

# Developing a room occupancy monitor using M5stack and AWS

## Objectives:

In this lab you will:

- Learn how to setup the M5Stack
- Register a thing in AWS IoT Core
- Publish MQTT Messages to the IoT Core
- Receive MQTT Messages from the IoT Core Broker

## Scenario:

In this lab we setup a scenario where we use a Time of Flight sensor with M5Stack to detect human motion while a person enters a room and maintain number of people who are present in the room and send this data to the AWS services using MQTT.

Since the ToF sensor does not give any information of the direction of the motion detected, to emulate the action of decrementing the counter storing the number of people inside the room when someone goes out of the room, we use MQTT subscription. We publish MQTT messages over a topic which the M5Stack is subscribed to and on receiving this trigger we reduce the counter storing the number of people in the room by 1.

Thus, using MQTT publish subscribe functionality we emulate a scenario to calculate the number of people inside a room.

## Requirements:

- **An AWS account**  
<https://aws.amazon.com/>
- **M5 Stack device**  
<https://devices.amazonaws.com/detail/a3G0h000007djMLEAY/M5Stack-Core2-ESP32-IoT-Development-Kit-for-AWS>  
or  
optionally any IOT device that can communicate over MQTT protocol and send data to AWS
- **Time-of-Flight(ToF) Sensor**  
<https://m5stack.hackster.io/products/time-of-flight-tof-v153l0x-laser-ranging-unit-mcp4725>

# CHAPTER 1: Setting up the device

## Pre-requisites for configuring the M5Stack

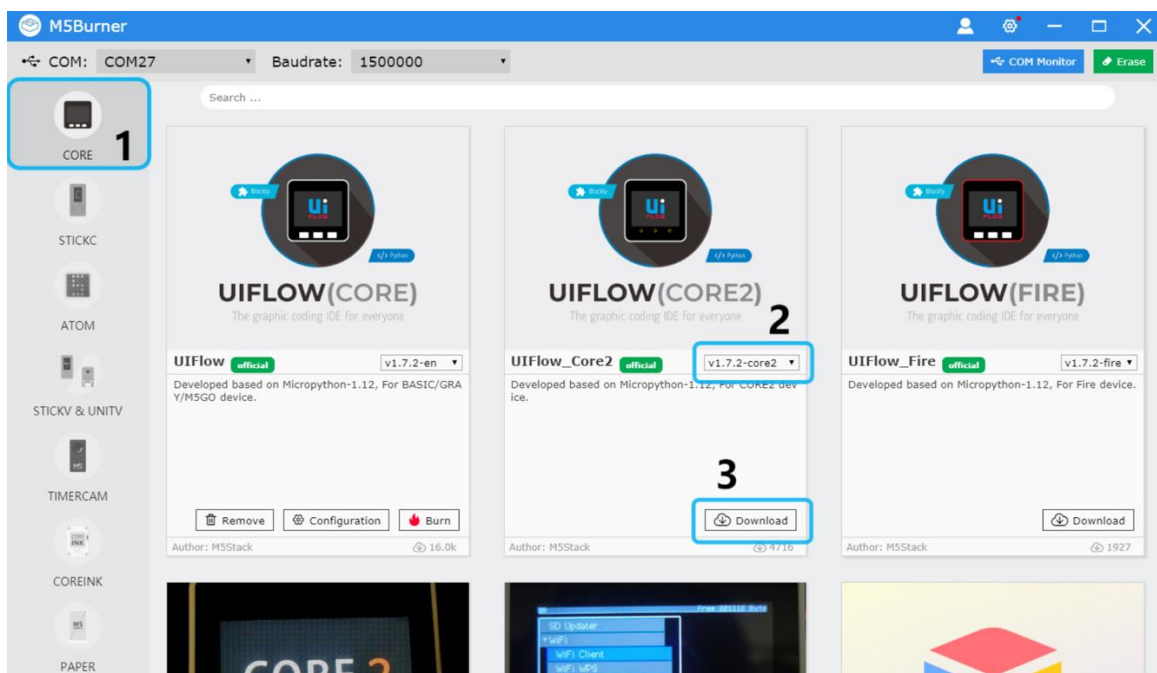
1. Download and Install the CP210x USB to UART Bridge VPC Driver  
Steps to install the driver: [https://m5stack.github.io/UIFlow\\_doc/en/en/base/Update.html](https://m5stack.github.io/UIFlow_doc/en/en/base/Update.html)

Links to download the required software:

- a. <https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers>
  - b. <https://shop.m5stack.com/pages/download>
2. Download and Install the M5 Burner: <https://shop.m5stack.com/pages/download>

## Burning the UIFlow Firmware onto the M5Stack

1.
  - Double-click to open the M5Burner
  - ① Select the corresponding device type in the left menu
  - ② Select the firmware version you want
  - ③ Click the download button to download.



In our case we will be downloading the UIFlow\_Core2 latest version to burn onto the M5Stack

2. Then connect the M5 device to the computer through the Type-C cable, select the corresponding COM port (For MAC devices the port may read something like

/dev/tty.SLAB\_USBtoUART), the baud rate can use the default configuration in M5Burner,

- ④ Click Erase on the right to erase Flash, close the current page when finished.
  - ⑤ Click "Burn" to start burning, you can input WIFI configuration information during burning.
3. When the burning log prompts **Burn Successfully**, it means that the firmware has been burned.

## Configuring the WIFI on the M5Stack

1. Click the power button on the left side of the device to turn it on. The UIFlow Logo appears on the screen.



2. After entering the main page, press the **Setup** button on the screen.



3. In the **WiFi** option, press the start button of the config Wi-Fi by web option, and the device will automatically restart, where selected Wi-Fi is the last connected WiFi.



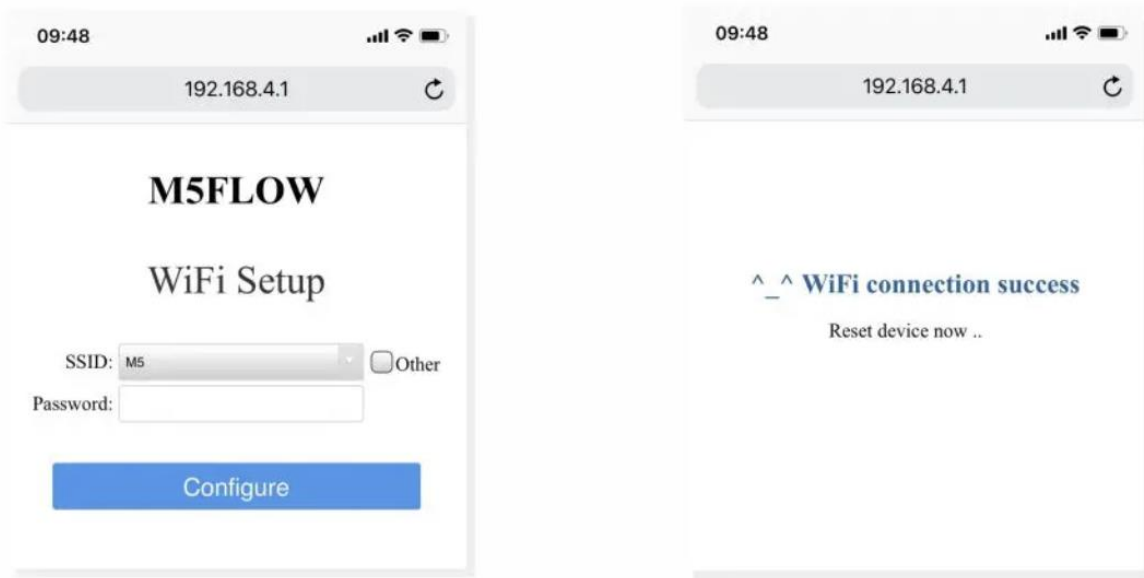
4. After the device jumps, the WiFi Config page will be displayed.



5. Follow the prompts to connect to the SSID hotspot through the WiFi of the mobile phone or computer.
  1. Open the browser to visit **192.168.4.1** and enter the WiFi information in the pop-up page to configure the network successfully.
  2. After the configuration is successful, the device will automatically restart. And enter the programming mode.



*Connect to the SSID shown on the M5Stack through your computer or Mobile*

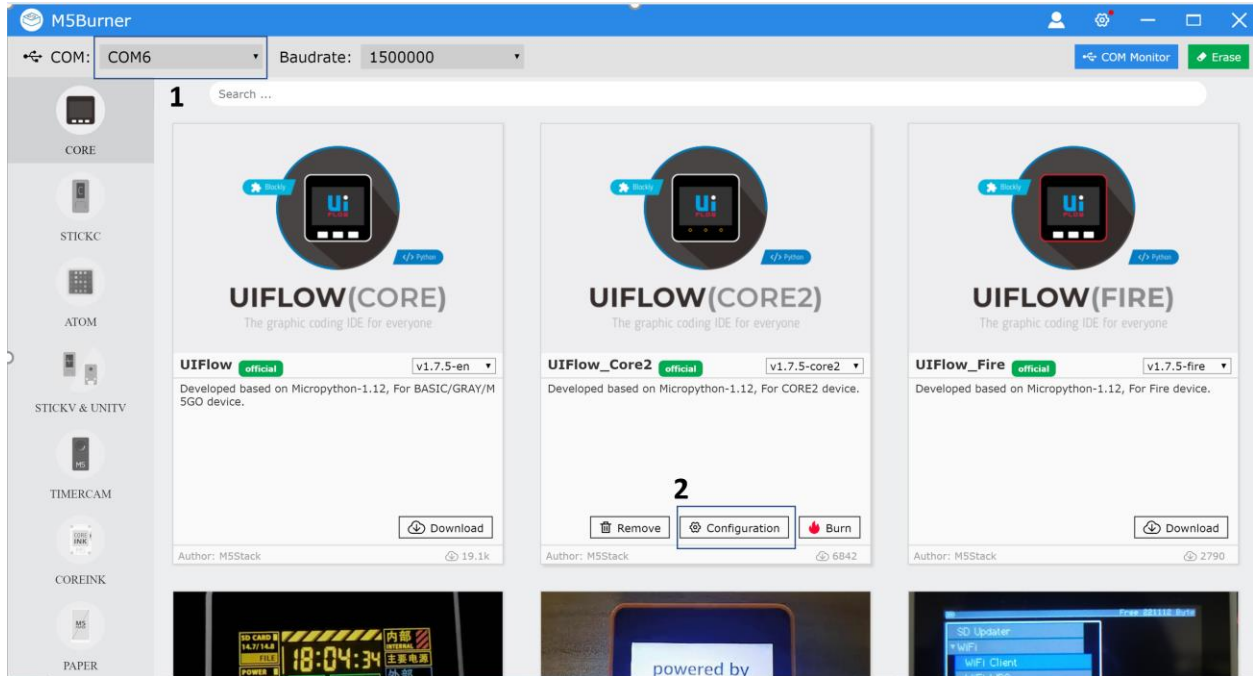


*Enter the SSID (Name of the WIFI) and password of the Wifi you want to connect your M5Stack to. (Note: The M5Stack may not allow you to connect to public WIFI's such as EduRoam or Clemson Guest, or even any 5Ghz networks. You may connect to your mobile hotspot or any private 2.4 Ghz Wifi such as your home WIFI).*

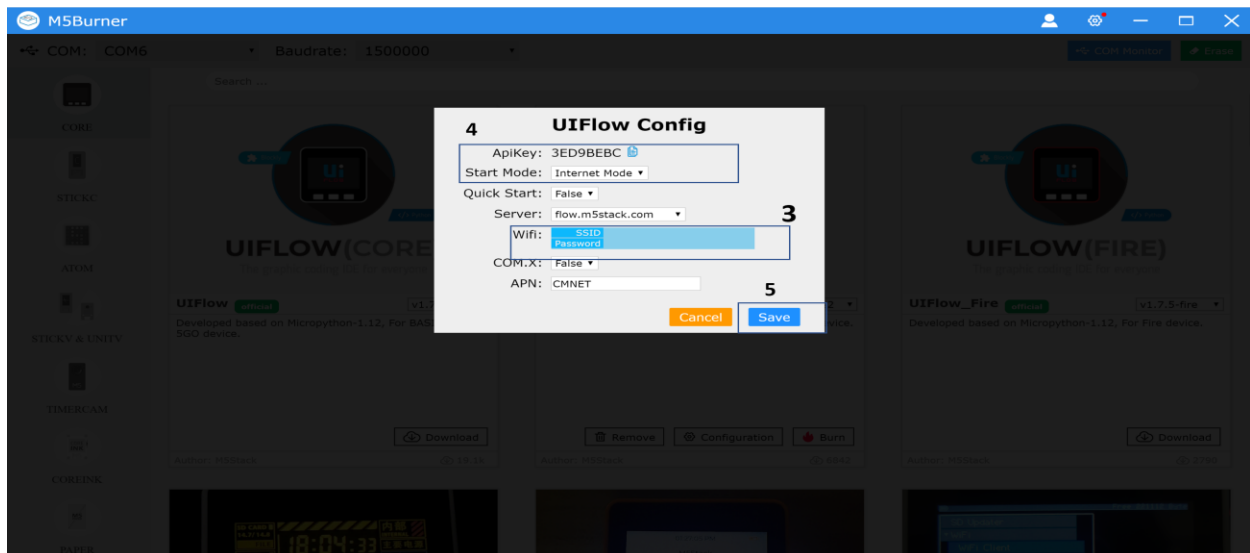
## **Alternate way of configuring the WIFI on the M5Stack using the M5 Burner**

Connect M5Stack to the computer and open the M5 Burner:

1. In the M5Burner on the top left of the application first ensure whether the right port is selected.
2. Then select the configuration button for the UIFLOW(Core 2) firmware option which has been installed on to the M5Stack in the previous step.



3. In the given text boxes, enter the Wifi SSD and Password.
4. Select the Start Mode as Internet Mode and Copy the API Key for later use.
5. After entering the right details enter the save button. Wait for a few seconds while the configuration is written to the M5Stack.

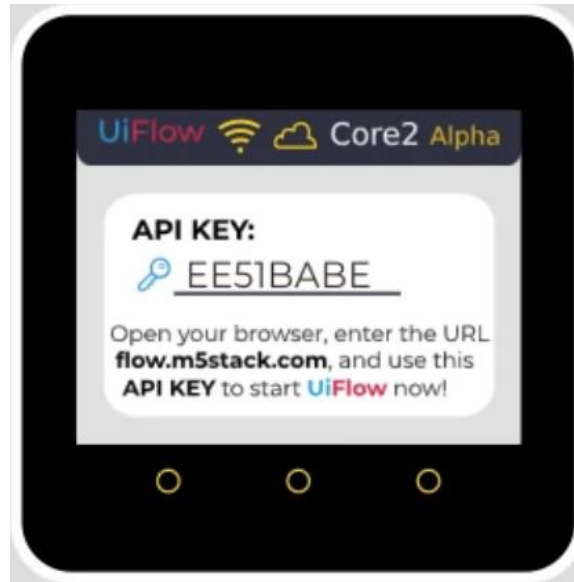


*Enter the SSID (Name of the WIFI) and password of the Wifi you want to connect your M5Stack to. (Note: The M5Stack may not allow you to connect to public WIFI's such as EduRoam or*

*Clemson Guest, or even any 5Ghz networks. You may connect to your mobile hotspot or any private 2.4 Ghz Wifi such as your home WIFI).*

## Connecting the M5Stack to UIFlow

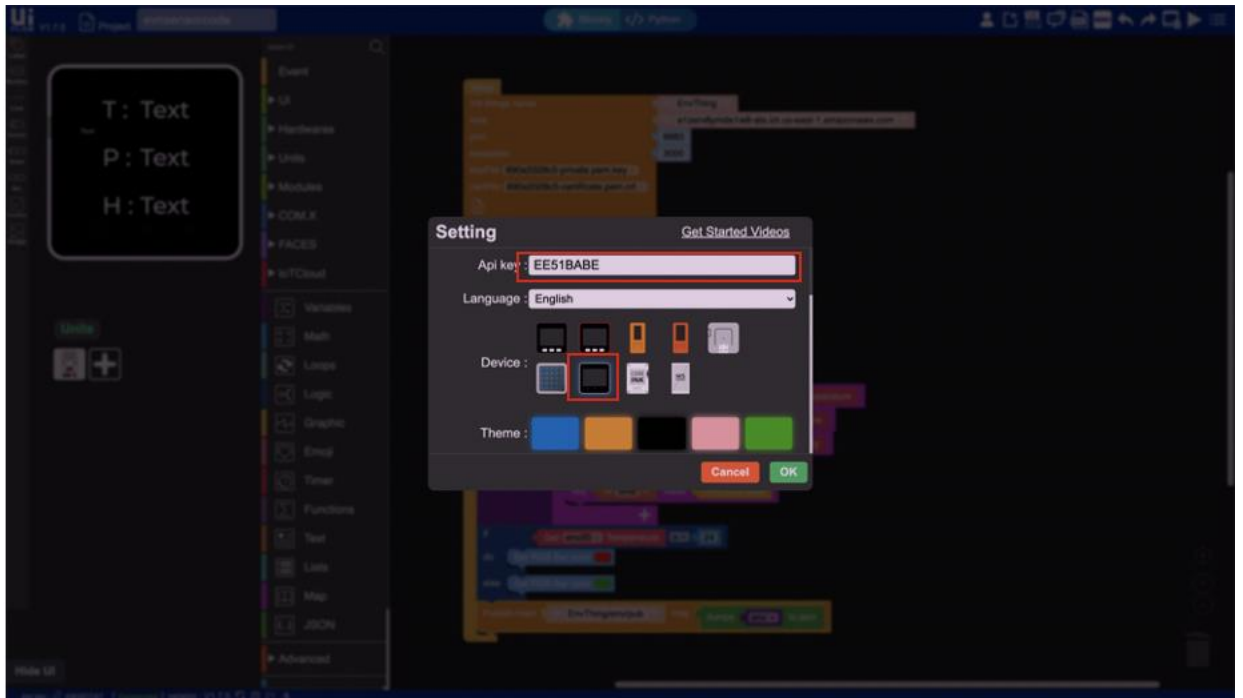
1. The device automatically restarts on a successful WIFI Connection and display the following screen:



Note: Network programming mode is a docking mode between M5 device and UIFlow web programming platform. The screen will show the current network connection status of the device. When the indicator is green, it means that you can receive program push at any time. Under default situation, after the first successful WiFi network configuration, the device will automatically restart and enter the network programming mode

2. Follow the instructions displayed on the device and go to [flow.m5stack.com](https://flow.m5stack.com) on your computer
3. Type in the matching API Key from the screen or if you have previously copied from the M5Burner into UIFlow and select the device type.





**Note:** API KEY is the communication credential for M5 devices when using UIFlow web programming. By configuring the corresponding API KEY on the UIFlow side, the program can be pushed for the specific device. The user needs to visit [flow.m5stack.com](https://flow.m5stack.com) in the computer web browser to enter the UIFlow programming page. Click the setting button in the menu bar at the upper right corner of the page, enter the API Key on the corresponding device, select the hardware used, click OK to save and wait till it prompts successfully connecting or displays connected in the bottom left on the status bar.

### References links for setting up the M5Stack

1. [https://docs.m5stack.com/en/quick\\_start/core2/m5stack\\_core2\\_get\\_started\\_MicroPython](https://docs.m5stack.com/en/quick_start/core2/m5stack_core2_get_started_MicroPython)

## CHAPTER 2: SETTING UP THE DEVICE IN AWS

### Registering a Thing in AWS

1. After you create your account in AWS, login to the console and navigate to the AWS IOT Core Dashboard.
2. In the [AWS IoT console](#), in the side navigation pane, choose **Manage**, and then choose **Things**.
3. If **You don't have any things yet** dialog box is displayed, choose **Register a thing**. Otherwise, choose **Create**.
4. On the **Creating AWS IoT things** page, choose **Create a single thing**.
5. On the **Add your device to the device registry** page, enter a name for your IoT thing (for this example enter the following as your device name, **“TOFThing”**), and then choose **Next**.  
You can't change the name of a thing after you create it. To change a thing's name, you must create a new thing, give it the new name, and then delete the old thing.
6. On the **Add a certificate for your thing** page, choose **Create certificate**.
7. Choose the **Download** links to download the certificate for the thing with the extension **\*\*\*\*\*.cert.pem**, and the private key. We will be using these certificates later.

**Important: This is the only time you can download your certificate and private key.**

8. Choose **Activate**.
9. Choose **Attach a policy**.
10. For now, we will not attach any policy and Choose **Register a thing**.

**Reference link for Registering a thing**

1. <https://docs.aws.amazon.com/iot/latest/developerguide/iot-moisture-create-thing.html>

## Creating a policy

In the project we will create a policy that allows us to connect with AWS and publish and subscribe to messages.

1. In the [AWS IoT console](#), in the side navigation pane, choose **Secure**, and then choose **Policies**.
2. Enter the following name to your policy “M5ToAWSPolicy”
3. Since we will be writing our own policy in the **Add Statements** section Choose **Advanced Mode** and copy/paste the code give below.
4. Replace **<Region-Name>** with the name of the region in which you are working such as **“us-east-1”**
5. Replace **<Account-No>** with your AWS account number which can be found on the Top right of the AWS Console by clicking on your account name
6. Replace **<ThingName>** with the name you gave your thing while registering it in the previous step (In our case the name is **TOFThing**).

Code to Copy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:Connect",
      "Resource": "arn:aws:iot:<Region-Name>:<Account-No>:client/<ThingName>"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Subscribe",
      "Resource": "arn:aws:iot: :<Region-Name>:<Account-No>:topicfilter/<ThingName>/sub"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Receive",
      "Resource": "arn:aws:iot: :<Region-Name>:<Account-No>:topic/<ThingName>/sub"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Publish",
```

```
"Resource": [
  "arn:aws:iot: <Region-Name>:<Account-No>:topic/<ThingName>/pub"
]
}
]
}
```

Example Code:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:Connect",
      "Resource": "arn:aws:iot:us-east-1:526762624946:client/TOFThing"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Subscribe",
      "Resource": "arn:aws:iot:us-east-1:526762624946:topicfilter/TOFThing/sub"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Receive",
      "Resource": "arn:aws:iot:us-east-1:526762624946:topic/TOFThing/sub"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Publish",
      "Resource": "arn:aws:iot:us-east-1:526762624946:topic/Room1/sensordata/pub"
    }
  ]
}
```

## Attaching the created Policy to the certificate

1. In the [AWS IoT console](#), in the side navigation pane, choose **Secure**, and then choose **Certificates**.
2. Click on the certificate you just created.
3. Then click on **Actions** and Click on **Attach Policy** from the drop-down menu.
4. Select the policy we just created and click **Attach**.

### References links about AWS Policies

1. Attach a policy to a client certificate:  
<https://docs.aws.amazon.com/iot/latest/developerguide/attach-to-cert.html>
2. Publish/Subscribe Policy Examples:  
<https://docs.aws.amazon.com/iot/latest/developerguide/pub-sub-policy.html>


### References

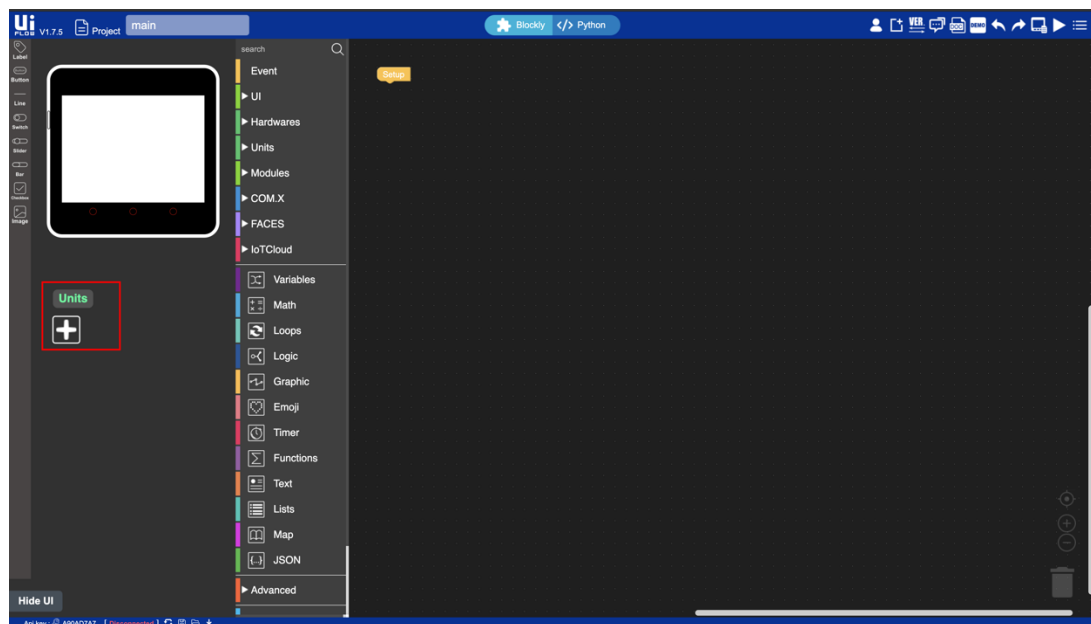
<https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers>  
<https://shop.m5stack.com/pages/download>

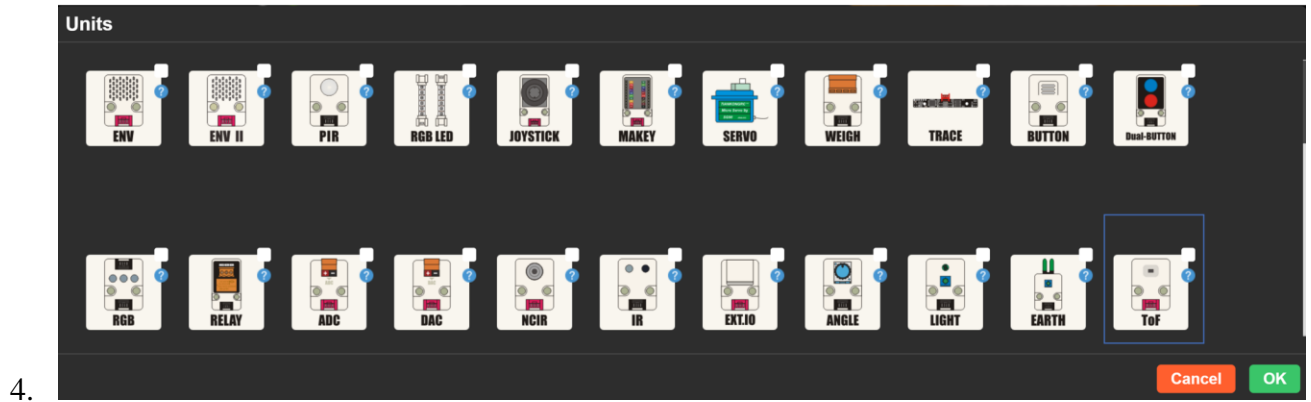
## Chapter 3: Programming and Publishing data to AWS

With the thing registered, certificates downloaded, and the relevant policies attached to the certificate we are now ready to program and publish data to AWS.

### Setting up the coding environment

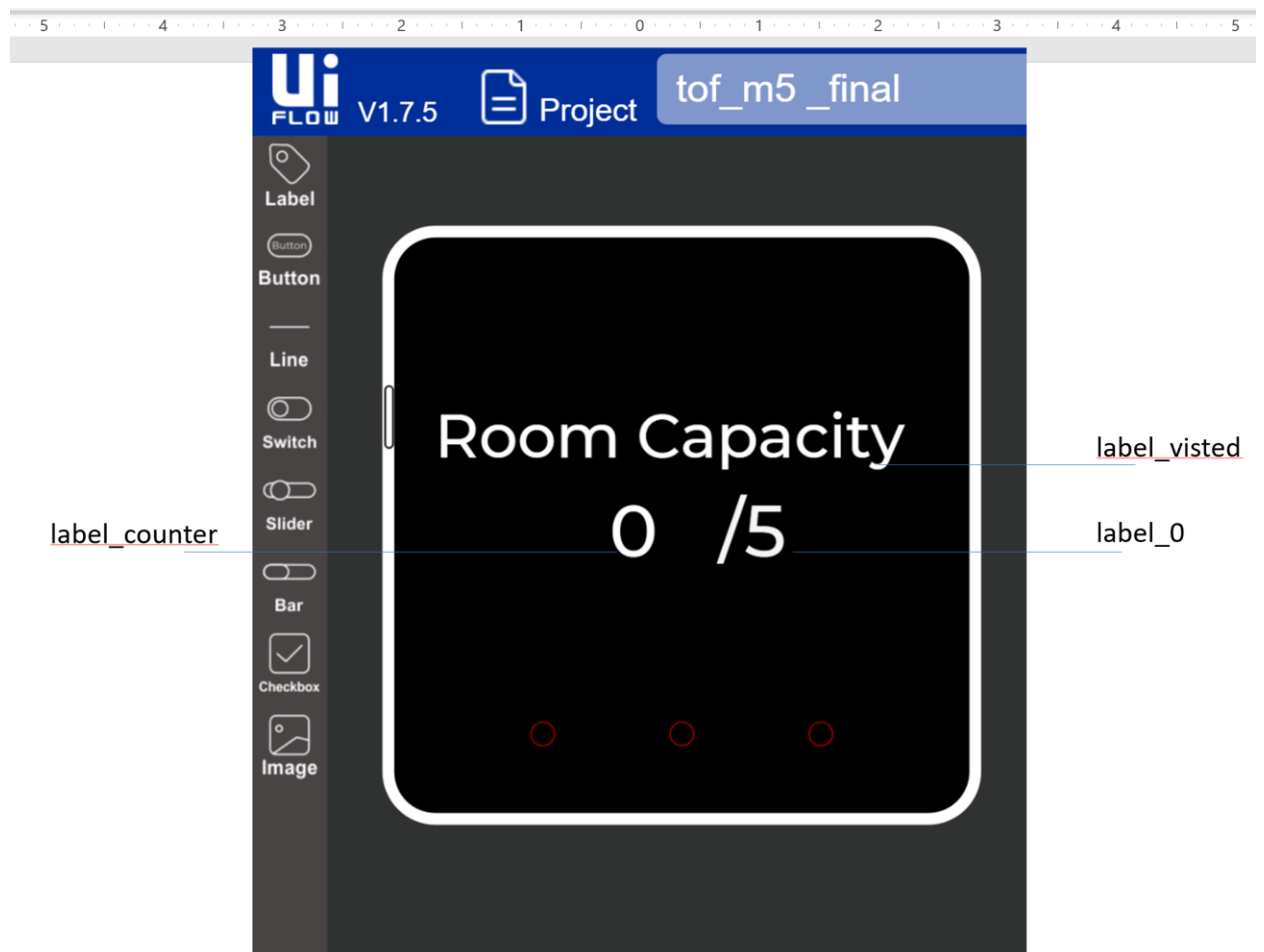
1. After connecting the M5Stack to UIFlow in chapter 1 we are now ready program, just hit the **refresh device status** button  in the status bar on the bottom to verify that the device is still connected.
2. Connect the ToF sensor to the Port A of the M5 Stack using the grove cable provided. (Port A is the port besides the charging port and it's the only Port the sensor will work on)
3. In the UI Flow screen click on the + sign below **Units**





### Setting up the layout for the assignment

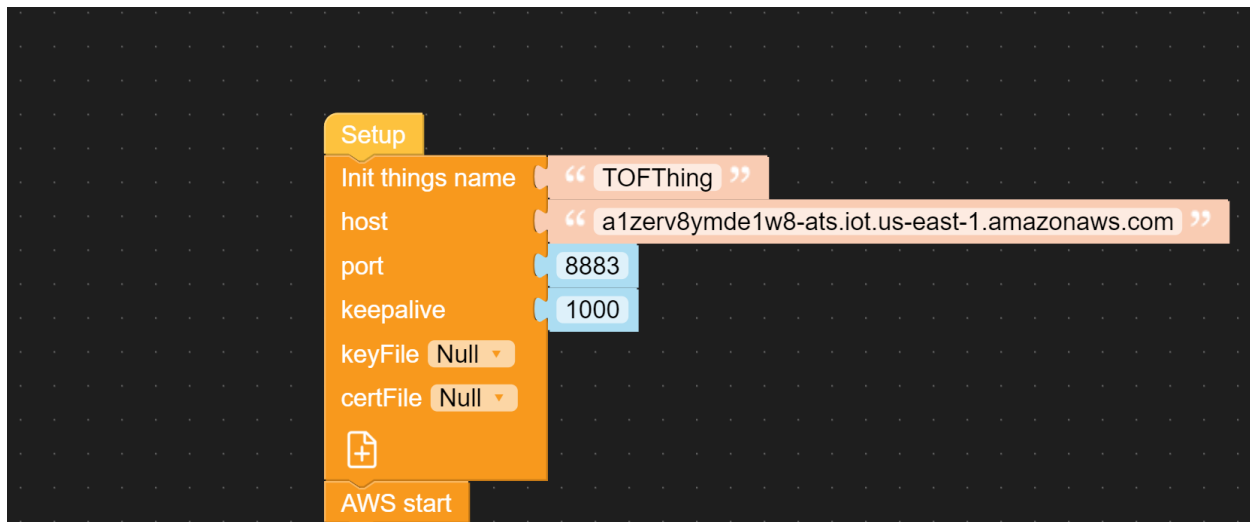
For the layout we have 3 labels to display 3 things on the screen.



Label	Description	Default Value
label_visited	Display the string room capacity	<b>Room Capacity</b>
label_counter	Display the number of people entering a room based on the motion detected by ToF	<b>0</b>
Label_0	Display maximum expected capacity inside the room which we can keep as 5 people.	<b>/5</b>

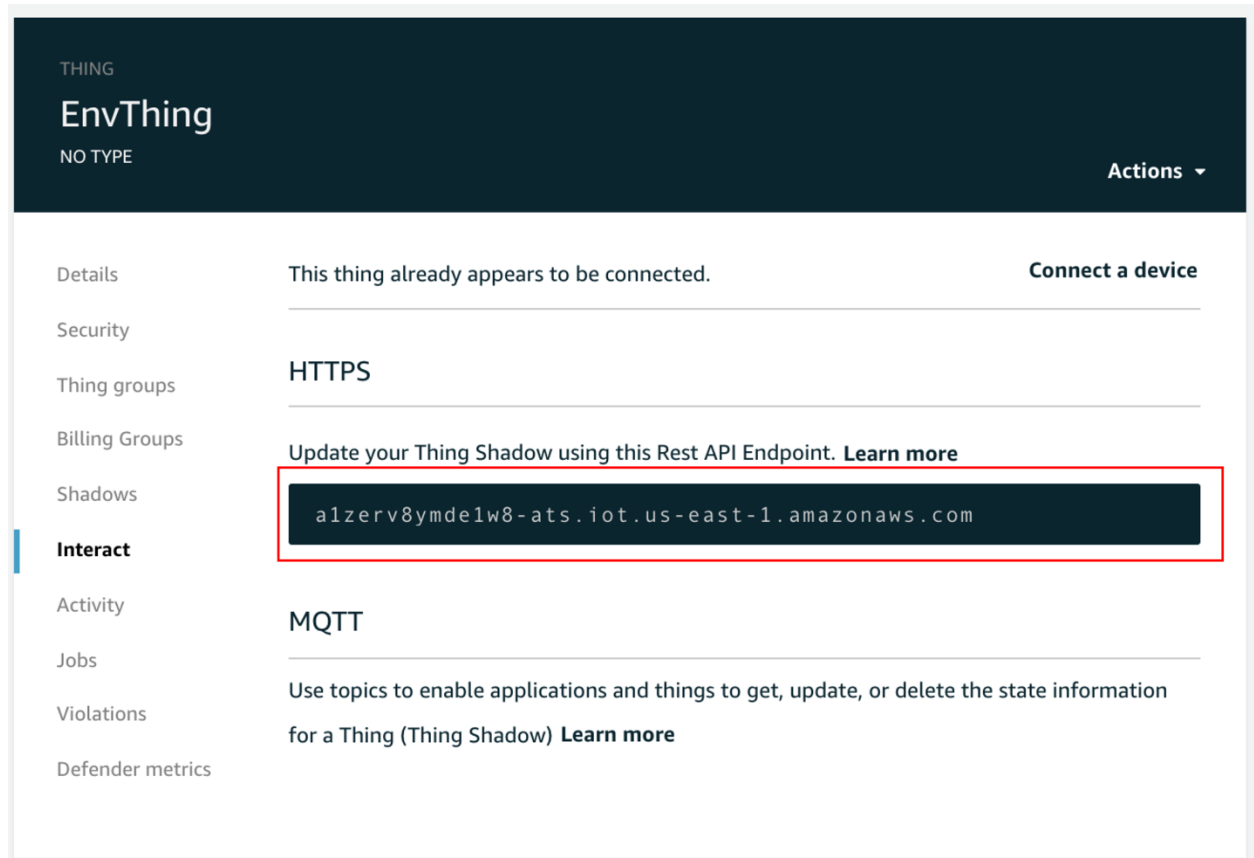
## Coding the solution

### 1. Setting up connection to AWS



- a. After setup add the AWS connection block and put in the following values as required:
  - i. **Init things name:** Add the name with which you had registered your thing in AWS. In our case the name was “TOFThing”
  - ii. **Host:** This is the endpoint to which you will be connecting to. This can be found by going to the AWS IoT Core dashboard.
    1. In the [AWS IoT console](#), in the side navigation pane, choose **Things**, and then click on the name of your thing (“TOFThing”).
    2. In the side navigation pane of your thing choose **Interact**.
    3. Copy and paste the endpoint given under HTTPS





- iii. **Port:** Put in 8883 as the port number. (8883 is the default port number for a secure mqtt connection)
- iv. **Keepalive:** This is the time for which the connection is to be kept alive. We took a large number like 300 so that the connection is active for a longer period of time.
- v. Hit the plus button to add files. Add the private key and certificate that we had downloaded while registering the thing.
- vi. **Keyfile:** From the drop down list select the added private key file.
- vii. **certFile:** From the drop down list select the added certificate key file.

b. Add the AWS Start block

## 2. Coding the business logic

There are total 3 main parts in the business logic:

- Block 1: Write logic for checking the distance reading of the ToF sensor and if its less than a threshold we declare that motion has been detected and increase the counter by 1.
- Block 2: Subscribe to an AWS topic and on receiving an MQTT message from the publisher reduce the counter value by 1.
- Block 3: Use the last touch button on the M5 Device to reset the counter.

Block 1:

For Block 1 let's set the following logic in the UI Flow:

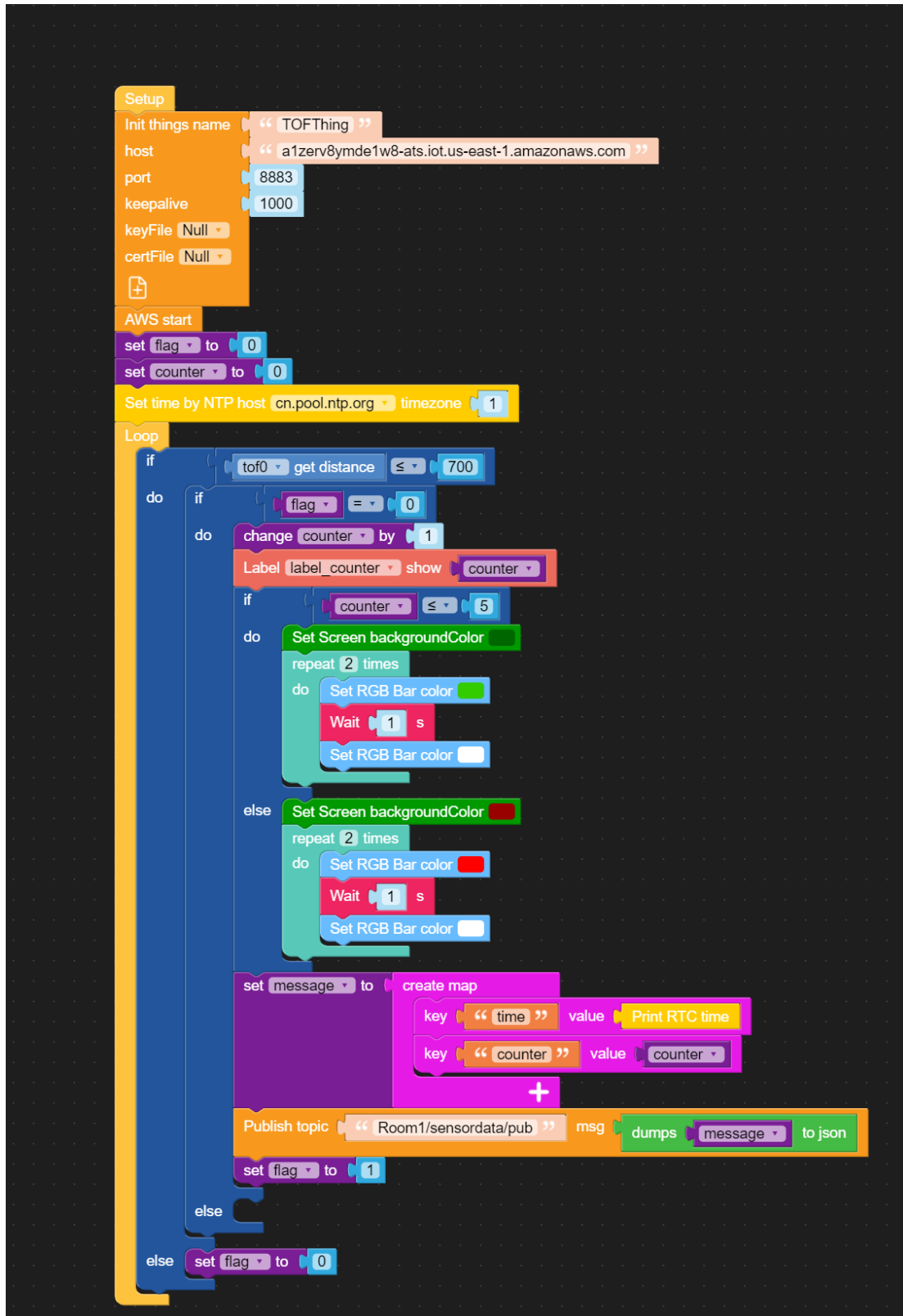
- Initialize variables counter and flag to 0. These variables are used to maintain counter value for number of people inside the room and flag is used to avoid multiple increment of the counter during a single event of motion detection.
- Set the RTC clock to UTC timezone to send in the MQTT messages to the IoT Core.
- The pseudo code for this block is as follows:

```
while True:
    if (distance_from_ToF < 700):
        if(flag == 0):
            // increase the counter by 1;
            counter= counter+1;
            //Set the label_counter to the new counter value.
            Show(label_counter,counter)
            if(label_counter<5):
                flash green led lights twice with a pause of 1 sec
                turn the screen background green.

            else:
                flash red led lights twice with a pause of 1 sec
                turn the screen background red.

            publish MQTT data of timestamp and counter
            flag=1
        else:
            flag=0
```

The blocky code for this part looks like the following in the UI Flow:



## Block 2:

Now for block 2 we want to simulate a functionality when someone walks outside a room and we decrease the counter holding the number of people inside the room by 1.

For this we use the MQTT subscription. We send a message over a topic to which the M5 stack device is subscribed to and once we receive that message the counter is reduced by 1.

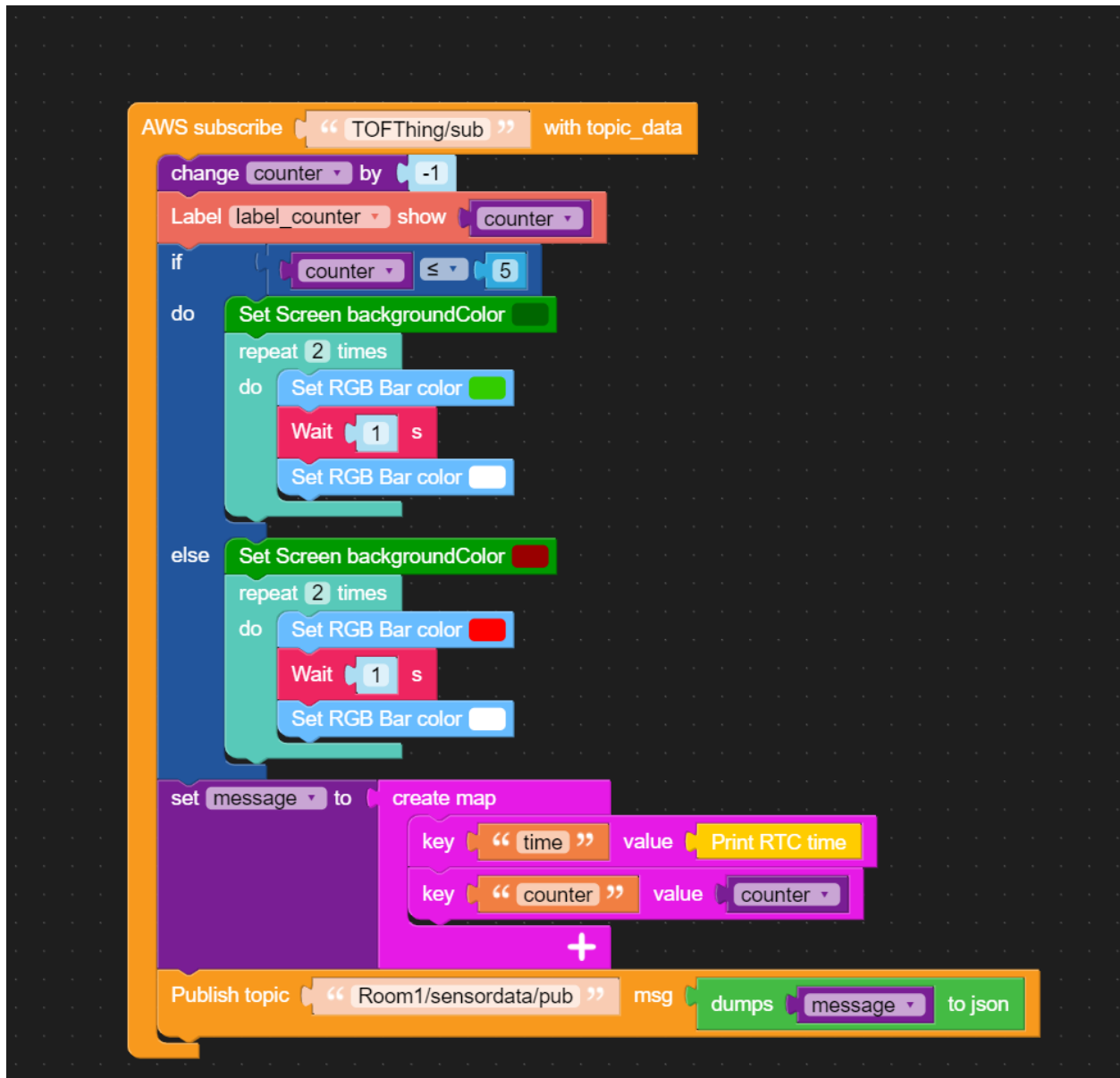
For this we make use of the MQTT subscribed block in the UI Flow blocky.

The pseudo code for this block would be like:

```
if(message received over the subscribed topic):
    counter=counter-1;
    Show(label_counter,counter-1)
    if(label_counter<5):
        flash green led lights twice with a pause of 1 sec
        turn the screen background green.
    else:
        flash red led lights twice with a pause of 1 sec
        turn the screen background red.

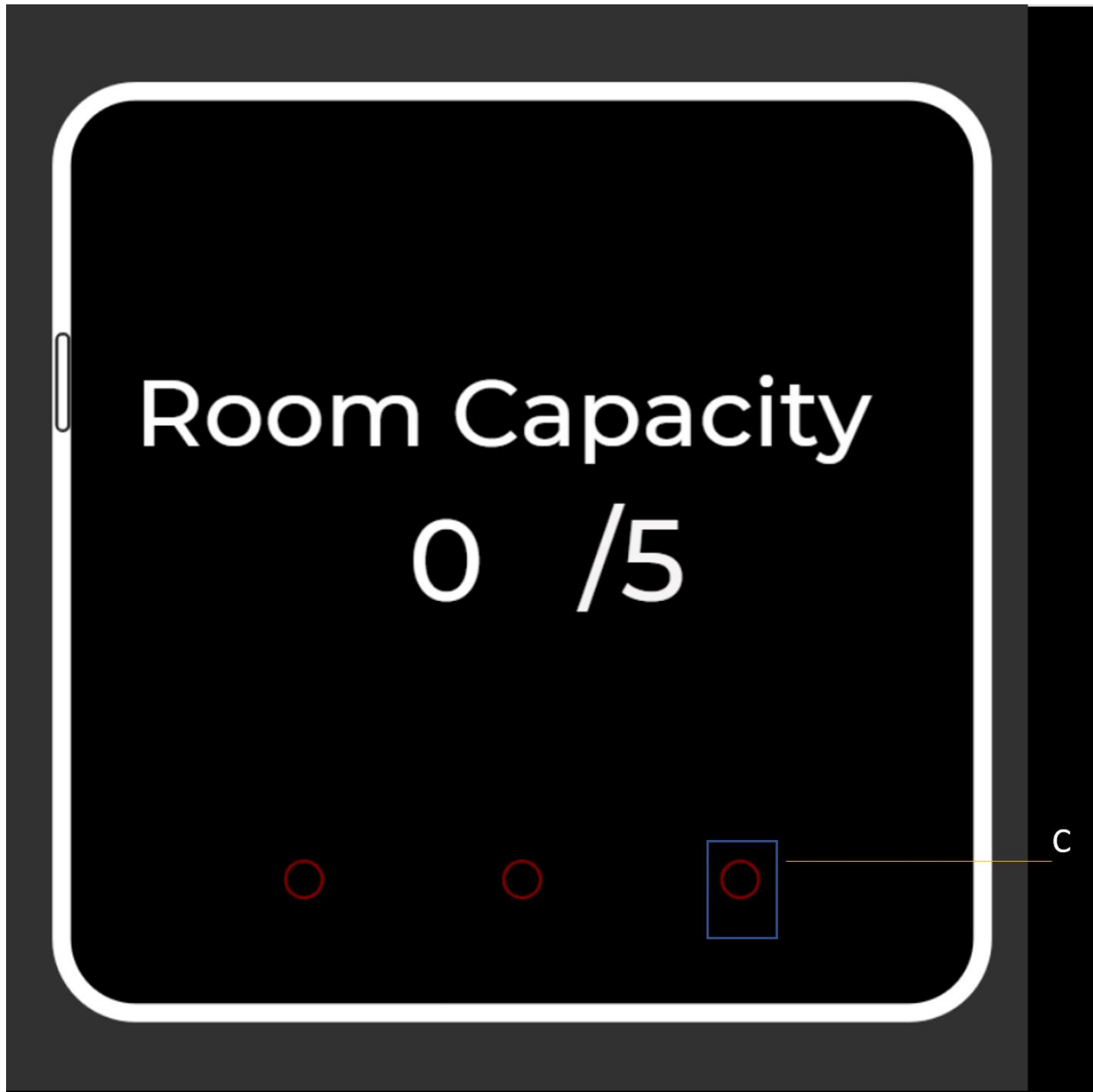
publish MQTT data of timestamp and new value counter
```

The Blockly code for this module would be like:



Block 3:

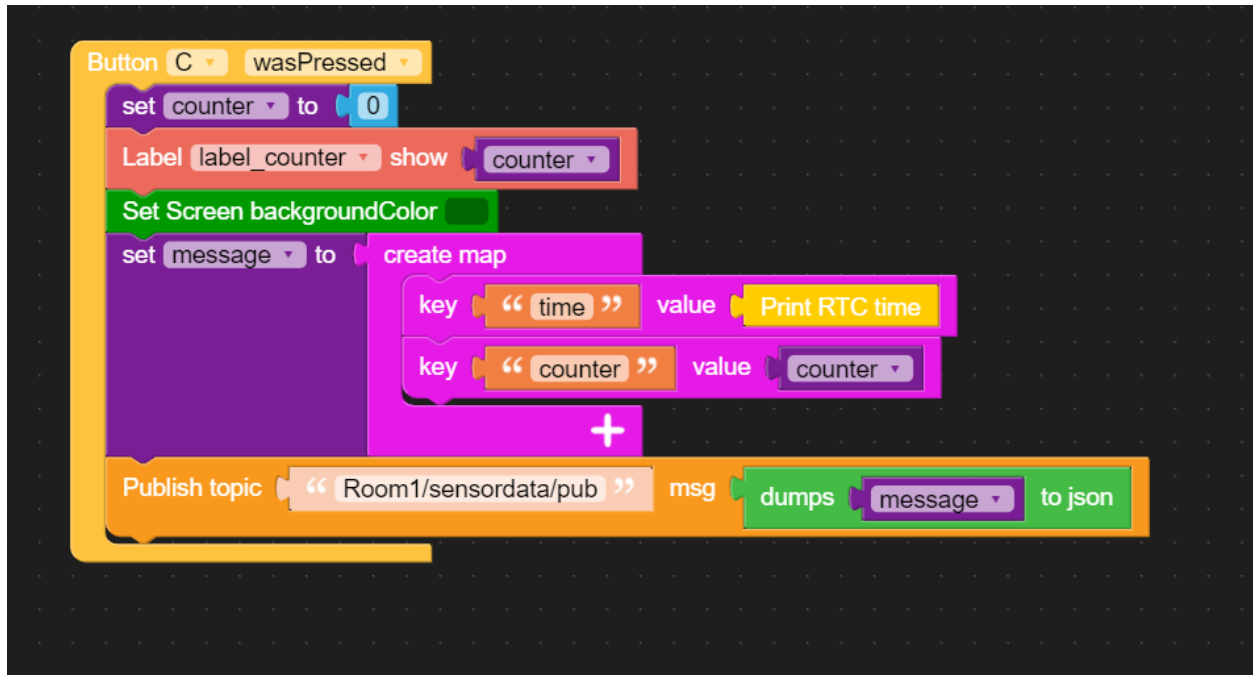
In this block we have to create a functionality where we can reset the counter to 0 on pressing the 3<sup>rd</sup> button on the touchscreen on the M5 stack.



The pseudo code for this block would be :

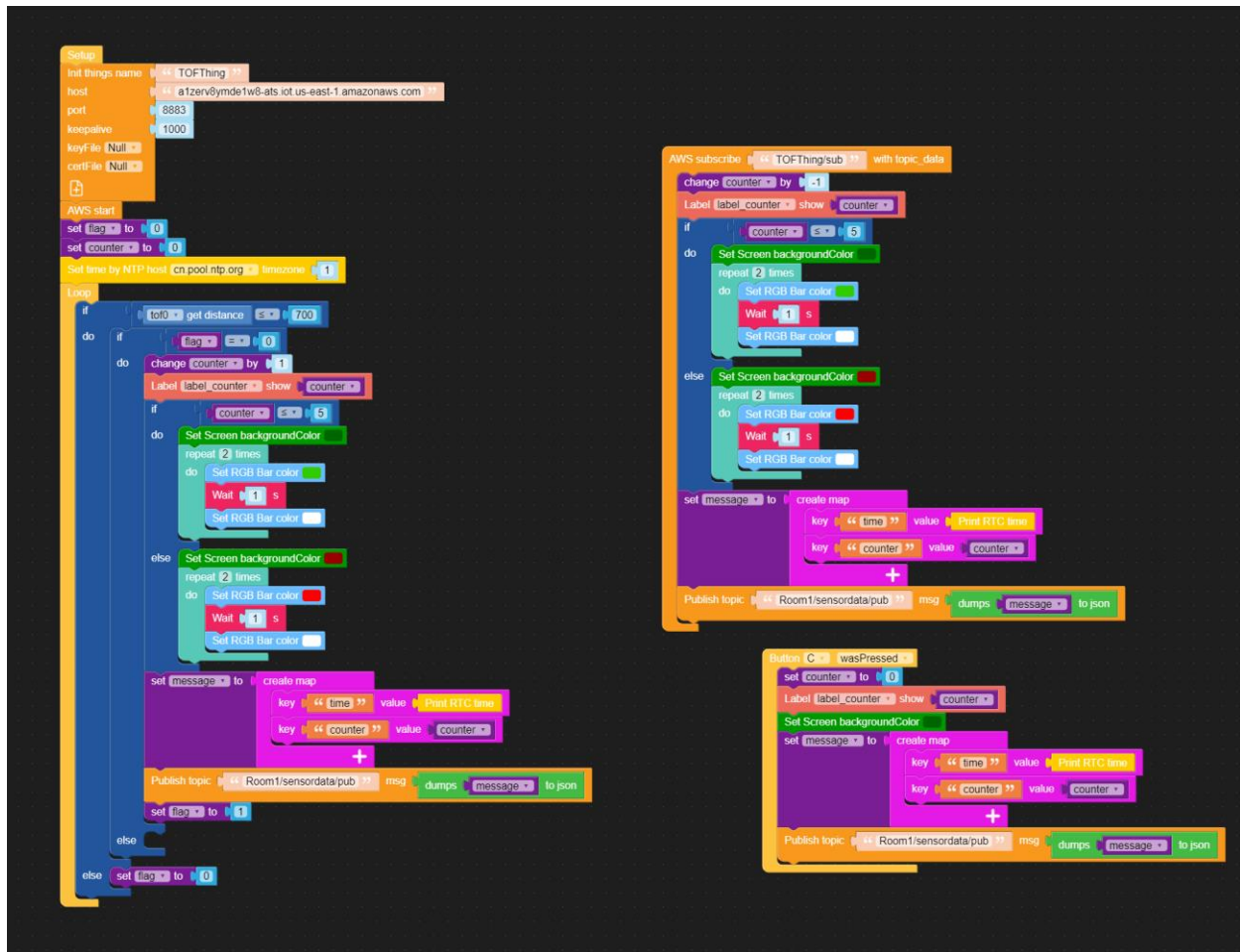
```
if(button C pressed):  
    Set counter to zero.  
    Turn the screen background green.  
    publish MQTT data of timestamp and new value of counter
```

We program this using an event on button pressed in the UI Flow and following is the blocky code for this:



### 3. Final code

#### Blocky Code



## MicroPython Code

```
from m5stack import *
from m5stack_ui import *
from uiflow import *
from IoTcloud.AWS import AWS
import json

import time
import unit

screen = M5Screen()
screen.clean_screen()
screen.set_screen_bg_color(0x000000)
tof0 = unit.get(unit.TOF, unit.PORTA)

counter = None
flag = None
```



```

message = None

label_visited = M5Label('Room Capacity', x=0, y=50, color=0xfefefe, font=FONT_MONT_40,
parent=None)
label_counter = M5Label('0', x=113, y=108, color=0xffffffff, font=FONT_MONT_48,
parent=None)
label0 = M5Label('/5', x=184, y=108, color=0xf6f4f4, font=FONT_MONT_48, parent=None)

from numbers import Number

def fun_TOFThing_sub_(topic_data):
    global counter, flag, message
    counter = (counter if isinstance(counter, Number) else 0) + -1
    label_counter.set_text(str(counter))
    if counter <= 5:
        screen.set_screen_bg_color(0x006600)
        for count in range(2):
            rgb.setColorAll(0x33cc00)
            wait(1)
            rgb.setColorAll(0xffffffff)
    else:
        screen.set_screen_bg_color(0x990000)
        for count2 in range(2):
            rgb.setColorAll(0xff0000)
            wait(1)
            rgb.setColorAll(0xffffffff)
    message = {'time':(rtc.printRTCtime()),'counter':counter}
    aws.publish(str('Room1/sensordata/pub'),str((json.dumps(message))))
    pass

def buttonC_wasPressed():
    global counter, flag, message
    counter = 0
    label_counter.set_text(str(counter))
    screen.set_screen_bg_color(0x006600)
    message = {'time':(rtc.printRTCtime()),'counter':counter}
    aws.publish(str('Room1/sensordata/pub'),str((json.dumps(message))))
    pass
btnC.wasPressed(buttonC_wasPressed)

```

```
aws = AWS(things_name='TOFThing', host='a1zerv8ymde1w8-ats.iot.us-east-1.amazonaws.com', port=8883, keepalive=1000, cert_file_path="/flash/res/c6eb468e52-certificate.pem.crt", private_key_path="/flash/res/c6eb468e52-private.pem.key")
aws.subscribe(str("TOFThing/sub"), fun_TOFThing_sub_)
aws.start()
flag = 0
counter = 0
rtc.settime('ntp', host='cn.pool.ntp.org', tzzone=1)
while True:
    if (tof0.distance) <= 700:
        if flag == 0:
            counter = (counter if isinstance(counter, Number) else 0) + 1
            label_counter.set_text(str(counter))
            if counter <= 5:
                screen.set_screen_bg_color(0x006600)
                for count3 in range(2):
                    rgb.setColorAll(0x33cc00)
                    wait(1)
                    rgb.setColorAll(0xffffffff)
            else:
                screen.set_screen_bg_color(0x990000)
                for count4 in range(2):
                    rgb.setColorAll(0xff0000)
                    wait(1)
                    rgb.setColorAll(0xffffffff)
            message = {'time':(rtc.printRTCtime()),'counter':counter}
            aws.publish(str('Room1/sensordata/pub'),str((json.dumps(message))))
            flag = 1
        else:
            pass
    else:
        flag = 0
    wait_ms(2)
```

4. Hit the run button on the top right of the UIFlow IDE.

### Download Code from GitHub

You can also download the code file from the following Github link and import it directly into your UI Flow:

<https://github.com/KhaladkarNikhil/M5StackUIFlow>

## Checking to see if your device is publishing to AWS over MQTT and receiving a message on a subscribed topic

1. In the [AWS IoT console](#), in the side navigation pane, choose **Test**, and then choose **MQTT test client**.
2. In the subscribe to a topic section, enter the topic filter from the code: “Room1/sensordata/pub” and hit Subscribe.
3. Now, place your hand near the ToF sensor so it detects the motion. Once the motion is detected the counter is increased by 1 and the data is sent to the MQTT broker. You can visualize the data in the MQTT test client as follows:

▼ Room1/sensordata/pub	April 29, 2021, 13:08:42 (UTC-0400)
<pre>{   "time": "2021-04-29 18:08:42 Thur",   "counter": 5 }</pre>	
▼ Room1/sensordata/pub	April 29, 2021, 13:08:39 (UTC-0400)
<pre>{   "time": "2021-04-29 18:08:39 Thur",   "counter": 4 }</pre>	
▼ Room1/sensordata/pub	April 29, 2021, 13:08:32 (UTC-0400)
<pre>{   "time": "2021-04-29 18:08:32 Thur",   "counter": 3 }</pre>	

4. Now open the tab “Publish to a topic” in the MQTT test client. Publish a message to the TOFThing/sub as follows:

### MQTT test client [Info](#)

You can use the MQTT test client to monitor the MQTT messages being passed in your AWS account. Devices publish MQTT messages that are identified by topics to communicate their state to AWS IoT. AWS IoT also publishes MQTT messages to inform devices and apps of changes and events. You can subscribe to MQTT message topics and publish MQTT messages to topics by using the MQTT test client.

Subscribe to a topic

**Publish to a topic**

Topic name  
The topic name identifies the message. The message payload will be published to this topic with a Quality of Service (QoS) of 0.

TOFThing/sub

Message payload

```
{
  "message": "out"
}
```

Additional configuration

**Publish**

Note: This message string can be anything. We are not checking the message string in the M5stack. We just reduce the counter if the M5Stack gets a message on the TOFThing/sub topic. For simplicity purpose the message string is kept as out since we are emulating a scenario when a person is going out of the room.

Now check the counter on the screen. It will be reduced by 1 when the message is received. Also, the TOFThing/pub is notified with the new counter value.

- Now, test the reset counter functionality. When you press the button c on the M5 stack you would see that the counter is reset to 0 and the new counter value is published to the MQTT broker.

▼ Room1/sensordata/pub	April 29, 2021, 13:46:09 (UTC-0400)
<pre>{   "time": "2021-04-29 18:46:10 Thur",   "counter": 0 }</pre>	
▼ Room1/sensordata/pub	April 29, 2021, 13:46:07 (UTC-0400)
<pre>{   "time": "2021-04-29 18:46:07 Thur",   "counter": 5 }</pre>	

## **Conclusion**

You have successfully managed to Setup the M5Stack, write a program to read sensor data, push sensor data to AWS and subscribe to an MQTT topic and perform action on receiving a message on the same topic.