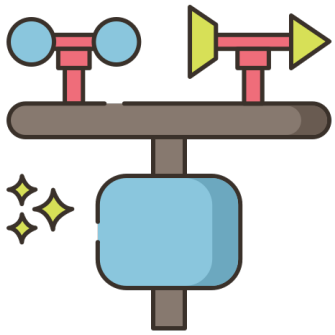


Station.Ino
Smart Weather Station

Petracci Riccardo matr. 114216

Corrado Pallucchini matr. 114192

Supervisors

Prof. Lorenzo Morresi

Dott. Andrea Piermarteri

Master of Science in Computer Science(LM-18)

Embedded System: Architecture(2020/2021)

Contents

Chapter 1

Introduction

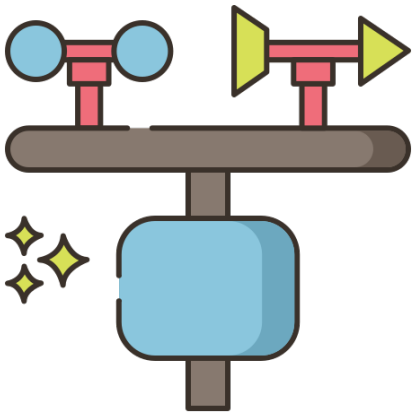
Station.Ino is a smart weather station and it is only a prototype.

The main function of Station.Ino is retrieve data from enviroment and push them over the Internet on Firebase Database. With a PWA called Station.Ino and a Telegram Bot, users can retrieve all sensors data.

To anticipate some details, Station.Ino project contains an Arduino that retrieve temperature, humidity and luminosity outside and with Bluetooth module send this data to other Arduino that is inside and have temperature and humidity sensors.

Internal Arduino sends its data and external Arduino's data to Raspberry Pi which has task of sending all data retrieved into Firebase Real Time Database.

Users can see real time data and some charts with the Station.Ino PWA and ask to Stationino bot some information.



Chapter 2

Project Structure

In this section we will show general structure of Station.Ino project describing both the electrical part (Arduino and its shields) and software one (PWA and the Telegram Bot).

Station.Ino, as we just said, is system aimed to catch enviromental data, in our case light, temperature and humidity, and then push on the user interface (PWA or Telegram Bot).

This documentation is divided into different section to understand better all topics:

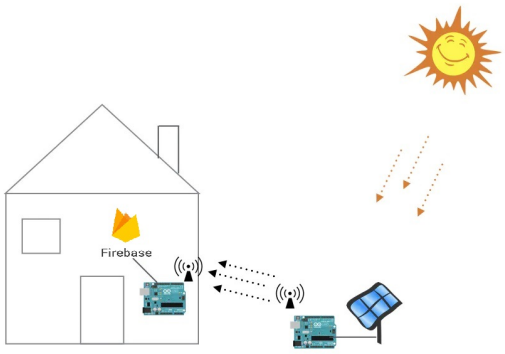
1. sensors section: where we talk about different sensors;
2. solar power section: where we talk about circuit that allow outside Arduino to stay on all time.
3. gateway section: where we talk about Raspberry PI used as a gateway to forward data on Firebase and host telegram bot;
4. application section: where we talk about Station.Ino PWA and Stationino Telegram Bot.

Chapter 3

Sensors

In Station.Ino project we have two different Arduino that retrieve same data but one Arduino is outside while other one is inside.

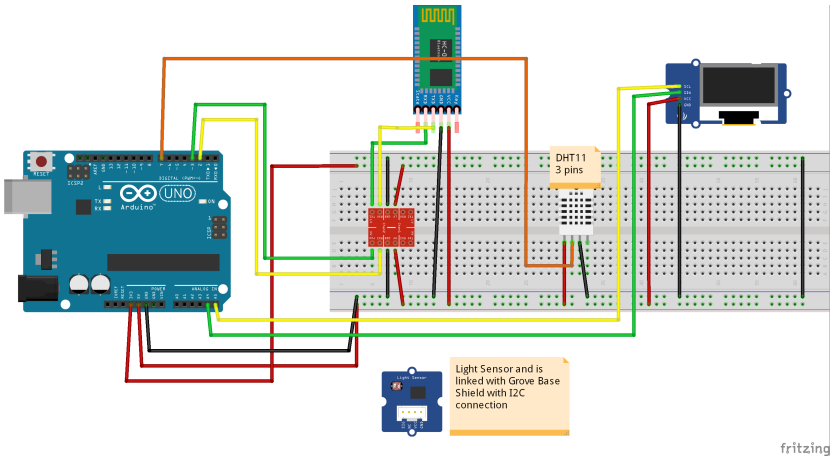
In this way we can have both data from inside our home and outside, where outside Arduino has also a light sensor.



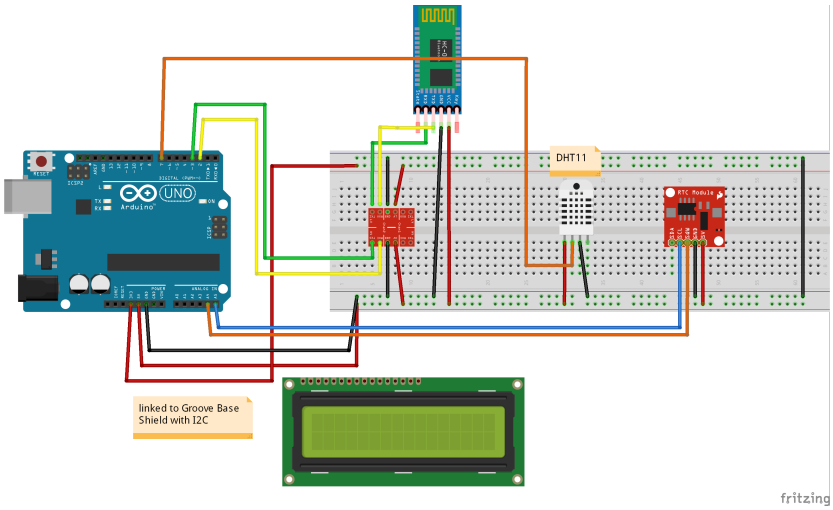
As we can see in the art image outside Arduino has a solar system with solar panel where it is not directly powered by solar but by rechargeable batteries.

The mechanism that will discuss in 4 in general is composed by batteries that power Arduino while solar panel charge them.

Talking about external Arduino' sensors we have a Grove Luminosity sensor, DHT11 3 pins for temperature and humidity, a Display OLED 0.96" 128x64 SSD1306, Grove Base Shield and Bluetooth Module HC-05 used with voltage converter 3,3V/5V.



Talking about internal Arduino' sensors we have DHT11 3 pins for temperature and humidity, Grove-16x2 LCD I2C, Grove Base Shield, Tiny RTC and Bluetooth Module HC-05 used with voltage converter 3,3V/5V.



3.1 Overall Arduino' Sensor

The components of external and internal Arduino has different task and we will talk about them in general.

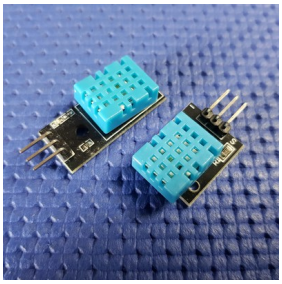
3.1.1 DHT11

The KY-015 temperature and humidity sensor module is part of the "37 In 1 Sensor Module Board Set Kit For Arduino" series, it looks like a small circuit,

the DHT11 sensor, two resistors, a led and a 3-pin connector.

This module is compatible with the most popular electronics platforms such as Arduino, Raspberry Pi and ESP8266.

The fields of application of the module can be: HVAC test and control equipment, auto motive, data logger, automatic control, Weather station, humidity controllers, dehumidifiers.



3.1.2 Bluetoot Module HC-05 and voltage converter 3,3/5 V

The HC-05 module is a module that allows you to convert a UART serial port into a Bluetooth port; this device can be used to allow communication between a microprocessor (MCU), mounted for example on an Arduino board, and a device equipped with Bluetooth communication, such as a PC, a Smartphone or a Tablet. This module is based on the CSR Bluecore 04 External chip with CMOS and AFH (Adaptive Frequency Hopping Feature) technology and is compatible with the Bluetooth V2.0 3Mbps + EDR (Enhanced Data Rate) protocol.

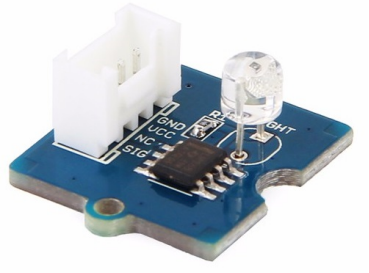
The main feature of the HC-05 module is the possibility of being set, through the AT commands, in two modes, according to the user's needs: as a Master device or Slave device; this will allow you to use the same type of module to create a network of devices.

Bi-directional logic level converter is a small device that safely steps down 5V signals to 3.3V AND steps up 3.3V to 5V at the same time. This level converter also works with 2.8V and 1.8V devices. What really separates this Logic level converter from our previous versions is that you can successfully set your high and low voltages and step up and down between them safely on the same channel. Each level converter has the capability of converting 4 pins on the high side to 4 pins on the low side with two inputs and two outputs provided for each side.

The level converter is very easy to use. The board needs to be powered from the two voltages sources (high voltage and low voltage) that your system is using. High voltage (5V for example) to the 'HV' pin, low voltage (3.3V for example) to 'LV', and ground from the system to the 'GND' pin.

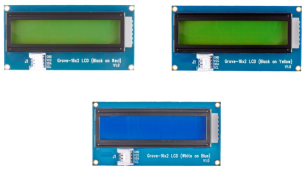
3.1.3 Grove Luminosity sensor

The Grove - Light sensor integrates a photo-resistor(light dependent resistor) to detect the intensity of light. The resistance of photo-resistor decreases when the intensity of light increases. A dual OpAmp chip LM358 on board produces voltage corresponding to intensity of light(i.e. based on resistance value). The output signal is analog value, the brighter the light is, the larger the value.



3.1.4 Grove-16x2 LCD I2C

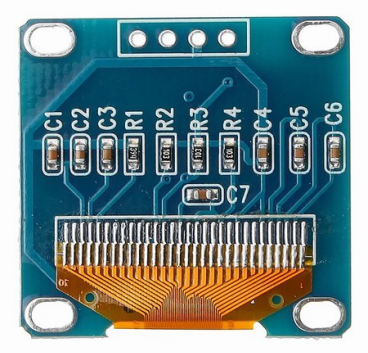
Grove - 16 x 2 LCD is a perfect I2C LCD display for Arduino and Raspberry Pi with high contrast and easy deployment. 16x2 means two lines and each line has 16 columns, 32 characters in total. With the help of Grove I2C connector, only 2 signal pins and 2 power pins are needed. You don't even need to care about how to connect these pins. Just plug it into the I2C interface on Seeeduino or Arduino/Raspberry Pi+baseshield via the Grove cable. There won't be complicated wiring, soldering, worrying about burning the LCD caused by the wrong current limiting resistor.



3.1.5 Display OLED 0.96" 128x64 -SSD1306

The OLED (Organic Light Emitting Diode) type indicates that not only is the display very bright and clear with a high contrast ratio, but it has a much lower power consumption than other display technologies such as the backlit LCD; it is also very visible in both total darkness and bright light conditions.

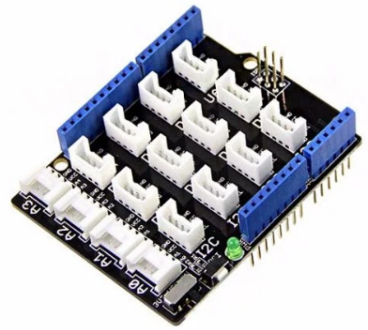
To interface the module, the display uses an I2C serial interface that requires only two data pins. The reset circuit is on board and means that an additional reset pin is not needed.



3.1.6 Grove Base Shield

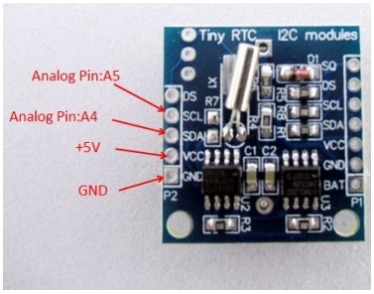
Base Shield provides a simple way to connect with Arduino boards and help you get rid of breadboard and jumper wires. With the 16 on-board Grove Connectors, you can easily connect with over 300 Grove modules! The pinout of Base Shield V2 is the same as Arduino Uno R3.

There are 16 on-board Grove connectors including 4 x Analog, 7 x Digital, 1 x UART, 4 x I2C. Apart from the rich Grove connectors, on the board you can also see an RST button, a green LED to indicating power status, ICSP pin, a toggle switch and four row of pinouts. There is no need to explain the RST button and LED.



3.1.7 Tiny RTC

This tiny RTC module is based on the clock chip DS1307 which supports the I2C protocol. It uses a Lithium cell battery (CR1225). The clock/calendar provides seconds, minutes, hours, day, date,month, and year information. The end of the month date is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12-hour format with AM/PM indicator.



3.1.8 How system work

Now that we have a general panoramic of sensor of the project we can talk about how they works together.

Talking about external Arduino it takes data from light, temperature and humidity then it shows on oled display to give to users briefly information if they are near it.

With Bluetooth module in Station.Ino there are two different role: Master and Slave that can be setupped with AT mode and setup code for Arduino. Master is internal Arduino that retrieve data from Slave Arduino (external Arduino).

Meanwhile with Bluetooth module every 2 minutes slave sends a JSON with enviromental information to master one that communicate these information with raspberry.

Internal Arduino is powered with direct current because it receive all data from external Arduino and it has task to send data to Rapsberry Pi with serial port.

Arduino Master create a JSON with its data and Slave one, so Master send JSON on serial port and Raspberry Pi collect informations with USB interface. Meanwhile inside Arduino, as external one, show their data on LCD (or OLED) to allow user to read data.

Chapter 4

Solar Power

Using solar power technique we can run our sensor related or any other stand-alone Arduino project for a long time without use direct current and place outside Arduino where we want. In fact without cable for direct current we can be free to choose our favorite position for our Arduino.

4.1 Drawback

The main drawback of battery operated device is that it will be depleted after a certain time. This drawback can be eliminated by using natural resources like solar, wind or hydro energy. The most obvious free source of energy to charge the batteries is solar energy that is also more easy to use.

Among the rechargeable battery, nickel metal hydride (NiMH) and Li-Ion battery are widely used for battery operated device, but in our project we opted for 3AA NiMH.

4.2 Facts of battery charging

The thumb rule for charging Ni-Mh batteries is 1/10th (commonly known as C/10). To charge the battery pack at 1/10th its rated current requires 16 hours of charge time. The solar panel receives optimal sunlight for only four hours per day, from 10 a.m. to 2 p.m. Thus, a totally ideal system would require four days to fully charge the battery pack.

4.2.1 How to choose solar panel

The main source for powering the sensor module is a solar panel. So it must be able to provide current for powering the Arduino as well as current to charge the battery pack during the day.

In general there are two factor to consider:

1. voltage: you should choose 1.5 times the battery pack voltage
2. current: you should consider the current taken by Arduino and current for charging batteries.

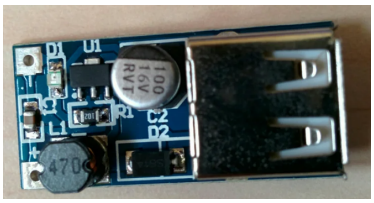
If we want do some calculation we need to consider:

1. A battery pack is made of 3 AA Ni-Mh batteries;
2. Battery voltage = 1.2 x 3= 3.6V
3. So required voltage for solar panel =3.6 x 1.5 = 5,4V
4. We can choose a 5V solar panel for it.
5. The sensor module along with Arduino taking 100mAh current.
6. Battery capacity is 2800mAh
7. C/10 = 280mAh
8. The solar panel has to provide current 100mAh for Arduino along with a current not more than 280mAh.
9. Lets take 200 mAh for charging the battery Total current required = 100+200=300mAh
10. From the above calculation, it is clear that we need no less than a solar panel with 5V and 300mAh.

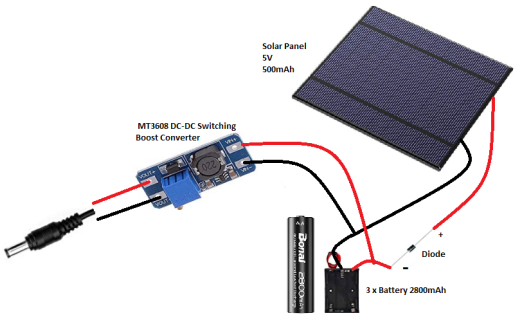
4.2.2 Ni-Mh Battery Charger

To power Arduino we decided to use 3AA to have more efficient systems because with 4AA we could have total voltage more voltage that we need (about more than 5V when battery totally charged).

Using 3AA we have to raise the voltage level to 5V by using a voltage booster circuit that standard sized nickel metal hydride (NiMH) cells with simple charge circuit.



At the end to understand the circuit made to recharge batteries we can see the picture below.



During the test with some calculation we estimate that the external Arduino with this technique can be stay on more or less 8/9 months because calculation is very hard to do with precision.

1. Iavg = (Ton*Ion) / (Ton) , where Ton is Time on and Ion is Intensity when Arduino is on;
2. Ton = 24h and Ion = 1μA
3. Iavg = 8640μA = 8.64 mA
4. Operating Voltage = 5 V
5. Pavg= V x Iavg = 5x 8.64= 43.2 mW
6. Ni-Mh battery capacity = 2800 mAh
7. Battery voltage = 3.6V
8. Power = 3.6 x 2800 = 10080 mWh
9. Battery life = 10080/43.2 = 233.33 hours = 9.7 months approximately

Chapter 5

Gateway

In this section we will talk about Raspberry Pi Model 3B, how it take data from Arduino and how it push data on Firebase Real Time Database.

Raspberry Pi is a single board computer developed in the UK by the Raspberry Pi Foundation. The presentation to the public took place on February 29, 2012.

The card was designed to accommodate Linux kernel or RISC OS operating systems. It is assembled in Wales, in the Sony UK Technology Center. A specially dedicated operating system was conceived, called Raspberry Pi OS.

The design is based on a Broadcom-manufactured system-on-a-chip (BCM2835, or BCM2836 for the Raspberry Pi 2, or BCM2837 for the Raspberry Pi 3), which incorporates an ARM processor, a VideoCore IV GPU, and 256 or 512 Megabyte or 1 Gigabyte of memory. The project includes neither hard disk nor solid state drive, relying instead on an SD card for boot and non-volatile memory.

For each revision several versions were produced different model: A, A+, B, B+ and other. In our case the model is Raspberry Pi Model B+ that has Bluetooth integrated but in our project we don't use it.

To use Rapsberry we installed inside an operating system called Raspberry Pi OS that until May 2020 was called Raspbian; it is an operating system based on the official Debian distributions for ARM Architecture, adapted for use on the Raspberry Pi.

All Raspberry Pi code used to do all gateway task are written in Python.

5.1 Catch Data From Arduino

Serial communication is simply a way to transfer data. The data will be sent sequentially, one bit at a time (1 byte = 8 bits), contrary to parallel communication, where many bits are sent at the same time.

More specifically, when you use Serial with Arduino and Raspberry Pi, you are using the UART protocol. UART means "Universal Asynchronous Reception and Transmission" and there are two ways to connect Raspberry Pi and Arduino for Serial communication:

1. Serial via USB;
2. Serial via GPIOs.

In our case, to use Raspberry as gateway we opted to use first choice "Serial via USB".

5.2 Detect Arduino board

When Arduino board is connected with USB cable, you should see it appear as /dev/ttyACM0, or /dev/ttyUSB0 (sometimes the number can be different, for example /dev/ttyACM1).

Simply run ls /dev/tty* and you should see it. At this point if you're not sure which device is the Arduino board, simply disconnect the board (remove the USB cable), and run ls /dev/tty* again. This way you will easily spot the serial device name of your Arduino.

At the beginning we tried to understand how to catch data using Python code and then we focused on take the data that we needed to push on Firebase that we will see below.

```
#!/usr/bin/env python3
import serial
if __name__ == '__main__':
    ser = serial.Serial('/dev/ttyACM0', 9600, timeout=1)
    ser.flush()
    while True:
        if ser.in_waiting > 0:
            line = ser.readline().decode('utf-8').rstrip()
            print(line)
```

5.3 Firebase

Firebase is a platform for building applications for mobile and web devices developed by Google. It was originally an independent company founded in 2011. In 2014 Google acquired the platform and it is now its flagship offering for app development. We used a particular function of a long list of them called Real Time Database that it is a cloud-hosted database. Data is stored as JSON and synchronized in realtime to every connected client. When developers build cross-platform apps with their iOS, Android, and JavaScript SDKs, all of clients share one Realtime Database instance and automatically receive updates with the newest data.

5.3.1 How it works

The Firebase Realtime Database lets you build rich, collaborative applications by allowing secure access to the database directly from client-side code. Data is persisted locally, and even while offline, realtime events continue to fire, giving the end user a responsive experience. When the device regains connection, the Realtime Database synchronizes the local data changes with the remote updates that occurred while the client was offline, merging any conflicts automatically.

The Realtime Database provides a flexible, expression-based rules language, called Firebase Realtime Database Security Rules, to define how your data should be structured and when data can be read from or written to. When integrated with Firebase Authentication, developers can define who has access to what data, and how they can access it.

```
{
  "rules": {
    "users": {
      "$user": {
        ".read": "auth.uid === $user",
        ".write": "auth.uid === $user"
      }
    }
  }
}
```

The Realtime Database is a NoSQL database and as such has different optimizations and functionality compared to a relational database. The Realtime Database API is designed to only allow operations that can be executed quickly. This enables you to build a great realtime experience that can serve millions of users without compromising on responsiveness. Because of this, it is important to think about how users need to access your data and then structure it accordingly.

5.3.2 Real Time Database on Raspberry Pi - Pyrebase

To use Real Time Database on Raspberry Pi we used Pyrebase that it is a Python wrapper for the Firebase API.

On our Raspberry Pi we used a Service Account that has administrative role so it can read and write bypassing all database roles.

To ensure Service Account we used a file .JSON with a unique code made by "Firebase Admin SDK" that it is a feature used to create service account usually used to configure "things" that are not person. In fact Raspberry does not have username or password because it is a "things" that have admin role to set and get data.

In this Firebase JSON file with a long name -we renamed config.JSON for convenience - there are all private settings that allow pyrebase code to send data with admin role and not have security problems.

```
#!/usr/bin/env python3
import serial
import simplejson as json
import time
import pyrebase
from datetime import date
from config import uid, uid2, db
from deviceInfo import info, IP_addres, h_name

ser = serial.Serial('/dev/ttyACM0',4800)

buffer = ""

while True:
    try:
        db.child("users/"+uid).child("device").child("information").set(info)
        buffer = ser.readline()
        print(buffer)
        data = json.loads(buffer)

        if ('extTemp' in data) == True:
            db.child("users/"+uid).child("raspberryRTSensorsExt").child("realTime").set(data
            db.child("users/"+uid).child("raspberrySensorsExt").child("storeData").push(data

        elif ('temperature' in data) == True:
            db.child("users/"+uid).child("raspberryRTSensors").child("realTime").set(data)
            db.child("users/"+uid).child("raspberrySensors").child("storeData").push(data)

        buffer = ""
        print(" ")
    except json.JSONDecodeError:
        print("Error : try to parse an incomplete message")
        continue
```

5.3.3 Real Time Firebase Structure

Real Time Database structure (as 5.3.3 figure shown) has a tree structure with Real Time Data and Stored Data (used to create charts that we will see at 6) and it is divide in users id because in an industrial case all users has own Arduino and raspberry and all of them could see their data on the app with their credential but in this prototyping case we have only 2 users that are me and corrado and there is not a real pairing (need if this prodact will industrialized).

In a industrial product gateway has a unique service account and they can create pairing connection with application to validated user identity, called uid. In our case, we are in a prototyping situation where pairing is not considered and there are only two users - Riccardo and Corrado authors of Station.Ino - and the UID is passed manually inside a configuration file.

Chapter 6

Application

In this section we will talk about applications of Station.Ino:

1. the web application made with Angualar Framework
2. Telegram Bot made with a famous application Telegram that allowed us to create a nice bot.

6.1 Angualar

The Station.Ino Frontend component is a Angular web application which retrieve the enviromental data requests to the Firebase Real Time Databse.

Angular is an open source framework for the development of web applications licensed by MIT, an evolution of AngularJS. Developed primarily by Google, its first release took place on September 14, 2016.

Angular has been completely rewritten compared to AngularJS and the two versions are not compatible. The programming language used for AngularJS is JavaScript while that of Angular is TypeScript.

Applications developed in Angular are executed entirely by the web browser after being downloaded from the web server (client side processing). This results in the saving of having to send the web page back to the web-server every time there is a request for action from the user. The code generated by Angular runs on all major modern web browsers such as Chrome, Microsoft Edge, Opera, Firefox, Safari and others.

Angular has been designed to provide a quick and easy tool for developing applications that run on any platform including smartphones and tablets. In fact, the web applications in Angular in combination with the Bootstrap open source toolkit become responsive, ie the design of the website adapts according to the size of the device used.

Another responsive design toolkit, Flex Layout, is under development, which is easier to use than Bootstrap and designed specifically for Angular.

Another toolkit that facilitates the design in Angular is Angular Material, a series of components that allows you to create a web page very quickly: with the combined use of Flex Layout and Angular Material you can create very advanced responsive web sites and applications based on Angular.

6.1.1 Frontend application itself

We first generated a dashboard home page application with a sidebar with some sections that represent the main interface components. Interfaces are three tabs are:

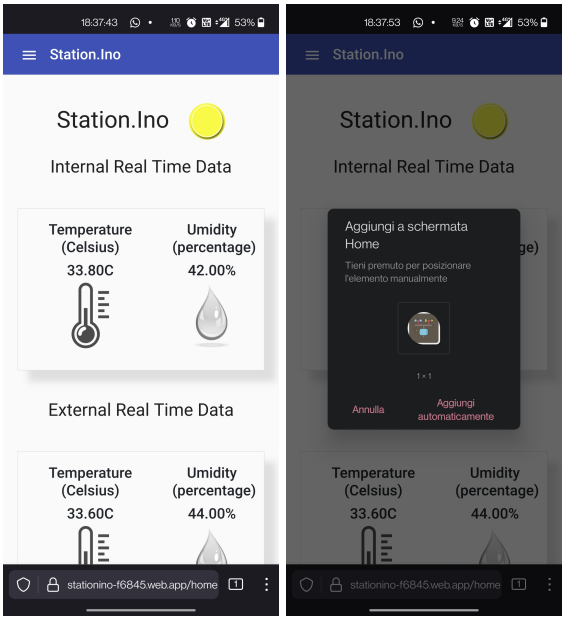
1. Home page
2. Analytics
3. About us
4. Profile
5. Gateway Info
6. Login

6.1.2 PWA

It is very important talk about PWA that means Progressive Web App that is different to real Android or IOs application and it is a term, originally coined by Google, which refers to web applications that are developed and loaded like normal web pages, but which behave similarly to native applications when used on a mobile device. Unlike traditional applications, progressive web apps are a

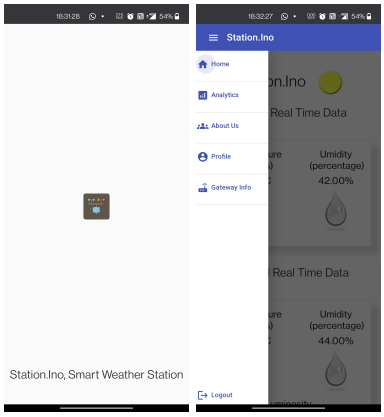
hybrid between normal web pages (or websites) and mobile applications. This application model seeks to combine the possibilities offered by most modern browsers with the benefits of mobile use.

The term Progressives refers to the fact that, from the point of view of the user experience, these applications can enable a series of additional features to normal web pages depending on the functionality offered by the device. For example, the browser can offer the user to save them on the home screen of the mobile terminal, to be perceived in all respects as native apps. Several companies have seen notable improvements in various key performance indicators, such as an increase in time spent on site or new user conversions.



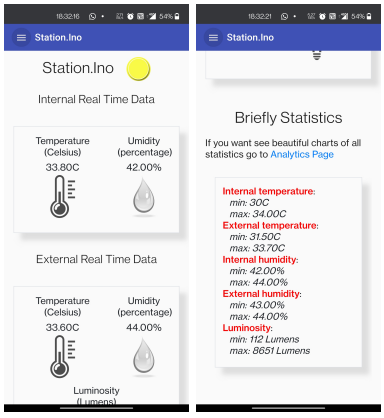
6.1.3 Components Overview

The Station.Ino application is presented by icon while application is loading and a sidebar to show various interfaces that are described below.



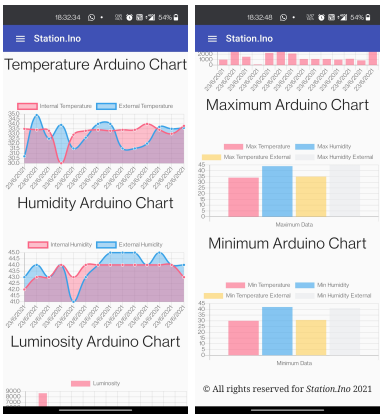
Home page

In the home page user can see real time data that are updated in the "users/uid/raspberryRTSensors/realTime" path of the firebase tree JSON. The home page is accessible only by authenticated users that can get their data and this is possible with security roles that we saw before 5.3.1.



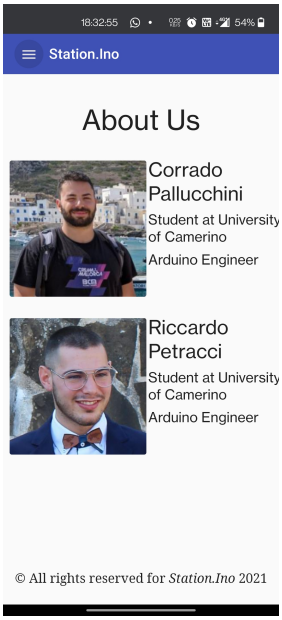
Analytics

In the analytics page user can see some chart about sensors values trend and some chart with maximum (or minimum) values of sensors.



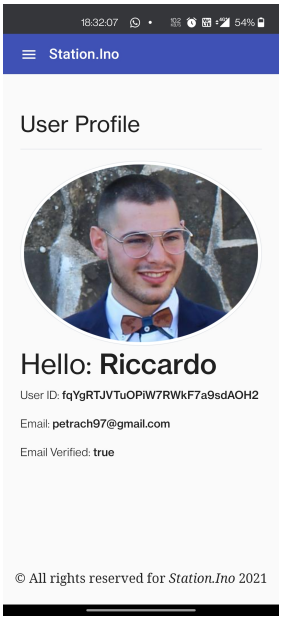
About Us

In the about us page we have general description of authors of the Station.Ino project Riccardo and Corrado with their photos.



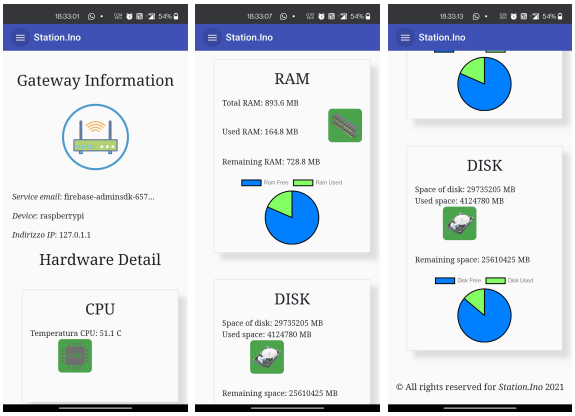
Profile

In the profile interface we have basic user information and it's photo.



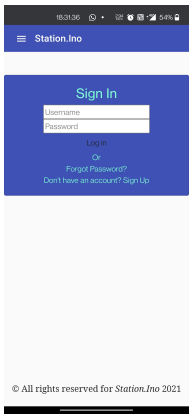
Gateway

In the gateway page we have all information of our gateway that is Raspberry Pi (in our case) and all its data from IP address to CPU temperature, RAM usage and disk usage.



Login

The login interface allow users to create a new account or authenticate to their account already exists. The login page is the first page that user will see at the first launch of the app to allow him to connect on sensors data of him account. The login page - also the About Us page - is the unique feature accessible to user that are not registered because Station.Ino app need to understand in which UID it need to retrieve data and it is possible only with login.



6.2 Telegram Bot

Telegram is a free and open source, cross-platform, cloud-based instant messaging (IM) software. This service also provides end-to-end encrypted video calling, VoIP, file sharing and several other features. It was launched for iOS on 14 August 2013 and Android in October 2013. The servers of Telegram are distributed worldwide to decrease data load with five data centers in different regions, while the operational center is based in Dubai in the United Arab Emirates. Various client apps are available for desktop and mobile platforms including official apps for Android, iOS, Windows, macOS and Linux (although registration requires an iOS or Android device and a working phone number). There are also two official Telegram web twin apps – WebK and WebZ – and numerous unofficial clients that make use of Telegram's protocol. All of Telegram's official components are open source, with the exception of the server which is closed-sourced and proprietary.

Telegram provides end-to-end encrypted voice and video calls and optional end-to-end encrypted "secret" chats. Cloud chats and groups are encrypted between the app and the server, so that ISPs and other third-parties on the network can't access data, but the Telegram server can. Users can send text and voice messages, make voice and video calls, and share an unlimited number of images, documents (2 GB per file), user locations, animated stickers, contacts,

and audio files. In January 2021, Telegram surpassed 500 million monthly active users. It was the most downloaded app worldwide in January 2021.

6.2.1 Bot Father

Bots are third-party applications that run inside Telegram. Users can interact with bots by sending them messages, commands and inline requests. User can control bots using HTTPS requests with Bot API.

At the core, Telegram Bots are special accounts that do not require an additional phone number to set up. Users can interact with bots in two ways:

1. Send messages and commands to bots by opening a chat with them or by adding them to groups.

2. Send requests directly from the input field by typing the bot's @username and a query. This allows sending content from inline bots directly into any chat, group or channel.

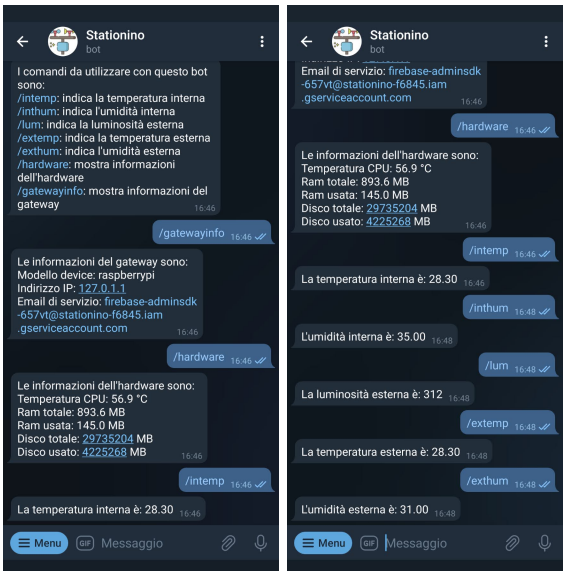
Messages, commands and requests sent by users are passed to the software running on servers. Intermediary server handles all encryption and communication with the Telegram API. Users communicate with this server via a simple HTTPS-interface that offers a simplified version of the Telegram API.

To create a bot there is a bot for that just talk to BotFather and follow a few simple steps. Once you've created a bot and received your authorization token, head down to the Bot API manual to see what you can teach your bot to do.

6.2.2 Stationino Bot

The bot has several commands with which you can interact:

- start: bot start command;
- help: lists the commands you can use and what they do;
- intemp: returns the temperature of the indoor station;
- inthum: returns the humidity of the indoor station;
- lum: returns the luminosity of the external station;
- extemp: returns the temperature of the external station;
- exthum: returns the humidity of the external station;
- hardware: shows Raspberry informations such as CPU temperature, total Ram and Ram used, Total disk and disk used
- gatewayinfo: shows gateway informations such as device model, IP address and service email.



Whenever a command is sent to the bot, it makes a get request directly to the Firebase database like this:

```
def intemp(update, context):
    intemp = db.child("users/fqYgRTJVTuOPiW7RWkF7a9sdAOH2/
    raspberryRTSensors/realTime/temperature").get()
    intemp1 = intemp.val()
    update.message.reply_text('La temperatura interna è: ' + str(intemp1))
```

Then, the commands are handled through the

```
dp.add_handler (CommandHandler(""))
```

function in the main method.

```
def main():
    updater = Updater("TOKEN", use_context=True)
    dp = updater.dispatcher
    dp.add_handler(CommandHandler("start", start))
    dp.add_handler(CommandHandler("help", help))
    dp.add_handler(CommandHandler("extemp", extemp))
    # other commands
    dp.add_error_handler(error)
    updater.start_polling()
    updater.idle()

if __name__ == '__main__':
    main()
```

6.3 Conclusions

The system we created is an experimental of Smart Weather Station that can be the based of a big project for the future.

Of course, everything we built up until now is a prototype and can be largely improved, but should work as a proof of concept for what can be done and how to achieve it.

For example, we have only two Arduino but in a large used we could have lot of Arduino. Other example can be the pairing that in this prototype is a manual pairing but in a industrialized product pairing should be done by user and harware.

We are very happy to followed this course because we learned lot of stuff about hardware scope and about software scope specially in a low level coding that is very important to realize projects as Station.Ino.

If you are interested to see entire project you can visit our GitHub page at this link https://github.com/riccardo-petracci/Station.Ino. Inside the GitHub Repository you can found schematic circuits, gateway code, sketch of arduino and front-end folder.

6.4 Bibliography

Arduino: https://www.arduino.cc
RaspberryPi: https://www.raspberrypi.org/
Wikipedia: https://it.wikipedia.org/wiki/
Firebase: https://firebase.google.com/
Angular: https://angular.io/
Angular Material: https://material.angular.io/
Python: https://www.python.org/
Pyrebase: https://pypi.org/project/Pyrebase/
Pyrebase GitHub: https://github.com/thisbejim/Pyrebase
Telegram: https://core.telegram.org/bots
Bootstrap: https://getbootstrap.com/