

Lab of Iot project documentation

Marialuisa Trerè, 0522500652 and Simona Santoro, 0522500651

University of Salerno, Salerno, Italy

2 dicembre 2019

1 Material used for the project

- **Arduino UNO WI-FI REV 2:** Arduino UNO WI-FI REV 2 is a micro-controller board based on ATmega328P(datasheet). It has 14 digital input/output pins — 5 can be used as PWM outputs with Pin 11 that is not PWM as on the UNO Rev3 — 6 analog inputs, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the micro-controller.
- **Arduino MKR WI-FI 1010:** The MKR WIFI 1010 is a significant improvement on the MKR 1000 WIFI. It's equipped with an ESP32 module made by U-BLOX. This board aims to speed up and simplify the prototyping of WiFi based IoT applications thanks to the flexibility of the ESP32 module and its low power consumption. The MKR WIFI 1010 also has WiFi and Bluetooth module based on ESP32. For this board you need a micro usb cable.
- **MQ-2 sensor:** This sensor is sensitive to smoke and other flammable gas, such as PG, Butane, Propane, Methane, Alcohol and Hydrogen.
- **Flame sensor:** These types of sensors are used for short range fire detection and can be used to monitor projects or as a safety precaution to cut devices off / on.
- **Passive buzzer:** A buzzer is an electronic audio signaling component: a small speaker capable of emitting tones at certain frequencies. There are active and passive buzzers, the main difference between them is that an "active" buzzer can generate tones independently, while the "passive" buzzer requires a micro-controller that generates a waveform, such as to cause the internal membrane of the component to vibrate, thus emitting the desired frequency.
- **Servomotors SG90:** A servomotor is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity and acceleration.
- **Light sensors BH1750FVI I2C:** BH1750FVI is an digital Ambient Light Sensor IC for I2C bus interface. This IC is the most suitable to obtain the ambient light data for adjusting LCD and Keypad backlight power of Mobile phone. It is possible to detect wide range at High resolution.
- **LEDs:** A LED is an optoelectronic device that exploits the ability of some semiconductor materials to produce photons through a spontaneous emission phenomenon when crossed by an electric current.
- **Alexa Echo:** Alexa is a cloud-based artificial intelligence that comes in the form of a voice assistant, a software that can be integrated into speakers and other smart devices. By interpreting natural language, he manages to interact with man.

2 Project functionality

The project is composed by two different type of Arduino: Arduino Uno WiFi Rev 2 and Arduino MKR WiFi 1010.

The main features of our project are based on *IFTTT* service, *CloudMQTT* and *Arduino Create*. **Their operation will be illustrated in the next section.**

Both the Arduino are connected to WiFi network without using an external module, because they already have this built-in functionality. They connect to the network using information in the file *arduino_secrets.h*, it contains the network's name and the password, and also information for the connection to *CloudMQTT*.

With **Arduino Uno WiFi Rev 2**, we are connected the Light sensors BH1750FVI I2C, a servomotor, and different LEDs, in order to form an energy saving system for your house.

The idea is to manage the system through an android application that is able to send and receive MQTT messages. The user can choose between three different modality: *free*, *manual* and *sleep*.

- **Sleep:** the first action that the system does in this mode is to turn off the light and close the curtains.
- **Manual:** in this modality the user is free to choose the best artificial and natural light that wants. This means that the user can open the curtains and can adjust the light intensity by using the android application.
- **Free:** this mode, is the very important one, for one of the goal of our project. The user hasn't the ability to adjust the curtains or the light intensity. This mode uses the data from the light sensor to automatically adjust the external and internal light, in this way will balance the artificial light with the natural one. In addition if the external light is sufficient to guarantee an optimal light for the user, the light goes out. If the light sensor value is greater than a threshold the light is turned off. If instead the value is less than a certain threshold then the light is turned on, otherwise the light intensity is reduced.

With **Arduino MKR WiFi 1010**, instead, we are connected the gas sensor, the flame sensor, a servomotor and a buzzer, in order to form an antigas and fire prevention system. Moreover, thanks to this type of Arduino, we are able to connect Alexa Echo with our system through *Arduino create*.

This Arduino takes care of the following functionalities:

- If the value of flame sensor is more than a chosen threshold - we have chosen 500 as threshold - the buzzer is activated by simulating an alarm, and is sent an alert email on your smartphone with IFTTT service. The buzzer is deactivated when the flame sensor value goes below the threshold.
- If the value of the gas sensor is more than a chosen threshold - we have chosen 500 as threshold - is activated a servomotor, which rotates 180 degrees to simulate the opening of a slot, useful for circulating air and to eliminate the gas. Moreover, as for the flame sensor, is sent an alert email on the smartphone with IFTTT service. When the gas sensor value decreases and goes below the threshold, the servomotor is activated again to simulate the closing of the previously slot.
- Connect Alexa Echo with our system, in this way, with a simple voice command, you can switch between the 3 different modes(free, manual, sleep). To do this, when Alexa receive the voice command, sends an MQTT message on the same topic where the other arduino is listening.

3 Services used

The system is implemented by using several services, such as: **If This Then That** (IFTTT), **CloudMqtt**, **Arduino Create**, and an android app called **IoT MQTT Panel**.

3.1 IFTTT

IFTTT is a free web service that allows the creation of simple chains of conditions, called *applets*. An applet is activated by other services such as Gmail, Facebook, Instagram or Pinterest and can for example send an e-mail message when the user uses a hashtag in a tweet or can send a copy of a Facebook photo to an archive when the user is tagged in it.

To access to IFTTT service you have to create an account. Then, you create two new applets, what we called *send_email_fire_sensor* and *send_email_gas_sensor*, both from the Webhooks service to e-mail service. In this way, every time a web request, in our case a POST request, is sent to IFTTT via arduino, IFTTT service sends an alert email to your email . This thanks to the key that we can find in the documentation of the webhooks service page, without this key will not be possible to trigger the event.

3.2 CloudMQTT

It gives use the possibility to avoid to manage the broker on scaling by giving us a Mosquitto broker fully managed. To access to CloudMQTT service you have to create an account.

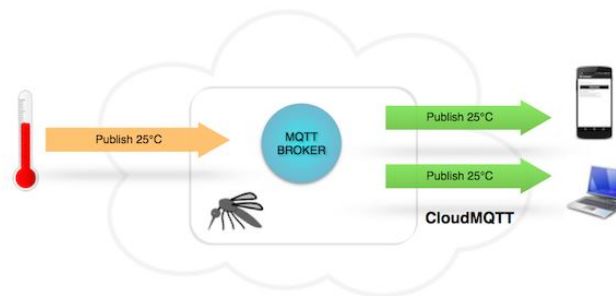


Figura 1: CloudMqtt

First of all you have to create a **new instance** and choose a name for it, we chosen the name *Project_LabOfIoT*, after choose the datacenter and also the region where your instance should be.

Plan Region Configure (Dedicated plans only) Confirm

Select a plan and name - Step 1 of 4


Name:

Plan:

Tags:

Tags are used to separate your instances between projects. This is primarily used in the project listing view for easier navigation and access control.

Tags allow admins to manage team members access to different groups of instances.

Plan: 
Cute Cat

See the [plan page](#) to learn about the different plans.

Cancel Select Region

Figura 2: Create an instance

Plan Region Configure (Dedicated plans only) Confirm

Select a region and data center - Step 2 of 4

Data center US-East-1 (Northern Virginia)

aws

« Back Cancel Review

Figura 3: Choose datacenter and Region

The instance is immediately provisioned and in the **details** window there are all the information about the instance such as: server, username, password, port and so on.

Instance info

Server

User Restart

Password

Port

SSL Port

Websockets Port (TLS only)

Connection limit 5

Active Plan

Cute Cat

Upgrade Instance

Figura 4: Details window

The **WebSocket UI** window provides a live view of the data that has been sent to a topic.

WebSocket

Messages are displayed in real-time as they are received by the broker. It's not possible to view historical data.

Send message

Topic

Message

Send

Received messages

Topic	Message
temp	17
light	50%
window	open
mode	free

Figura 5: WebSocket UI window

After the environment configuration, we have to connect the Arduino to the server. In order to do that, we used **PubSubClient** library that give us the ability to create an Arduino Client for MQTT, we used this library for both the Arduino. The library provides several method:

1. **setServer(server,port)**: this method gives us the possibility to set up the server we have configured or want to use. All the information needed are in the detail window *Figure 4*.

2. **connected()**: Checks whether the client is connected to the server. This return **true** if the client is still connected, otherwise return **false**.
3. **connect (clientID, username, password)**: Connects the client with a username and password specified (*Figure 4*). Return **true** if connection succeeded, otherwise **false**.
4. **publish(topic,message)**: allow us to send a message to a specific topic, for example the information about the light or some critical information. In this way all the client subscribed to the topic will receive the information.
5. **subscribe(topic,QoS)**: The client can subscribe to a topic by using this function, also can specify the *quality of service*. The Quality of Service (QoS) level is an agreement between the sender of a message and the receiver of a message that defines the guarantee of delivery for a specific message.
 - **QoS=0**: At most once
 - **QoS=1**: At least once
 - **QoS=2**: Exactly once

If the Client is registered to a specific topic and another one publish some data on this topic, the client receives this information and the callback is called.

6. **setCallback(callback_function)**: The callback is the function called that trigger something when the Client receive some data. So, this method allow us to set a function (*callback_function*) created by the programmer, in this function we have to set the behaviour of the system when receive some information.
7. **disconnect()**: disconnects the Client.

3.3 IoT MQTT Panel

IoT MQTT Panel is an android application that allow us to manage and visualize the IoT project based on MQTT protocol. MQTT is a machine-to-machine (M2M)/“Internet of Things” connectivity protocol. It was designed as an extremely lightweight publish/subscribe messaging transport. The application provide some panels such as: Button, switch, slider and so on.

The screenshot shows the 'Create connection' screen of the IoT MQTT Panel application. The form is organized as follows:

- Connection name ***: Text input with 'Test' and a blue question mark icon.
- Client ID**: Text input with 'test' and a blue question mark icon.
- Broker Web/IP address ***: Text input with a blue question mark icon.
- Port number ***: Text input.
- Network protocol**: Dropdown menu with 'TCP' selected and a blue question mark icon.
- Device list**: Section with a '+' button and a list item 'Test' with a home icon and a blue question mark icon.
- Advanced options**: Collapsible section (expanded) showing:
 - Connection timeout**: Text input with '30' and a blue question mark icon.
 - Keep alive**: Text input with '60' and a blue question mark icon.
 - Username**: Text input.
 - Password**: Password input with masked characters and a blue question mark icon.

Figura 6: Create connection

The first action to do to use the application is to create a connection, in this phase is necessary to insert the information about the server, the username and the password; we can find this information in the details window on cloudMQTT (*Figure 4*). The connection name is arbitrary in the example on *Figure 5* we choose Test, also for the ClientID obviously must be unique. You have to create at least one device, if you don't do that the app gives you an error and you can't go on. You can add a device by push on the "plus" button near the voice device list (*Figure 6*). After, you have to choose and insert the device name like in *Figure 7*.

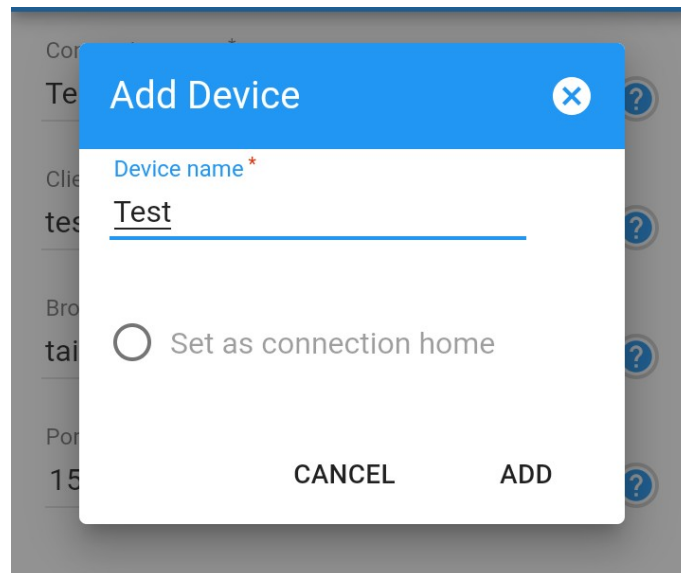


Figura 7: Create a device

So you have created the connection to the server. Now to establish a connection you have to push on the "cloud" symbol. If the symbol is crossed out means that you are not connect to the server (*Figure 8*)

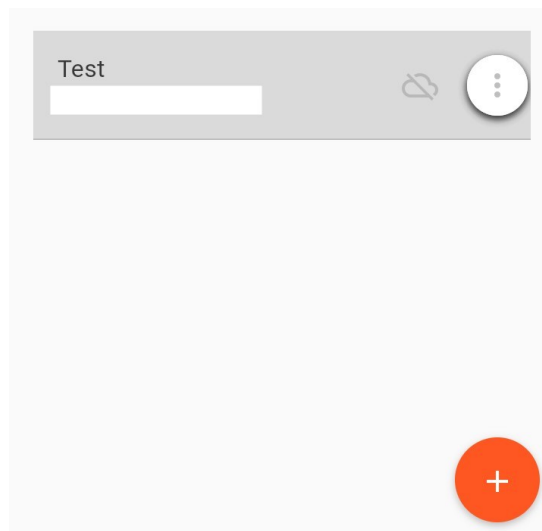


Figura 8: Client isn't connect

Otherwise if it appear just like in *Figure 9* means that the connection is successfully done.

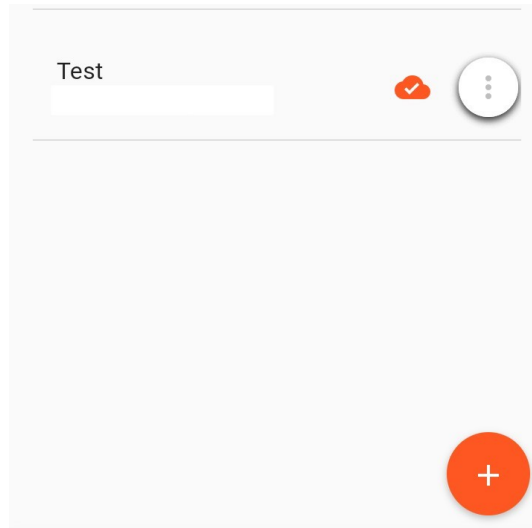


Figura 9: Client connect

Now, the configuration is done and the client is also connected to the server; is possible to create the panel. By pressing on the created client we enter the area where is possible to create the panel by adding all the necessary features.

The *Figure 10* is our example of a panel.

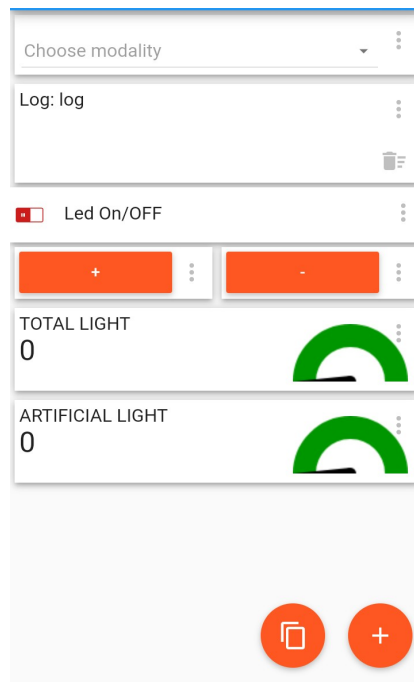


Figura 10: IoT Mqtt Panel example

When a new feature is added is necessary to specify the panel name, the topic where the data will be published, and the payload that is the message that will be sent to the topic. If the feature, for example a slider or Gauge, isn't able to publish something means that the feature have to receive only the data so is necessary to specify the topic from which the Client will receive the data.

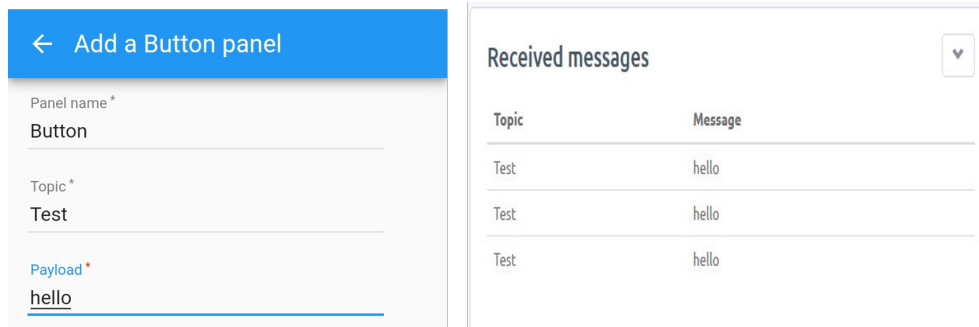


Figura 11: Example Button Panel and received message on CloudMQTT

In the example, *Figure 11*, is created a button panel where the panel name is Button, the topic is Test and the payload is “Hello”, in this way every time the button is pushed the client send the payload to the Test topic as we can see always in *Figure 11*.

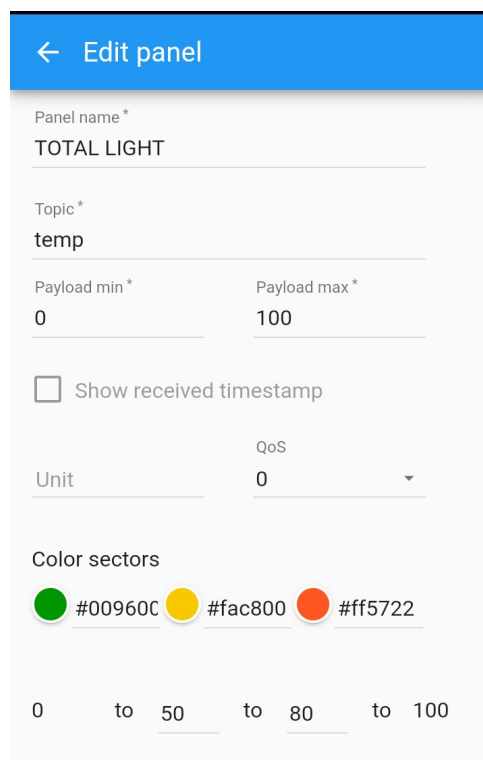


Figura 12: Gauge Panel

In *Figure 12* there is an example of Gauge panel, this is different from the button because is only able to receive the data from a topic. In the example, the Panel name is *TOTAL LIGHT*, the topic is *temp*. Is necessary also to set a range of value, in the example the range goes from 0 to 100. There is also the possibility to select different color for different range of values.



Figura 13: Gauge Panel example

3.4 Arduino Create

Arduino Create is an integrated online platform that enables Makers and Professional Developers to write code, access content, configure boards, and share projects. We used it to connect Alexa to our project, considering that Arduino Create allows to create devices that Alexa can connect and control. To access to Arduino Create you have to create an account.

With Arduino Create you can use an online IDE, connect multiple devices with the Arduino IoT Cloud, browse a collection of projects on Arduino Project Hub, and connect remotely to your boards with Arduino Device Manager.

The Arduino Web Editor allows you to write code, save it to the cloud and upload sketches to any Arduino board and Intel®-based platforms from your web browser (is recommended Chrome, but you can also use Firefox, Safari and Edge), after installing a simple plug-in, called **ArduinoCreateAgent**. It is hosted online, therefore it will always be up-to-date with the latest features. Your Sketchbook will be stored in the cloud and is accessible from any device.

Through the voice **Device Manager**, we added a new board that we called *MKR_WiFi_1010_2* (we chosen Arduino MKR WiFi 1010 because the Arduino Uno WiFi Rev 2 is not compatible yet with Arduino Create). Arduino Device Manager allows you to connect remotely to cloud-enabled boards and manage the boards and devices you have added to Arduino Create. You can check your board settings, modify the network configuration, start and stop sketches, as well manage packages on the device's operating system. Using the Web Editor you can write and upload sketches to you cloud-enabled board remotely, anywhere in the world.

Than, in the voice **Arduino IoT Cloud**, we created a new thing called *Alexa*. In this thing, we created 3 properties which are equivalent to the 3 modes (free, manual, sleep), and therefore to the devices to which Alexa will be able to connect.

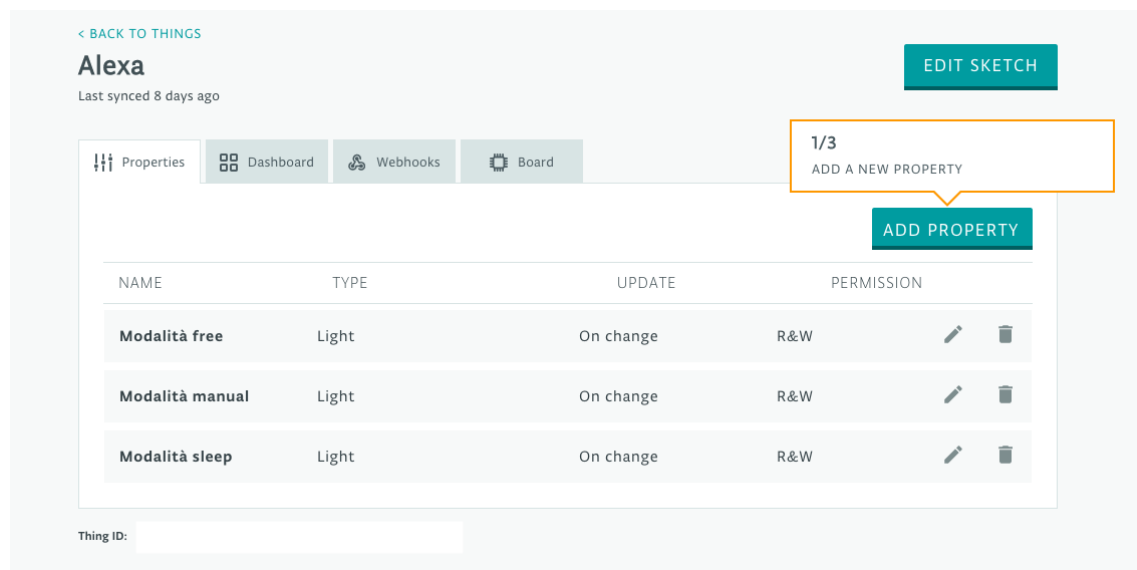


Figura 14: Arduino IoT Cloud

In the Figure 14, we can see the screen of Arduino IoT Cloud, with the 3 properties, and below we find the **Thing ID** that is very important to connect the device, as we'll see.

In the voice **Dashboard**, we can see a representation of the type of property chosen and we also have the possibility to change their status. We chosen the property *Light* because is easier to control, but there are also many other types of properties.

If you click on the button **Edit Sketch**, Arduino Create will open the online IDE (Figure 15) where part of the code will be generated, based on the previously created properties.

In the *Secret* file, you need to insert the network information (name and password), in *thingProperties.h* we have the connection to the network and the initialization of the various properties, included the Thing ID. Instead, in the sketch *alexa_nov15b.ino* the online IDE automatically creates a function for each property, which is executed whenever its state changes. You can insert your code in this sketch, and obviously the physical board must be connected to the PC. Finally, you have to save, verify and upload the sketch on the board (using the button with the arrow, on top left).

Then, from the Alexa app, you need to download the Arduino skill, than you need to go in *Devices*, click on *Add Devices*, then click on the type of devices, in this case *Light*, and search for devices; Alexa will find the devices that correspond to the properties on Arduino Create, add this devices, and finally you can communicate with devices via the Alexa voice commands.

In our project, every time that Alexa receives the command voice, and the status of one of this devices(properties) is changed, the corresponding function is executed through which the MQTT request is sent. The other Arduino reads the message on the same topic and changes to the mode required.

If you want use Arduino IDE instead of the online IDE, you need to download some libraries: *Arduino_ConnectionHandler*, *RTCZero*, *ArduinoIoTCloud*, *ArduinoCloudThing*.

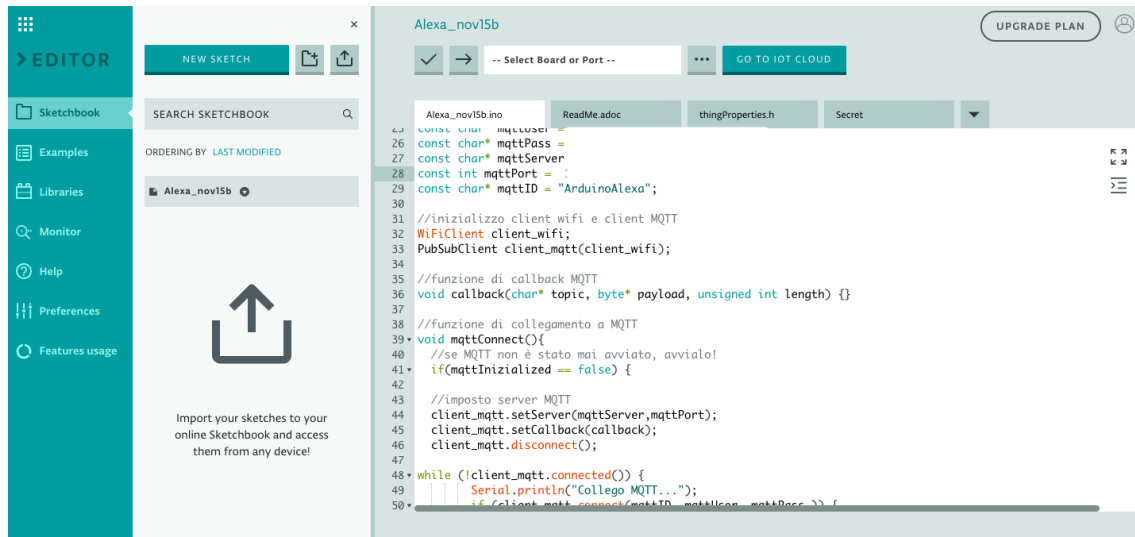


Figura 15: Online IDE

3.4.1 Keep attention

You should have some little problems with the Arduino MKR WiFi 1010 board.

- If you have problems with the configuration of the board with Arduino Create, go to Arduino IDE, upload on the board a sketch called *FirmwareUpdater* that you can find in the example sketch of Wi-FiNINA(in the tools item). Then in the Tools menu, open *WiFi101/WiFiNINA FirmwareUpdater*, select the port, select the most recent firmware of the board and click on *Update Firmware*. After, select the domain *arduino.cc*, and click on *Upload Certificates to WiFi module*. This procedure will update the board's firmware.
- If you have problems with the POST request for IFTTT on this board, perform the same procedure as in the previous point, but after you click on the button *Update Firmware*, click on *Add domain*, insert *maker.ifttt.com*, click "ok", select both domains, and finally click on *Upload Certificates to WiFi module*. Without this procedure it is possible that the request is not successful, as you do not have the necessary permissions.

4 Wiring of components

4.1 Arduino MKR WiFi 1010

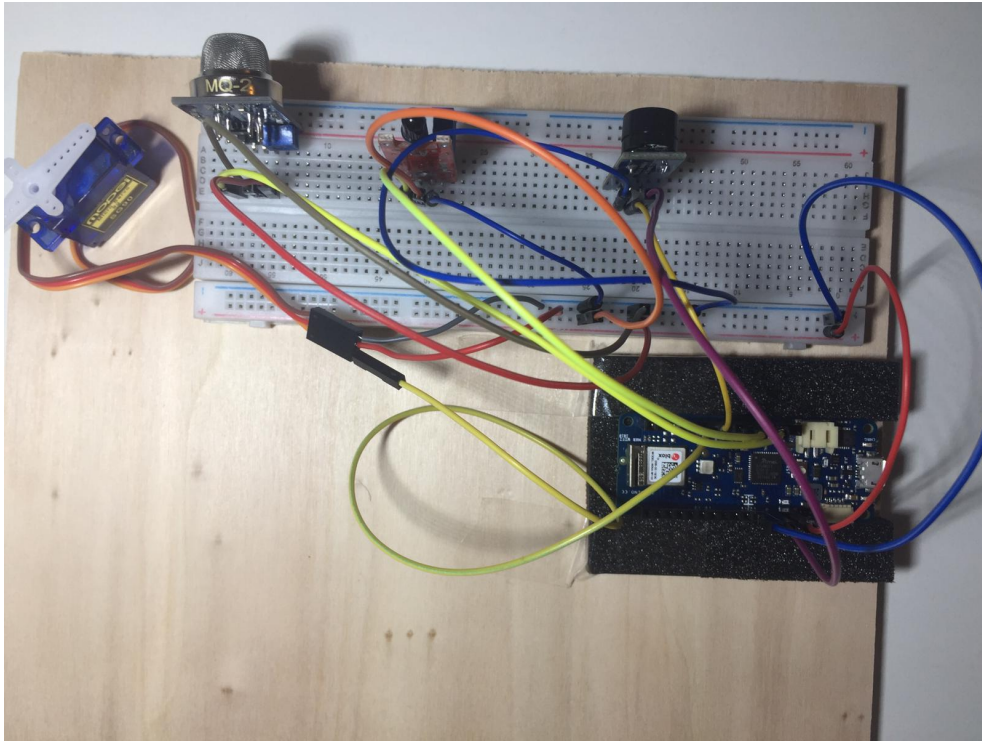


Figura 16: Wiring Arduino MKR WiFi 1010(from above)

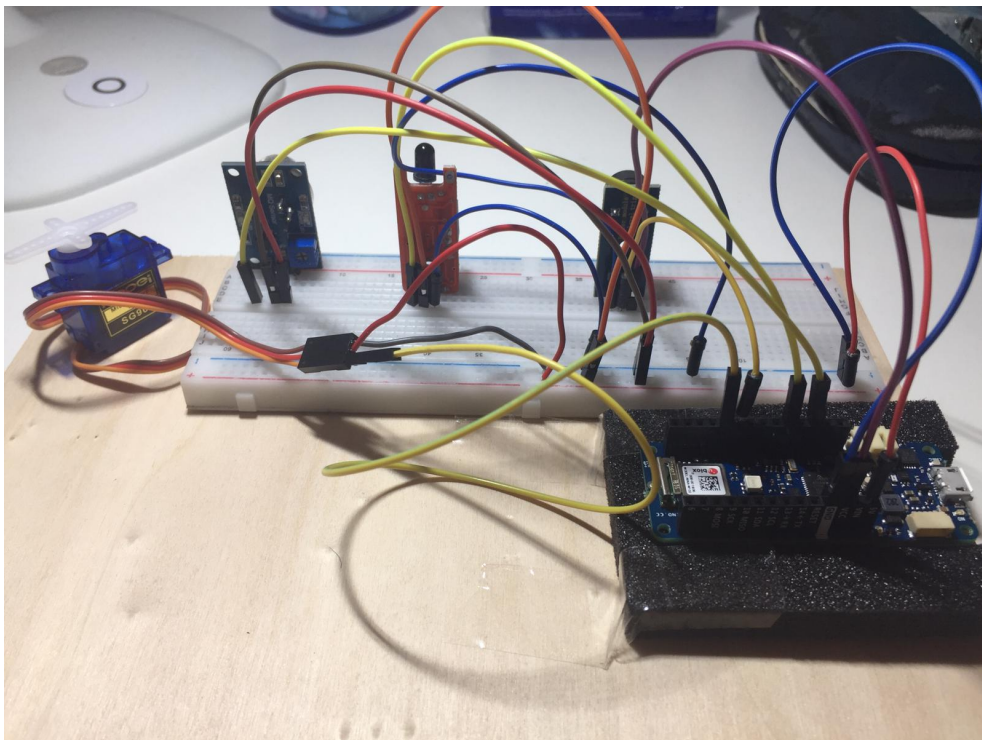


Figura 17: Wiring Arduino MKR WiFi 1010(from behind)

With this arduino, we connected its GND and V5 entry with the board, respectively to the negative and positive poles, through red and blue wires.

With **gas sensor**, the analog entry is connected with the analog pin A1(through the yellow wire), the GND of the sensor is connected to the negative pole on the board(through the brown wire), and the VCC entry of the sensor is connected to the positive pole on the board(through red wire).

With the **flame sensor**, the digital entry is connected with the analog pin A3(through the yellow wire), the GND of the sensor is connected to the negative pole on the board(through the brown wire), and the VCC entry of the sensor is connected to the positive pole on the board(through orange wire).

With the **buzzer**, the I/O exit is connected with the digital pin 0(through the yellow wire), the GND entry of the sensor is connected to the negative pole on the board(through the blue wire), and the VCC entry of the sensor is connected to the VCC of the Arduino(through the purple wire), because in this way the buzzer receives 3.3v.

With the **servomotor**, the entry with the orange wire is connected with the digital pin 0(through the yellow wire), the entry with the red wire is connected with the positive pole on the board(through the red wire), and the entry with the brown wire is connected with the negative pole on the board.

4.2 Arduino UNO WiFi Rev 2

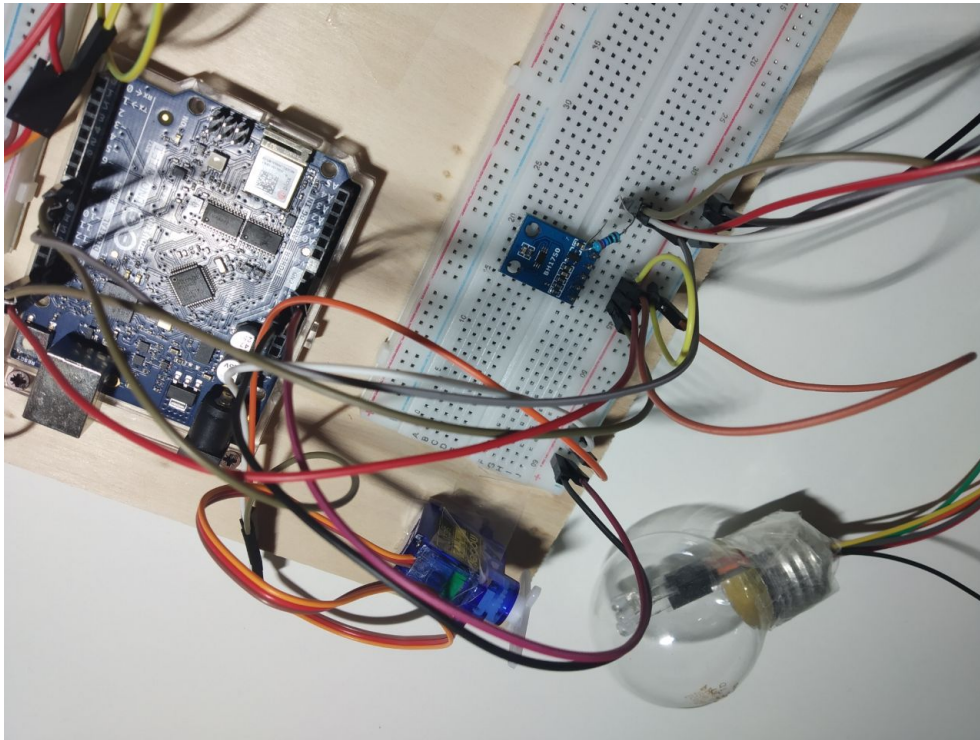


Figura 18: Wiring Arduino UNO WiFi Rev 2(from above)

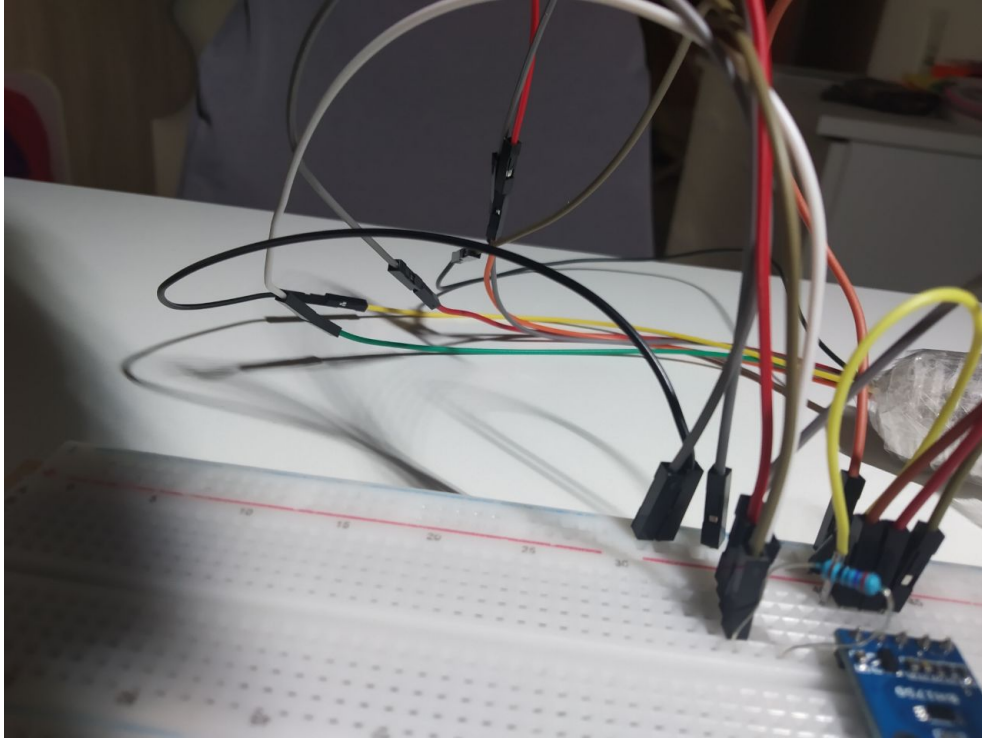


Figura 19: Wiring Arduino UNO WiFi Rev 2, LED parallel connection

Just like with the Arduino MKR WiFi 1010, we connected the GND and the voltage (5V) entry respectively to the negative and positive poles of the board. Connected to this Arduino we have, three white LED with one resistor, the light sensors and one servo motor. In order to avoid to use one PIN for each led, we connect the led in **parallel**, as we can see in *Figure 19*.

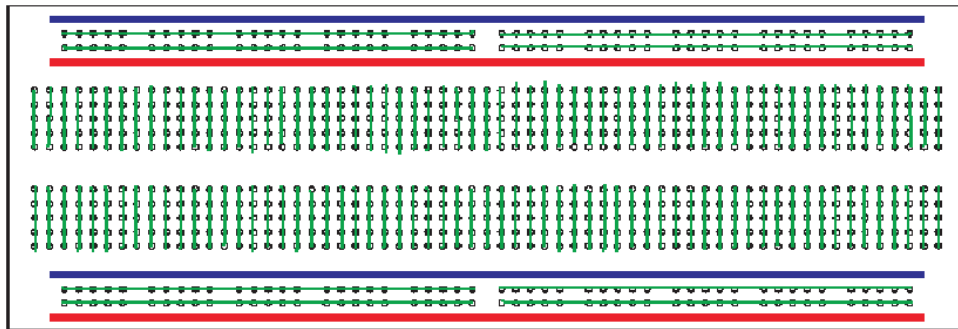


Figura 20: Bread Board-Internal connections

As we can see in the figure above, we can connect the LEDs in parallel since the entry of a column, generally five, in the bread board are internally connected to each other. In the *Figure 18*, we can see three cables connected with the respective anode, while the cathodes are connected to the GND. Just one cable (the one near the resistor) is connected to the PIN in order to share the signal with the other three components, in this way the LEDs work simultaneously. The PIN used in our project is the **PIN 10**, that can be used as PWM outputs.

For the **light sensor**, we have a **BH1750** sensor, the connection are written on the sensor. In the *Figure 18* shows how we connect the light sensor, the first to the right is the **addr** and we doesn't use it. The next two are connected to **SDA** and **SCL**. The SDA is the Serial data that carries the data, while the SCL is the Serial Clock is the clock signal wich synchronize the data transfer between devices. The last two (always to the right) are connected to the GND and voltage.

The last one components is the **servomotor SG90** this object present three cables. The *brown* cable need to be connected to the GND, while the *red* one to the voltage and the orange to the PIN, in our project for the servomotor we use the PIN 8.