Informatica - Area scientifica
Dipartimento di Scienze matematiche, informatiche e multimediali
Università di Udine

Progetto di Internet of Things

"Save the Plant"

Bellotto Chiara (142982)
Da Re Davide (141976)
Galvan Matteo (142985)
Giabardo Diletta (144176)

Anno accademico 2020/2021

Indice

Introduzione

Per il corso di Internet of Things abbiamo deciso di realizzare una PWA (Progressive Web App) che permetta all'utente di controllare alcuni parametri vitali di una pianta da appartamento. Il tutto è stato realizzato tramite l'utilizzo della scheda ESP32 e di appositi sensori collegati ad essa. Questi raccolgono i dati relativi alla pianta grazie al modulo Wi-Fi integrato nella scheda. Successivamente viene creata una connessione alla rete Wi-Fi casalinga che invia i dati raccolti al server a cui è collegata la PWA. La nostra applicazione, oltre a poter essere visitata nel web, potrà essere scaricata nel proprio dispositivo.
I parametri che abbiamo deciso di monitorare sono i seguenti:

- umidità della terra
- umidità e temperatura della stanza
- livello dell'acqua
- luminosità

Nella seguente relazione parleremo nel dettaglio dei dispositivi utilizzati per la realizzazione del progetto, della scrittura del codice per ESP32, della comunicazione tra ESP32 e il server, della creazione della PWA, della raccolta dei dati e delle problematiche che abbiamo avuto durante lo sviluppo del progetto.

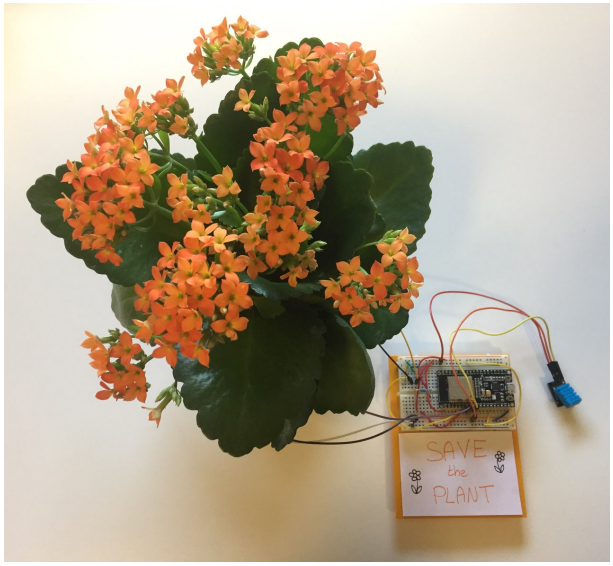

Figura 1: Save the Plant

1 Componenti

Le componenti che abbiamo utilizzato all'interno del nostro progetto sono: ESP32, sensore di luminosità (LDR), sensore di umidità della terra (cavi jumper), sensore di umidità e temperatura della stanza (DHT11) e il sensore del livello dell'acqua (Water Level Sensor).

1.1 ESP-WROOM-32

ESP32 è un chip microcontrollore con Wi-Fi 2.4 GHz e Bluetooth integrato. Può essere considerato come il successore dell' ESP8266: le maggiori differenze consistono nella presenza nell' ESP32 di più pin GPIO con i quali sviluppare dei progetti più complessi senza dover ricorrere ad un GPIO expander, la presenza di connettività Bluetooth e maggiore memoria.

Figura 1.1: ESP-WROOM-32

In particolare, abbiamo deciso di utilizzare l' ESP-WROOM-32, un modello potente della famiglia ESP32-Az Delivery. Si tratta di un modulo versatile e può essere utilizzato per reti a bassa potenza, ma anche per compiti impegnativi. Si presenta di piccole dimensioni e facile da incorporare ad altri prodotti. Inoltre, integra una vasta gamma di periferiche. E' proprio per questi motivi che abbiamo optato per l'utilizzo di ESP32 piuttosto che Arduino Uno o Rasberry Pi. In particolare, l'ultima board si prestava ad essere più strutturata e complessa di quello che ci serviva per la realizzazione del nostro progetto e perciò abbiamo deciso di affidarci a qualcosa di più semplice ma comunque efficiente. Inoltre, tra ESP32 e Arduino Uno abbiamo scelto il primo poichè dotato di Wi-Fi integrato.

1.2 Bright Sensor: LDR (Light Dependent Resistor)

Il Light Dependent Resistor (LDR), anche conosciuto come fotoresistore, serve per misurare la quantità di luce all'interno di un determinato ambiente. Il valore di questo cambia in base alla quantità di luce esposta. LDR restituisce un voltaggio analogico quando viene connesso ai 3.3V, e i dati riportati variano in proporzione all' intensità della luce. L' ESP32, che è stato costruito in ADC(analog-to-digital converter), converte il risultato analogico(che va da 0 a 3.3V) in digitale, con un range che va tra 0 e 4095. Lo 0 rappresenta la luce massima, e il 4095 rappresenta la luminosità minima.
Il dato ottenuto viene poi inviato al server.

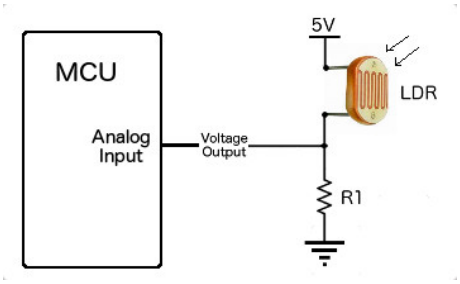

Figura 1.2: Bright Sensor(LDR)

1.3 Cavi jumper

Abbiamo utilizzato i cavi jumper per rilevare l'umidità della terra della pianta. In particolare, abbiamo collegato alla board due cavi femmina. Le estremità di questi le abbiamo inserite nella terra per leggere il suo valore di resistenza dalla quale si potrà capire l'umidità della terra. I valori sono direttamente proporzionali e meno la terra sarà umida, più fatica farà la corrente a transitare e restituirà perciò un valore analogico proporzionale al dato rilevato.

Figura 1.3: Cavi jumper

1.4 Sensore di Temperatura ed Umidità (DHT11):

Il DHT11 è un sensore digitale di umidità e temperatura dell' aria costituito da una parte resistiva che si occupa del rilevamento dell' umidità e da un NTC che rileva la

temperatura. Queste due parti vengono gestite da un microcontrollore che è parte integrante del sensore stesso. I dati rilevati vengono passati in digitale, quindi non è necessaria una conversione da parte dell'ESP32 utilizzato. Il sensore utilizza una tecnica digitale che, unita alla tecnologia di rilevamento dell'umidità, ne garantisce l'affidabilità e la stabilità. I suoi elementi sensibili sono connessi con un processore 8-bit single-chip ed ogni sensore di questo modello è compensato in temperatura e calibrato in un'apposita camera. Questa determina in modo preciso il valore di calibrazione, il cui coefficiente viene salvato all'interno della memoria.

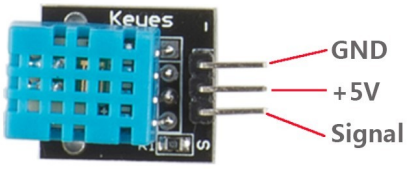

Figura 1.4: DHT11

Gli intervalli che riesce a percepire sono i seguenti:

- Temperatura: da 0 a 50 ºC +- 2 ºC
- Umidità: da 20 a 90% +- 5%

1.5 Water Level Sensor

Il Water Level Sensor è un sensore utile per misurare il livello dell'acqua che si trova in un recipiente. In particolare, lo abbiamo utilizzato per rilevare la quantità di acqua presente nel piatto della pianta in modo tale da sapere quando questo dovrà essere riempito. Il sensore in questione è composto da dieci tracce in rame esposte, cinque delle quali sono tracce di potenza e le altre cinque di rilevamento. Queste non sono connesse a niente e si trovano nella parte del sensore che verrà immerso nell'acqua. Inoltre, è presente un LED di alimentazione che si accende quando la scheda è alimentata. Per collegare il sensore si utilizzano i tre PIN a disposizione: S (signal), + (VCC) e - (GND). Il primo è un'uscita analogica che deve essere collegata a uno degli ingressi analogici dell' ESP32, il secondo fornisce alimentazione al sensore e il terzo è un collegamento a terra.

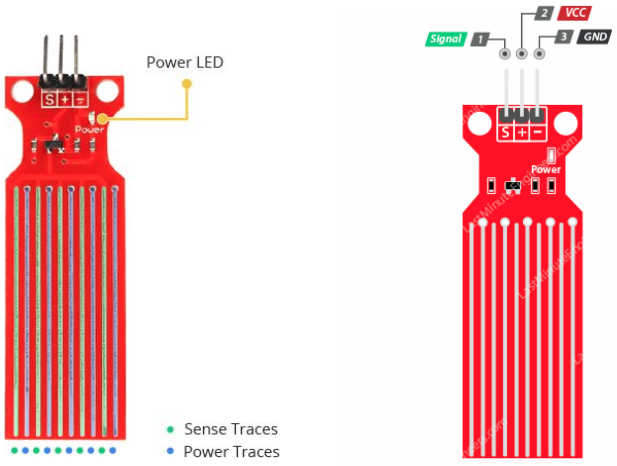

Figura 1.5: Water Level Sensor

La serie di tracce parallele esposte, insieme, agiscono come un resistore variabile, la cui resistenza varia in base al livello dell'acqua. La variazione che si ottiene corrisponde alla distanza dalla parte superiore del sensore alla superficie dell'acqua. Il sensore produce una tensione di uscita in base alla resistenza che possiamo misurare e capire così il livello dell'acqua. La resistenza e l'altezza del livello dell'acqua sono inversamente proporzionali: infatti, più il sensore è immerso, migliore sarà la conduttività e minore sarà la resistenza. Al contrario, con un sensore poco immerso si otterrà una scarsa conduttività e una maggiore resistenza. In particolare, il sensore restituirà 0 come valore quando l'acqua non sarà presente.

2 Progettazione fisica

Di seguito si può visualizzare l'immagine rappresentante il circuito costruito per il nostro progetto.

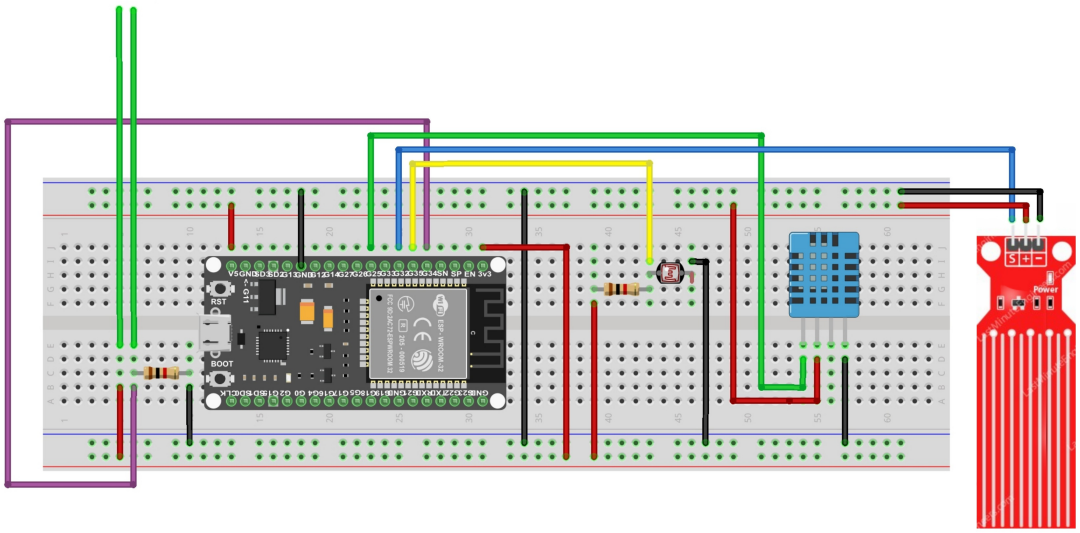

Figura 2.1: Circuito fisico

3 Implementazione

Spiegheremo di seguito come abbiamo programmato l' ESP32 in vista del nostro progetto. Abbiamo utilizzato come ambiente di sviluppo l' IDE di Arduino e riportiamo di seguito il codice seguito da una breve spiegazione.

```
#include "WiFi.h" // ESP32 WiFi include
#include <DHT.h>
#define DHTPIN 25
#define DHTTYPE DHT11
const uint16_t port = 8090;
const char * host = "172.16.151.125";
float valueBright;
float valuewaterLevelSensor;
int temperatureRoomSensor;
int humidityRoomSensor;
float valuehumidityGroundSensor;
float minore = 2700;
float maggiore = 3500;
WiFiClient client;
DHT dht(DHTPIN, DHTTYPE);

void setup()
{
  dht.begin();
  ConnectToWiFi();
}
void ConnectToWiFi()
{
  WiFi.mode(WIFI_STA);
  WiFi.begin("WiFi", "password");

  uint8_t i = 0;
  while (WiFi.status() != WL_CONNECTED){
  }
}

void loop()
{
  WiFiClient client;
  if (!client.connect(host, port)) {
    delay(2000);
    return;
  }

  valueBright = analogRead(35);
```

```
  int brightSensor = (int) ((valueBright / 4095) *100);
  brightSensor = 100 - brightSensor;
  valuewaterLevelSensor = analogRead(32);
  int waterLevelSensor = 0;
  if(valuewaterLevelSensor >=minore){
    waterLevelSensor = (int)
    (((valuewaterLevelSensor-minore)/(maggiore-minore))*100);
  }
  temperatureRoomSensor = dht.readTemperature();
  humidityRoomSensor = dht.readHumidity();
  valuehumidityGroundSensor = analogRead(34);
  int humidityGroundSensor = (int) ((valuehumidityGroundSensor / 1350) *100);
  if (client.connected()) {
    client.print(String(brightSensor) + ";"
    + String(waterLevelSensor) + ";" +
    String(temperatureRoomSensor) + ";" +
    String(humidityRoomSensor) + ";" +
    String(humidityGroundSensor) + ";" + "#");
  }
  delay(30000);
}
```

All'inizio abbiamo dichiarato l'indirizzo IP e la porta a cui collegarsi per il caricamento dei dati e le variabili che identificano i vari sensori collegati all'ESP32.

Nel *void setup()* abbiamo scritto le istruzioni che vengono eseguite all' inizio del programma. In particolare, abbiamo inizializzato la porta seriale e il DHT (il sensore della temperatura e uminità dell'ambiente).

Nella funzione *void ConnectToWifi()* abbiamo inizializzato la connessione al Wi-Fi locale definendone il nome e la password.

Il *void loop()* è la parte principale del programma, ripetuta in continuazione fino a quando non viene tolta l'alimentazione o non viene premuto il tasto reset. In questa funzione abbiamo creato la connessione al Wi-Fi e una volta che ci si riesce a connettere all'host e alla porta iniziano le varie misurazioni da parte dei sensori collegati e il tutto viene stampato. La raccolta dei dati si ripete ogni 30 secondi.

I dati raccolti sono stati opportunamente normalizzati prima di iniziare le misurazioni effettive in modo tale da avere un range più significativo a cui fare riferimento per controllare, successivamente, i parametri vitali della pianta.

4 Trasmissione dei dati

Spiegheremo di seguito come abbiamo implementato il codice per la comunicazione tra l'ESP32 e il server con il relativo invio dei dati.

Utilizziamo un file txt creato precedentemente, "linechartdata.txt", nel quale andremo a inserire i dati delle ultime 20 misurazioni: viene eliminata la prima riga, viene sovrascritto lo stesso file incollando le righe del precedente file unite a quella nuova.
Tramite FTP viene caricato all'interno del sito galvan.altervista.org un file "linechartdata.csv" che viene creato copiando il contenuto del file "linechartdata.txt".

Viene creato un file txt nel quale andremo a inserire i dati delle misurazioni se la seconda cifra dei minuti sarà un "1". Questa caratteristica permette di salvare solo una parte dei dati di tutta la giornata in modo tale da non averne un'eccessiva quantità.
Tramite FTP viene caricato all'interno del sito galvan.altervista.org un file <dd_mm_YY>.csv che viene creato copiando il contenuto del file txt. Il nome del file consiste nel giorno in cui sono state effettuate le misurazioni.

Prima di effettuare la connessione con la socket effettuiamo dei controlli, in particolare se la connessione è già avvenuta o se ci sono stati dei problemi nella creazione di questa andiamo a stamparli a schermo. Ogni qualvolta viene creata una connessione viene generato un thread di servizio che dialoga con l'ESP32 e riceve i dati. La connessione viene terminata nel momento in cui i file vengono caricati tramite FTP. Per eseguire lo shutdown del server sarà sufficiente premere la combinazione di tasti "Ctrl+C".

5 Sito web

All'interno del php del sito web possiamo trovare diverse sezioni implementate:

1. **Home:** all'interno di essa vengono realizzati dei bottoni per la navigazione in una sidebar, che permettono all'utente di spostarsi tra i diversi grafici del sito. Inoltre, vengono mostrati all'utente le ultime rilevazioni effettuate dai sensori affiancati da un cerchietto colorato. Questo viene colorato di verde se i dati sono regolari, altrimenti diventa rosso per segnalare un' abbassamento o alzamento critico dei dati.

2. **Grafici:** per tutti e quattro i sensori vengono utilizzati dei grafici a linea appartenenti alla libreria Graph.js. Sopra i grafici viene implementato anche un bottone "Calendario", che ha la funzione di permettere all'utente di visualizzare i grafici di altri giorni.

3. **Creazione dei grafici:** viene utilizzata la libreria d3.js per estrapolare i dati dai .csv e per chiamare la funzione di creazione del grafico che sto visualizzando in quel momento. Ogni 10 secondi i dati vengono riletti e i grafici vengono ricreati.

5.1 Strumenti per la realizzazione del sito web

All'interno del nostro sito web abbiamo utilizzato diversi strumenti per la realizzazione dei nostri obiettivi. Di seguito spieghiamo i più importanti.

Bootstrap4
Bootstrap è un framework HTML, CSS e Javascript che dà la possibilità di creare pagine responsive. Utilizza quello che è chiamato "Grid system", ovvero lavora tramite una griglia formata da righe (row) e colonne (col). Ciò permette la creazione di pagine adatte a tutti i dispositivi desktop, tablet o smartphone. Bootstrap viene utilizzato quando si desidera creare un nuovo sito web senza dover scrivere tutto da zero e perciò consente di sviluppare un progetto in modo più rapido. Tuttavia, non deve essere considerata la soluzione definitiva ad ogni problema di web-design, ma deve essere valutato il suo utilizzo in base alle esigenze di sviluppo. Per la realizzazione del nostro sito web abbiamo deciso di utilizzare Bootstrap perchè, come detto prima, consente uno sviluppo più immediato.

Graph.js
Graph.js è una libreria open-source, disponibile su GitHub, che aiuta a visualizzare i dati usando JavaScript. In particolare permette di creare dei grafici per rappresentare al meglio i nostri dati. Graph.js supporta 8 tipi di grafici differenti come quelli

costituiti da barre, linee, punti, ecc...
In modo particolare, Graph.js, è stata utilizzata all'interno del nostro progetto per creare dei grafici adeguati ai diversi tipi di dati in input. Difatti, abbiamo deciso di utilizzare il grafico a linee in modo tale da poter rappresentare al meglio lo sfasamento tra dati e la distribuzione di questi.

jQuery
jQuery è una libreria JavaScript per applicazioni web distribuita come software libero. Nasce con l'obiettivo di semplificare la selezione, la manipolazione, la gestione degli eventi e l'animazione di elementi DOM in pagine HTML, nonché semplificare l'uso di funzionalità AJAX, la gestione degli eventi e la manipolazione dei CSS.
In particolare serve per astrarre le interazioni a basso livello con i contenuti delle pagine HTML. L'approccio di tipo modulare di jQuery consente la creazione semplificata di applicazioni web e contenuti dinamici versatili.
jQuery è stata utilizzata all'interno del nostro progetto per rendere più semplice la creazione di contenuti dinamici all'interno del nostro sito.

5.2 PWA

La Progressive Web App (PWA) nasce con l'obiettivo di portare un sito web ad avere un comportamento simile a quello di una applicazione mobile. Essendo gli smartphone sempre più diffusi, l'implementazione di una PWA per visualizzare i dati è una parte importante.
L'installazione è molto semplice: la prima volta che si entrerà nel sito, comparirà un pulsante per scaricare la PWA. Nel momento in cui si cliccherà sul pulsante, il browser chiederà se può installare la PWA sul nostro device. Dopo aver accettato, l'applicazione verrà installata e sarà presente tra le nostre applicazioni.
Così facendo, non sarà più necessario utilizzare Chrome per accedere al sito tramite il link "https://galvan.altervista.org", ma sarà sufficiente cliccare sulla PWA precedentemente installata.
I requisiti minimi che il sito dovrà avere per poter essere definito come una PWA sono:

- Protocollo SSL Attivo
- Sito responsive
- Presenza del Service Worker e funzionamento offline
- Presenza del file manifest

Possiamo trasformare il sito web in una PWA tramite creazione del file manifest e creazione del Service Worker.

WebManifest Il WebManifer è un file JSON che dice al browser che sto usando come installare e utilizzare la PWA. Il file conterrà tutte le informazioni di base della Progressive Web Application. Un WebManifest tipico include il nome dell'applicazione, la sua icona, e l' URL che verrà aperto quando l' applicazione verrà lanciata.

Service Worker Il Service Worker lavora in background e si pone tra il sito web e la rete. Esso ci consentirà di offrire i contenuti anche in condizioni di assenza di rete e rendere, quindi, fruibile il contenuto del sito anche quando è offline o il dispositivo non è connesso a Internet. Include funzionalità come le notifiche push, la gestione della cache, la navigazione offline e la sincronizzazione in background.

Di seguito mostriamo delle foto che illustrano l'installazione e la visualizzazione della PWA su un dispositivo mobile.

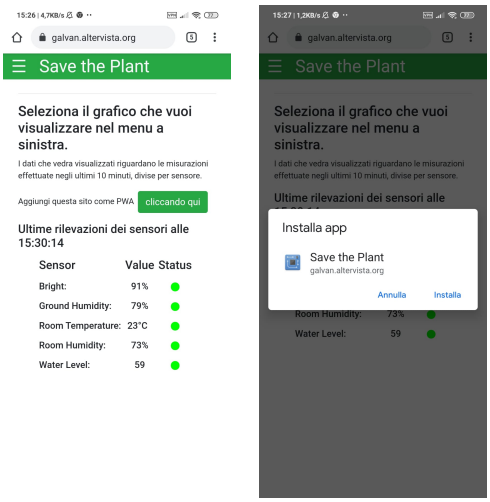

Figura 5.1: Sito visualizzato in Chrome (sinistra) e installazione PWA (destra)

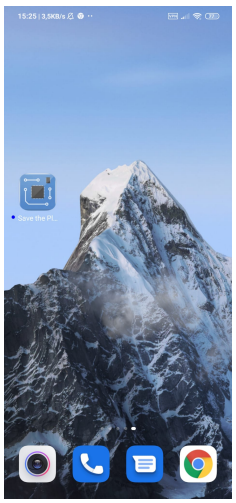

Figura 5.2: PWA installata

Mostriamo, inoltre, alcune foto della nostra PWA realizzata.

Figura 5.3: Home della PWA

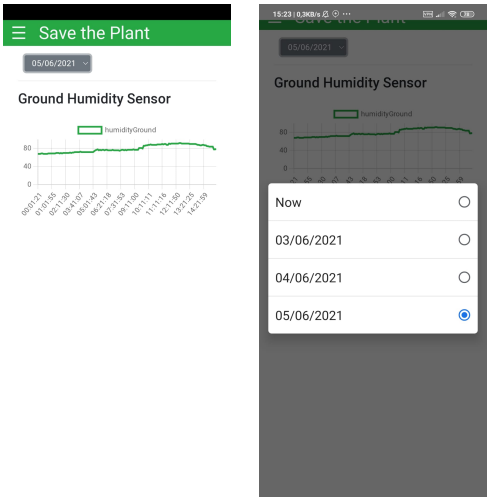

Figura 5.4: Grafico visualizzato all'interno della PWA (sinsitra) e selezione della data (destra)

6 Problematiche riscontrate

Durante lo sviluppo del nostro progetto abbiamo riscontrato delle problematiche alle quali abbiamo trovato delle soluzioni.

Innanzitutto, la nostra idea iniziale consisteva nell'utilizzo di Arduino Uno e della scheda Wi-Fi ESP8266-01s per la realizzazione del progetto. Tuttavia, abbiamo incontrato delle difficoltà per quanto riguarda il sensore di luminosità: programmando direttamente l'ESP8266 e avendo già collegato il sensore di umidità e temperatura non siamo riusciti ad inviare i dati raccolti dal foto-resistore al modulo Wi-Fi. Perciò, abbiamo optato per l'utilizzo di una board con il Wi-Fi già integrato e la nostra scelta è ricaduta, appunto, sull'ESP32.

Un'altra problematica che abbiamo incontrato è stata relativa al Water Level Sensor. Dopo una prima notte in cui abbiamo raccolto i dati relativi alla pianta, abbiamo ritrovato il sensore ossidato e i dati acquisiti, come si può vedere nell'applicazione, sono sballati. Questo sensore, infatti, è noto per la sua breve durata se esposto a un ambiente umido: l'alimentazione applicata alla sonda accelera costantemente il tasso di corrosione in modo significativo. E' perciò consigliato non alimentare costantemente il sensore, ma di alimentarlo solo quando si effettuano le letture.

7 Conclusione

Lo scopo del progetto, proposto dal corso Internet of Things, è quello di permettere allo studente di interfacciarsi direttamente con strumenti in grado di interagire con la realtà. Nel nostro caso abbiamo scelto di realizzare un "Salva piantina", che, attraverso strumenti informatici, permetta all'utente un controllo semplice e diretto di alcuni parametri vitali della pianta che altrimenti sarebbero difficili da mantenere sotto controllo. Il tutto è stato realizzato grazie a strumenti semplici e poco costosi, in modo tale da permettere anche all'utente medio di poterlo realizzare in casa. Difatti, anche molte parti del codice da noi presentate sono open-source, come librerie e framework.
In conclusione, speriamo che questo progetto, che si trova su GitHub (https://github.com/galvan29/PWAIOT.git) e di conseguenza accessibile a chiunque, venga realizzato e utilizzato da altri utenti.

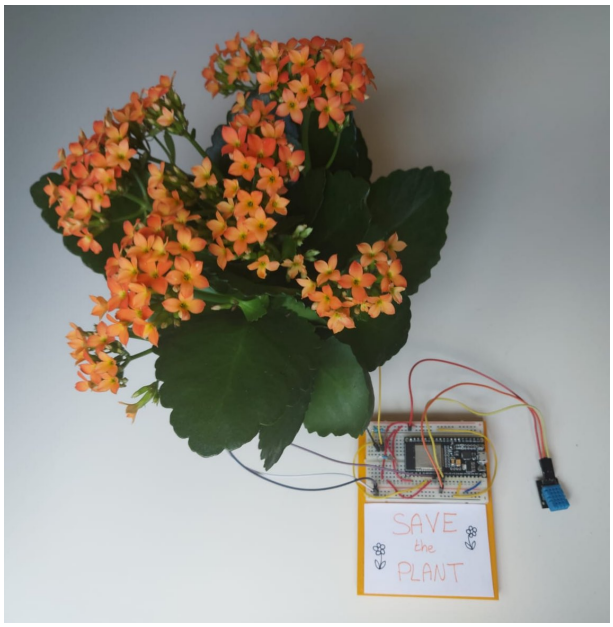

Figura 7.1: Save the Plant

8 Sitografia

[1] https://lastminuteengineers.com/water-level-sensor-arduino-tutorial

[2] http://www.elenet.net/index.php?qa=1341qa_1=cose-bootstrap

[3] https://www.w3schools.com