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

Settembre 2022

Progetto e implementazione di un sistema per il monitoraggio dello stato di salute di una pianta

Candidato: Hanna De Maria

Relatore: Prof. Riccardo Berta

Correlatore: Dott. Luca Lazzaroni

Sommario

La presente tesi ha come scopo la progettazione e la realizzazione di un sistema embedded per il monitoraggio remoto delle condizioni di salute e ambientali di una pianta. Tale sistema permette inoltre il mantenimento del benessere della pianta. La progettazione è basata sulla scheda Arduino MKR WiFi 1010 che gestisce un sensore di luminosità TSL2561, un sensore capacitivo di umidità del terreno e una pompa sommergibile che permette l'irrigazione della pianta. La comunicazione è basata sulla connessione Wi-Fi, la scheda si occupa dell'acquisizione dei dati e dell'invio di questi al cloud, tramite le API rilasciate dal framework Measurify, dove sono controllati e memorizzati, ma anche dell'irrigazione della pianta se necessario. Measurify svolge il ruolo di interfaccia tra il sistema di misurazione di Arduino e l'applicazione per smartphone. Quest'ultima permette all'utente di accedere ai dati memorizzati e visualizzarli per giorno, settimana o mese.

INDICE

1. Introduzione

1.1. Internet of Things

Con **Internet of Things (IoT)** ci si riferisce a un ecosistema di oggetti fisici in comunicazione tra loro, tramite connessione internet, allo scopo di raccogliere ed elaborare informazioni.
L'Internet of Things o Internet delle cose è il risultato dell'evoluzione di internet avvenuto negli ultimi anni ed è diventato parte integrante della società moderna. Il termine IoT è stato introdotto alla fine degli anni 90, tuttavia Cisco System stima la sua concretizzazione attorno agli anni 2008/2009, quando il numero di dispositivi connessi ad internet superò il numero degli esseri umani. Attualmente si stimano nel mondo più di 8 miliardi di dispositivi IoT connessi alla rete. Il concetto di IoT ha avuto uno sviluppo preponderante nei settori dell'automotive, dell'e-health, della domotica e dell'industria. In questo modo, gli elettrodomestici e oggetti comuni come automobili, orologi o macchine industriali hanno acquisito la possibilità di raccogliere, elaborare e condividere dati in un ecosistema di reti.
La diffusione dell'IoT è stata inoltre favorita dai miglioramenti tecnologici che hanno permesso di creare sensori in scala microscopica. Inoltre, hanno avuto un ruolo decisivo le reti Wi-Fi e la tecnologia Bluetooth, necessarie per regolare la grande quantità di dati da trasferire. Infine, perché l'IoT funzioni correttamente, entra in gioco l'Edge computing, che aggrega ed elabora dati in tempo reale. È grazie alle nuove tecnologie e al potenziamento della connettività wireless, quindi, che l'Internet of Thing ha preso vita, diventando oggi un fenomeno di portata mondiale e determinante per lo sviluppo della società nel suo insieme. Il mondo IoT è tuttora in continuo sviluppo e si prevede di raggiungere quota 29,4 miliardi di dispositivi connessi entro il 2030. **[1]**

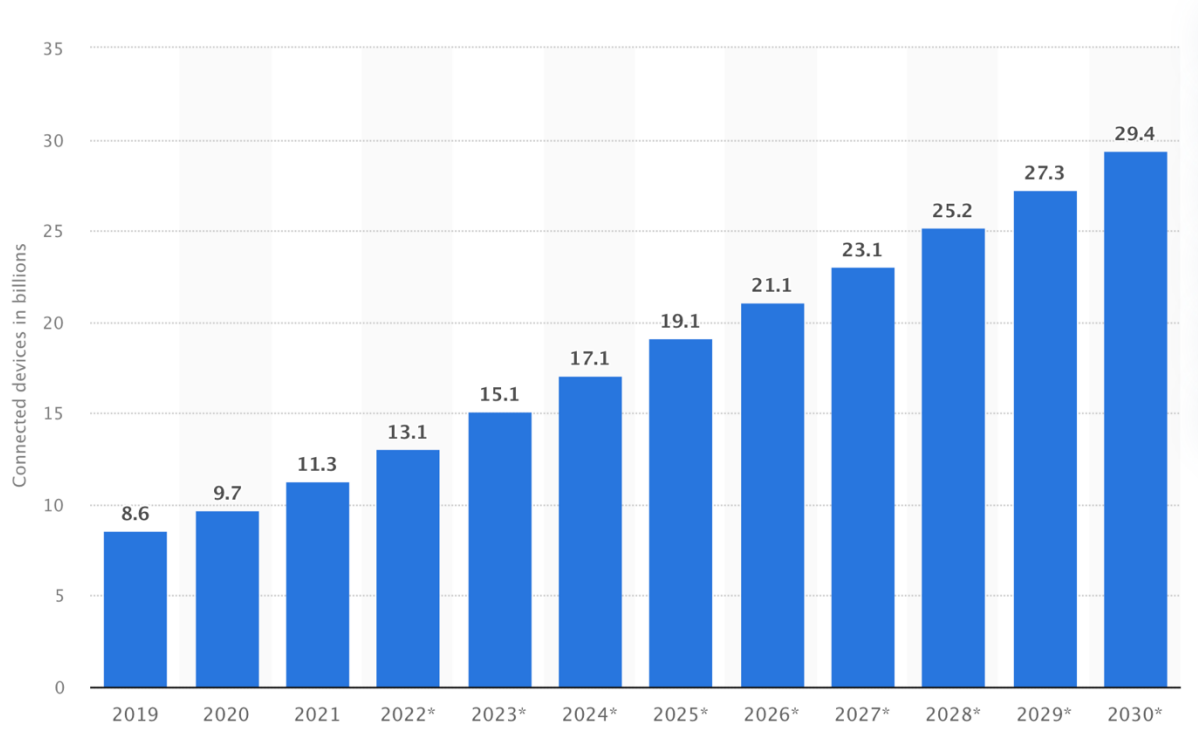

Figura 1: Numero di dispositivi IoT connessi alla rete dal 2019 al 2021, con una previsione degli anni successivi fino al 2030.

Nell'Internet of Things, tutte le Things (cose) che vengono connesse a Internet possono essere suddivise in tre categorie:

1. Things che raccolgono informazioni e poi le inviano.
2. Things che ricevono informazioni e poi agiscono su di esse.
3. Things che fanno entrambe le cose.

I sensori sono innumerevoli, tra i tanti possiamo trovare sensori di temperatura, sensori di movimento, sensori di umidità, sensori di qualità dell'aria, sensori di luce, etc. Tali sensori, assieme a una connessione, consentono il raccoglimento automatico di informazioni sull'ambiente circostante, permettendoci di prendere decisioni più intelligenti.

Ad esempio, in un'azienda agricola, ottenere automaticamente informazioni sull'umidità del suolo può dire agli agricoltori esattamente quando le loro piantagioni hanno necessità di essere annaffiate. Invece di annaffiare troppo (che può essere un uso eccessivo dei sistemi di irrigazione) o annaffiare troppo poco (che può essere una costosa perdita di raccolti), l'agricoltore può garantire che le piantagioni ottengano esattamente la giusta quantità di acqua. Ciò consente all'azienda agricola di aumentare la resa del raccolto riducendo le spese associate e allo stesso tempo diminuire gli sforzi dedicati all'osservazione e cura delle piantagioni.

Tale tesi ha l'obiettivo di risolvere il problema sopracitato, osservando tale problematica dal punto di vista domestico e amatoriale. Infatti, lo scopo della tesi consiste nella progettazione e realizzazione di un sistema embedded per il monitoraggio remoto delle condizioni di salute e ambientali di una pianta, permettendo inoltre il mantenimento del benessere della stessa. La progettazione del sistema embedded è basato su una scheda Arduino MKR WiFi 1010 che legge i dati di umidità del terreno e luminosità ambientale rispettivamente attraverso un sensore capacitivo di umidità del suolo e un sensore di luminosità TSL2561. I dati raccolti sono inviati attraverso le API di Measurify al cloud, dove vengono immagazzinati per permetterne consultazioni future. Successivamente all'analisi dei dati raccolti, il sistema embedded aziona un relè che gestisce una pompa sommergibile che permette l'irrigazione della pianta. Lo schema complessivo è mostrato in figura 2.

Il vantaggio di tale sistema è mantenere in condizioni ottimali di salute una pianta, riducendo al tempo stesso l'intervento necessario da parte dell'utente, che dovrà occuparsi solamente di riempire periodicamente il serbatoio dell'acqua. Inoltre, l'utente può monitorare da remoto le condizioni di salute della pianta attraverso un'applicazione cross-platform (desktop e mobile).

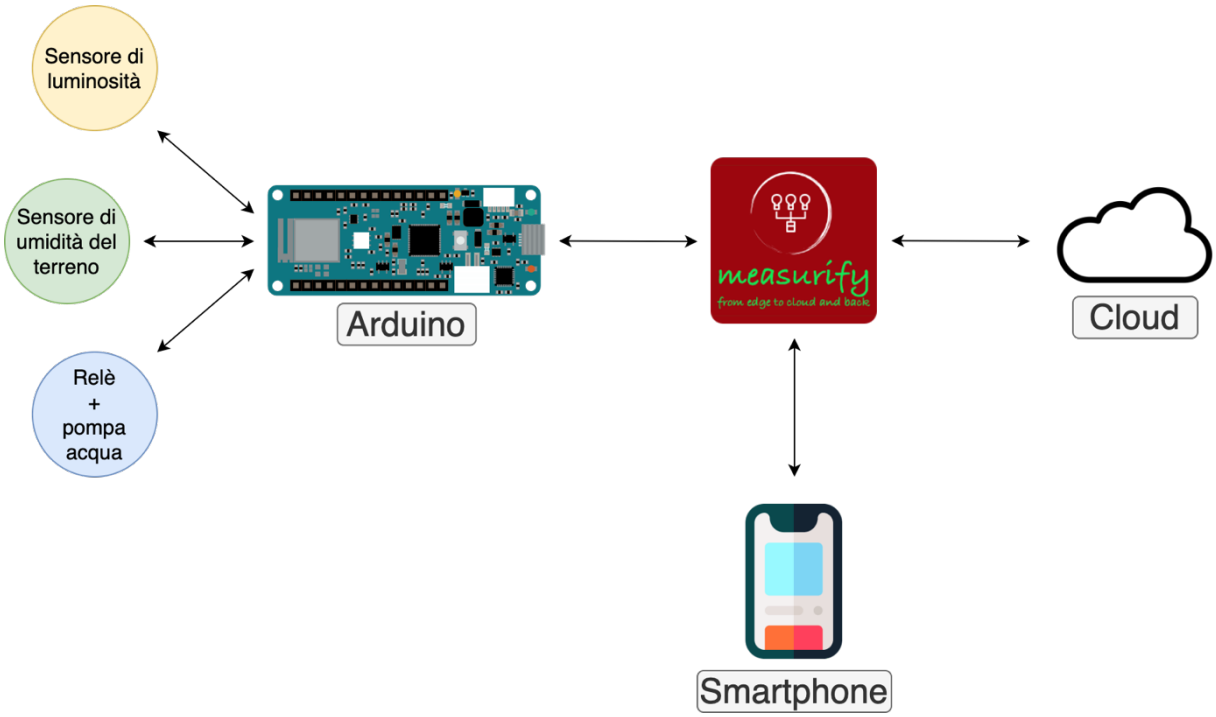

Figura 2: Overview del sistema

2. Metodi e strumenti utilizzati

2.1. Arduino MKR WiFi 1010

Per sviluppare il progetto di tesi è stata utilizzata la scheda Arduino MKR Wi-Fi 1010 **[2]**.
Questa scheda ha dimensioni miniaturizzate e contiene un processore ARM Cortex-M0 32-bit SAMD21. La connettività Wi-Fi e Bluetooth è implementata dal modulo NINA-W102, che opera con le reti di banda 2,4 GHz. Per utilizzare all'interno del proprio codice (sketch) la comunicazione Wi-Fi è necessario utilizzare la libreria Wi-Fi di Arduino oppure la libreria di EdgeEngine.

Per comunicare in maniera sicura, la scheda utilizza il chip crittografico ATECC508, fornito da Microchip Technology. Inoltre, la scheda è dotata di 28 pin I/O (suddivisi in analogici, digitali o alcuni utilizzabili in entrambe le modalità) che permettono il collegamento di sensori e altri dispositivi. In particolare, 10 pin possono essere usati come uscite PWM (pulse width modulation), che permettono in uscita l'erogazione di un segnale digitale in cui il periodo dell'impulso a livello alto varia rispetto al periodo del segnale.

Arduino dispone di una porta di comunicazione micro-USB, utilizzata per alimentare il microcontrollore con una tensione di 5V, effettuare il caricamento dello sketch creato dallo sviluppatore e per comunicare con l'IDE in fase di debug. Infine, è programmabile tramite il linguaggio C++ con l'uso del popolare IDE di Arduino.

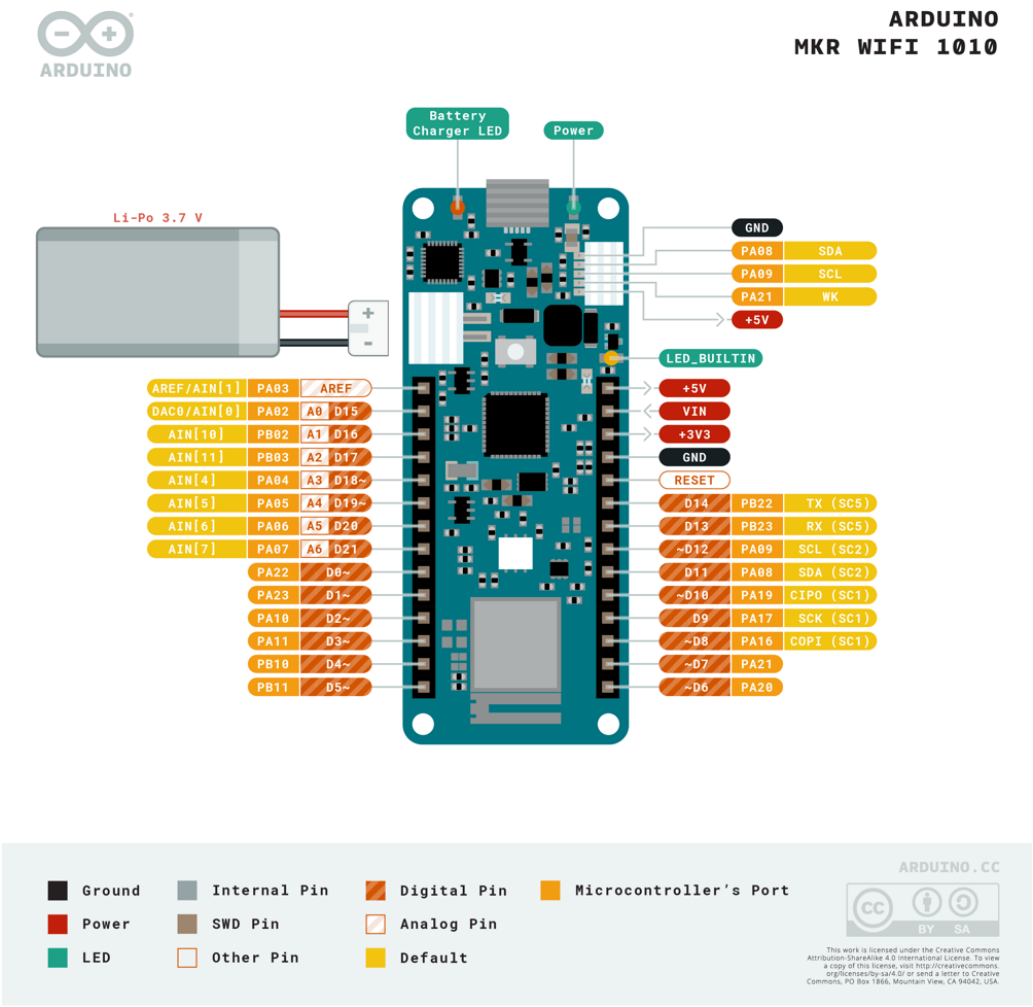

Figura 3: Pin-Out Arduino MKR Wi-Fi 1010

2.2. Sensore di luminosità TSL2561

Il sensore di luminosità TSL2561 **[3]**, realizzato da Electrow, è un convertitore che trasforma l'intensità della luce in un segnale digitale. Tale sensore ha una risposta piatta sulla maggior parte dello spettro visibile e a differenza dei sensori più semplici, misura sia la luce infrarossa (IR) che quella visibile per approssimare meglio la risposta dell'occhio umano.

Inoltre, il TSL2561 è un sensore integratore (assorbe la luce per un periodo di tempo predeterminato) che, modificando il tempo di integrazione, è in grado di misurare piccole e grandi quantità di luce, con misurazioni nel range compreso tra i 0.1 a più di 40k Lux. Vi sono anche due convertitori analogico-digitali (ADC) che sono in grado di integrare contemporaneamente le correnti provenienti dai due fotodiodi.

Questo sensore è dotato anche di un interrupt programmabile ed è in grado di comunicare con il microcontrollore sfruttando il protocollo di comunicazione I2C. La sua tensione di funzionamento varia tra i 2.7V e i 3.6V.

Questo sensore è dotato di cinque pin, partendo dall'alto verso il basso troviamo:

- **VCC:** Pin di tensione di alimentazione della scheda che deve essere pari a 3V per far funzionare correttamente il sensore.
- **GND:** Pin che rappresenta la massa (terra).
- **SCL e SDA:** Sono i pin dedicati alla comunicazione tramite il protocollo I2C, è tramite questi che il sensore riceve i comandi e trasmette le misurazioni al microcontrollore.
- **INT:** Si tratta del pin dedicato all'interrupt e va collegato al microcontrollore solo se necessario. In questo progetto è stato scollegato, in quanto le misurazioni vengono effettuate periodicamente e non vi sono valori per i quali bisogna subito avvisare l'utente.

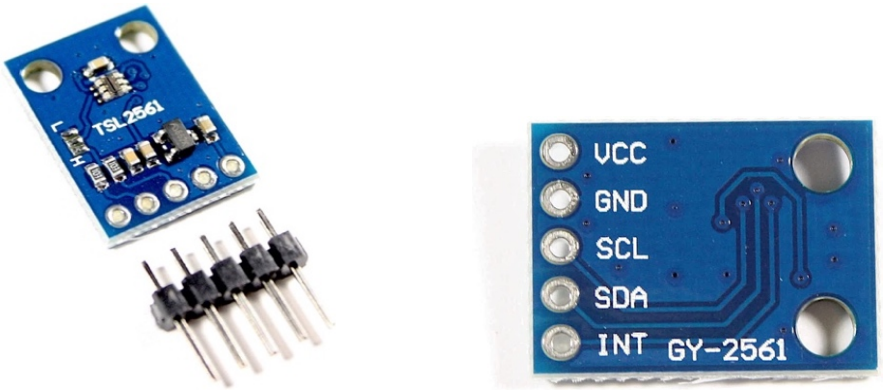

Figura 4: Sensore di luminosità TSL2561

2.3. Libreria Adafruit TSL2561

Per la scrittura del codice, necessario alla gestione da parte di Arduino del sensore di luminosità, è stata utilizzata la libreria TSL2561_U di Adafruit **[4]**.

Di seguito vengono descritti i principali elementi di questa libreria:

- **Adafruit UTSL2561 Unified(addr, id):** Tale classe permette di istanziare un oggetto di tipo Adafruit_TSL2561_Unified, che fornisce accesso a tutti i metodi di gestione del sensore all'interno dello sketch. Il costruttore della classe prevede in ingresso due argomenti. Il primo è l'indirizzo I2C del dispositivo. Il secondo è l'id univoco che viene utilizzato per identificare il dispositivo nei data log.
- **begin():** Questo è un metodo fondamentale, in quanto stabilisce la connessione tra Arduino e il sensore.
- **enableAutoRange(bool):** Abilita la modalità auto-range del sensore, consentendogli di adattare automaticamente la sua sensibilità in base alle condizioni ambientali. Come argomento prende in ingresso un booleano (true = modalità abilitata).
- **setIntegrationTime(time):** Metodo che imposta il tempo di integrazione del sensore e quindi la risoluzione delle misurazioni. Ho deciso di utilizzare la risoluzione migliore corrispondete a un tempo di 402 ms, in quanto non è necessario effettuare le misurazioni velocemente.
- **getEvent(event):** Il metodo effettua la misurazione dal sensore, inserendo il risultato all'interno del campo light dell'argomento event.

2.4. Sensore di umidità del terreno

Per misurare l'umidità del terreno ho utilizzato un sensore di tipo capacitivo **[5]**, che rispetto al sensore di tipo resistivo presenta vantaggi significativi, quali tensione di uscita pressoché lineare, elevata stabilità di funzionamento nel tempo, ampia gamma di misura RH (umidità relativa) e costo contenuto.

Il sensore di tipo resistivo sfrutta la differente resistività del terreno data dalla diversa umidità, infatti se il terreno è secco è meno conduttivo rispetto al terreno umido. Questo sensore ha due problemi:

- la resistività del terreno dipende anche dal tipo di terreno oltre che dall'umidità.
- la corrosione; tale sensore è formato da due lingue metalliche e conduttrici poste ad una distanza di circa 0.5/1 cm l'una dall'altra. In queste due linguette scorre una corrente usata per misurare la resistività. Quando scorre corrente si ha un effetto elettrolisi sui contatti, portando all'ossidazione di uno e alla riduzione dell'altro, fino a falsare completamente le misurazioni.

Il sensore di tipo capacitivo non ha la necessità di avere parti metalliche direttamente a contatto con il terreno umido, risultando quindi molto meno soggetto alla corrosione.

Il sensore da me utilizzato è il KeeYees KYES516 che utilizza, per il collegamento con Arduino, un'interfaccia a 3 pin:

- **GND e VCC:** Sono i pin di alimentazione della scheda, GND è la massa mentre VCC è la tensione di alimentazione. Quest'ultima deve essere compresa tra 3.3V e 5.5V per far funzionare correttamente il sensore.
- **AOUT:** Il pin dedicato alla trasmissione dei dati, va collegato a un pin analogico del microcontrollore.

È importante sottolineare che questo tipo di sensore non è totalmente impermeabile. Infatti, prima dei componenti elettronici vi è una riga bianca perpendicolare alla lunghezza del dispositivo che rappresenta il limite fino al quale si può inserire il sensore nel terreno (si veda *Figura 5*).

La lettura dei valori di umidità può essere effettuata in tempo reale e i valori variano nel range 310-700.

Per determinare tali valori sono stati effettuati due test:

- **Test in acqua:** Consiste nel verificare quale valore riporta il sensore mentre è immerso in acqua. Il valore ottenuto in questo modo è stato 310.
- **Test in aria:** Consiste nel verificare il valore riportato dal sensore mentre è esposto all'aria. La misura ottenuta è stata di 700.

Queste due misure ottenute rappresentano il 100% (310) e lo 0% (700) di RH. Per convertire tali valori in umidità relativa, è stata usata la seguente formula:

$$\frac{700 - N}{700 - 310} \bullet 100$$

Dove N rappresenta la misura che si vuole convertire in %RH.

Figura 5: Sensore capacitivo di umidità del terreno

2.5. Kit irrigazione

L'Arduino gestisce l'irrigazione della pianta attraverso 3 componenti: un relè, una pompa dell'acqua ad immersione e 1 metro di tubo flessibile.

Il **relè** è un dispositivo che permette di collegare un interruttore a un circuito di corrente. Si tratta di un elemento fondamentale all'interno di un circuito elettrico e in uso da moltissimo tempo.
Negli usi più comuni, il relè è un elemento essenziale quando si vuole un sistema per accendere o spegnere un dispositivo in maniera alternativa, ossia senza utilizzare l'interruttore classico.
Un relè, infatti, è in grado di aprire o chiudere un circuito tramite un impulso elettrico. Alla base del suo funzionamento e quindi del suo utilizzo, vi è una variazione di corrente: il relè devia il flusso di corrente senza interromperlo, permettendo quindi il fluire della corrente stessa su un altro circuito.
I relè vengono suddivisi in tre differenti generazioni (prima, seconda e terza generazione) e classificati in base a una serie di parametri fisici ed elettrici. Tuttavia, dal punto di vista tecnico, possono essere sempre considerati simili, ossia caratterizzati da tre elementi principali: una bobina, una ruota dentata e un contatto elettrico.
Quando viene azionata la bobina, si genera un campo elettromagnetico in grado di mettere in movimento la ruota dentata che, a sua volta, azionerà il contatto elettrico che collegandosi con l'interruttore (o gli interruttori in caso di relè a più contatti) aprirà o chiuderà il circuito.

In questo progetto di tesi il relè utilizzato è a un canale e ha una VCC di 5V.
Tale relè è stato utilizzato per azionare e spegnere la pompa dell'acqua.

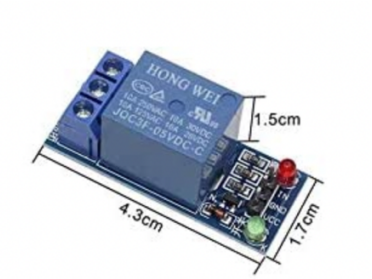

Figura 6: Relè 1 Canale 5V

Una **pompa dell'acqua** è un dispositivo in grado di generare un flusso di liquido utilizzando l'energia cinetica. Pertanto, ha alcuni elementi di base:

- *Ingresso*: dove viene assorbito il liquido.
- *Motore + Elica*: quello incaricato di generare l'energia cinetica che estrae l'acqua dall'ingresso e la invia attraverso l'uscita.
- *Produzione*: è il passaggio da cui uscirà il liquido spinto dalla potenza della pompa dell'acqua.

La pompa dell'acqua utilizzata è di tipo a immersione e funziona con tensione in ingresso compresa tra i 3.3V e i 5V. Tale pompa viene azionata esclusivamente a seguito di valori di umidità bassi per la sopravvivenza della pianta.

Figura 7: Pompa dell'acqua a immersione

Infine collegato alla pompa dell'acqua vi è un 1 metro di **tubo flessibile** in PVC che permette lo scorrimento dell'acqua e l'irrigazione della pianta.

Figura 8: Tubo flessibile in PVC

2.6. Measurify

Un altro importante aspetto da trattare è quello del server Measurify **[6]**, che ha il compito di memorizzare i dati e successivamente elaborarli.
Measurify è un API cloud (di tipo RESTful) orientata alla raccolta e all'elaborazione di dati per la gestione di oggetti intelligenti negli ecosistemi IoT. Si concentra sul concetto di misurazione, questo tipo di dato è infatti molto comune nell'IoT, il che rende più semplice ed efficace il processo di astrazione necessario per mirare a diversi settori e contesti operativi.
Measurify introduce i seguenti concetti:

- **THING**: si tratta di un oggetto per cui si sta operando una misurazione. Nel mio caso la thing è la pianta da monitorare.
- **FEATURE*:*** si tratta della grandezza fisica che si intende misurare, come l'umidità del terreno e la luminosità.
- **DEVICE*:*** si tratta del dispositivo (hardware/software) o il sensore che permette la misurazione di una certa grandezza fisica (feature) su un determinato oggetto (thing).
- **MEASUREMENT*:*** si tratta di una misura effettuata da un dispositivo (device) per una certa grandezza fisica (feature) su un determinato oggetto (thing). Essa contiene, oltre tutte le informazioni sulla misura effettuata, anche informazioni riguardo la data e l'ora di inizio e fine della misura, la thing sulla quale è stata eseguita la misura, il device che l'ha rilevata, la feature che rappresenta e il valore della misurazione attraverso un array di samples.

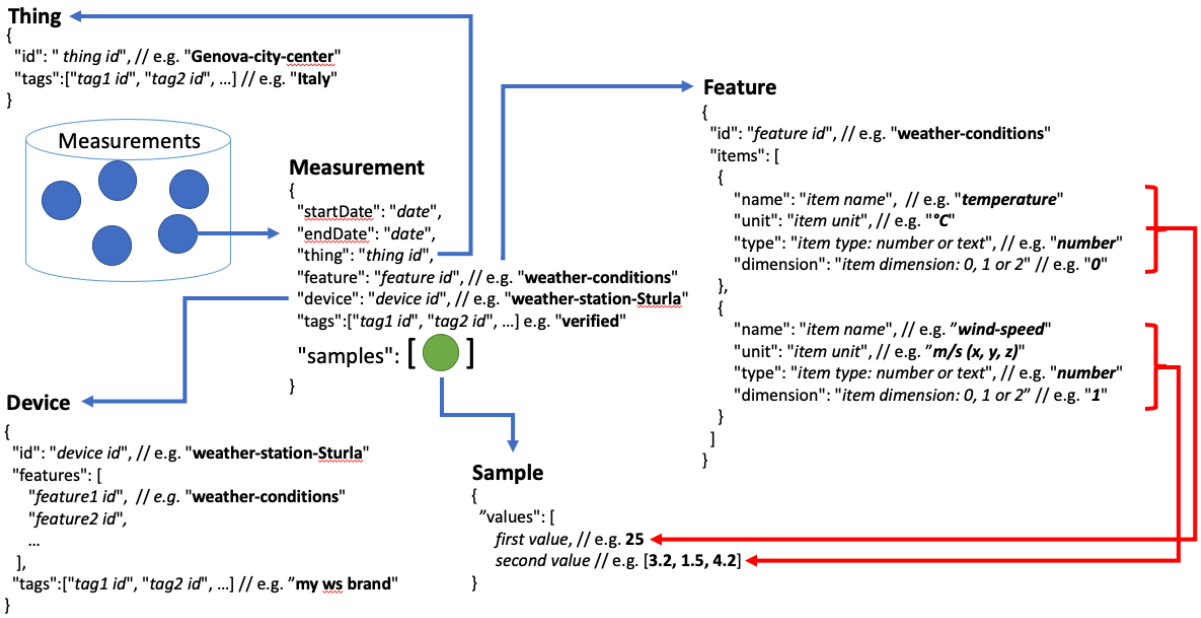

Figura 8: Schema di funzionamento di Measurify

I concetti di Measurement e Feature permettono astrazione rispetto ai valori memorizzati e richiesti dal cloud. Questo è possibile soltanto grazie al fatto che i valori all'interno delle Measurement devono corrispondere alla descrizione rappresentata dalla Feature: ogni volta che Measurify riceve una Measurement, viene utilizzato lo schema della Feature per controllare che i valori siano strutturati correttamente e in caso di successo, tali valori sono archiviati all'interno di un database che l'utente potrà interrogare e modificare attraverso richieste HTTP (GET, POST, PUT, DELETE) differenti in base a ciò che si vuole effettuare. Per effettuare le varie richieste HTTP si è utilizzato Postman.

2.7. Edge Engine

Edge Engine è una libreria scritta in codice C standard creata dall' Elios Lab. Può essere compila per board con microcontrollore (Arduino) o per piattaforme desktop (Linux, MacOS, Windows).
Questa libreria elabora i flussi dei dati provenienti dai sensori tramite script o esegue comandi utilizzando gli attuatori ad esso collegati. Gli script sono costituiti da un insieme di operazioni note che possono essere utilizzate insieme per eseguire calcoli, anche complessi, su ciascun flusso di misura.
L'Edge Engine può essere configurato per ottenere dal cloud (Measurify) gli script da eseguire localmente e controllare periodicamente se questi sono cambiati o se ci sono nuovi script da eseguire.

2.8. Postman

Postman **[7]** è una piattaforma API pensata per gli sviluppatori per progettare, costruire e testare le API. Si tratta di uno strumento che permette di eseguire richieste HTTP ad un server di back-end.
È possibile specificare manualmente il verbo, gli headers e il body della richiesta e analizzare la risposta ricevuta dal server.
Ho utilizzato Postman per interfacciarmi con il server di Measurify. Ad esempio, per ottenere le misurazioni inviate da Arduino al server ho costruito per prima cosa una richiesta di login al seguente URL:

https://students.measurify.org/v1/login

utilizzando il verbo POST e inserendo all'interno del body le credenziali di accesso.

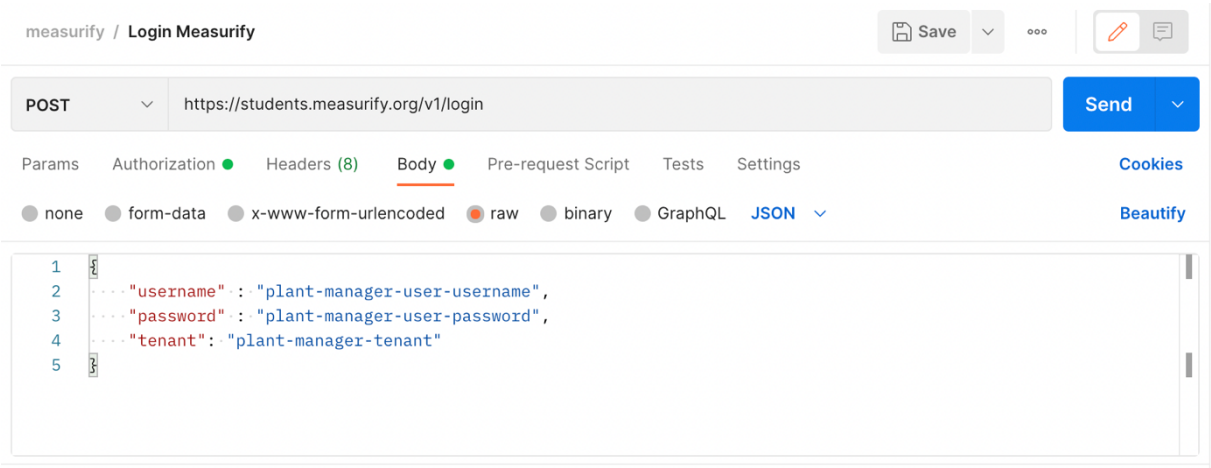

Figura 9: Richiesta di Login

In un modello di comunicazione client-server è spesso necessario, per questioni di sicurezza, che il server debba identificare il client. L'identificazione può avvenire attraverso delle credenziali che dovrebbero essere specificate in ogni richiesta inviata dal client al server, ma questo comporta una probabilità alta che tali informazioni vengano rubate da client malevoli. Per ovviare a questa problematica sono state sviluppate diverse tecniche di verifica, tra cui il modello di autenticazione **token-based** utilizzato anche da Measurify: se il server verifica che le credenziali specificate dal client sono corrette, nella risposta da parte del server vi sarà un token che il client potrà utilizzare nelle richieste successive per identificarsi senza la necessità di specificare nuovamente le credenziali. Solitamente è previsto che il token abbia una durata di validità molto limitata, in modo che un possibile client esterno che ne entra in possesso abbia un tempo molto limitato per eseguire azioni malevole.

Nel caso di Measurify, il token ha durata di 30 minuti. Se l'autenticazione ha successo, il body della risposta ricevuta da Measurify contiene una stringa formattata in JSON e tra i suoi membri possiamo trovare il token di identificazione. Il client, per essere identificato dal server nelle richieste successive, dovrà utilizzare tale informazione, in particolare dovrà essere specificato all'interno dell'header con chiave "Authorization".

Una volta ottenuto il token dalla richiesta è stato possibile interrogare Measurify, utilizzando il verbo GET per ottenere i valori delle misurazioni inviati da Arduino, all'URL:

https://students.measurify.org/v1/measurements

In Figura 10 è presente un esempio di risposta ricevuta su Postman da Measurify, che rappresenta una misurazione effettuata il 21-08-2022 (con orario GMT) sulla 'plant B' con valori di luminosità e umidità del suolo rispettivamente di 38 e 315.


```
{
    "visibility": "private",
    "tags": [],
    "_id": "6301ba49f859ae0040b22a1c",
    "thing": "plant B",
    "feature": "plant-state",
    "device": "plant-monitor",
    "samples": [
        {
            "values": [
                38,
                315
            ]
        }
    ],
    "startDate": "2022-08-21T04:53:32.489Z",
    "endDate": "2022-08-21T04:53:32.489Z"
}
```


Figura 10: Esempio risposta ricevuta da Measurify su Postman

Vi è la possibilità di filtrare i dati richiesti utilizzando un solo filtro o combinare i vari filtri insieme tra loro per limitare le misurazioni a quelle di nostro interesse. Ad esempio "*filter={"thing":"plant B"}&limit=25&page=2"* è una combinazione di tre filtri che permettono di visualizzare solo i dati relativi alla pianta B, con un limite di 25 misurazioni per pagina mentre si sta richiedendo la seconda pagina tra tutte quelle disponibili.

Postman è risultato molto utile anche durante lo sviluppo dell'applicazione per smartphone, mi ha permesso di effettuare una comparazione con i dati mostrati dall'app, ma anche capire come effettuare le richieste di misurazioni, eventuali filter e come elaborare le risposte JSON. Possiede inoltre la comoda possibilità memorizzare variabili di ambiente e usarle nelle successive chiamate, cosa che è stata fatta con il token in modo da non doverlo inserire in ogni richiesta.

2.9. Flutter

L’evoluzione continua dei dispositivi mobile ha spinto gli sviluppatori a studiare a fondo i meccanismi e le linee guida di sviluppo di applicazioni per i principali sistemi operativi e piattaforme, in modo da poter raggiungere più utenti possibile. Con il passare degli anni lo sviluppo di applicazioni è diventato più semplice, ma resta comunque difficile per un unico sviluppatore riuscire a realizzare e mantenere nel tempo un'app nativa per diversi sistemi operativi e piattaforme. In questa ottica nasce **Flutter [8]**, un framework open-source e cross-platform di Google per la creazione di applicazioni multipiattaforma compilate in modo nativo da un'unica base di codice.

Flutter si basa su alcuni elementi principali:

- **Dart:** è un linguaggio di programmazione sviluppato da Google con lo scopo di sostituire JavaScript per lo sviluppo Web. Il compilatore di Dart permette di scrivere programmi sia per web sia per desktop, smartphone e server, tramite due diverse piattaforme. L'intento di Dart è quello di risolvere i problemi di JavaScript, offrendo al tempo stesso migliori prestazioni, la possibilità di sviluppare più facilmente strumenti utili alla gestione di progetti di grandi dimensioni e migliori funzionalità legate alla sicurezza.
- **Flutter Engine:** è scritto in C++ e fornisce supporto per il rendering a basso livello utilizzando la libreria grafica di Google, Skia Graphics. Inoltre, si interfaccia con l’SDK della piattaforma specifica come quelli di Android e iOS. Una particolarità molto apprezzata del Flutter engine, grazie al codice scritto in Dart, è quella di poter effettuare un "hot-reload" dell'applicazione dove la modifica del codice viene iniettata immediatamente all'interno dell'applicazione così da visualizzare all'istante le modifiche effettuate senza un riavvio completo o un cambio di stato.
- **Foundation Library:** Si tratta di una libreria scritta in Dart, che fornisce tutte le classi e i metodi di base per sviluppare applicazioni tramite Flutter.
- **Widget:** La progettazione dell'interfaccia utente in Flutter prevede l'assemblaggio e/o la creazione di vari widget. Un widget in Flutter rappresenta una descrizione immutabile dell'interfaccia utente; grafici, testo, forme e animazioni vengono creati utilizzando i widget. È possibile creare widget più complessi combinandone diversi semplici.

L’editor utilizzato in questa tesi è Visual Studio Code, realizzato da Microsoft, che implementa e si interfaccia con l’SDK.

3. Sperimentazioni e risultati

3.1. Acquisizione dati dalla scheda Arduino

Il sistema complessivo necessario all'acquisizione dei dati e al loro invio a Measurify è composto da un microcontrollore, Arduino MKR Wi-Fi 1010, al quale sono stati collegati tutti i sensori e i dispositivi descritti precedentemente.

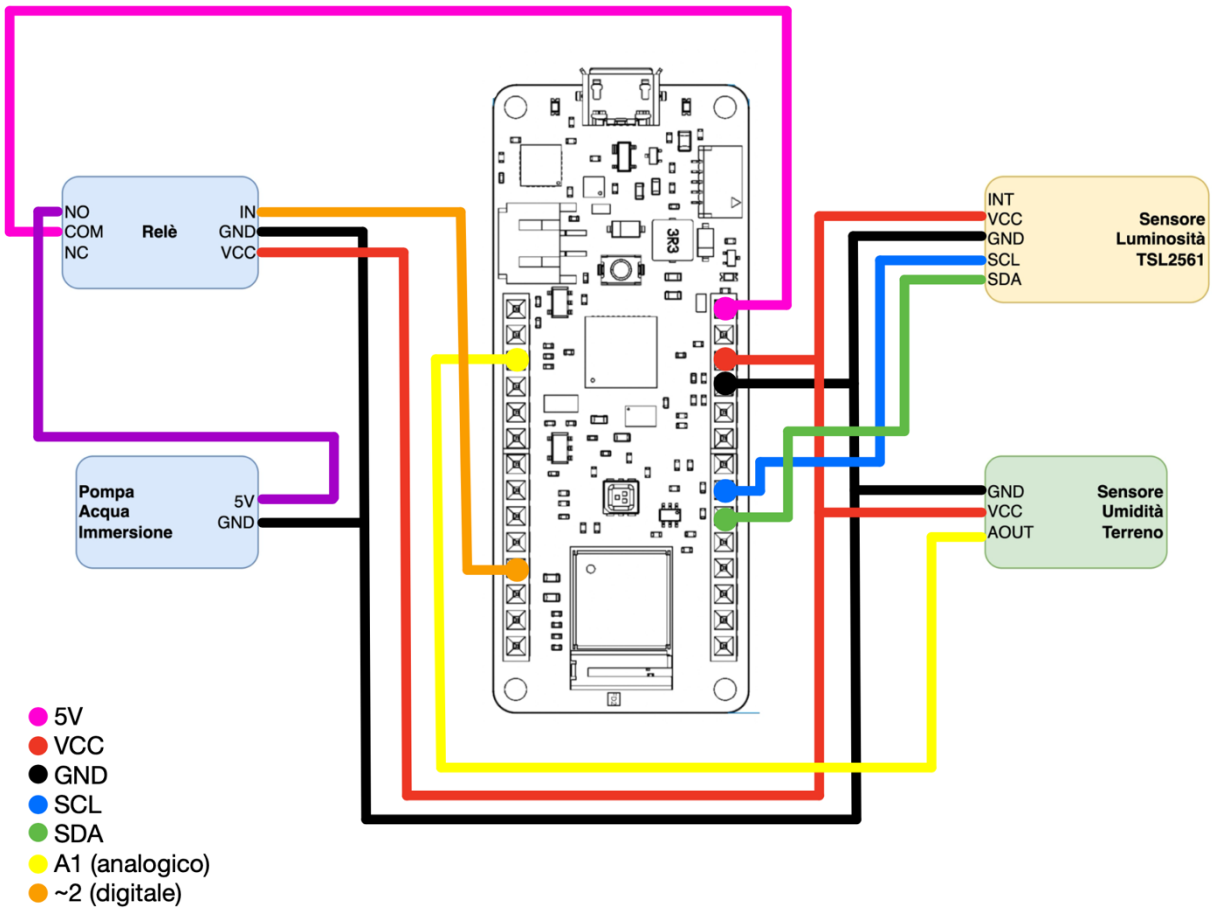

Figura 11: Diagramma di connessione Arduino

- **Cablaggio Sensore di Luminosità:**
  Il lato sinistro corrisponde ai pin del sensore di luminosità, mentre il lato destro corrisponde alla board di Arduino.

SDA → SDA

SCL → SCL

VCC → VCC (3.3V)

GND → GND

- **Cablaggio Sensore di Umidità:**
  Il lato sinistro corrisponde ai pin del sensore di umidità, mentre il lato destro corrisponde alla board di Arduino.

AOUT → Pin A1 (analogico)

VCC → VCC (3.3V)

GND → GND

- **Cablaggio Relè:**
  Il lato sinistro corrisponde ai pin del relè mentre il lato destro corrisponde alla board di Arduino.

IN → Pin ~2 (digitale)

VCC → VCC (3.3V)

GND → GND

COM → 5V

- **Cablaggio pompa dell'acqua ad immersione:**
  Il lato sinistro corrisponde ai capi della pompa dell'acqua, mentre il lato destro corrisponde al relè e al GND comune.

5V → Normally Open Port

GND → GND

Per l'alimentazione della scheda Arduino ho utilizzato inizialmente il computer e successivamente un caricabatterie portatile da 10000 Mah collegato alla sua porta USB.
Il sistema Arduino esegue le istruzioni che sono racchiuse all'interno di uno sketch, che viene caricato su Arduino attraverso l'Arduino IDE disponibile su PC.
Lo sketch è composto da due fasi, quella di setup che è la preparazione e quella di loop che è l'esecuzione. Entrambe le funzioni sono necessarie per l'esecuzione del programma.

3.1.1. Setup

La fase di setup () è posta sempre all'inizio del programma e viene eseguita solamente una volta all'avvio del sistema. In questa funzione vengono dichiarati gli oggetti utilizzati per interagire con i sensori e vengono avviati tutti i collegamenti ai dispositivi collegati al microcontrollore.
Nel caso in cui il sensore di luminosità fallisse la fase di setup, ho predisposto l'accensione di un led rosso per notificare la condizione di errore.

Inoltre, nella funzione di setup si stabilisce la connessione di rete attraverso il Wi-Fi e viene inizializzata la connessione verso Measurify attraverso la libreria di Edge Engine.

3.1.2. Loop

La funzione loop () fa girare consecutivamente, per tutta la durata di funzionamento del sistema, il programma contenuto all'interno delle parentesi graffe (lettura ingressi, attivazione uscite ecc), permettendo al programma di scambiare, rispondere e controllare la scheda Arduino.

Di seguito analizzo l'algoritmo in dettaglio:

- **Rilevazione dei dati dai sensori:** All'inizio della funzione di loop viene effettuata per prima la misurazione dal sensore di umidità attraverso la lettura dal pin A1 di un valore compreso tra 310 e 700. Tale valore verrà poi convertito in percentuale di umidità attraverso una funzione di mapping. Successivamente, viene eseguita la lettura del sensore di luminosità attraverso l'utilizzo della funzione **sensorLight.getEvent()** descritta nel paragrafo precedente.
- **Innaffiatura:** a partire dalla misurazione di umidità svolta in precedenza, uno statement condizionale determina se l'umidità rilevata è tale da necessitare l'accensione della pompa dell'acqua. Nel mio caso quando l'umidità del terreno è inferiore al 75%, allora l'algoritmo aziona il relè, per un tempo di 2500 ms, che fornisce corrente alla pompa dell'acqua.
- **Invio dei dati a Measurify:** Una volta ultimati tutti gli step precedenti l'algoritmo istanzia una struttura dati compatibile con Measurify contente tutte le misurazioni di luce e umidità da salvare sul cloud. Infine, l'algoritmo chiama le API di Edge Engine per il caricamento dei dati (funzione POSTMeasurement).
- **Clean up e Attesa:** Le ultime due operazioni effettuate dall'algoritmo sono: rimozione dalla memoria degli oggetti non più utili attraverso la funzione clean() e chiamata alla funzione delay() per la messa in stand-by di Arduino per 20 minuti.

3.2. Applicazione Smartphone

La seconda parte della tesi consiste nella realizzazione di un'applicazione smartphone con lo scopo di monitorare la pianta da remoto e comprendere se necessita di cure (ad esempio essere spostata in un luogo più luminoso). L'applicazione è stata sviluppata mediante l'utilizzo del framework open-source Flutter, realizzato da Google, tramite l'editor Visual Studio Code **[9]**.
L'applicazione è stata chiamata "SafePlant" e supporta i sistemi operativi mobile (iOS, Android) e desktop (Windows, Linux e MacOS).

Appena avviata l'applicazione mostra una schermata di **Login** dove vengono richieste le tre credenziali di Measurify: username, password e tenant per accedere al proprio account e all'applicazione stessa. Il primo elemento della schermata, guardando dall'alto al basso, è un'immagine che riporta il logo dell'app. Sotto questa vi è una Card che racchiude vari widgets. Sono stati alternati widgets di tipo Text (es. "Username", "Password"..) e TextField (es." Enter your username") dove l'utente deve inserire il dato richiesto. Per permettere all'utente di capire dove sta inserendo i dati, nel momento in cui viene toccato un campo il suo bordo si illumina di verde, restando illuminato per tutto il tempo dell'inserimento.
L'ultimo elemento della Card è un Button che se premuto tenta il login tramite una richiesta POST a Measurify.
Una volta inseriti tutti i dati richiesti, nel caso in cui questi fossero errati (la risposta alla richiesta POST non è andata a buon fine per vari motivi) l'applicazione mostra un Alert di errore permettendo all'utente il reinserimento dei dati, in caso contrario invece si accede al Cloud e, dopo una breve schermata di caricamento dati, viene aperta la pagina Home.

La **Home** contiene tutte le informazioni raccolte dai sensori di Arduino. La pagina contiene tre widget di tipo Card. La prima mostra, colorati in verde, i valori in percentuale dei dati dell'ultima misurazione effettuata da Arduino. Sono stati utilizzati due widget di tipo Text che mostrano sempre l'ultimo valore aggiornato.
La seconda card contiene i grafici temporali di luminosità e di umidità. Tali grafici sono stati realizzati attraverso l'utilizzo della libreria draw-graph, sono interattivi e consentono all'utente la visualizzazione sovrapposta dei dati di misurazione, oppure singola. Per visualizzare il grafico di una misurazione singolarmente è sufficiente cliccare il nome corrispondente alla misurazione dalla legenda.
Vi è la possibilità di selezionare la modalità di visualizzazione dei dati per giorno, settimana e mese, consentendo lo scorrimento avanti e indietro dei dati sulla base della modalità selezionata.
Infine l'ultima Card presenta due pulsanti, uno che porta alla schermata Luminosity_info e l'altro alla PlantCareTips.

La **Luminosity_info** contiene informazioni utili per la comprensione e la lettura del grafico di luminosità (che è stato mappato in maniera non lineare secondo la guida di Microsoft **[10]**), contiene infatti una tabella che rappresenta soglie approssimative per le condizioni di illuminazione comuni e il passaggio di illuminazione corrispondente. In questo caso, ogni passaggio di illuminazione rappresenta un cambiamento nell'ambiente di illuminazione.

Infine la **PlantCareTips** contiene dei consigli utili sulle azioni che solo l'utente può gestire, come il luogo dove posizionare la pianta, l'utilizzo del fertilizzante e capire quando alla pianta mancano dei nutrimenti in base a come si presentano le foglie.

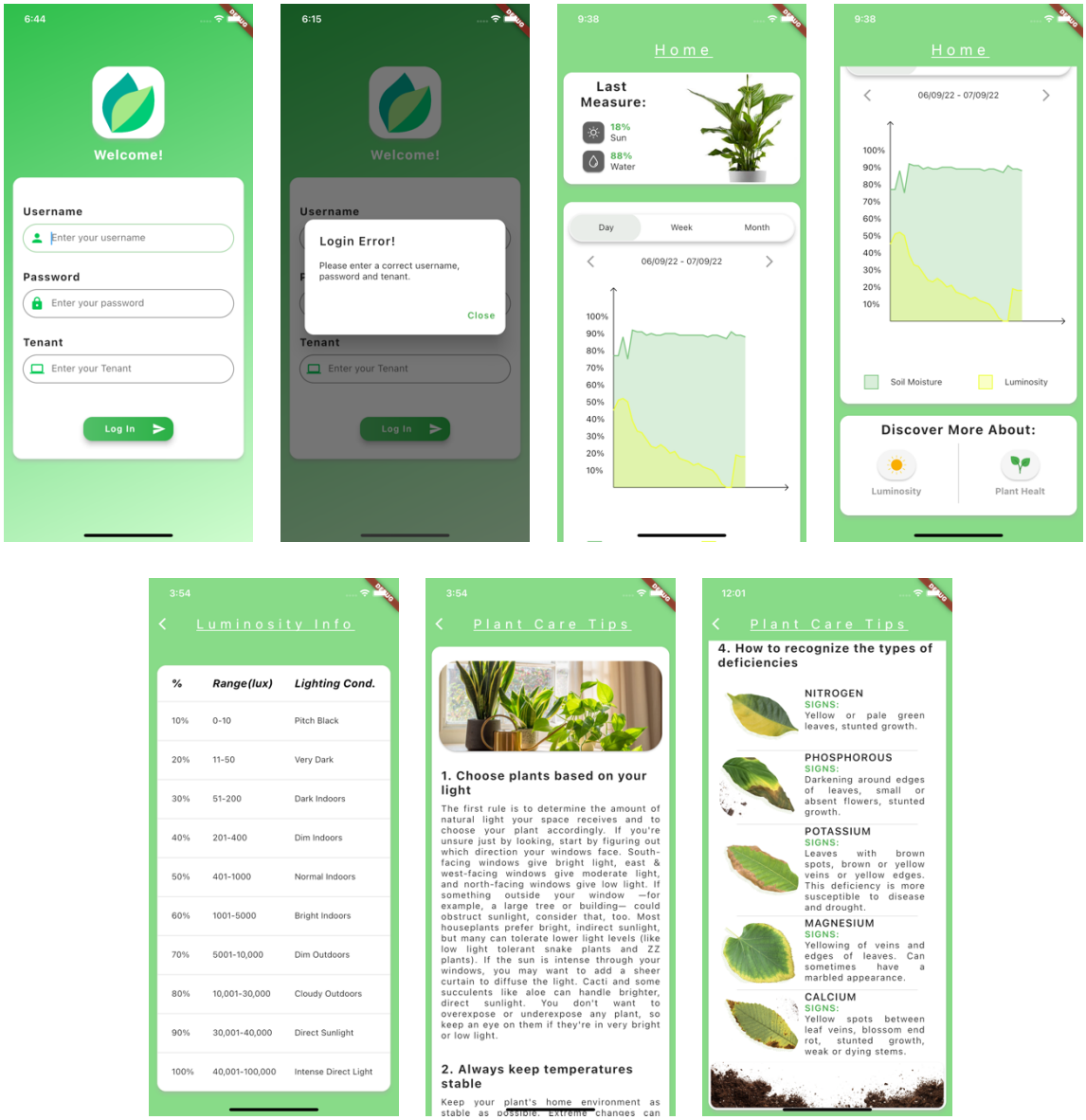

Figura 11: Schermate dell'applicazione

4. Contributo personale e considerazioni conclusive

L' obbiettivo della tesi è stato quello di sviluppare un sistema embedded per il monitoraggio e il mantenimento in salute di piante, ma anche lo sviluppo di un'applicazione smartphone, in modo da poter rendere consultabili i dati da remoto.

I dati acquisiti dal sistema sono stati inviati al cloud, attraverso le RESTful API messe a disposizione dal framework open-source Measurify. Il supporto di Measurify mi ha permesso inoltre di accedere ai dati del cloud attraverso l'applicazione smartphone.
L'utilizzo di Measurify è stato semplice e intuitivo, anche grazie alla grande quantità di materiale disponibile sulla repository.

Per sviluppare lo sketch è stato utilizzato l'IDE di Arduino, questo ha facilitato lo sviluppo del codice grazie a un'ottima documentazione e un'ampia quantità di materiale disponibile online. Tuttavia, ciò che più ha facilitato e velocizzato la programmazione è stato è il gestore delle librerie dell'ambiente di sviluppo che permette, con pochi e semplici passaggi, di cercare e installare librerie per qualsiasi funzionalità e sensore.

Per quanto riguarda lo sviluppo dell'applicazione ho utilizzato Flutter, il framework open-source realizzato da Google. Ho trovato l'utilizzo di Flutter semplice, intuitivo e allo stesso tempo divertente, permettendomi di sviluppare interfacce complesse a partire da semplici widget. A tal scopo, è stata eseguita una ricerca e uno studio della documentazione del framework che ho trovato essere molto ricca e in continuo aggiornamento.
La possibilità di avere un'applicazione cross-platform rappresenta il vantaggio di poterla utilizzare su più piattaforme, rendendo versatile l'applicazione.
Ho ancora apprezzato la possibilità di racchiudere il layout e la parte di funzionamento nello stesso file. Durante lo sviluppo è risultata molto comoda la funzionalità di hot-reload del codice che permette di vedere immediatamente le modifiche apportate.
Per la stesura effettiva del codice è stato utilizzato l'IDE Visual Studio Code, che è stato sviluppato da Microsoft, mentre per la fase di test è stato utilizzato il simulatore di XCode provando l'applicazione su diversi modelli disponibili.

Come sviluppi futuri del progetto potrebbe essere molto interessante aggiungere al serbatoio, da cui la pompa attinge, un galleggiante per la verifica del livello dell'acqua. Tale modifica permetterebbe di monitorare ed eventualmente programmare da remoto l'intervento umano per il serbatoio, mentre l'aggiunta di elettrovalvole permetterebbe la gestione di più piante in contemporanea.
Tra le migliorie vi sarebbe anche la possibilità di ricevere dati meteo per pianificare l'innaffiatura della pianta, in modo da preservare la scorta di acqua disponibile all'interno del serbatoio ed evitare anche un'irrigazione eccessiva.
Un'altra miglioria riguarda il risparmio energetico, come la disattivazione del Wi-Fi o l'invio dei dati solo quando necessario (quando si verifica una variazione consistente delle misurazioni) in quanto attualmente il dispositivo dipende da una batteria esterna.
Infine, l'implementazione di un pannello solare renderebbe quasi totalmente autonomo il sistema.

Complessivamente, sono molto soddisfatta del risultato ottenuto tramite questa tesi sia dal punto di vista della formazione personale sia dal punto di vista del prodotto realizzato. In particolar modo, sono contenta di essere venuta a conoscenza di Flutter in quanto lo reputo uno strumento molto interessante e con un grande potenziale per lo sviluppo di applicazioni mobile, ma anche per aver creato un sistema embedded tramite l'utilizzo di Arduino, che dispone di moltissimi sensori differenti e progetti interessanti, utili anche per la vita di tutti i giorni.

5. Riferimenti Bibliografici

[1] Stime crescita IoT, https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/

[2] Arduino MKR1010, https://docs.arduino.cc/hardware/mkr-wifi-1010

Arduino, https://www.arduino.cc

[3] Sensore luminosità TSL2561, https://www.elecrow.com/wiki/index.php?title=File:Luminosity_Sensor-_TSL2561_Breakout.jpg

[4] Adafruit TSL2561, https://github.com/adafruit/Adafruit_TSL2561/blob/master/examples/sensorapi/sensorapi.ino

[5] Sensore umidità terreno, https://makersportal.com/blog/2020/5/26/capacitive-soil-moisture-calibration-with-arduino

[6] Measurify, https://measurify.org

[7] Postman, https://www.postman.com/product/what-is-postman/

[9] Visual Studio Code, https://code.visualstudio.com

[8] Flutter, https://flutter.dev

[10] Guida Microsoft, https://docs.microsoft.com/en-us/windows/win32/sensorsapi/understanding-and-interpreting-lux-values