

SMARTLAB DIY Sensor Workshops

SMART  LAB

Author

Daniel Nuñez H. (UL)

Contact

hello@smartlablimerick.ie

Disclaimer

This document contains information that reflects only the authors' views, and the Sustainable Energy Authority of Ireland is not responsible for any use that may be made of the information it contains.

Table of Contents

Introduction.....	2
Workshop Methodology	3
1.1 Activity description.....	4
1.1.1 Internet Of Things.....	5
1.1.2 Graphing data in the cloud	5
1.2 IoT Kits description.....	5
1.3 Arduino Cloud.....	6
1.4 Workshop structure	7
1.4.1 Code details.....	23
References.....	29

Introduction

The SMARTLAB project is assessing methods for implementing the Smart Readiness Indicator (SRI) within existing building stock in Limerick city centre. The SRI standard is intended to support efforts to improve efficiency, comfort and performance of buildings and contribute to targeted reductions in carbon emissions across society. The vision for this project is mapping a process where ordinarily 'non-smart' buildings can be brought up to a basic level of 'smart readiness' (O'Flaherty, 2023). The process of change encompassed by this aim is a socio-technical one, involving both technological and social innovations. A key aspect of this work is engaging with project participants to better understand how they view sensor technology and to demystify that technology so that it can be made more useful and impactful. In light of this, the project's [WP1-D1 Plan for Stakeholder Engagement](#) describes how a Do It Yourself (DIY) approach is integrated into project delivery. This work is directly related to project objectives around the empowerment of smart energy citizens and the deployment of smart infrastructure.

Although DIY sensors will not be installed as the default option in SMARTLAB participant buildings, due to ease of calibration of the factory-made devices and the need for a CE mark on the devices used, a core communication function of the project is to build and share DIY toolkits which are easy to use and understand by non-technical stakeholders (WP2, WP4). This is to support capacity building amongst stakeholders on the potential of smart building technologies and to make these technologies more accessible.

The project purchased a number of Arduino kits which can collect data critical to SRI evaluation and ongoing smart building services. There is potential for DIY sensors to contribute to an improved SRI rating in any building assessment, as it is the functionality of the technology, which is measured, and DIY sensors can potentially be more easily adapted to include control add-ons than proprietary technology. The availability of DIY kits will address several technical and financial barriers to the uptake of smart technologies already identified within project research, in the SMARTLAB deliverable [WP1-D3 Barriers to Improving the Smartness of Buildings](#).

This report outlines the work involved in developing project activities to support a Do-It-Yourself approach. It covers the development of sensor demystification workshops with project participants. They are intended to be introductory lessons to familiarise participants with the relevant technology, facilitating the adoption of technology and supporting broader understanding of the functioning of the Internet of Things (IoT). Work in this area, carried out in Fab Lab Limerick and in the Citizen Innovation Lab and tested in preliminary workshops with Limerick's "Maker Community" in November and December 2022, has been published on GitHub for further refinement by online communities: [SMARTLAB Limerick on GitHub](#).

Workshop Methodology

The workshops outlined below are designed to allow for demystification of sensor technologies among potential participants in the SMARTLAB project. To achieve this objective, Do-It-Yourself sessions are proposed. Users will experiment with different technological devices that allow them to enter the IoT world, understand the operation and control certain aspects that will enable them to adopt the technology.

For this purpose, IoT Kit belonging to the Arduino brand will be used. The specific kit is called [Explore IoT Arduino](#). It aims to enable the first steps in internet-connected objects. It is an educational tool used in high schools and colleges.

The kit has a teacher's guide, with a series of activities that take students through different IoT concepts and knowledge in a straightforward way. Some of the activities that can be developed with the kit are:

1. Introduction to an Internet of Things
2. Get to know the kit
3. Graphing Data in the cloud
4. Storing our data
5. Remote triggers
6. Home security alarm
7. Classroom tracker
8. Urban farming 101
9. Greenhouse web server
10. International weather application

The SMARTLAB project can be explained dynamically with activities one and three. Suggestions for facilitators of activities using these kits are as follows:

- Facilitation: Your role as a "facilitator" is vital for the students to find the right approach
- Troubleshooting: If activities or projects do not work as expected, help the students find the right path to detect mistakes.
- Sharing: There are multiple ways to write the same program, and students can learn a lot from each other by sharing their sketches and presenting their solutions for the given challenges.
- Providing additional knowledge: Take the opportunity to extend explanations and add real-life examples and facts during the lessons.
- Self-assessment: Encourage the students to think about what they are doing, how they perform, and how their work could be improved.
- Time management: Make sure that the students accomplish something by the end of each session for positive assurance, even if they do not cover all of the content.
- Personal growth: See mistakes (both yours and those of the students) as opportunities to learn. Promote lifelong learning and self-reliability.

Currently, there are 7 Kits IoT, and the maximum recommended range for kits is two people. Therefore, technological introductory activities can only be for a maximum of fourteen people.

The following are required for the workshops:

1. Computers according to the number of kits available. Basic features, +13 or similar, +4GB ram, internet connection available (can be HotSpot from the phone)
2. A facilitator and a technical assistant to answer the queries.
3. Have electrical connection points
4. Display or projector for presentation

Additionally, the workshops must have exemplary elements of the technology, such as: showing installed Milesight sensors and the IES platform.

1.1 Activity description

The activities are intended to cover expected events described in WP1-D1 chapter 3.1.

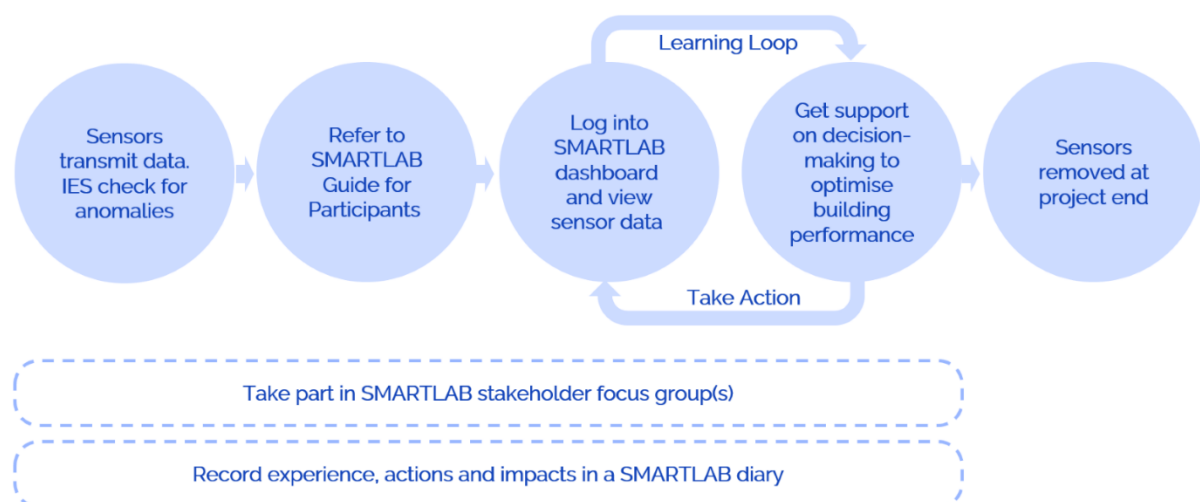


Fig. 53 SMARTLAB Participant Journey: Demonstration of energy management/other smart services (Trejo-Rangel et al., 2022, p. 20)

Activities one and three of the Explore IoT Kit, Internet of things and Graphing data in the cloud are sufficient to reach the "Learning Loop" and "Take Action" zone. Allowing participants to log into a dashboard and view their sensor data and, at the same time, understand how their decisions could improve the performance of their building.

It is recommended to divide the activity into two parts, one of exposition of concepts and first steps in the platform, and the second in creating the dashboard. These two activities comprise a total time of 60 min, maximum. This is considering the simplification in the explanation of codes and use. Additionally, participants can explore actions and consequences that allow them to be viewed on the platform. For example:

- Place the heating system near the equipment
- Breathing close
- Move through space or outside.

With this, participants will clearly understand how the SMARTLAB project seeks to work and help them in energy decision-making.

1.1.1 Internet Of Things

This activity is described in the learning course provided by Explora IoT Kit, [link](#). It seeks to explain how connected physical targets communicate and share data over the internet or other networks. The topics covered by this activity are:

- What is the Internet of Things
- How extensive is the network
- Where can we use these networks of connected objects

This activity should not take more than 8 minutes since it is only an introduction to concepts.

1.1.2 Graphing data in the cloud

This activity is described in the learning course provided by Explora IoT Kit, [link](#). It seeks that participants can visualise the data obtained by their sensors by having a dashboard platform with which they can observe and understand how their space behaves. The topics covered by this activity are:

- Configuring the device
- Creating a dashboard
- Computational thinking through the code

1.2 IoT Kits description

The IoT kit or Arduino Explore IoT Kit is a tool that allows entry into the digital world of connected objects—introducing the fundamentals of the Internet of Things quickly and simply.

The kit contains ten activities with approximately 45 min per activity. This time can be reduced by eliminating parts of the session of detailed explanation of the technology, such as programming or understanding the code.

Additionally, the kits include the following:

- Online learning platform
- 12 months free trial for Arduino Create Maker Plan
- Arduino MKR WiFi 1010
- MKR IoT Carrier with built-in sensors and actuators
- PIR sensor + plug-and-play cable
- Moisture sensor + plug-and-play cable
- Micro USB-cable
- Case for the Arduino MKR Wi-Fi 1010 and MKR IoT Carrier

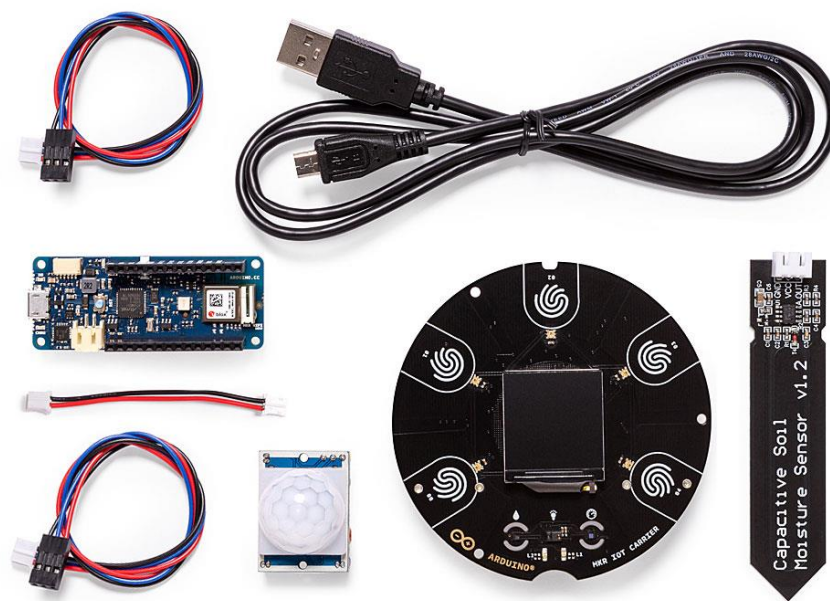


Fig. 54: Arduino Explore IoT Kit

Link: <https://explore-iot.arduino.cc/guide/teacher-guide>

1.3 Arduino Cloud

Arduino Cloud is an online platform enabling anyone to write code, access educational content, set up boards and share projects. It allows free access and can access web editors to program, connect multiple devices with Arduino IoT Cloud and view information.

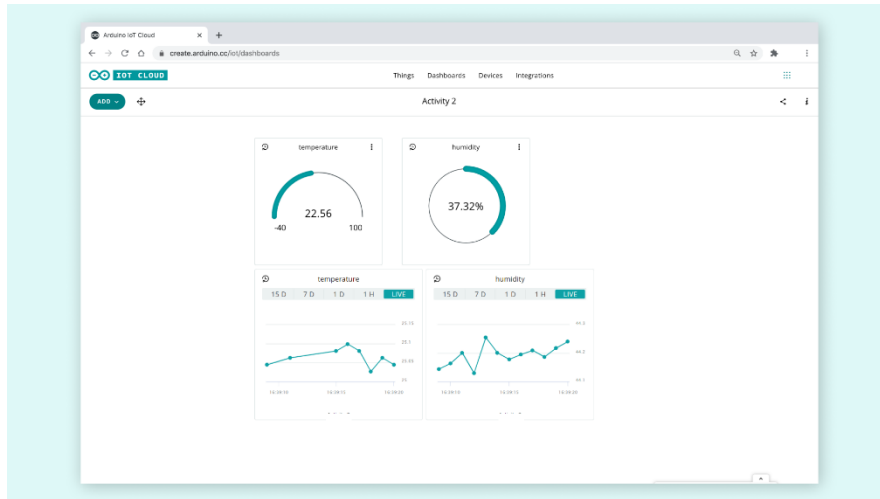


Fig. 55: Arduino IoT Cloud

Link: <https://explore-iot.arduino.cc/guide/teacher-guide>

Activity number 3 thoroughly explains how to access this platform and use it with clear examples and step-by-step to create a solution containing temperature and humidity monitoring—demonstrating the incredible versatility of the kit and the ease users and students can get into the technology quickly.

1.4 Workshop structure

The structure and steps for the realization of the workshop is designed as follows:

1. Arrangement of the computers (one for each kit), tested with the platform. Devices ready and connected.
2. Every Arduino kit must be already ready, that is: armed and connected to the computer to be used. This avoids possible errors of recognition of the computer and the equipment.
3. Have Google Chrome open on the Arduino.cc page, within the account with which you will work (a test was already carried out with the same account open on multiple computers).

Name	Device	Variables	Last Modified
Guacamole	ASSOCIATE DEVICE	-	15 Dec 2022 22:43:59
Miguel MXN	Miguel Arduino MKR WiFi 1010	weather_report +4	15 Dec 2022 22:43:59
Smartlab	ASSOCIATE DEVICE	humidity +1	15 Dec 2022 22:03:41
smartlab02	ASSOCIATE DEVICE	temperature +4	09 Jan 2023 12:31:00
Untitled	Raynell Arduino MKR WiFi 1010	energy +2	09 Jan 2023 11:02:02
Untitled 2	smartlab02 Arduino MKR WiFi 1010	temperature	09 Jan 2023 12:40:29

Fig. 56: Arduino web, IoT Cloud front page

4. Have a demo device ready. That shows in case of connection errors, how the data and behavior are displayed before external stimuli (use the Heater Fan).
5. Enable a HotSpot point for Wi-Fi, without password.

These are the previous elements to take into account for the realization of the workshops. The following are the steps for realization:

1. Introduction to key IoT concepts. (8 min approx).
2. Introduction to the Arduino IoT cloud (2 min aprox). Use [link](#) to explain graphically how the platform works and why we are using this to explain SMARTLAB project.

"The Arduino IoT Cloud is an all in one solution for IoT development, where you can build visual dashboard to monitoring your devices, control your data remotely, share life updates with other surfaces and much more. It is an excellent tool to introduce you in the Internet of things concept and start to develop your DIY solutions."

"SMARTLAB project use a similar concept taking you in a journey through the technology to improve the behaviour of your building. Allowing you to understand the environment around you and how they react to your actions. It knowledge permit you and us make and take better decisions in Energy consuming, Heating and Cooling, Ventilation, Lightning and other."

3. Start by creating a new project. In IoT Cloud, "Things" tab, click "Create".

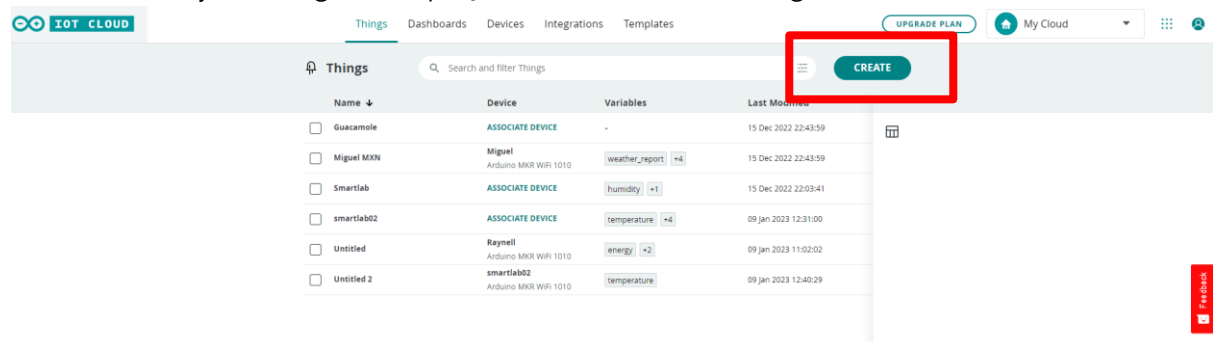


Fig. 57: Arduino web, IoT Cloud, create project

4. We created the name of the project.

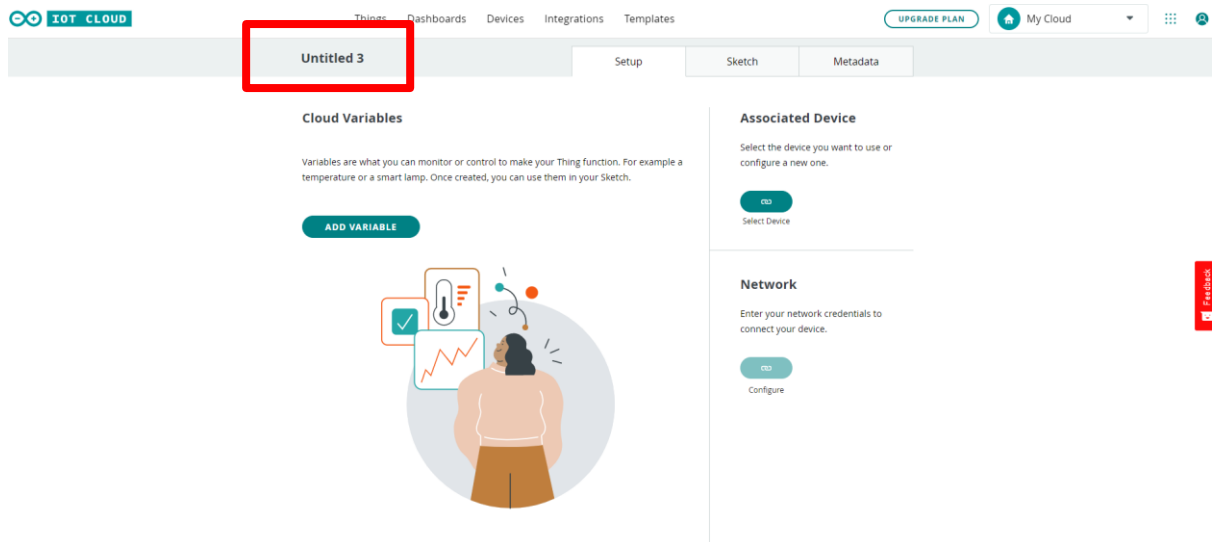


Fig. 58: Arduino web, IoT Cloud, project name

5. We associate the device with the one we are going to work on (it must already be connected to the computer)

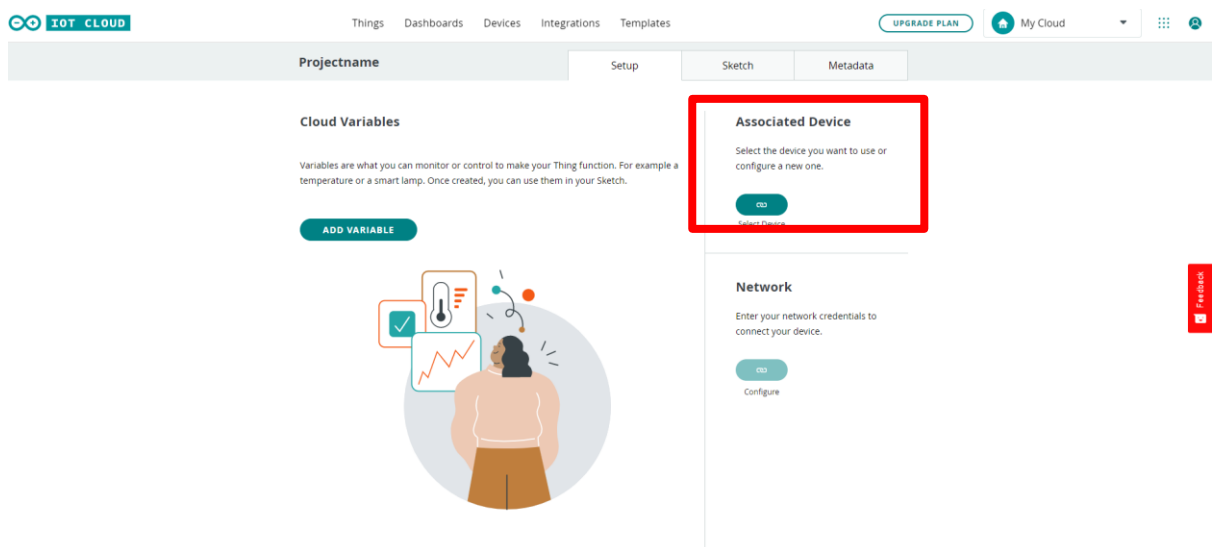


Fig. 59: Arduino web, IoT Cloud, associate device

6. Set up a new device.

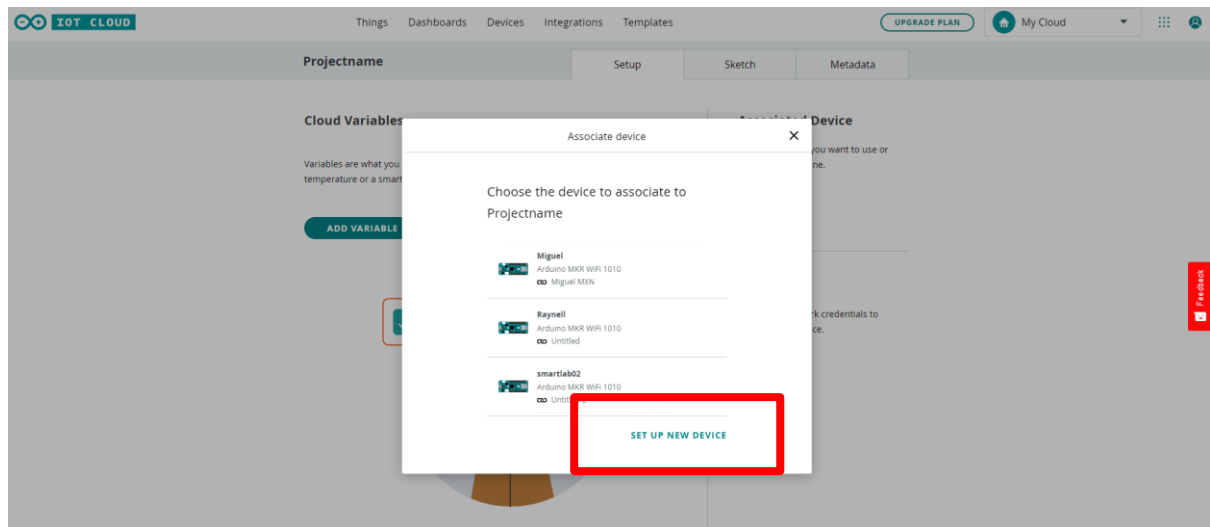


Fig. 60: Arduino web, IoT Cloud, set up new device

7. Select Arduino device.

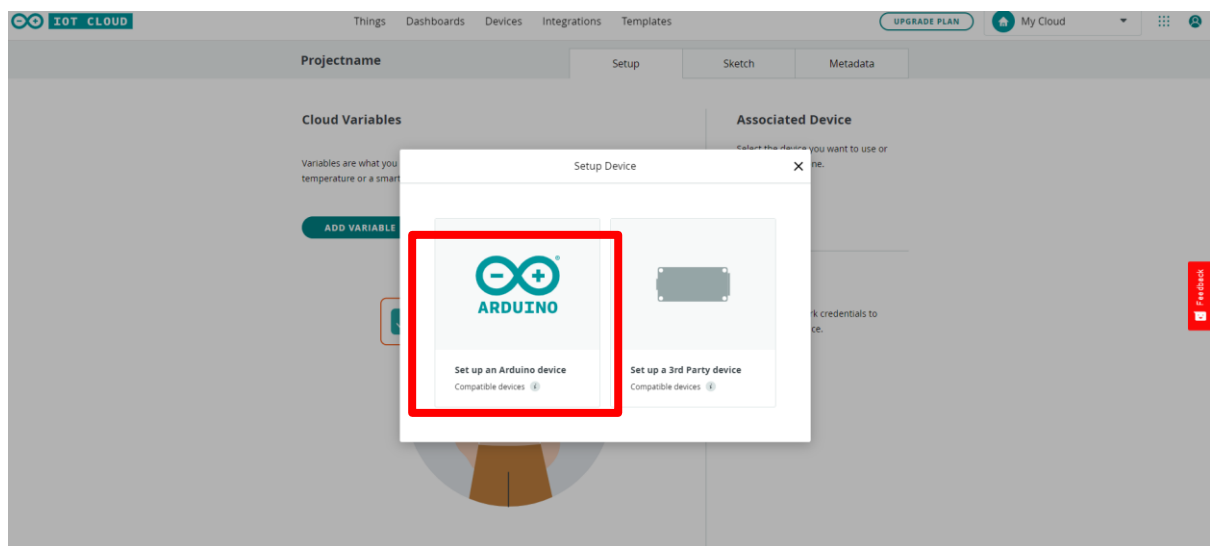


Fig. 61: Arduino web, IoT Cloud, Arduino device

8. The device will be recognized in a short time, when it happens it will give us the following confirmation:

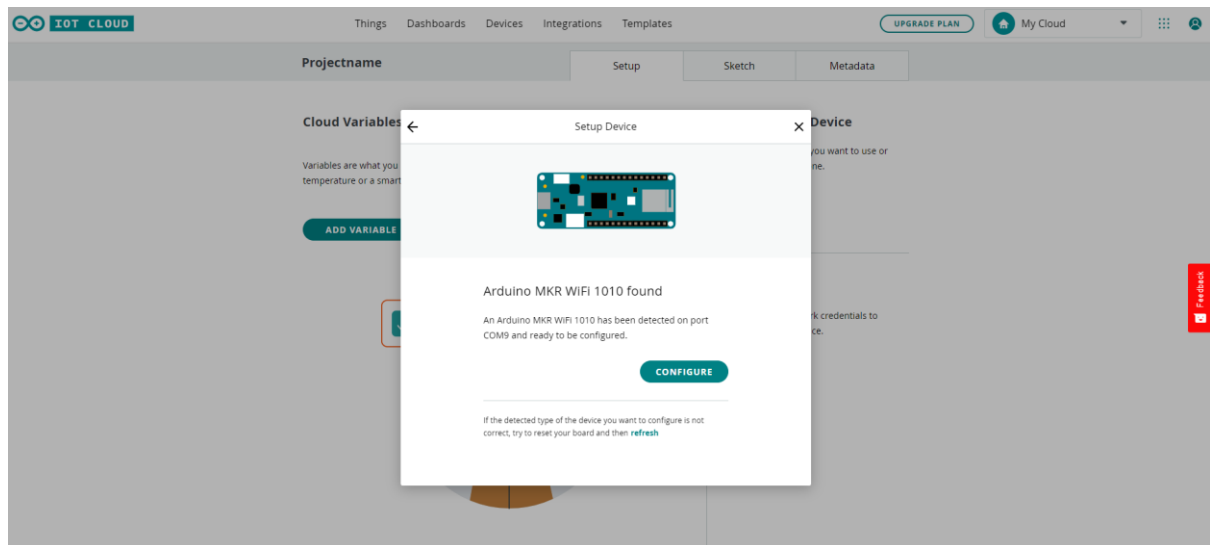


Fig. 62: Arduino web, IoT Cloud, Device found

9. It will ask us to enter a number to the device, an ID that we can recognize.
10. Once done we are ready to enter our variables. Variables are all the elements we want to know. For this workshop, we will try to obtain the Temperature and Humidity from the air near the equipment.

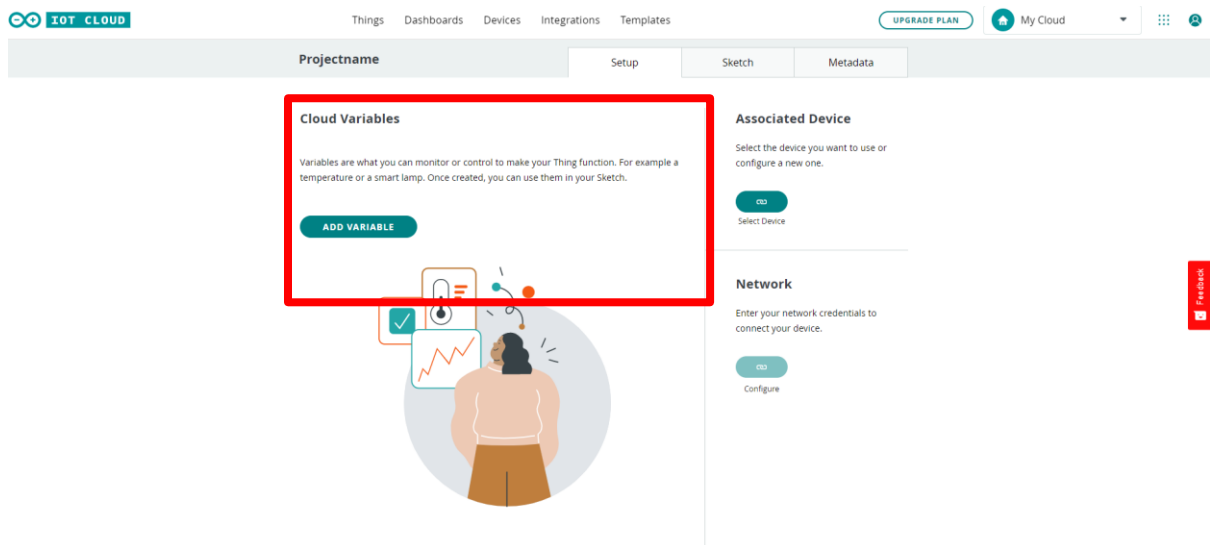




Fig. 63: Arduino web, IoT Cloud, add variables

11. This will be our configuration for the temperature variable

Add variable


Name

temperature

 Sync with other Things 


Floating Point Number


eg. 1.55



Declaration


`float temperature ;`



Variable Permission 

☐ Read & Write

☒ Read Only

Variable Update Policy 

☐ On change

☒ Periodically

Every

1

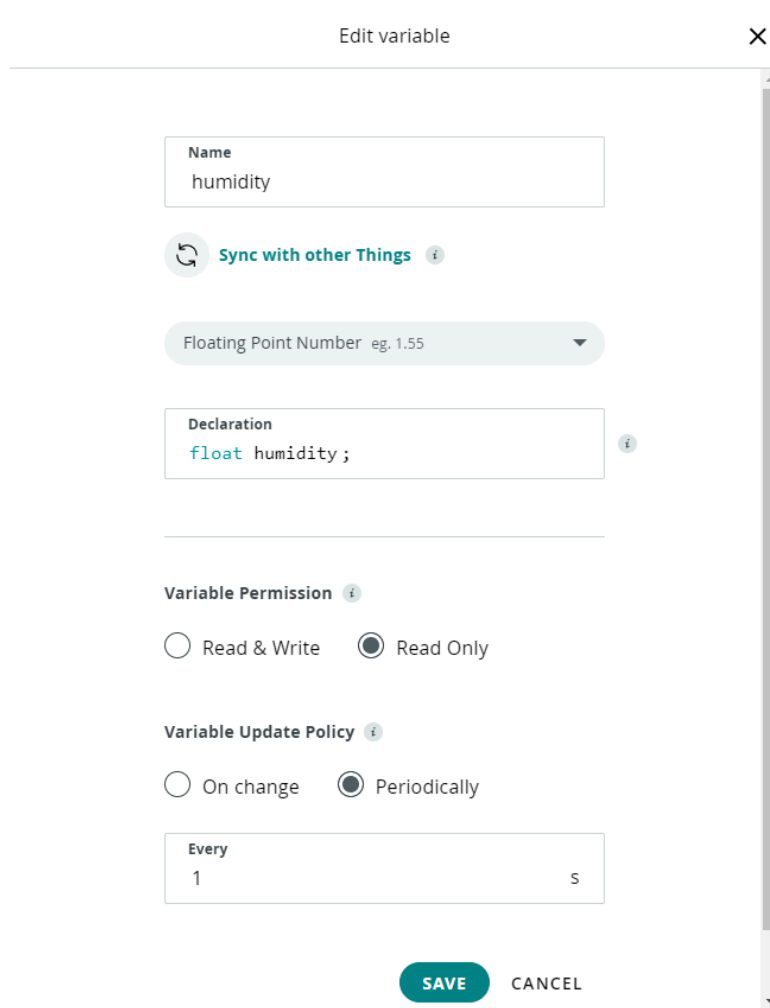
S

ADD VARIABLE

CANCEL

Fig. 64: Arduino web, IoT Cloud, temperature set up

12. This will be the configuration for the humidity variable



Edit variable

Name
humidity

Sync with other Things

Floating Point Number eg. 1.55

Declaration
`float humidity;`

Variable Permission

☐ Read & Write ☒ Read Only

Variable Update Policy

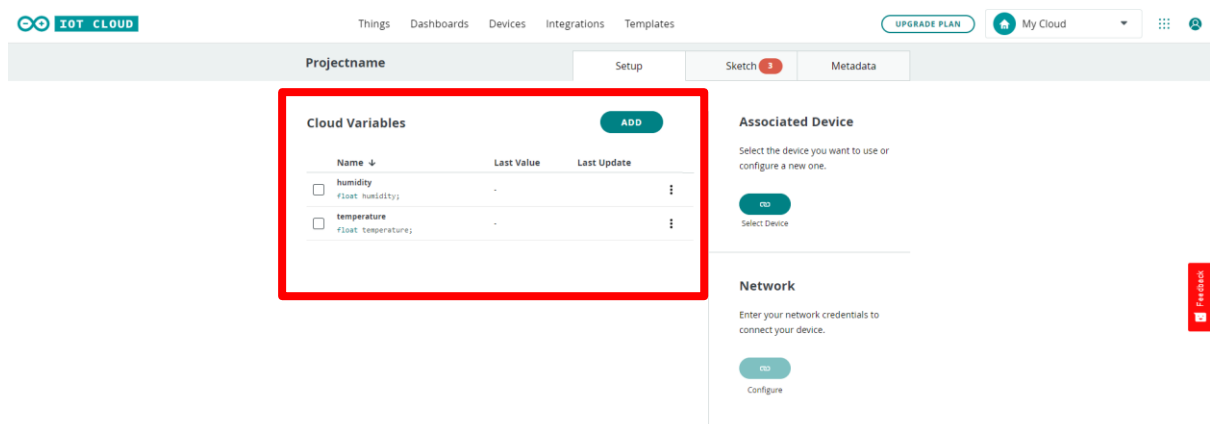
☐ On change ☒ Periodically

Every
1 s

SAVE CANCEL

Fig. 65: Arduino web, IoT Cloud, temperature set up

13. With this we already have our variables configured and ready to be used.



IoT CLOUD

Things Dashboards Devices Integrations Templates

UPGRADE PLAN My Cloud

Projectname Setup Sketch Metadata

Cloud Variables

Name	Last Value	Last Update
<input type="checkbox"/> humidity float humidity;	-	-
<input type="checkbox"/> temperature float temperature;	-	-

Associated Device

Select the device you want to use or configure a new one.

Select Device

Network

Enter your network credentials to connect your device.

Configure

Fig. 66: Arduino web, IoT Cloud, temperature set up

14. Let's proceed with the creation of our dashboard. Click on dashboard at the top.

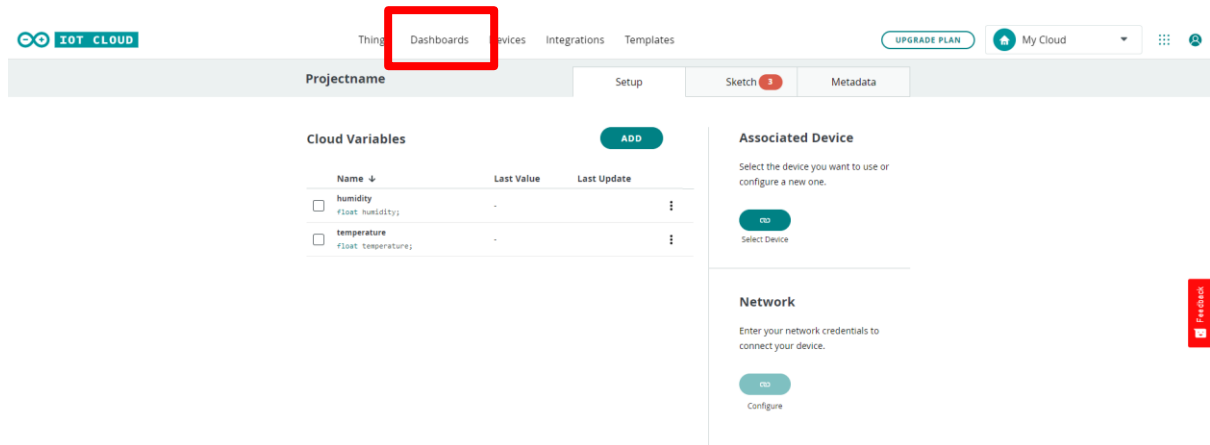


Fig. 67: Arduino web, IoT Cloud, dashboard head up

15. In dashboard, click on create.

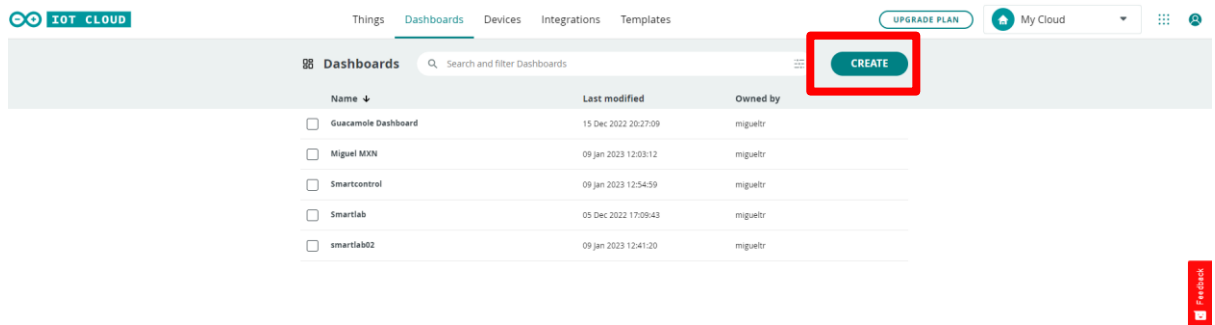


Fig. 68: Arduino web, IoT Cloud, dashboard create

16. Create a name for the dashboard

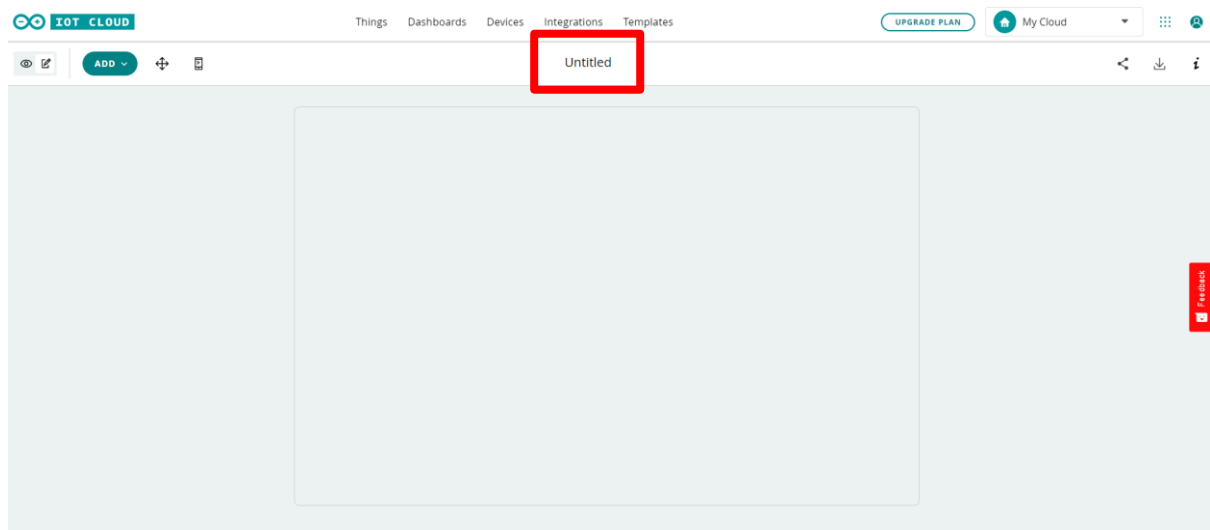


Fig. 69: Arduino web, IoT Cloud, dashboard name

17. Now we will add elements to our dashboard such as values, charts or indicators. Click on ADD, widgets.

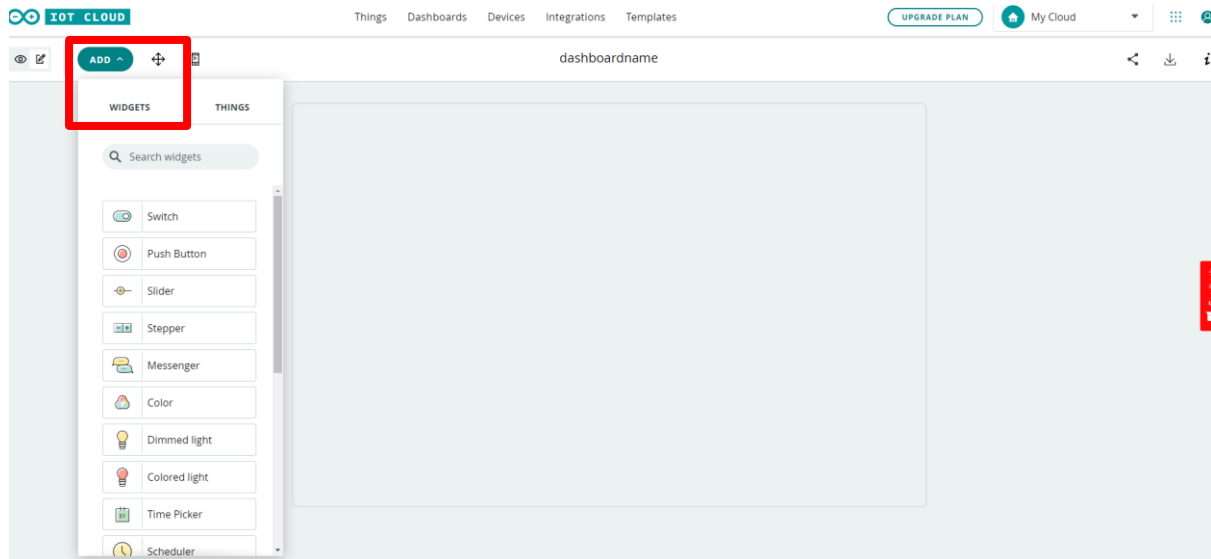


Fig. 70: Arduino web, IoT Cloud, Add elements

18. We look for the widget called "Value" and give it the name temperature

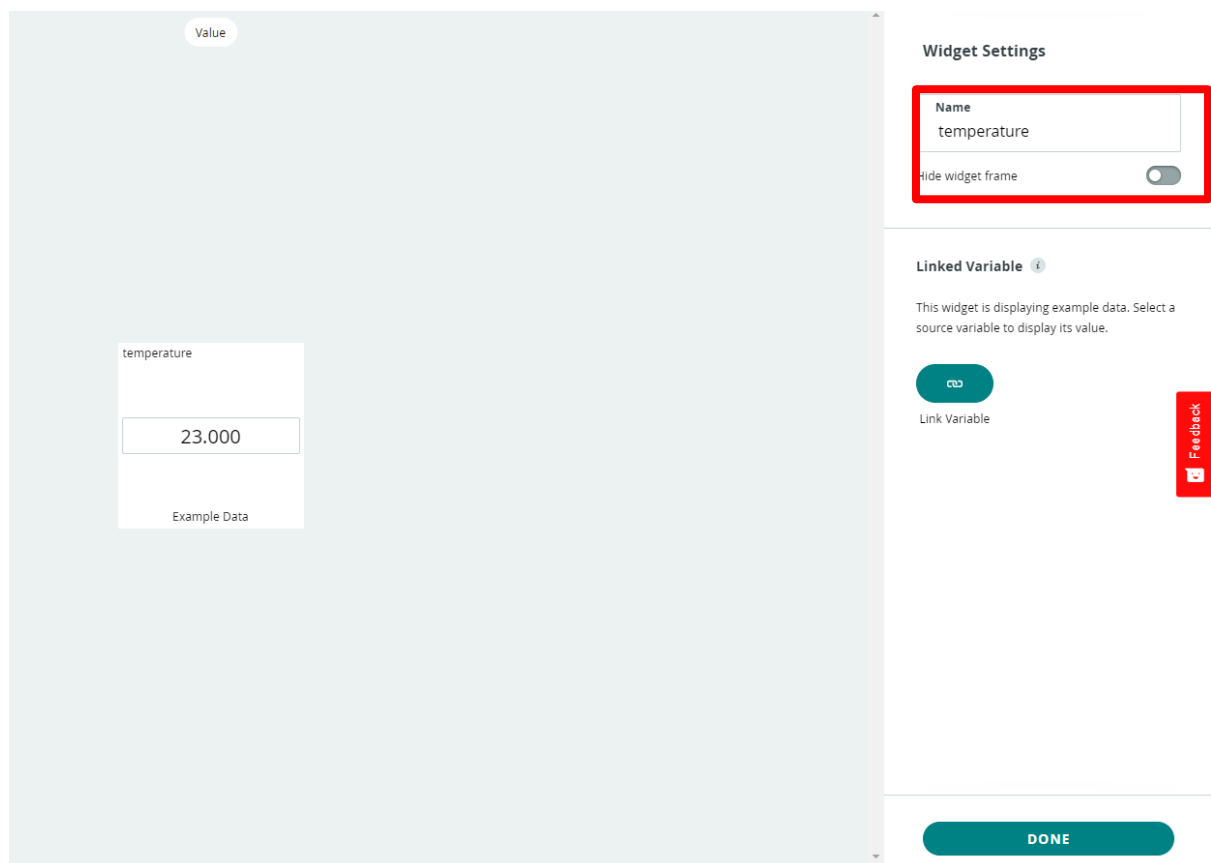


Fig. 71: Arduino web, IoT Cloud, dashboard temperature value

19. We now need to make a link with the previously created variable, click on "variable link."

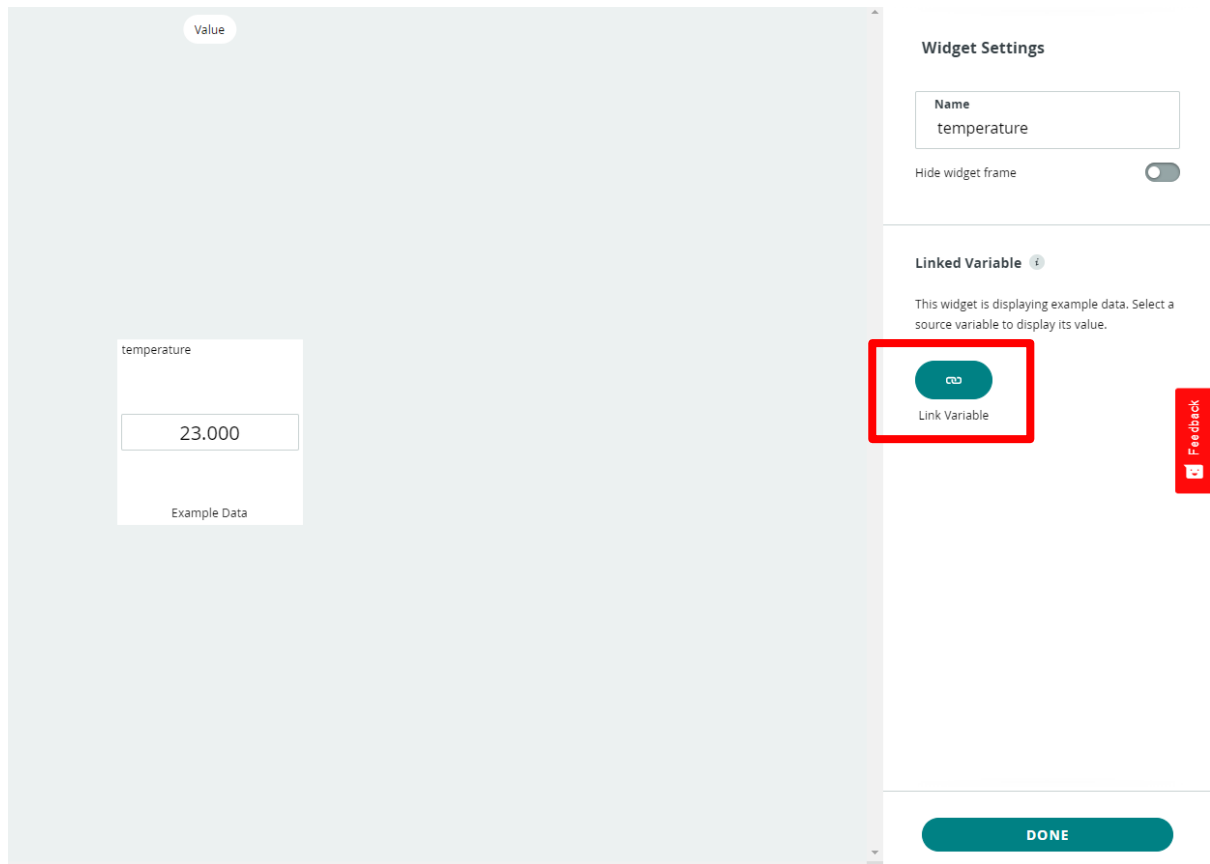


Fig. 72: Arduino web, IoT Cloud, dashboard value name

- Find the project we create and select the variable corresponding to the created element.

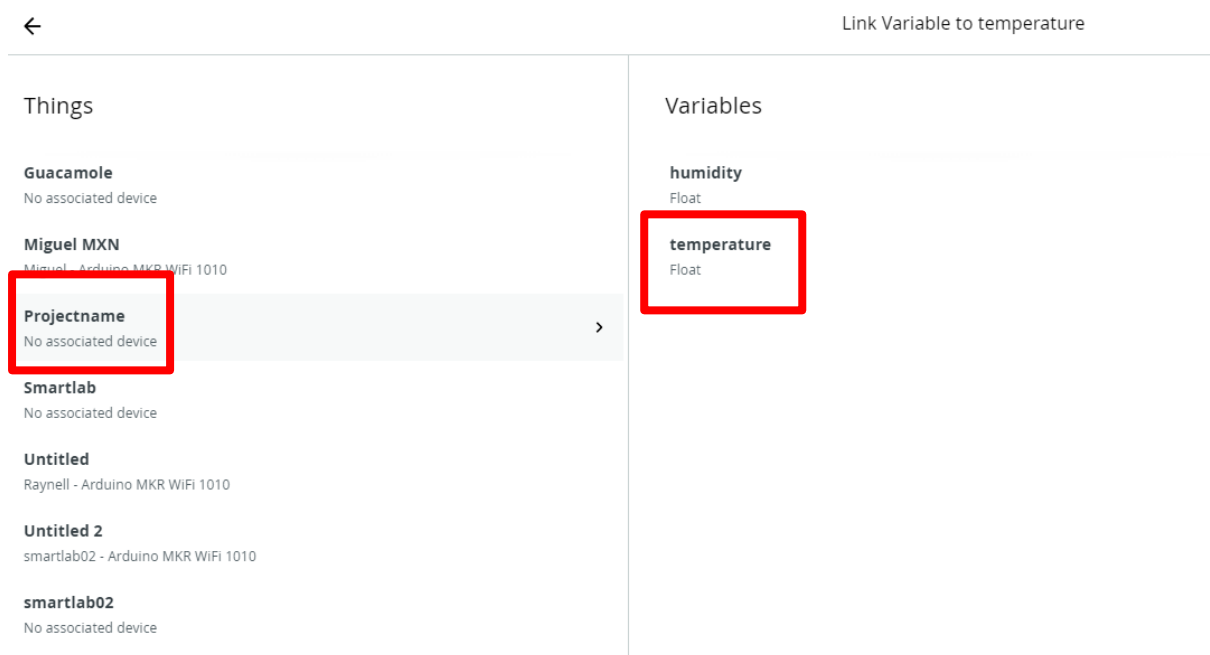


Fig. 73: Arduino web, IoT Cloud, variable linked

21. With this we already have our variable linked to our temperature element in the dashboard. Once we finish this process, we will automatically see the temperature and humidity data in the dashboard that we will create.
22. The same process should be performed for moisture. The result of completing this should result in this.

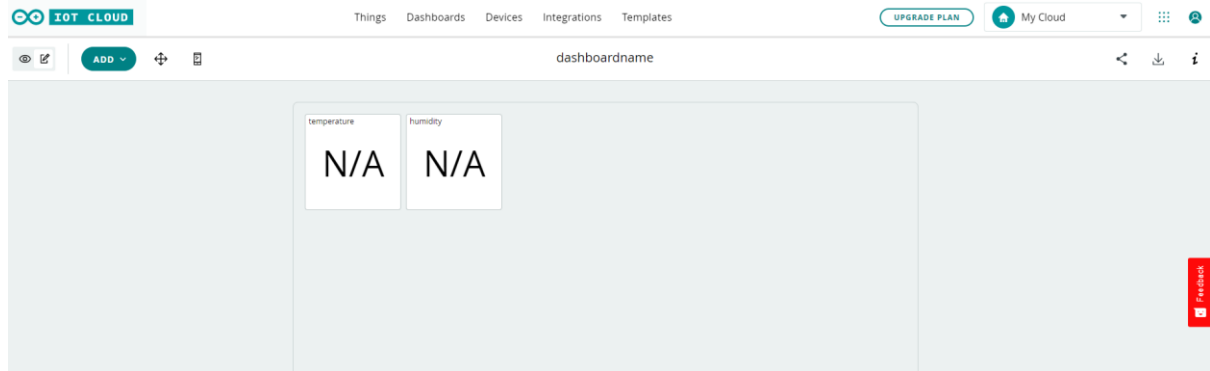


Fig. 74: Arduino web, IoT Cloud, dashboard with temperature and humidity value

23. We can add graphs that allow us to know the behavior of our environment in a period of time of days, weeks, months or in real time. The creation process is the same as creating the value widgets. The result should be as follows.

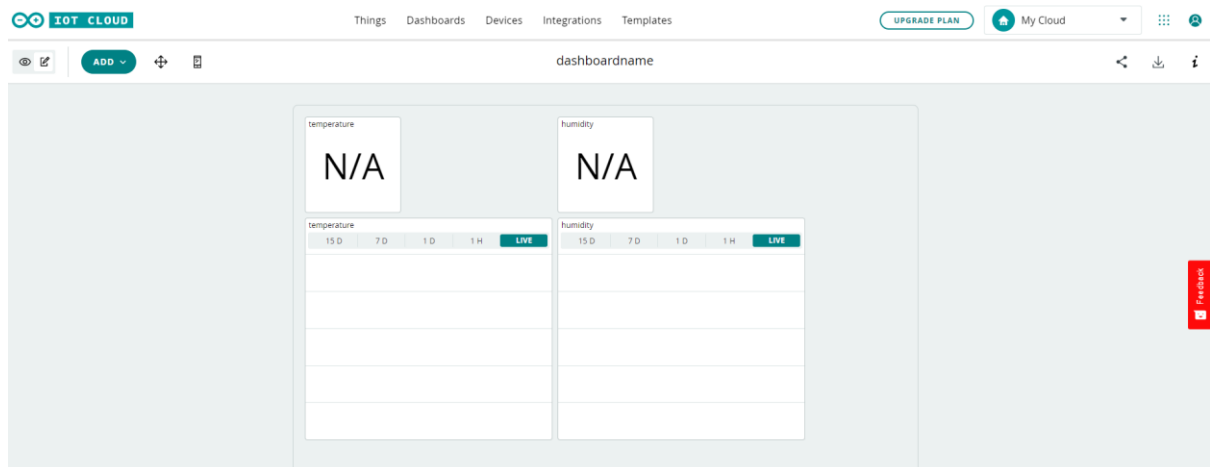


Fig. 75: Arduino web, IoT Cloud, dashboard set up

24. We are ready to start sending data to our platform, for this we need to connect our Arduino to a Wireless Network. In this case we will use Wi-Fi. Let's go back to the "Things" tab and select our project.

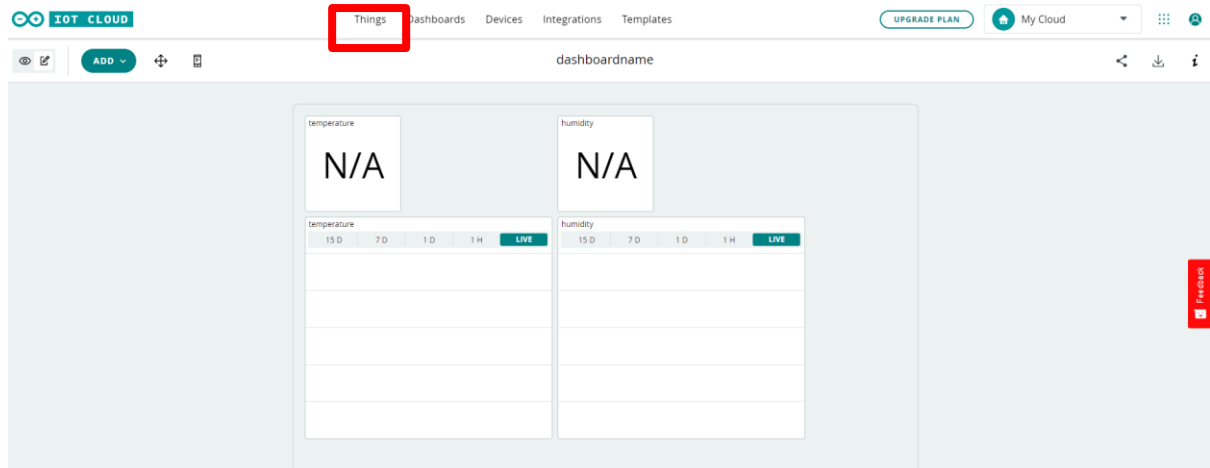


Fig. 76: Arduino web, IoT Cloud, come back to things

25. Once in the "things" page of our project, let's go to network.

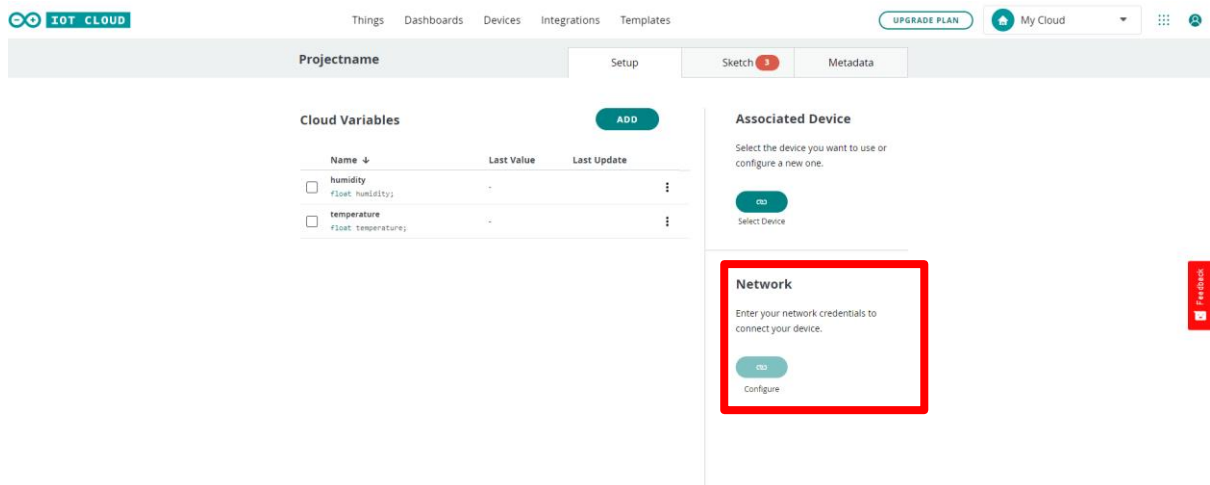


Fig. 77: Arduino web, IoT Cloud, network

26. Write the Wi-Fi credentials. I recommend use a HotSpot connection.

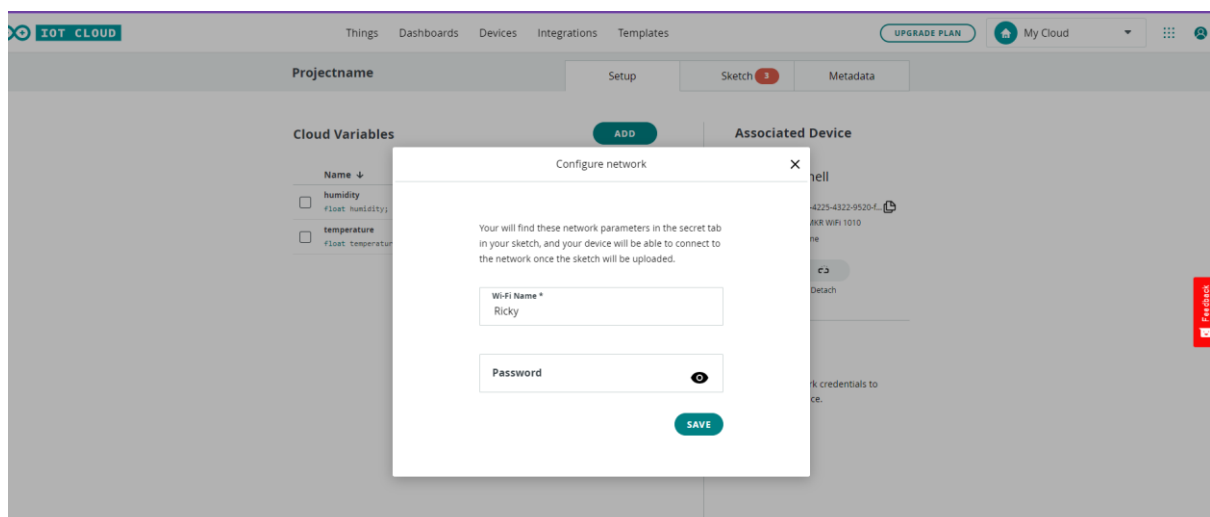
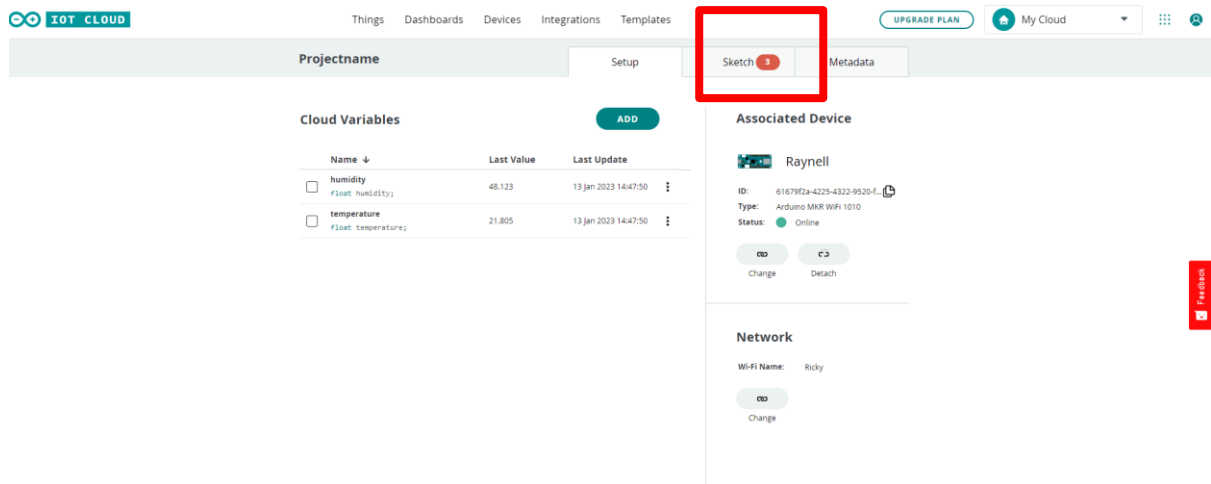


Fig. 78: Arduino web, IoT Cloud, Wi-Fi network

27. Once our connection is configured, we would only need to load the code that will allow our device to read all the data and show them in the dashboard we create. For this go to the stinking "sketch".

*Fig. 79: Arduino web, IoT Cloud, Sketch*

28. Once inside, delete the code that appears on the screen and replace with the following:

```
#include "thingProperties.h"
#include <Arduino_MKRIoTCarrier.h>
MKRIoTCarrier carrier;

void setup() {
  // Initialize serial
  Serial.begin(9600);
  // Defined in thingProperties.h
  initProperties();

  // Connect to Arduino IoT Cloud
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);

  CARRIER_CASE = false;
  carrier.begin();

  //Get Cloud Info/errors , 0 (only errors) up to 4
  setDebugMessageLevel(4);
  ArduinoCloud.printDebugInfo();

  //Wait to get cloud connection to init the carrier
  while (ArduinoCloud.connected() != 1) {
    ArduinoCloud.update();
  }
}
```

```
carrier.display.setTextSize(3);
carrier.display.setCursor(20, 70);
carrier.display.println("Waiting For");
carrier.display.setCursor(5, 110);
carrier.display.println("Connection...");
delay(500);
}

}

void loop() {
  static unsigned long timer=millis();
  static unsigned long timerscreen=millis();

  if (millis()-timer >= 4000){
    timer = millis();
    ArduinoCloud.update();
    // Your code here

    temperature = carrier.Env.readTemperature();
    humidity = carrier.Env.readHumidity();

  }

  if (temperature < 24 && temperature > 18){

    carrier.leds.setPixelColor(0, 0, 255, 0);
    carrier.leds.setPixelColor(1, 0, 255, 0);
    carrier.leds.setPixelColor(2, 0, 255, 0);
    carrier.leds.setPixelColor(3, 0, 255, 0);
    carrier.leds.setPixelColor(4, 0, 255, 0);
    carrier.leds.show();

  }

  if (temperature >= 24){

    carrier.leds.setPixelColor(0, 255, 0, 0);
    carrier.leds.setPixelColor(1, 255, 0, 0);
    carrier.leds.setPixelColor(2, 255, 0, 0);
    carrier.leds.setPixelColor(3, 255, 0, 0);
    carrier.leds.setPixelColor(4, 255, 0, 0);
    carrier.leds.show();

  }

  if (temperature <= 18){
```

```
carrier. leds. setPixelColor(0, 0, 0, 255);
carrier. leds. setPixelColor(1, 0, 0, 255);
carrier. leds. setPixelColor(2, 0, 0, 255);
carrier. leds. setPixelColor(3, 0, 0, 255);
carrier. leds. setPixelColor(4, 0, 0, 255);
carrier. leds. show();

}

if (millis()-timerscreen > 0 && millis()-timerscreen <= 100 ){
  carrier. display. fillScreen(ST7735_BLACK);

}

if (millis()-timerscreen > 100 && millis()-timerscreen <= 2000 ){

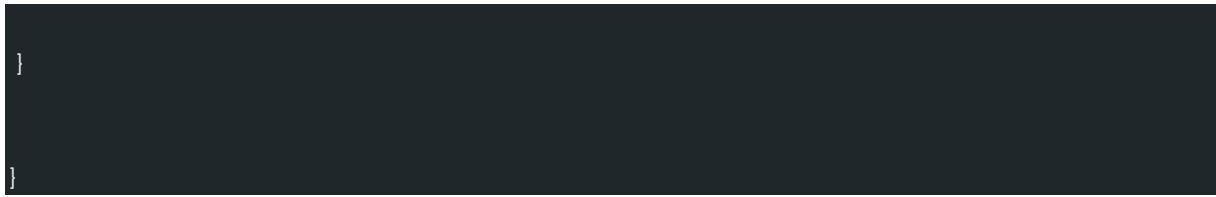
  carrier. display. setCursor(20, 60);
  carrier. display. setTextColor(ST77XX_WHITE);
  carrier. display. setTextSize(3);
  carrier. display. println("Limerick");
  carrier. display. setCursor(20, 90);
  carrier. display. println("Temperature: ");
  carrier. display. setCursor(60, 130);
  carrier. display. println(temperature);
  carrier. display. setCursor(50, 160);
  carrier. display. println("Celsius");
}

if (millis()-timerscreen > 2000 && millis()-timerscreen <= 2100 ){
  carrier. display. fillScreen(ST7735_BLACK);
}

if (millis()-timerscreen > 2100 && millis()-timerscreen <= 4000 ){

  carrier. display. setCursor(20, 60);
  carrier. display. setTextColor(ST77XX_WHITE);
  carrier. display. setTextSize(3);
  carrier. display. println("Limerick");
  carrier. display. setCursor(20, 90);
  carrier. display. println("Humidity: ");
  carrier. display. setCursor(60, 130);
  carrier. display. println(humidity);
  carrier. display. setCursor(50, 160);
  carrier. display. println("%");
}

if (millis()-timerscreen > 4000 ){
  timerscreen = millis();
}
```



29. Now we need the code to be loaded on our device. To do this, click the icon with the green arrow.

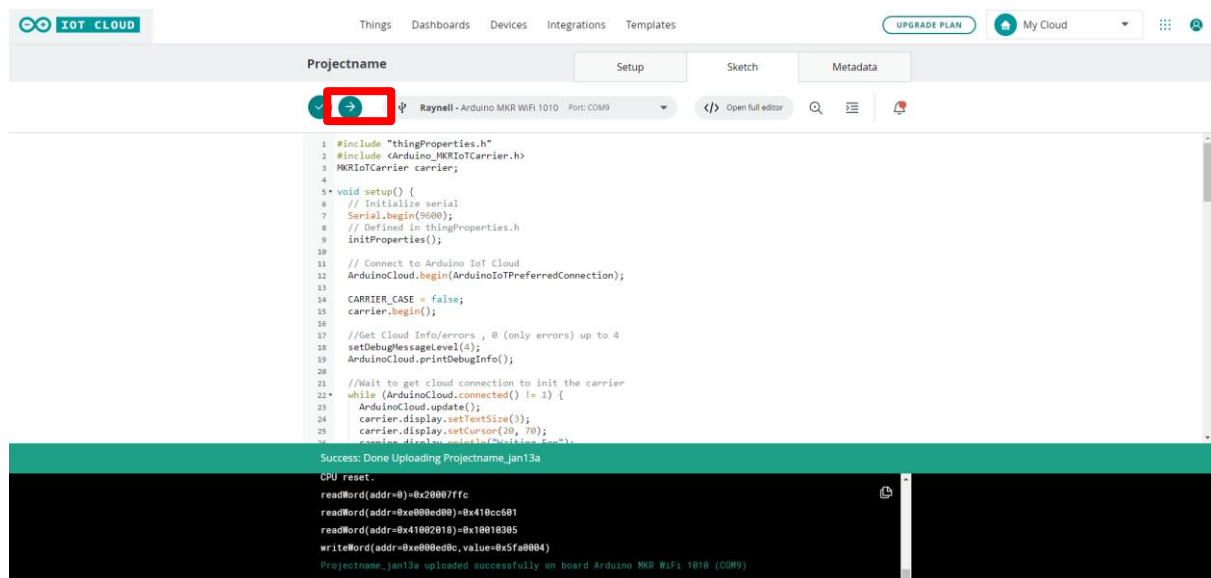


Fig. 80: Arduino web, IoT Cloud, Upload sketch

30. With the code loaded, and our Arduino working. We go to our dashboard and it should already be showing our data.

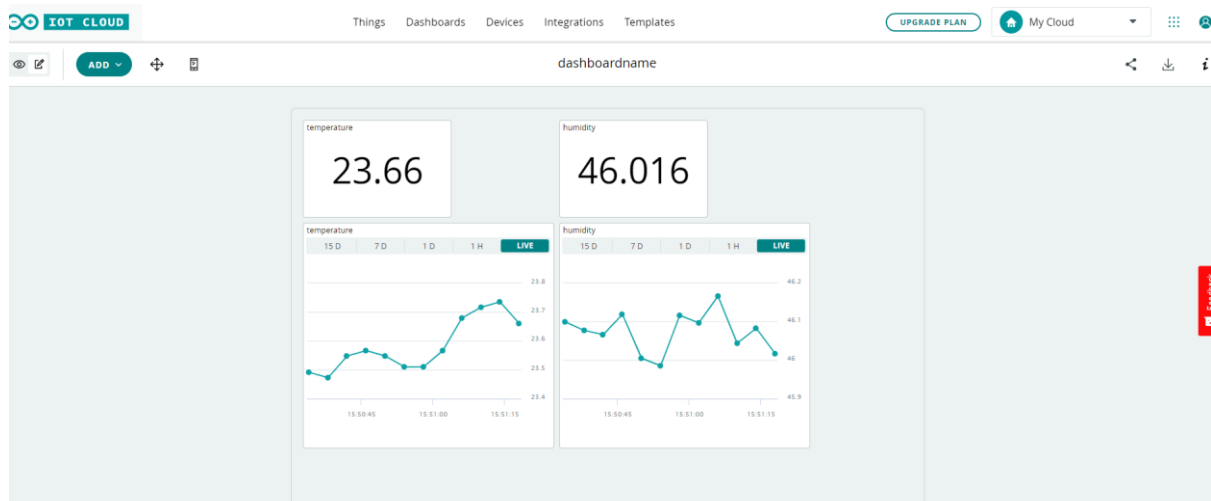


Fig. 80: Arduino web, IoT Cloud, Smart sensor working

31. Congratulations! They have made their dashboard. Now to experiment with the device, place hot and cold elements and check what happens.

1.4.1 Code details

The above code has some additional considerations to make the user experience in the workshop more entertaining.

I will explain part by part what the code does, so it is also free to experiment.

1. Automatic libraries

```
#include "thingProperties.h"
#include <Arduino_MKRIoTCarrier.h>
MKRIoTCarrier carrier;
```

Arduino automatically creates the libraries necessary for the operation of the device. There is nothing to modify here. Some sensors use their own libraries, and for their use you should look in their documentation which would be necessary.

2. Void Setup

```
3. void setup() {
4.   // Initialize serial
5.   Serial.begin(9600);
6.   // Defined in thingProperties.h
7.   initProperties();
8.
9.   // Connect to Arduino IoT Cloud
10.  ArduinoCloud.begin(ArduinoIoTPreferredConnection);
11.
12.  CARRIER_CASE = false;
13.  carrier.begin();
14.
15.  //Get Cloud Info/errors , 0 (only errors) up to 4
16.  setDebugMessageLevel(4);
17.  ArduinoCloud.printDebugInfo();
18.
19.  //Wait to get cloud connection to init the carrier
20.  while (ArduinoCloud.connected() != 1) {
21.    ArduinoCloud.update();
22.    carrier.display.setTextSize(3);
23.    carrier.display.setCursor(20, 70);
24.    carrier.display.println("Waiting For");
25.    carrier.display.setCursor(5, 110);
26.    carrier.display.println("Connection...");
27.    delay(500);
28.  }
29.
30.}
```


Void setup is a function that Arduino performs only when starting its operation. In this code Arduino configures the internet connection and tries to connect to the network that we give it.

3. Void Loop

```
void loop() {  
  static unsigned long timer=millis();  
  static unsigned long timerscreen=millis();  
  
  if (millis()-timer >= 4000){  
    timer = millis();  
    ArduinoCloud.update();  
    // Your code here  
  
    temperature = carrier.Env.readTemperature();  
    humidity = carrier.Env.readHumidity();  
  
  }  
  
  if (temperature < 24 && temperature > 18){  
  
    carrier.leds.setPixelColor(0, 0, 255, 0);  
    carrier.leds.setPixelColor(1, 0, 255, 0);  
    carrier.leds.setPixelColor(2, 0, 255, 0);  
    carrier.leds.setPixelColor(3, 0, 255, 0);  
    carrier.leds.setPixelColor(4, 0, 255, 0);  
    carrier.leds.show();  
  
  }  
  
  if (temperature >= 24){  
  
    carrier.leds.setPixelColor(0, 255, 0, 0);  
    carrier.leds.setPixelColor(1, 255, 0, 0);  
    carrier.leds.setPixelColor(2, 255, 0, 0);  
    carrier.leds.setPixelColor(3, 255, 0, 0);  
    carrier.leds.setPixelColor(4, 255, 0, 0);  
    carrier.leds.show();  
  
  }  
  
  if (temperature <= 18){  
  
    carrier.leds.setPixelColor(0, 0, 0, 255);
```

```
carrier. leds. setPixelColor(1, 0, 0, 255);
carrier. leds. setPixelColor(2, 0, 0, 255);
carrier. leds. setPixelColor(3, 0, 0, 255);
carrier. leds. setPixelColor(4, 0, 0, 255);
carrier. leds. show();

}
if (millis()-timerscreen > 0 && millis()-timerscreen <= 100 ){
carrier. display. fillScreen(ST7735_BLACK);

}
if (millis()-timerscreen > 100 && millis()-timerscreen <= 2000 ){

carrier. display. setCursor(20, 60);
carrier. display. setTextColor(ST77XX_WHITE);
carrier. display. setTextSize(3);
carrier. display. println("Limerick");
carrier. display. setCursor(20, 90);
carrier. display. println("Temperature: ");
carrier. display. setCursor(60, 130);
carrier. display. println(temperature);
carrier. display. setCursor(50, 160);
carrier. display. println("Celsius");
}

if (millis()-timerscreen > 2000 && millis()-timerscreen <= 2100 ){
carrier. display. fillScreen(ST7735_BLACK);
}
if (millis()-timerscreen > 2100 && millis()-timerscreen <= 4000 ){

carrier. display. setCursor(20, 60);
carrier. display. setTextColor(ST77XX_WHITE);
carrier. display. setTextSize(3);
carrier. display. println("Limerick");
carrier. display. setCursor(20, 90);
carrier. display. println("Humidity: ");
carrier. display. setCursor(60, 130);
carrier. display. println(humidity);
carrier. display. setCursor(50, 160);
carrier. display. println("%");
}

if (millis()-timerscreen > 4000 ){
timerscreen = millis();

}
}
```

```
}
```

This section is a little more complex, Void loop is a function that works continuously. It repeats again and again what is inside it.

In this code, the first thing you do is define certain time variables that will be used to control the visualization of data and the sending of them. All times are in milliseconds, which means that the value 1000 equals 1000 milliseconds or 1 second.

```
static unsigned long timer=millis();
static unsigned long timerscreen=millis();

if (millis()-timer >= 4000){
  timer = millis();
  ArduinoCloud.update();
  // Your code here

  temperature = carrier.Env.readTemperature();
  humidity = carrier.Env.readHumidity();
}
```

Additionally, the code in this section reads the sensor data every 4 seconds and updates this data in the Arduino Cloud.

The next section is the temperature alert control.

```
if (temperature < 24 && temperature > 18){

  carrier.leds.setPixelColor(0, 0, 255, 0);
  carrier.leds.setPixelColor(1, 0, 255, 0);
  carrier.leds.setPixelColor(2, 0, 255, 0);
  carrier.leds.setPixelColor(3, 0, 255, 0);
  carrier.leds.setPixelColor(4, 0, 255, 0);
  carrier.leds.show();

}

if (temperature >= 24){

  carrier.leds.setPixelColor(0, 255, 0, 0);
  carrier.leds.setPixelColor(1, 255, 0, 0);
  carrier.leds.setPixelColor(2, 255, 0, 0);
  carrier.leds.setPixelColor(3, 255, 0, 0);
  carrier.leds.setPixelColor(4, 255, 0, 0);
}
```

```
carrier. leds. show();  
  
}  
  
if (temperature <= 18){  
  
carrier. leds. setPixelColor(0, 0, 0, 255);  
carrier. leds. setPixelColor(1, 0, 0, 255);  
carrier. leds. setPixelColor(2, 0, 0, 255);  
carrier. leds. setPixelColor(3, 0, 0, 255);  
carrier. leds. setPixelColor(4, 0, 0, 255);  
carrier. leds. show();  
  
}
```

In this section we have written three conditions for which we must be given a color alert.

First conditions: If the temperature is less than 24 and greater than 18 degrees Celsius (Proposed Thermal Comfort Zone), the device will show green light, indicating that the space is in good thermal condition.



Fig. 81: Arduino sensor, Green temperature condition

If the temperature is greater than or equal to 24 degrees Celsius (above the proposed thermal comfort zone), the device will show us the red color. Indicating that we are in an overheated space.



Fig. 82: Arduino sensor, Red temperature condition

Third condition: If the temperature is less than or equal to 18 degrees Celsius (Under the proposed thermal comfort temperature), the device will show us the blue color. Indicating that we are in a cold space.



Fig. 82: Arduino sensor, Blue temperature condition

The rest of the Code is the necessary settings to display the data on the screen, which will do so for two seconds for temperature and another 2 seconds for humidity.

References

O'Flaherty, C. (2023). W2-D2: Approved List of Sensor Devices to Match the Data Requirements under the SRI and Enable Smart Building Monitoring across the Living Lab including through DIY Kits. University of Limerick. Report. <https://doi.org/10.34961/researchrepository-ul.22770353.v1>

Trejo Rangel, M. A., Lyes, M., Nuñez, D., Fitzgerald, H., & Kinsella, S. (2022). WP1-D1 Plan for Stakeholder Engagement and SMARTLAB Journeys Including SMARTLAB Calendar of Events. University of Limerick. <https://doi.org/https://doi.org/10.34961/researchrepository-ul.22202677.v1>

Trejo-Rangel, M. A., Lyes, M., Fitzgerald, H., & Kinsella, S. (2023). WP1-D3: Barriers to Improving the Smartness of Buildings and Associated Barriers to the Deployment of Smart Technologies and Services. University of Limerick. <https://doi.org/https://doi.org/10.34961/researchrepository-ul.22561156.v1>

Wolf, M., & McQuitty, S. (2011). Understanding the do-it-yourself consumer: DIY motivations and outcomes. *AMS Review*, 1(3–4), 154–170. <https://doi.org/10.1007/s13162-011-0021-2>