.h" e verrà memorizzato in memoria da Arduino.

Al termine di questo step, il loop ripartirà dal primo passaggio.




3.2 Applicazione Web

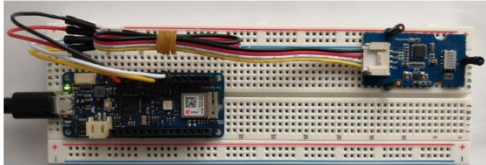
Il monitoraggio dei dati acquisiti avviene attraverso una semplice applicazione web. Questa, sviluppata sfruttando dei semplici codici HTML, JavaScript e CSS, si suddivide in 4 pagine differenti.

Homepage: ha lo scopo di presentare, molto brevemente, la web application stessa:

Home **Tabelle** Grafici Intervallo

Air Quality Monitor





Attraverso questa applicazione web è possibile monitorare le quantità di Metano, Diossido di Azoto e Monossido di Carbonio, riferite ad un ambiente nel quale è stato installato il sistema di acquisizione dati (Arduino + Sensore Multicanale).
Oltre alla homepage, sono consultabili altre 3 pagine: Tabelle, Grafici ed Intervallo Rilevamenti.

Tabelle: vengono visualizzate 3 tabelle distinte, una per ciascun gas, all'interno delle quali vengono raggruppate le corrispondenti misurazioni, ciascuna accompagnata dalla data e dall'orario in cui è stata effettuata.

Grafici: vengono visualizzati 3 grafici distinti, uno per ciascun gas, al cui interno è tracciato l'andamento della concentrazione, in ppb, nel tempo. Sull'asse delle ascisse troviamo il riferimento temporale (data e ora di misurazione) mentre sulle ordinate la concentrazione misurata dal sensore.

Intervallo: è possibile visualizzare l'intervallo temporale, settato in precedenza dall'utente, tra due misurazioni consecutive. Inoltre da qui è possibile modificare tale intervallo attraverso gli specifici bottoni di selezione.

Tabelle: vengono visualizzate 3 tabelle, una per ciascun gas trattato, al cui interno sono riportate le diverse misurazioni accompagnate dalla data e dall'orario in cui sono state effettuate.

Home **Tabelle** Grafici Intervallo

Air Quality Monitor




Tabella Misurazioni CO

Data Misurazione	Orario Misurazione	Concentrazione misurata [ppb]
21-1-2020	19:47	6044.16
21-1-2020	19:18	6044.16
21-1-2020	18:48	6193.68
21-1-2020	18:18	6472.43
21-1-2020	17:48	7155.01
21-1-2020	17:18	8858.53

Per la costruzione di questa pagina, in fase di sviluppo, è stata fondamentale l'implementazione di uno script che consentisse l'invio di alcune richieste GET dall'applicazione ad Atmosphere, il quale risponderà con una lista delle misurazioni memorizzate al suo interno, consentendo il riempimento delle tabelle.

Le richieste, costruite sfruttando un oggetto XMLHttpRequest e relativi metodi associati, verranno inviate ad un'URL di questo tipo:

<http://test.atmosphere.tools/v1/measurements?filter=%7B%22feature%22:%22CO%22%7D&limit=6&page=1>

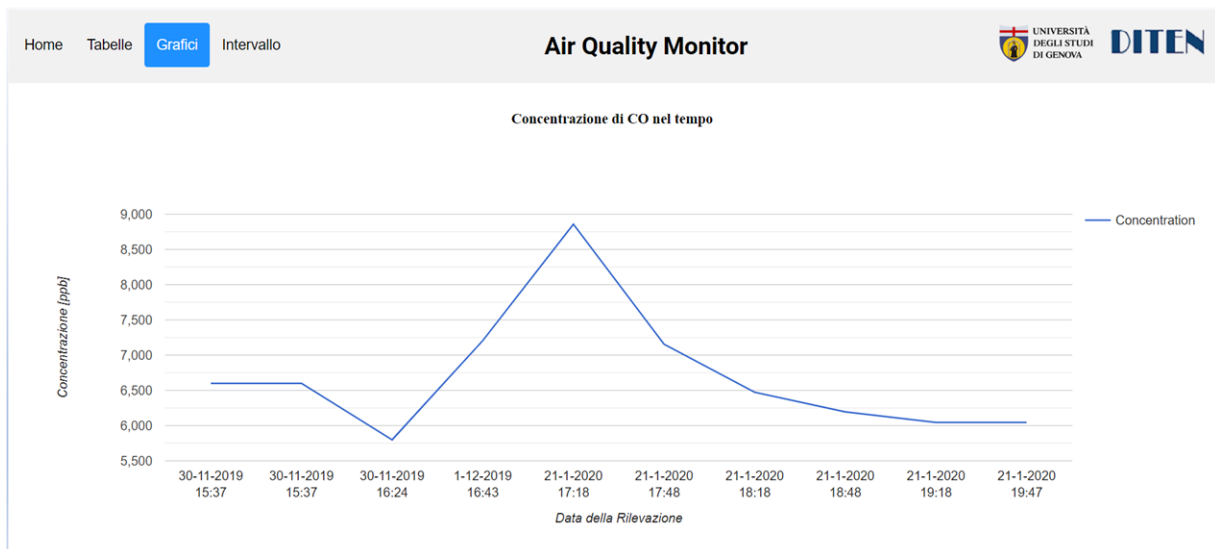
Come si può notare, al classico percorso necessario per raggiungere l'API sono state aggiunte alcune query (evidenziate in giallo):

- *feature:CO* : fa sì che le misurazioni ricevute come risposta da Atmosphere facciano riferimento al solo Monossido di Carbonio. Per le tabelle relative agli altri due gas trattati, ovviamente, CO verrà sostituito con NO₂ e CH₄.
- *limit=6* : fissa il numero di misurazioni inviate dall'API, nella risposta, a 6, essendo ciascuna tabella presente nella pagina composta da 6 righe. Questo porterà Atmosphere a suddividere i dati in pagine differenti, tutte contenenti 6 rilevazioni.
- *page=1* : determina quale pagina si vuole ricevere in risposta e quindi quale pagina visualizzare in tabella. Ovviamente, questo parametro potrà essere settato dall'utente, al quale è permesso passare da una pagina all'altra cliccando sui tasti "next" e "prev".

I metodi, legati all'oggetto XMLHttpRequest, utilizzati in fase di costruzione della richiesta sono i seguenti:

- *onload()*: gestisce la risposta dell'API, non appena questa viene resa disponibile. Per quanto riguarda questa pagina, viene verificato, al suo interno, lo status code. Nel caso in cui sia 200 (OK) allora la risposta viene trasformata in un oggetto JSON e passata come argomento ad una funzione di callback che si occuperà di andare a riempire le celle della tabella con i dati ottenuti. Quando invece lo status code è pari a 401 (Unauthorized), viene inviata una richiesta POST necessaria ad ottenere ed aggiornare il token di autorizzazione per poi, subito dopo, costruire ed inviare nuovamente la richiesta GET descritta precedentemente.
- *open(method: string, url: string)* : apre una richiesta HTTP e riceve come argomento il metodo desiderato (in questo caso GET) e l'URL a cui la richiesta deve essere inviata.
- *setRequestHeader(name: string, value: string)*: aggiunge l'header desiderato alla richiesta, utilizzando nome e valore specificati negli argomenti.
- *send()*: invia la richiesta HTTP definita dalle funzioni precedenti.

Grafici: vengono visualizzati 3 differenti grafici che mostrano l'andamento delle concentrazioni dei gas nel tempo.



Anche per questa pagina si è costruito uno script funzionale all'invio di una richiesta GET ad un'API associata ad una risposta contenente la lista delle misurazioni ricevute da Arduino. In questo caso, tuttavia, l'URL inserito è leggermente differente poiché l'unica query utilizzata è quella necessaria a filtrare le sole misurazioni relative al gas trattato nel grafico. Ad esempio, per la costruzione del grafico relativo alla CO, si ha:

<http://test.atmosphere.tools/v1/measurements?filter=%7B%22feature%22%3A%22CO%22%7D>

La richiesta è stata costruita in modo simile alla precedente, l'unica differenza risiede nel metodo `onload()`, il quale ha sempre un comportamento dipendente dallo script che si vuole implementare. Nel caso di questa pagina, dopo aver controllato che lo status code sia pari a 200, la risposta verrà trasformata in un oggetto JSON e per ciascuna misurazione al suo interno verranno memorizzati valore in ppb e data. Questi saranno inseriti in un array contenente le coppie `[data, valore]`, corrispondenti alle coordinate dei punti necessari a tracciare il grafico.

Per tracciare i grafici è stata utilizzata una semplice API di Google chiamata "Google Charts". Al fine di includere quest'ultima nello script sono stati aggiunti il TAG, nell'header del file HTML associato, e l'istruzione, nello script stesso, qui sotto riportati:

```
<script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
```

```
google.charts.load('current', { packages: ['corechart'] });
```

Dopo aver incluso l'API ed aver gestito il riempimento degli array con tutte le coppie necessarie al tracciamento dei grafici, sono state definite tre funzioni di callback (una per ciascun grafico), da lanciare una volta caricata l'API stessa, nel seguente modo:

```
google.charts.setOnLoadCallback(COdrawChart); // riportiamo il solo caso della CO
```

Ogni funzione di callback ha lo scopo di disegnare il grafico sulla pagina e definire le varie opzioni di visualizzazione. Al loro interno, è stato dichiarato un oggetto "chart", relativo al grafico, in questo modo:

```
let chart = new google.visualization.LineChart(COchartDiv); // verrà disegnato nel div COchartDiv
```

Una volta istanziato l'oggetto, è stata inserita l'istruzione `chart.draw(data, options)`, utile al vero e proprio tracciamento del grafico, dove l'argomento `data` fa riferimento ad un particolare oggetto contenente tutte le informazioni relative ai dati (array dei valori, tipo di dato sulle ascisse, tipo di dato sulle ordinate) mentre all'interno di `options` sono presenti tutte le opzioni di visualizzazione del grafico.

Intervallo: vengono visualizzati 3 bottoni. I primi due, se cliccati, permettono di scegliere le ore e i minuti che intercorrono tra una rilevazione e quella successiva. Ad ogni aggiornamento della pagina le ore e i minuti impostati di default saranno quelli memorizzati per ultimi su Atmosphere.

Il bottone "invia", invece, memorizza sull'API l'intervallo selezionato e lo renderà, quindi, disponibile alla consultazione da parte del sottosistema che si occupa dell'acquisizione dei dati dall'ambiente.

Per lo sviluppo di questa pagina è stato implementato uno script al cui interno sono gestite sia richieste GET che richieste PUT, entrambe facenti riferimento al medesimo URL:

<http://students.atmosphere.tools/v1/scripts/air-quality-script>

La richiesta GET, necessaria a selezionare dei valori di default adeguati per i bottoni di ore e minuti, avrà una struttura identica alle precedenti e, come sempre, a variare saranno la URL e il metodo *onload()*. Quest'ultimo, dopo aver verificato che lo status code sia 200, trasformerà la risposta in un oggetto JSON e andrà a memorizzare all'interno di una variabile "*millis*" il valore dell'intervallo, fornito in millisecondi. A partire da quest'ultimo valore si potrà ottenere il numero di ore, memorizzate in una variabile "*hours*", e di minuti, memorizzati in "*minutes*", ad esso corrispondente, attraverso queste tre operazioni:

```
let hours = Math.floor(millis / 3600000);  
millis -= hours * 3600000;  
let minutes = Math.floor(millis / 60000);
```

Il metodo *Math.floor(number)* restituisce l'intero, arrotondato per difetto, del numero che viene passato come parametro.

Il contenuto di *hours* e *minutes* viene confrontato con tutti i possibili valori selezionabili dai bottoni corrispondenti e al verificarsi di un match, il valore di default viene impostato.

La richiesta PUT, necessaria all'aggiornamento dell'intervallo, avrà, invece una implementazione di poco differente da quelle precedenti.

In questo caso, una volta cliccato il bottone "invia", vengono eseguiti i calcoli inversi a quelli visti sopra, ovvero, a partire da *hours* e *minutes*, selezionati dall'utente, si passa a "*millis*":

```
let millis = (hours * 3600000) + (minutes * 60000);
```

Una volta calcolato il valore da inviare all'API, si può procedere alla costruzione della richiesta.

A differenza delle precedenti, ovviamente, il metodo HTTP specificato in *open* sarà PUT e, inoltre, andrà passato, come argomento del metodo *send*, il body della richiesta, il quale avrà la seguente struttura:

```
{"code": {"interval": 300000}}
```

4. Contributo personale e considerazioni conclusive

L'obiettivo di questa tesi era realizzare un sistema embedded per la rilevazione di dati utili al monitoraggio della qualità dell'aria ambientale e sviluppare una web application dove questi potessero essere consultati.

I dati acquisiti dal sistema dovevano essere inviati, tramite RESTful API, ad un cloud chiamato Atmosphere dove potevano essere, sempre attraverso API, richiesti ed utilizzati dalla web application.

Per semplicità si è utilizzato un sensore multicanale utile alla rilevazione di più gas, garantendo così la possibilità di avere un quadro più ampio circa le condizioni di qualità dell'aria, ma al tempo stesso non è possibile garantire un'elevata precisione nelle misurazioni. Questo progetto, infatti, deve essere visto come il prototipo di un sistema più accurato, dove per ogni gas utile alla valutazione della qualità dell'aria ambientale è utilizzato uno specifico sensore ad alta precisione.

Detto ciò, i dati raccolti risultano affidabili nel caso di Monossido di Carbonio (CO) e Diossido di Azoto (NO₂) mentre nel caso del Metano (CH₄) l'affidabilità si abbassa notevolmente, essendo il sensore sensibile a concentrazioni di Metano superiori ai 1000 ppm.

Per sviluppare lo sketch caricato su Arduino si è utilizzato l'IDE ufficiale di quest'ultimo ed inoltre è stato indispensabile l'utilizzo di alcune librerie utili ad esempio al controllo del sensore, a settare la connessione ad Internet tramite WiFi e alla costruzione di richieste HTTP.

Per costruire l'applicazione web è stato utilizzato l'editor Visual Studio Code, dedicato alla programmazione e allo sviluppo web. Con l'ausilio di quest'ultimo si sono creati due file per ogni pagina: uno in formato HTML, dove era specificata la struttura di base della pagina, ed uno scritto in Javascript, collegato al file precedente tramite specifico TAG nell'header, dove veniva implementato uno script utile alla completa costruzione dinamica della pagina stessa. Un ultimo file, scritto sempre con Visual Studio Code, è quello in formato CSS dove si sono definite tutte le caratteristiche relative allo stile e alla posizione degli elementi nella pagina.

Potrebbero essere apportate, in futuro, diverse migliorie, oltre a quelle già citate precedentemente relative all'utilizzo di sensori ad alta precisione. Si può valutare, ad esempio, la possibilità di ampliare la mole di informazioni raccolte rilevando un numero superiore di gas oppure utilizzando altre tipologie di sensori tra cui quello di umidità (anch'essa parametro importante in quest'ambito).

Il sistema realizzato, infine, può risultare molto utile e trovare applicazione in determinati ambienti di lavoro, nei quali la concentrazione di specifici gas deve rimanere al di sotto di una certa soglia onde evitare possibili danni alla salute dei lavoratori.

5. Riferimenti bibliografici

- [1] Marijn Haverbeke, Eloquent JavaScript: A Modern Introduction to Programming, No Starch Press, 2018, 3 edizione.
- [2] Leonard Richardson, RESTful Web APIs, O'Reilly, 2013, 1 edizione.
- [3] Dominique Guinard, Building the Web of Things, Manning, 2016.
- [4] Michael Margolis, Arduino Cookbook, O'Reilly, 2011.
- [5] HTML/JavaScript Tutorials. URL: <https://www.html.it>.
- [6] Grove - Multichannel Gas Sensor Datasheet.
URL: http://wiki.seeedstudio.com/Grove-Multichannel_Gas_Sensor.
- [7] Arduino MKR WiFi 1010 Documentation. URL: <https://store.arduino.cc/arduino-mkr-wifi-1010>.
- [8] CSS toolkit for building web designs. URL: <https://docs.siimple.xyz>.