JOHANNES KEPLER
UNIVERSITÄT LINZ

SMART HOME USE CASES – USER GUIDE (UIFLOW V1.15.3)

TABLE OF CONTENTS

1. Introduction

This user guide presents the implementation and usage of a smart home system based on the M5Stack platform. The system demonstrates how physical devices such as sensors and actuators can be integrated into a cyber-physical system (CPS) and controlled through a digital interface.

The main objective of this project is to provide a practical introduction to smart home concepts, including motion detection, environmental monitoring, and security control. The system uses real-time communication via MQTT to connect the M5Stack device with a backend service and a frontend application, enabling remote monitoring and control.

This guide is structured into three main use cases. The first use case focuses on motion detection and automatic lighting. The second use case demonstrates how temperature and humidity are measured and displayed. The third use case introduces security monitoring for doors, windows, and locks.

The document is also designed for users or testers and provides step-by-step instructions for setting up the hardware, configuring the software, and running the system. In addition, the complete source code is provided to allow users to easily replicate the system by adjusting only a few configuration parameters, such as Wi-Fi credentials at option A section 2.4.1 (Running the system).

For this project, a cloud-based architecture was used to simplify deployment and accessibility. The frontend application was developed using Flutter and deployed via Firebase Hosting, while the backend service (Spring Boot) was deployed using Railway. Communication between the system components is handled through the public HiveMQ MQTT broker.

This setup allows users to access the system remotely via a mobile web browser without requiring any local installation of the frontend or backend components.

2 M5Stack Setup & Installation

2.1 Device Preparation

These contain all the necessary preparations to implement the use case with M5Stack. Before starting, ensure the following requirements are met:

● M5Stack UIFlow installed (web-based or desktop version)

UIFlow can be installed and used locally on the computer system online at https://flow.m5stack.com/.

● Connect the M5Stack device to the computer via USB

● USB driver for M5Stack devices installed

- Stable Wi-Fi connection

2.2 Install Drivers and UIFlow

Step 1

Open your web browser and visit the M5Burner website:
https://docs.m5stack.com/en/uiflow/m5burner/intro

Step 2

On the website, locate the section “UIFlow Firmware Burning Tool” and download the version that matches your operating system, as shown in Figure 1.

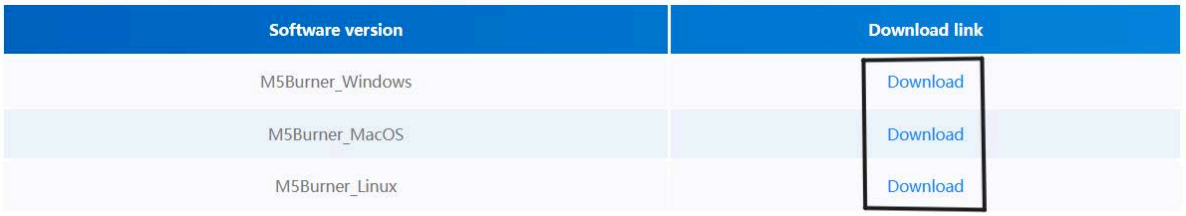


Figure 1: UIFlow firmware download (M5Burner)

Step 3

Visit the link https://docs.m5stack.com/en/download

Step 4

Afterwards under “USB DRIVER”, select the version “CP210x_VCP_” for your operating system (for Windows download the version *CP210x_VCP_Windows* , for macOS download the version *CP210x_VCP_MacOS* or if you are using Linux download the version CP210x_VCP_Linux) as shown in figure 2 below.

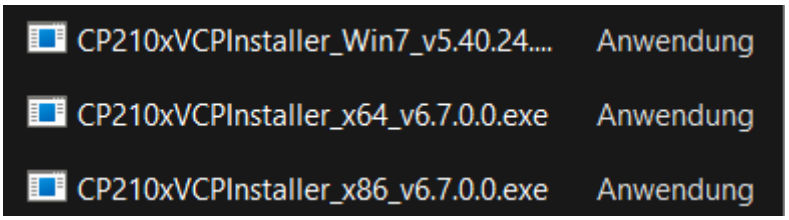
NO	Name	Download
1	CH9102_VCP_SER_MacOS v1.7	
2	CH9102 MacOS(M1 Silicon)	
3	CH9102_VCP_SER_Windows	
4	CH9102_VCP_CDC_Windows	
5	CP210x_VCP_Windows	
6	CP210x_VCP_MacOS	
7	CP210x_VCP_Linux	
8	SR9900_Windows	
9	SR9900_MacOS	
10	FTDI Drivers	

Figure 2: CP210x driver download

2.3 Firmware Installation

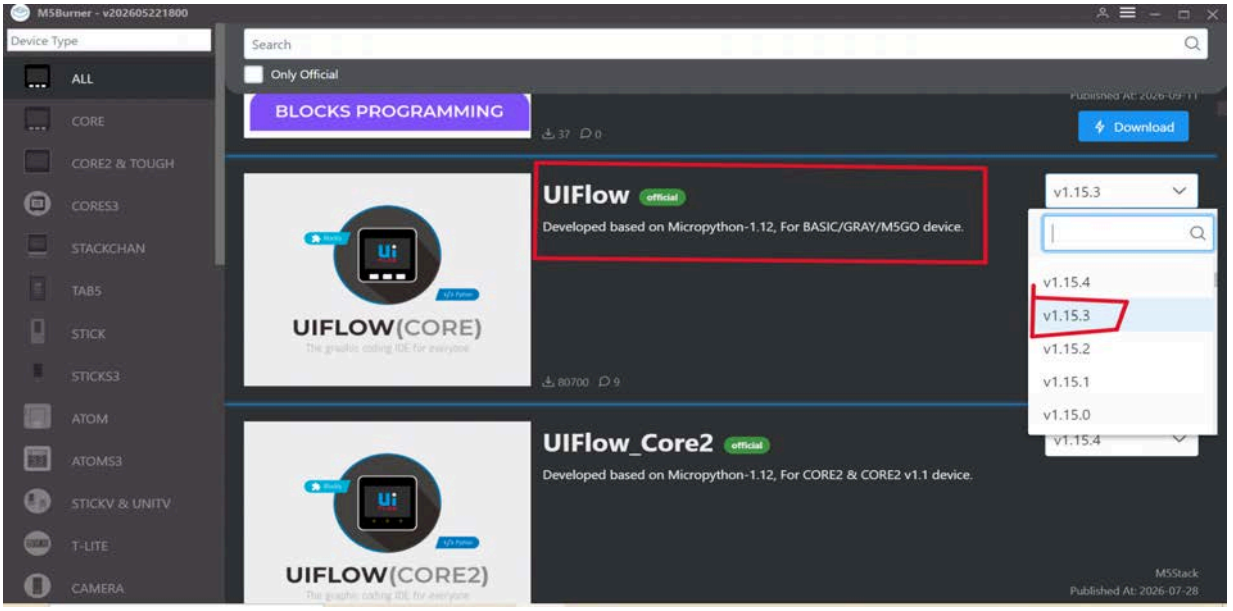
Step 1

After the download, open the extracted folder and run the installation file for your system ("CP210xVCPInstaller_x64…").



Step 2

Extract and Open the extracted folder and run "M5Burner.exe" which was downloaded under step 2 figure 1 . Then scrol down and choose the version UIFlow official v1.15.3 from the drop-down menu and click "Download" as shown with the red rectangle in figure 3 below.



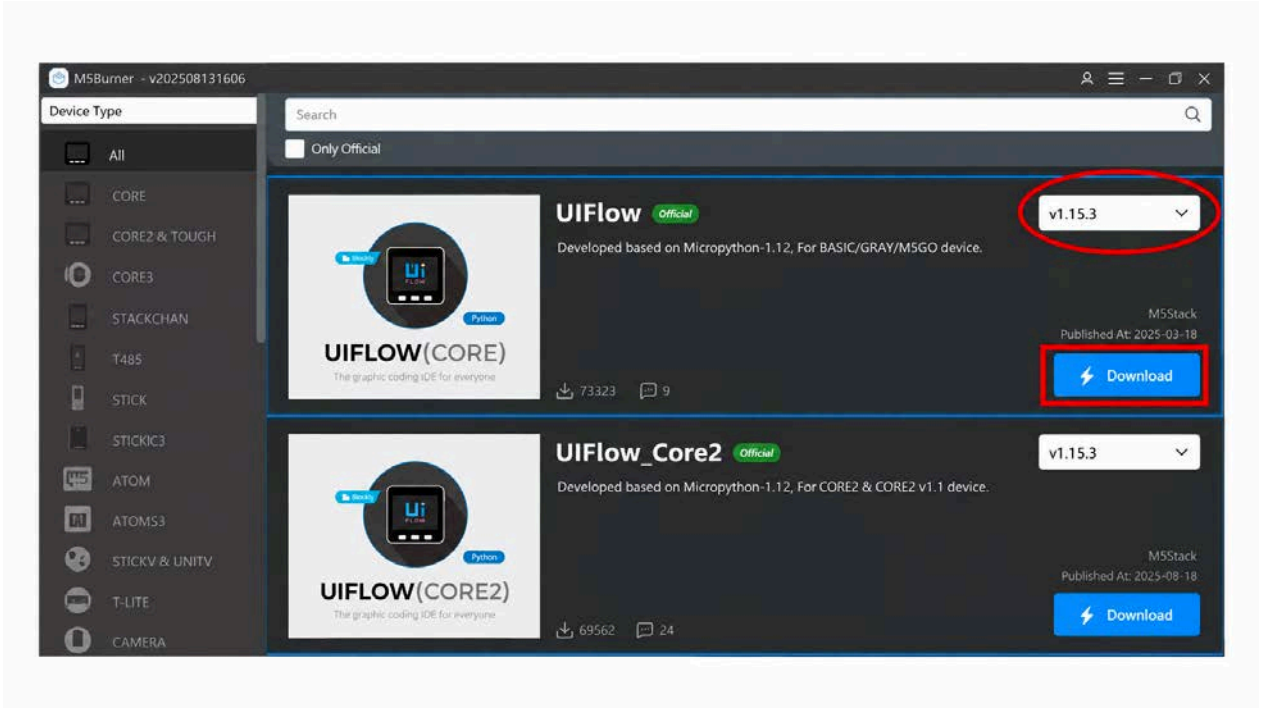


Figure 3: Download UIFlow official v1.15.3

Step 3

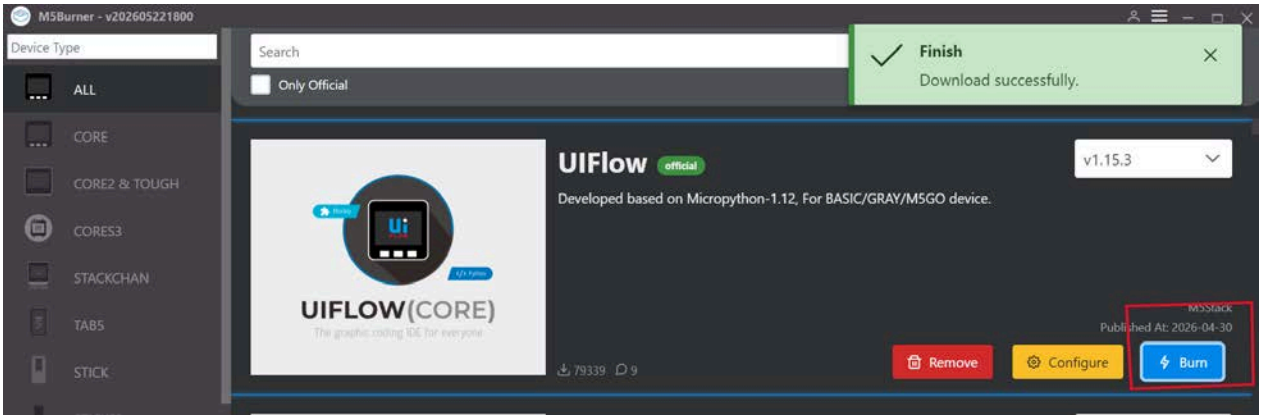
After a successful download ,Connect the M5Stack to your PC using the included USB Type-C to Type-A cable as demonstrated in figure 3.1 below.

Note:The USB connection allows the computer to communicate with the M5Stack and transfer the required UIFlow firmware to the device.

Figure 3.1: M5stack and PC connection via USB

Step 3.1

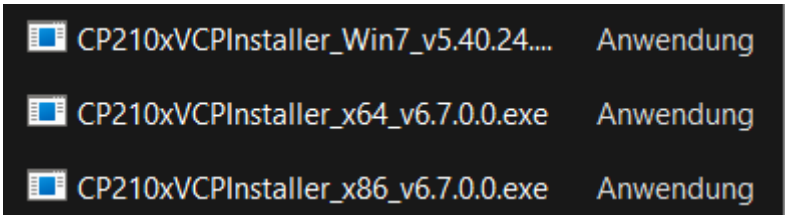
Then click "Burn" as demonstrated below.In this context, **burning** means installing the UIFlow firmware onto the M5Stack device. This firmware is required so that the M5Stack can communicate with UIFlow and execute the programs created later in this guide.



Step 4

Once the M5Stack is connected to the computer, it is automatically detected and assigned a **COM port**. Select the COM port that appears after connecting the M5Stack. In this example as shown in Figure 4 below, the connected M5Stack is detected as **COM3**. Keep the **BaudRate** at **1500000**, and then click **Next** to continue.

Note: COM locates your m5stack name if the driver below is well installed under section 2.3 Firmware Installation Step 1



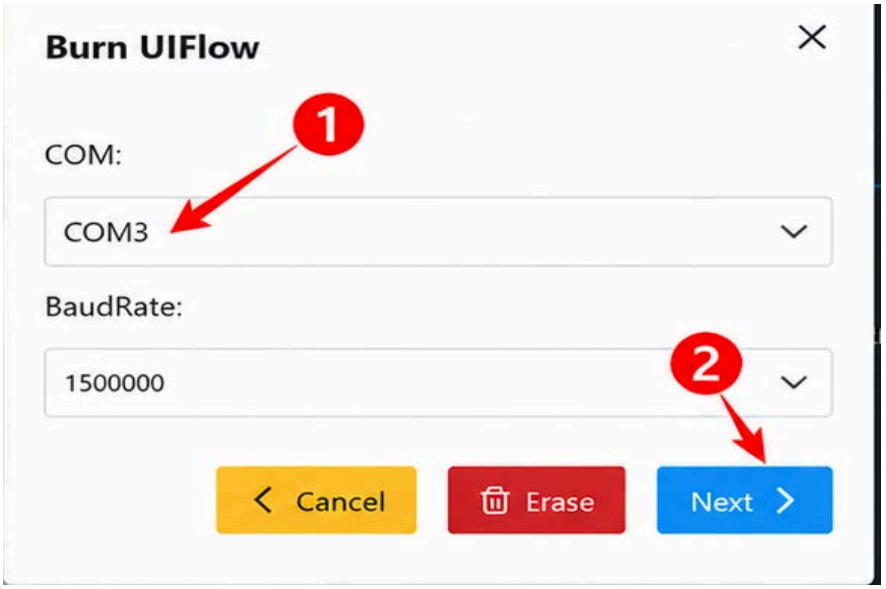


Figure 4: M5stack flashing setup

Troubleshooting: If the wrong COM port is selected, UIFlow Burner may display an error and the burning process will not complete. Return to the COM selection and identify the correct port by disconnecting the M5Stack and checking which COM port disappears. Reconnect the M5Stack and select the COM port that reappears. Then click **Next** and repeat the burning process.

Step 5

Make sure to fill in the correct Wi-Fi configuration that the device needs to connect to, including Wi-Fi SSD and Wi-Fi Password, then click next to start burning as shown below.

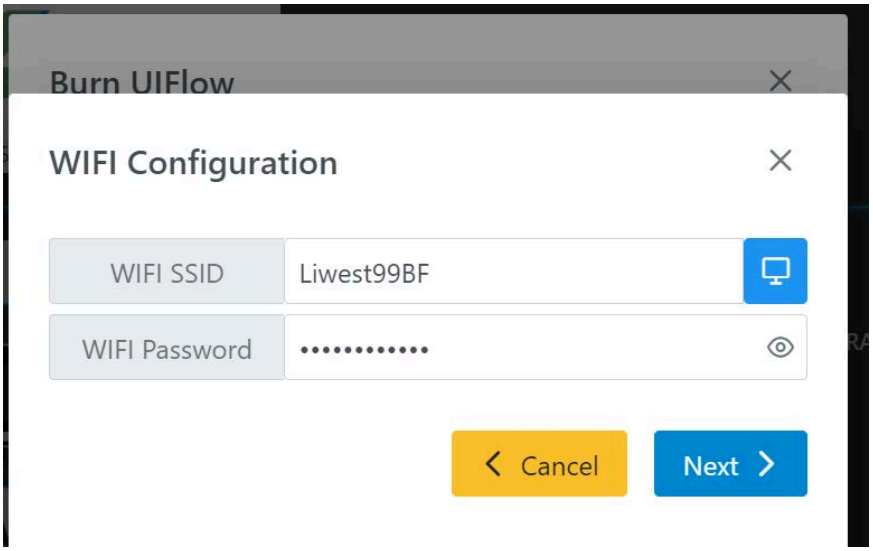


Figure 5: WiFi configuration

Step 6

After clicking “next”, wait until the installation is completed.

- Do not disconnect the M5Stack during this process.

A message such as “Burn Successfully” appears as shown in figure 6 below which means your m5stack is now connected.

Figure 6:Installation process for burning m5stack

Troubleshooting – COM Error

If a **COM error** occurs during the burning process:

1. Make sure the M5Stack is properly connected to the computer using the USB cable.
2. Close the error message and return to the **COM** selection.
3. Disconnect the M5Stack and check which COM port disappears from the list.
4. Reconnect the M5Stack and select the COM port that reappears.
5. Make sure the required **CP210x USB driver** is installed under section 2.3 Firmware Installation Step 1.
6. Start the burning process again.

Note: Do not disconnect the M5Stack while the firmware is being burned.

Troubleshooting – Screen Error After Burning Due to Incorrect Wi-Fi Configuration

If the M5Stack displays a connection error after the firmware has been successfully burned, the **Wi-Fi SSID or password may have been entered incorrectly** during the Wi-Fi configuration.

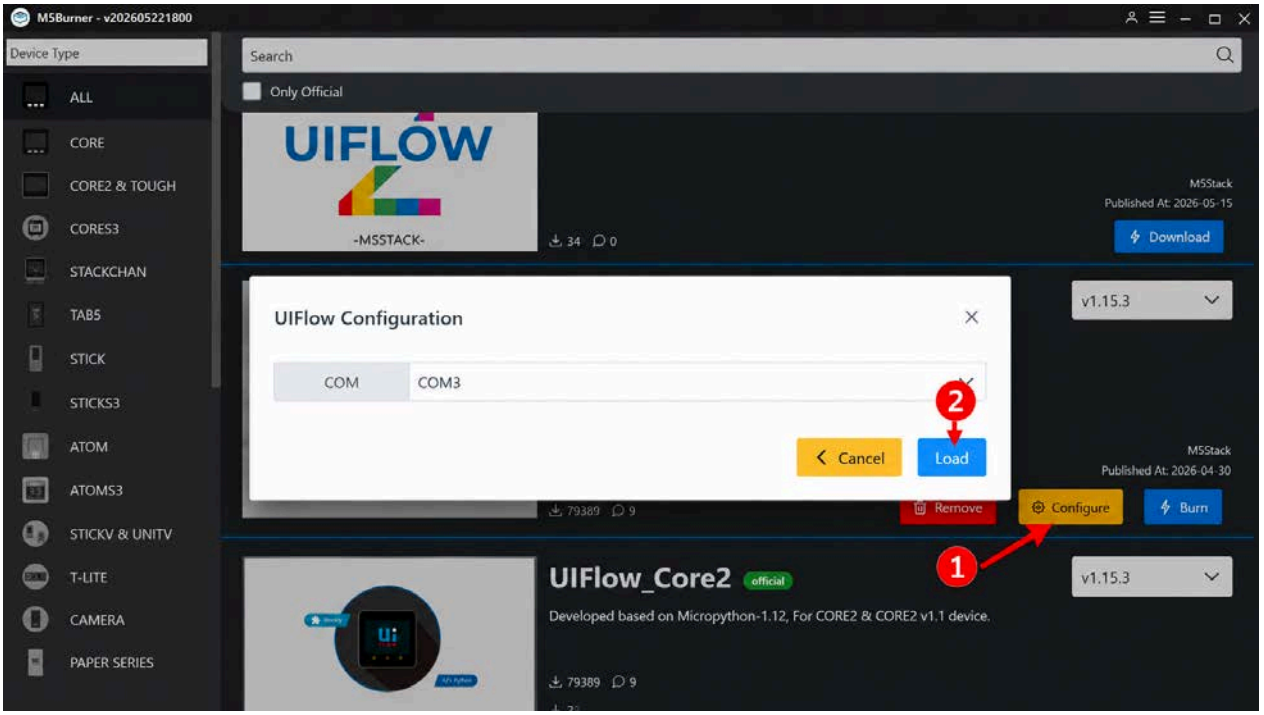
To resolve this issue:

1. Connect the M5Stack to the PC using the USB cable.
2. Open **M5Burner** and select **Burn** again.
3. Enter the correct **Wi-Fi SSID and password**. Ensure that the Wi-Fi name is entered exactly as displayed, including spaces and capital letters.
4. Complete the burning process again.
5. Restart the M5Stack and check whether it successfully connects to the configured Wi-Fi network.

Note: An incorrect Wi-Fi name or password does not necessarily mean that the firmware burning itself failed. The firmware may be installed successfully, but the M5Stack will be unable to connect to the Wi-Fi network afterward.

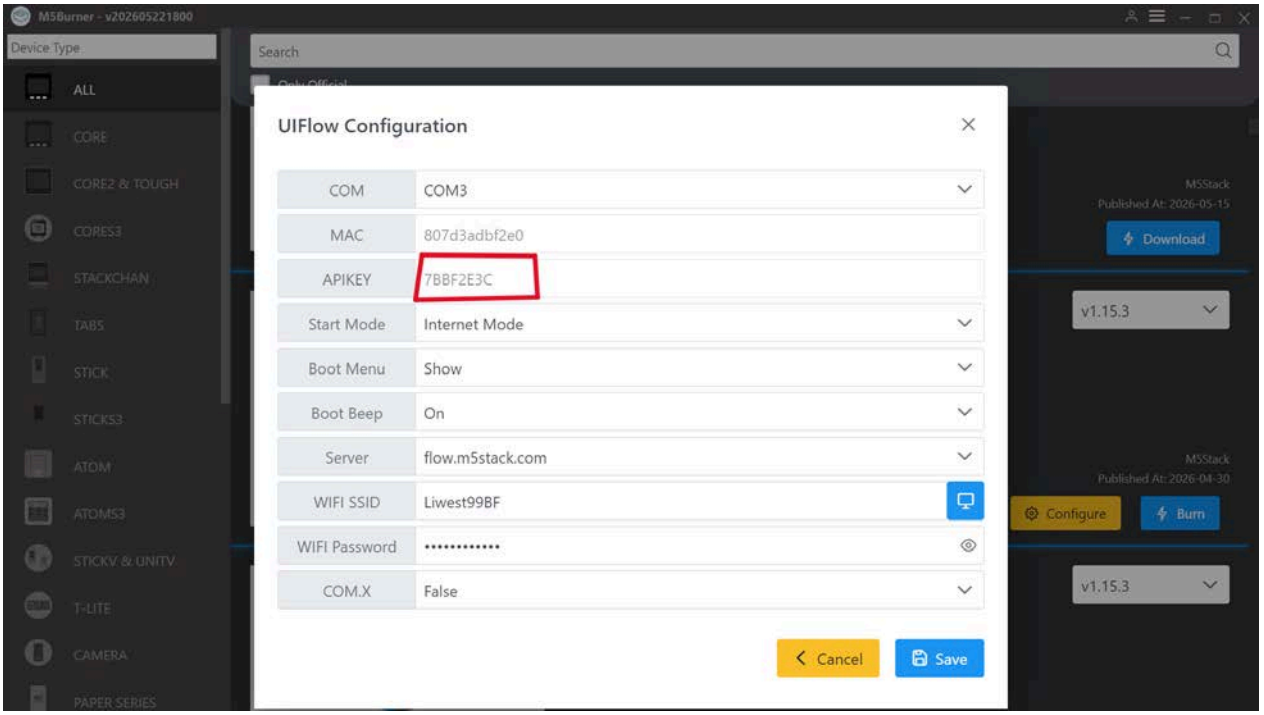
Step 7

After a successfully burn ,click on configure then click on load as shown below.



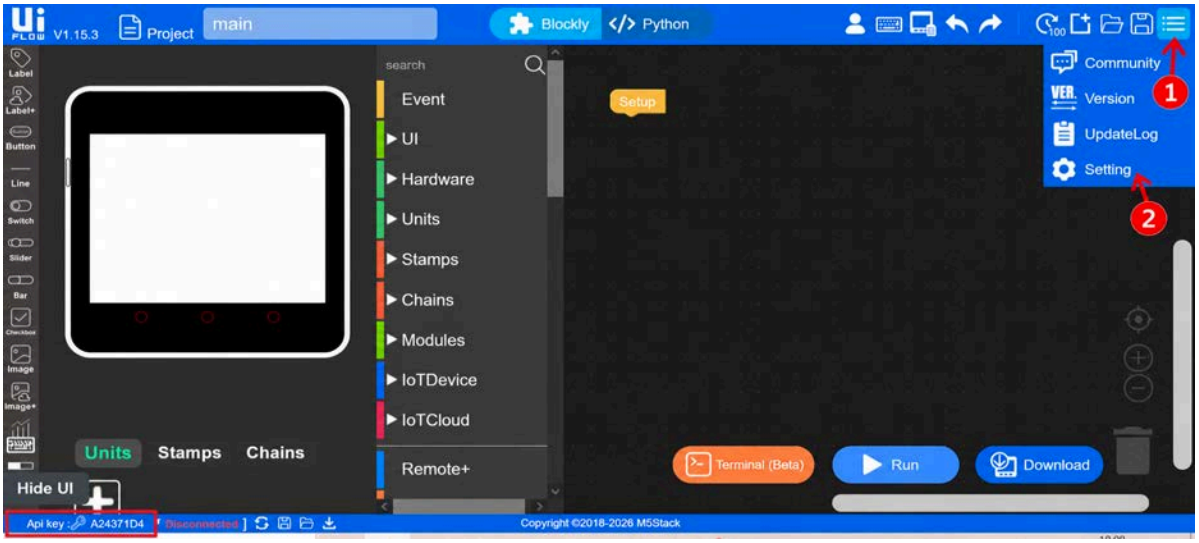
Step 8

This directs you to the page below. Now copy the API KEY generated by your console which will be used in the step (Step 10). Then click save. The API KEY will also appear on the M5stack screen after successful burn.



Step 9

Open your web browser and go to https://flow.m5stack.com/ and Click the 3 horizontal lines at the top right corner and click Setting (as shown below).



Step 10

In the Api key field:

- Insert the API KEY generated that can be seen on the M5stack screen or shown in step 8 for example "7BBF2E3C".
- Then press OK as shown in figure 7 below.

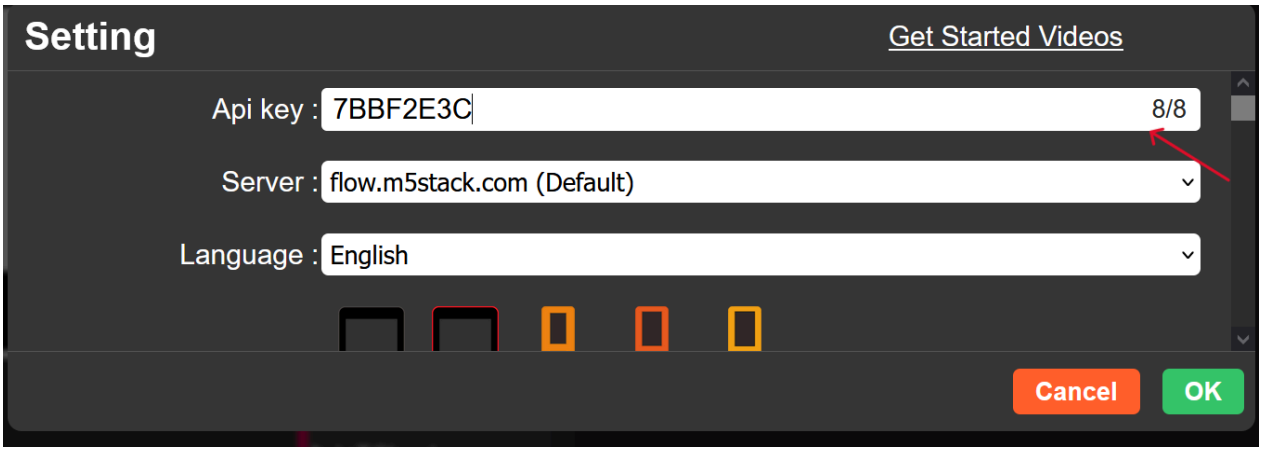


Figure 7: Uiflow configuration and connect to m5stack

Once a connection is successfully established, the M5Stack and UIFlow are ready to use. This is indicated by the status "Connected" in the lower left corner of the screen. This indicator is shown in the next image below in figure 8.

Figure 8: Successful connectivity

2.4 First Connection (UIFlow) and Running the system

After successfully connecting the M5Stack to UIFlow, you can proceed in one of two ways namely:

Option A – Run the Completed Smart Home System (Recommended for Testers and End Users).
Users who only want to run and test the completed Smart Home System can follow these steps (option A)

Option B – Recreate the System Step by Step.

Recommended for understanding how the Smart Home system was developed or recreate the project yourself

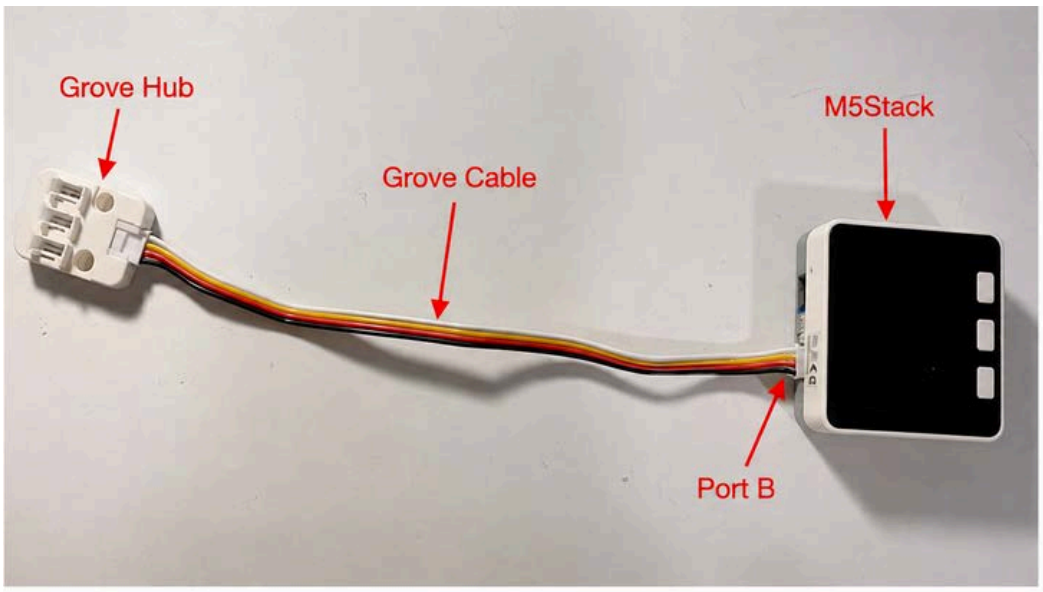
Option A – Run the Completed Smart Home System (Recommended for Testers and End Users)

Users who only want to run and test the completed Smart Home System can follow the steps below.

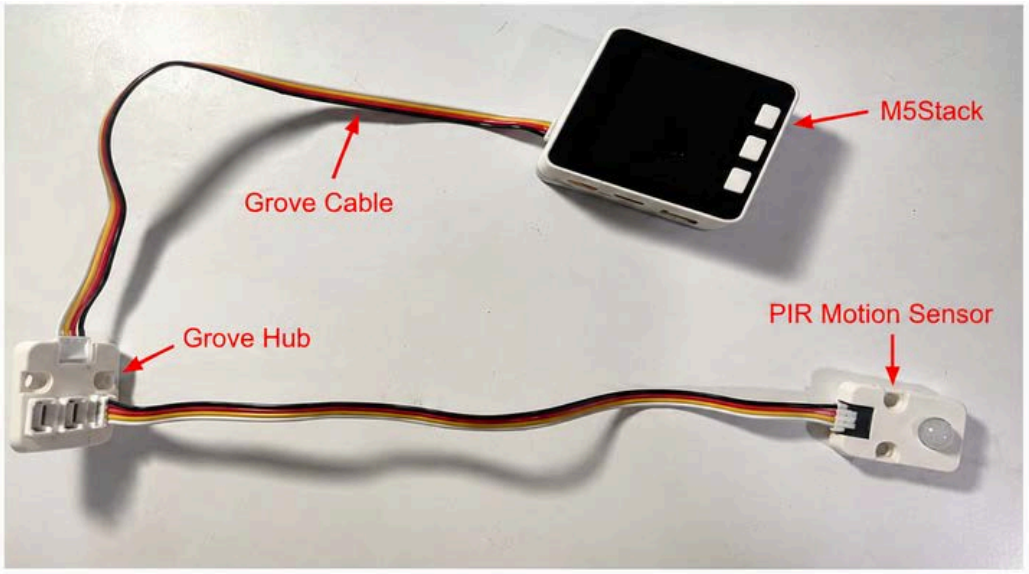
Hardware Setup

Before running the system, connect all hardware components as follows:

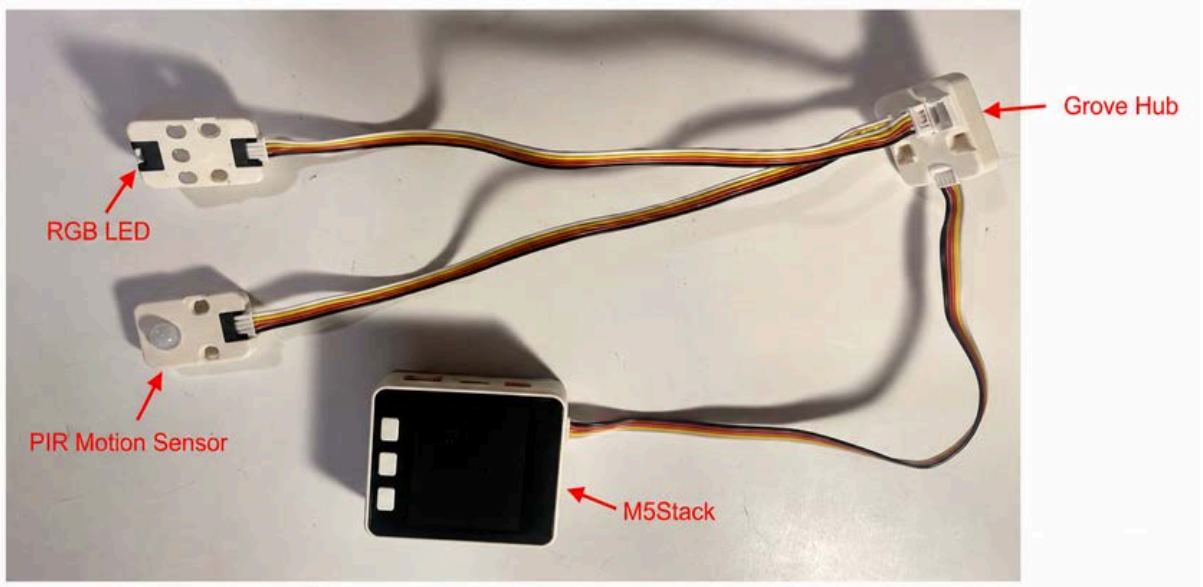
1. Connect the Grove Hub to **Port B** of the M5Stack using a Grove cable. This enables other components to be connected together.



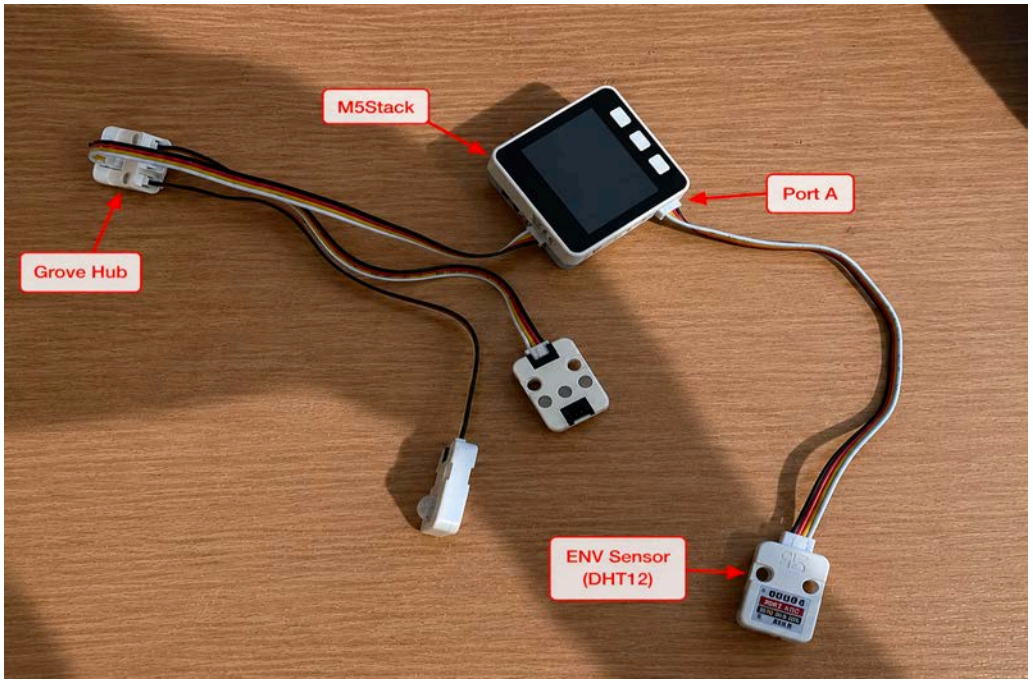
2. Connect the PIR motion sensor to Port B via the Grove Hub using a Grove cable.



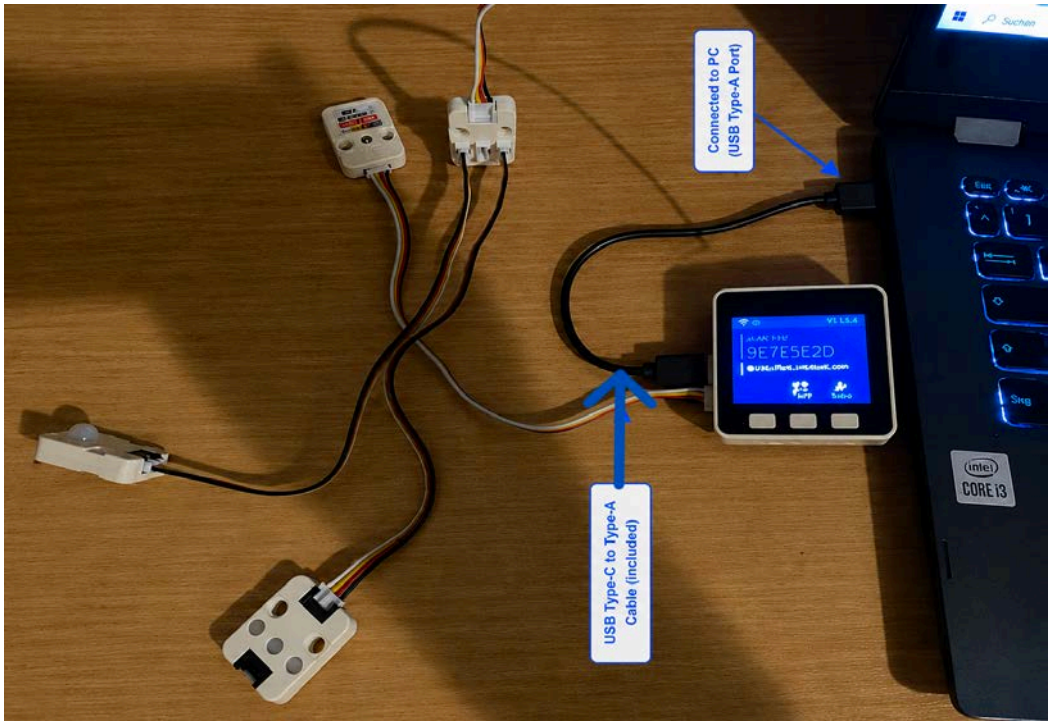
3. Connect the RGB LED Unit to another free port on the Grove Hub.



4. Connect the ENV Sensor (DHT12) to **Port A** of the M5Stack using a Grove cable and ensure that all cables are firmly connected.



5. Connect the M5Stack to your PC using the included USB Type-C to Type-A cable



Note: Once the M5Stack is connected to the computer via the USB cable, the screen will turn on. If the firmware was burned successfully, the API key will be displayed on the M5Stack screen.

2.4 .1 Running the System for option A

After the above hardware setup, follow the steps below to run the system. Please make sure that the m5stack device and its components are all connected to your computer via the usb cable as illustrated under **Hardware Setup** .

Video Tutorial (Recommended)

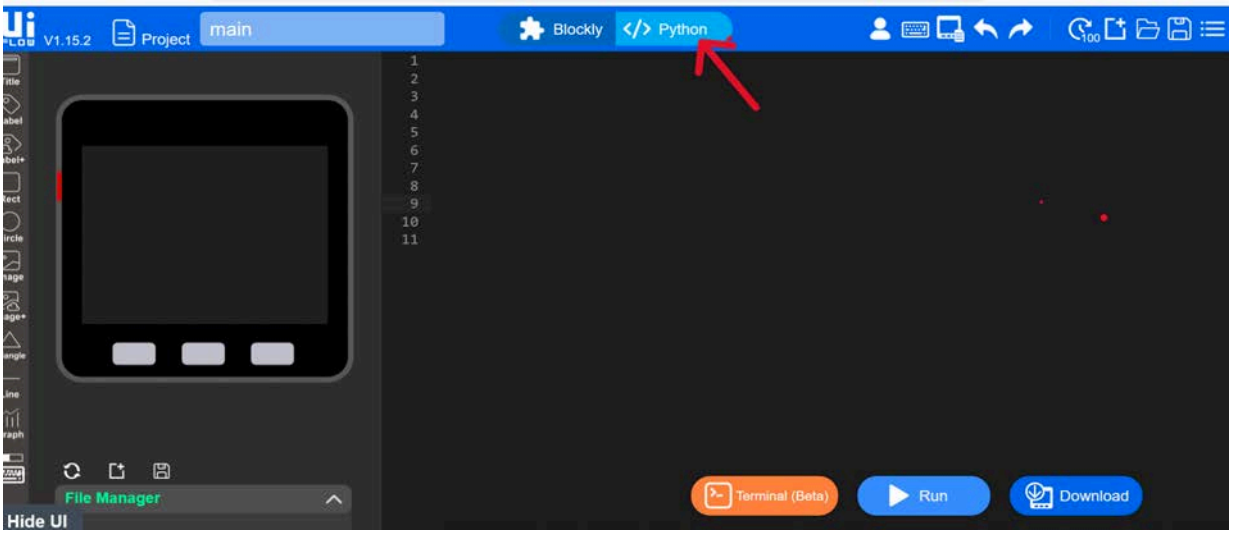
A complete video walkthrough demonstrating the code upload, Wi-Fi configuration and system execution is available here: If you encounter any difficulties while following the written instructions and steps below , it is recommended to watch the video tutorial first.

Step 1 Open UIFlow

Go to: https://flow.m5stack.com and make sure your M5Stack device and its components are all connected via USB as demonstrated at option A under **Hardware Setup** . Before that make sure your m5stack is already flashed or burned as described in section 2.

Step 2

In UIFlow, click the **Python** icon and ensure that the editor is empty, as shown below



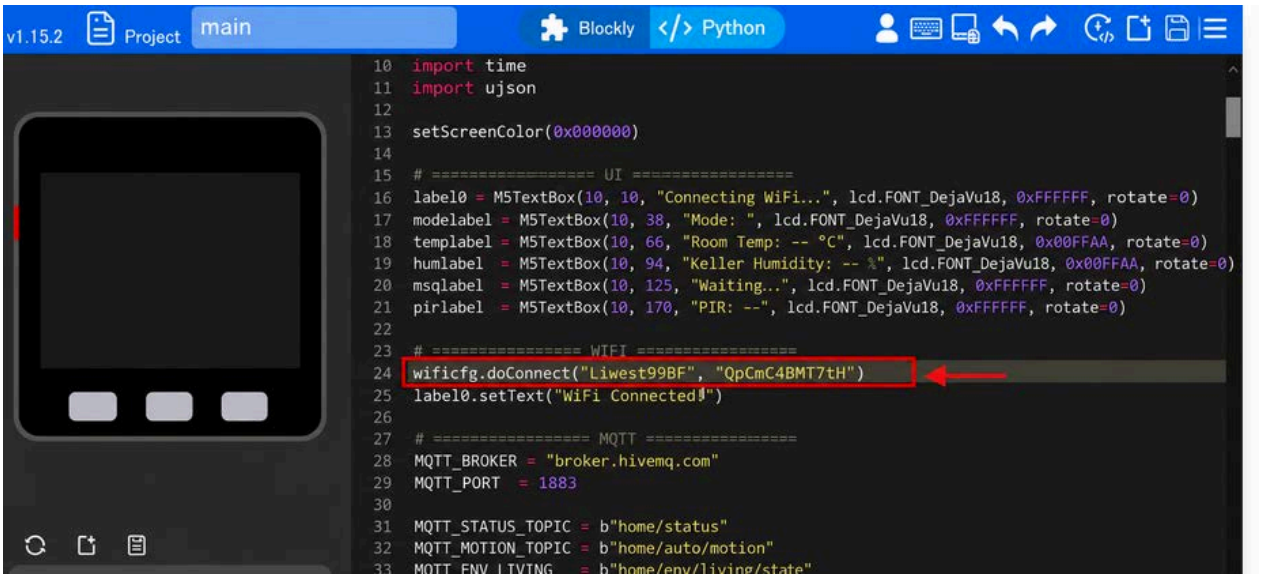
Step 3 Insert the Python Code

Download and copy the provided Python code here link: Complete Source Code Repository and paste it into the empty Python editor in UIFlow .

Step 4 Configure Wi-Fi

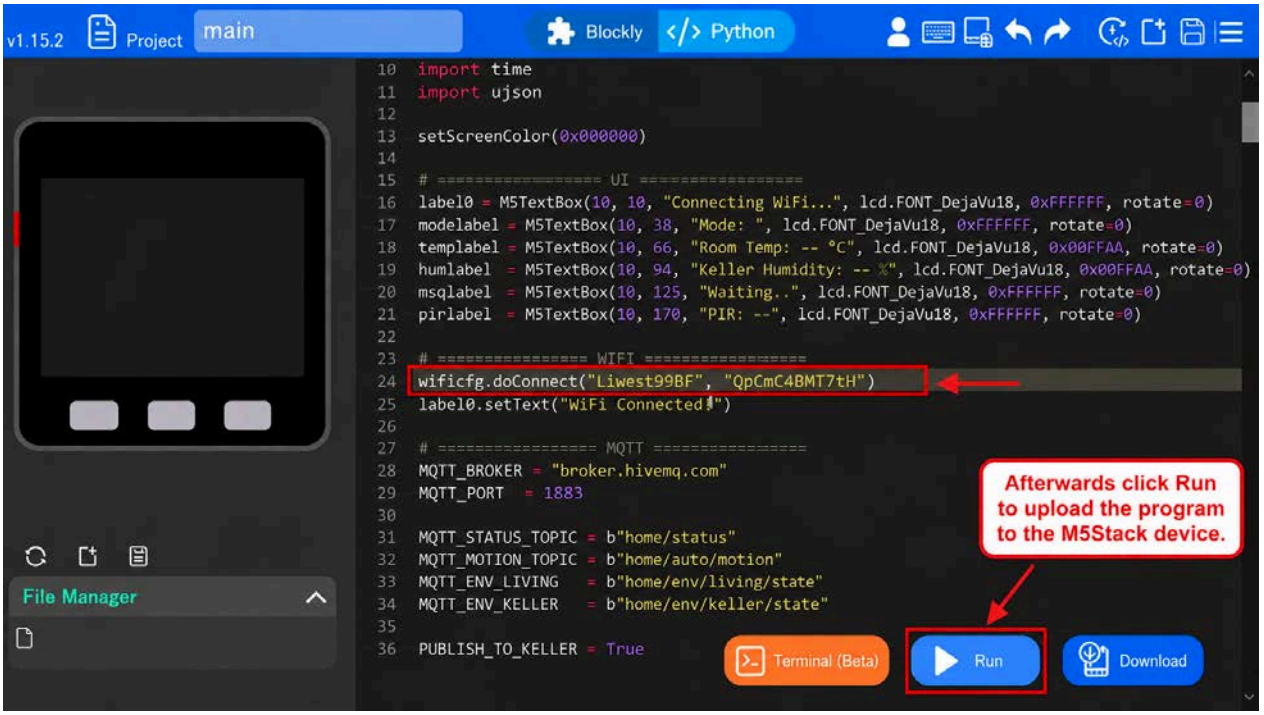
Inside the code, update the following line with your own Wi-Fi credentials at line 22:
wifiCfg.doConnect("`WIFI_NAME`", "`PASSWORD`")

Note: On the figure below my `WIFI_NAME is` Liwest99BF and `PASSWORD is` QpCmC4BMT7tH. And the WiFI credentials should be the same as the m5stack device WIFI configure as demonstrated at section 2.2 Install Drivers and UIFlow under step 5 else the program will not function on the m5stack screen



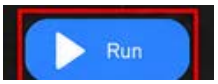
Step 5

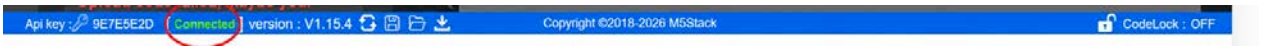
Afterwards click Run to upload the program to the M5Stack device as shown below.



TroubleShooting

If the connection status shows **“Disconnected”**, as indicated in the image with the red arrow, click the **Refresh icon** next to it to reconnect the M5Stack device indicated in red rectangle.

If it reconnects click on the Run icon 



After pressing run, your m5tack screen should display the screenshot below in few seconds

Step 5 Access the Application

Open the frontend application on a mobile device or web browser on laptop using the following URL: :https://smarthome-app-865f2.web.app

Step 6 – Test the Smart Home System

After opening the frontend application, verify that all system components are working correctly by testing the following functions or use cases.

Use Case 1 – Motion Detection and Automatic Lighting

This use case verifies the automatic detection of movement in the cellar and store room using PIR motion sensors. When motion is detected, the corresponding room light is automatically switched ON. If no motion is detected for 10 seconds, the light is automatically switched OFF. The following steps demonstrate the execution of this use case.

Note: The prototype uses **one M5Stack device and one PIR motion sensor** to simulate motion detection in both rooms. The same PIR sensor is used for both scenarios; the active room is selected using the buttons on the M5Stack.

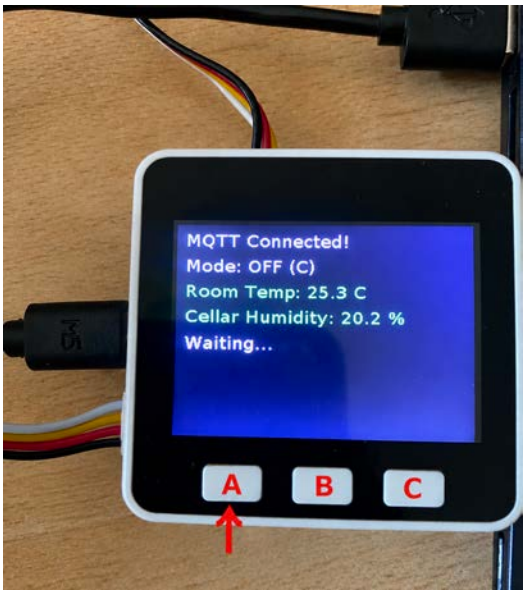
- The m5stack has 3 buttons A,B and C.

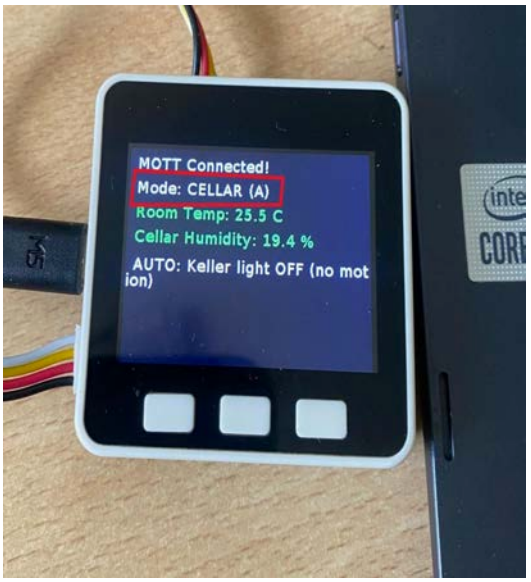
A Button (Cellar mode)

Step 1: Activate cellar mode

A. Press **Button A** on the M5Stack device. This switches the system from **OFF mode** to **CELLAR mode**

B. After pressing **Button A**, the display should update and show **"Mode: CELLAR (A)"**





Step 2: Trigger Motion Detection

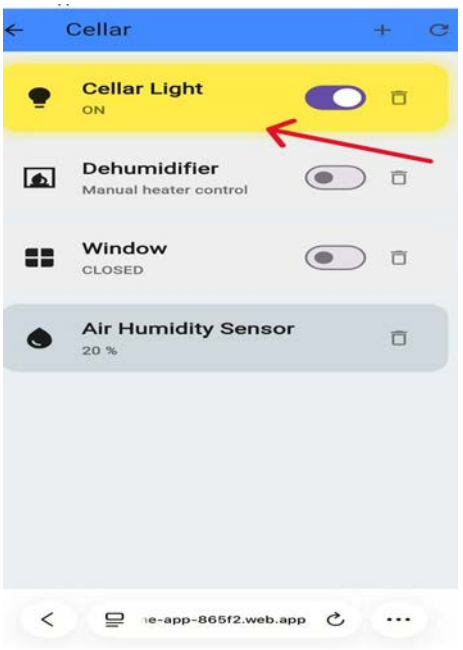
A. Move or pass an object in front of the **PIR Motion Sensor**(Red arrow indicates the PIR sensor). In a real-world scenario, this represents a person entering the cellar room. Anything motion is been detected light turns on automatic

- Expected results : the sensor will detect movement and the m5stack screen will display keller(cellar light is ON) as shown below in the red rectangle.

B. On the Smart Home App via the link:https://smarthome-app-865f2.web.app, click on cellar icon 

to open the cellar room view

- Expected results : the **Smart Home app updates the light status to ON** as shown below. This indicates that the light has automatically turned ON due to motion detection.



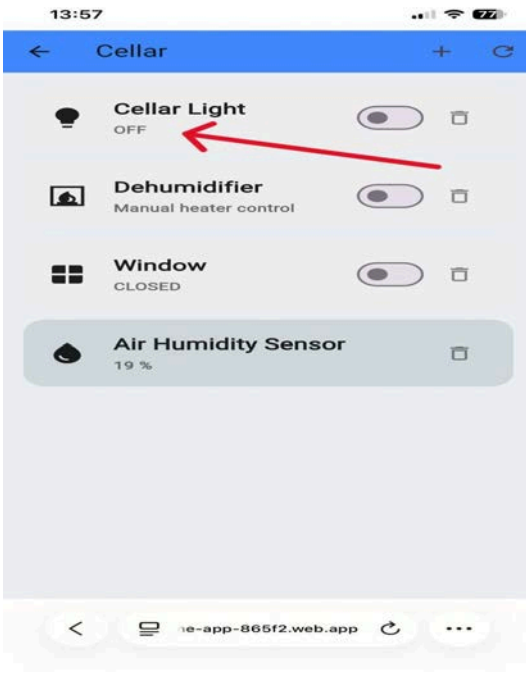
Step 3: Automatic Light Deactivation After No Motion Is Detected in cellar

A. After 10 seconds of no movement detection the light automatically turns off in cellar. Or move away from the pir motion sensor for some seconds.

- Expected results: The M5Stack display shows **"Keller light OFF"** as shown below highlighted in red rectangle.

B. The Smart Home application automatically updates the cellar light status to **OFF**.

- Expected results: The cellar light status in the application changes to **OFF** as shown below highlighted in red arrow . This indicates that the light has automatically turned OFF due to no motion detection.

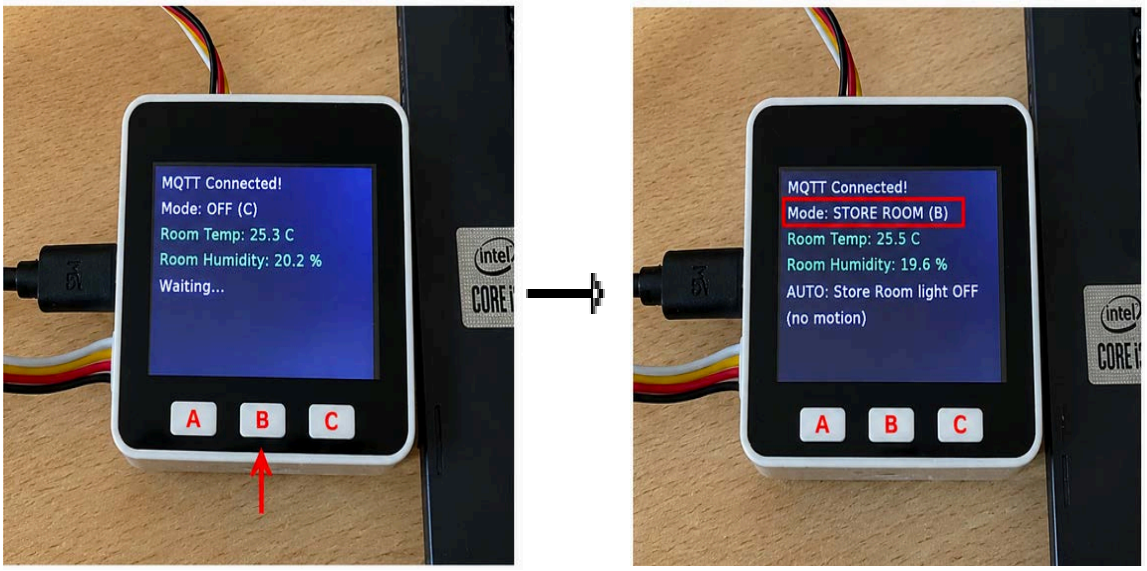


Button B (Store Room Mode)

Step 1: Activate Store Room Mode

A. Press Button B on the M5Stack device.This switches the system from **OFF mode** to **STORE ROOM mode**

B. After pressing **Button B**, the display should update and show **"Mode: STORE ROOM (A)"** as illustrated below in the red rectangle



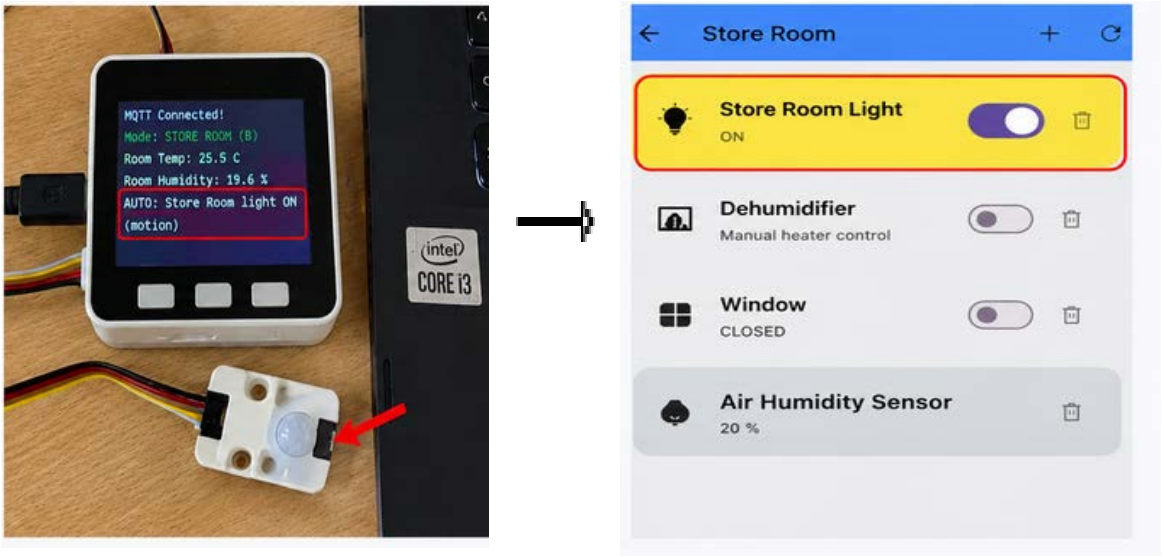
Step 2: Trigger Motion Detection

A. Move or pass an object in front of the PIR Motion Sensor(the red arrow indicates the PIR sensor). Once motion is detected, the store room light turns ON automatically.

- **Expected Result:** The PIR Motion Sensor detects movement and the M5Stack display shows **"Store Room light ON"** as shown below in the red rectangle .

B. On the Smart Home App, click on the **Store Room** icon to open the store room view.

- Expected Result: The Smart Home application automatically updates the light status to ON as shown below.



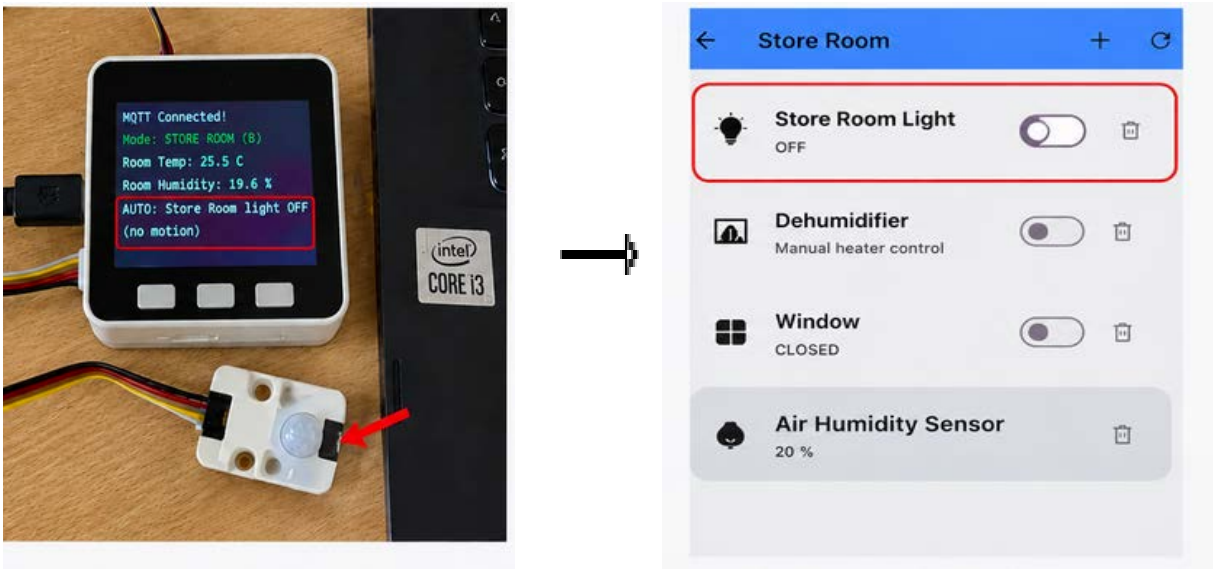
Step 3: Automatic Light Deactivation After No Motion Is Detected in store room

A. After 10 seconds without any movement being detected, the store room light automatically turns OFF. Or move away from the PIR Motion Sensor and wait for approximately 10 seconds.

- Expected Result: The M5Stack display shows "Store Room light OFF" as shown below in red rectangle.

B. Verify the light status in the store room Smart Home App

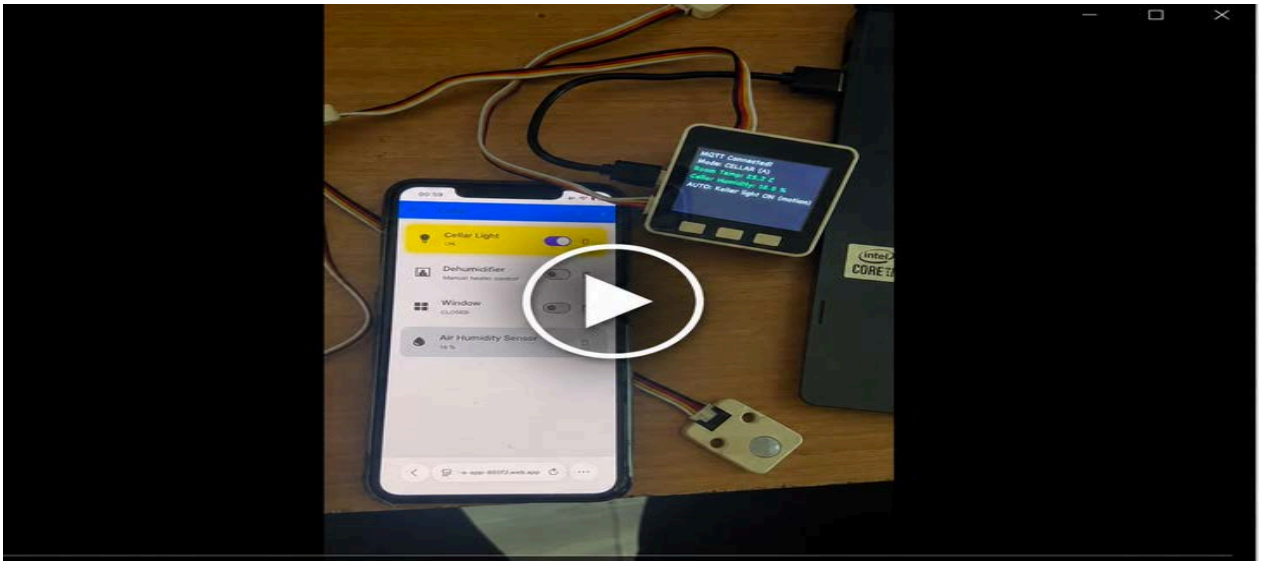
- Expected Result: The Smart Home application automatically updates the store room light status to OFF as shown below.



Button C (OFF)

When you press **Button C**, the system switches to **Mode: OFF (C)**. In this mode, the automatic cellar-light behavior is disabled, so the PIR motion sensor should **not automatically turn the cellar light on**.

The following video demonstrates how to test the automatic light control using the PIR motion sensor.



The video demonstrates that pressing the first button (**Button A**) switches the M5Stack to **Cellar Mode**. Moving a hand in front of the PIR motion sensor simulates a person entering the cellar room and automatically turns the **Cellar light ON**. At the same time, the light status is updated in the **Smart Home mobile application**. If the user presses **Cellar** in the mobile application, they are directed to the **Cellar room**, where the Cellar light is shown as **ON/yellow (automatic)**. If no further motion is detected for **10 seconds**, the Cellar light automatically turns **OFF**, and the updated OFF status is also shown in the mobile application and on the m5stack screen.

The same process applies to **B (Store Room)**. Pressing the second button (**Button B**) switches the M5Stack to **Store Room Mode**. When the PIR motion sensor detects movement, simulating a person entering the store room, the **Store Room light automatically turns ON**. The status is simultaneously updated in the Smart Home mobile application. By pressing **Store Room** in the application, the user is directed to the Store Room page, where the automatic light status can be viewed. If no motion is detected for **10 seconds**, the Store Room light automatically turns **OFF**, and the change is also reflected in the application.

This demonstrates that the physical motion detected by the **PIR sensor** is correctly processed by the M5Stack and reflected in the **Smart Home frontend application in real time**.

Note: If all test steps are completed successfully, the Motion Detection and Automatic Lighting use case is considered fully implemented and functioning as expected. The system correctly detects motion, automatically controls the lighting, and updates the light status in the Smart Home application in real time.

Use Case 2 – Environmental Monitoring

This use case verifies that temperature and humidity data collected by the ENV Unit (DHT12 sensor) are correctly processed by the M5Stack device and accurately displayed in the Smart Home application. It also confirms that changes in environmental conditions are reflected in real time on both the m5stack device and the frontend application.

Note The same ENV Unit (DHT12 environmental sensor) measures both temperature and humidity values in real time, meaning that separate sensors are not required for these two measurements. These values are processed by the system and displayed directly on the M5Stack screen, allowing the user to observe current room conditions at any time.

Step 1 – Check Sensor Values on the M5Stack

A. Observe the M5Stack display and check the current temperature and humidity values.

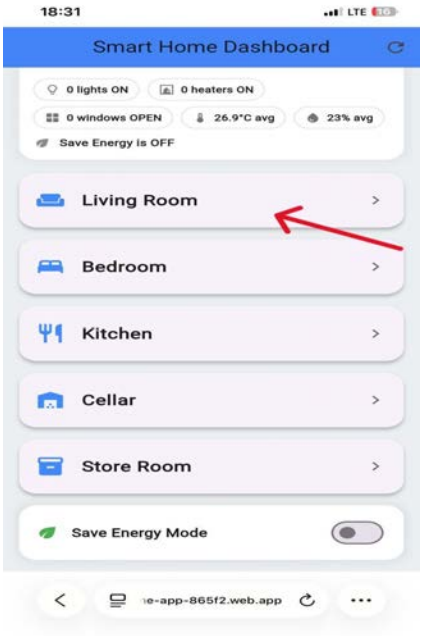
- Expected Results: The M5stack display shows the current room temperature and cellar humidity values as shown below

B. Verify Sensor Values in the Smart Home App by opening the URL link :https://smarthome-app-865f2.web.app

- Expected Results: Then select the living room or cellar to view the temperature and the humidity displayed in the frontend application and should correspond to the values shown on the M5Stack display as illustrated below .

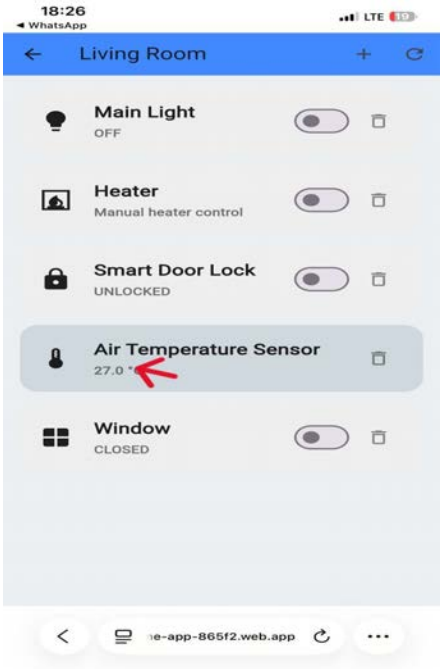
1.

1.click on the Living room icon



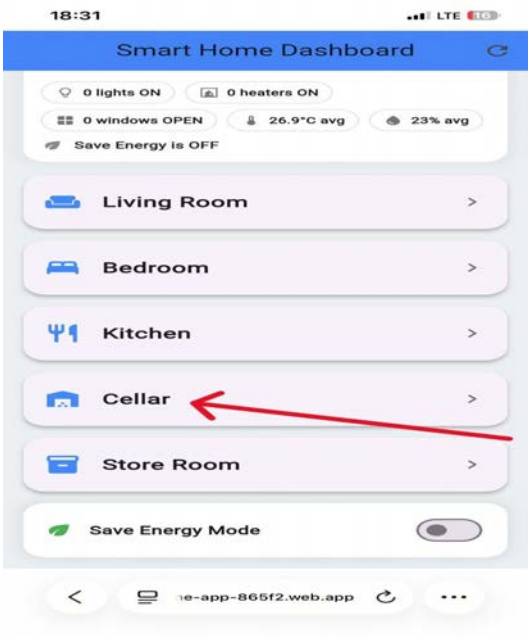
2. Room temp on m5stack indicated below

2. It directs you here
Below which show you the room temperature

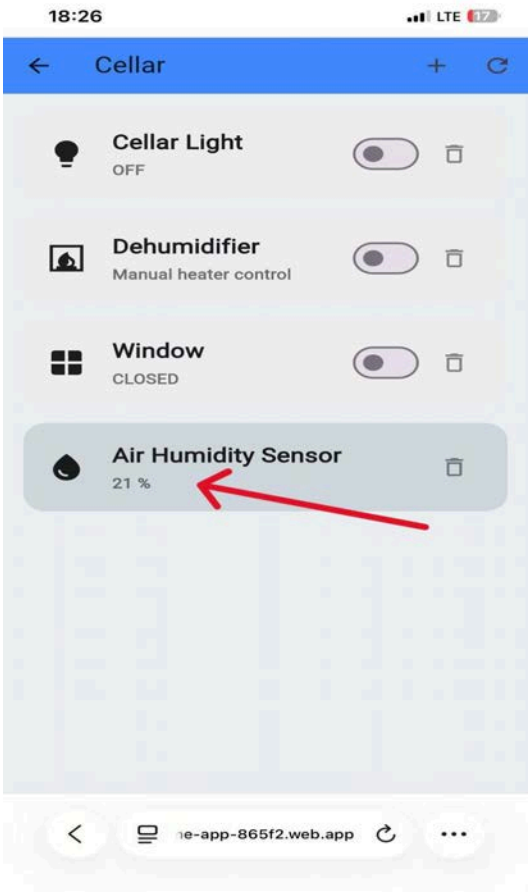


3. Cellar humidity on m5stack indicated below

3. Or click on the cellar icon



4. It directs you here
Below which show you the cellar humidity



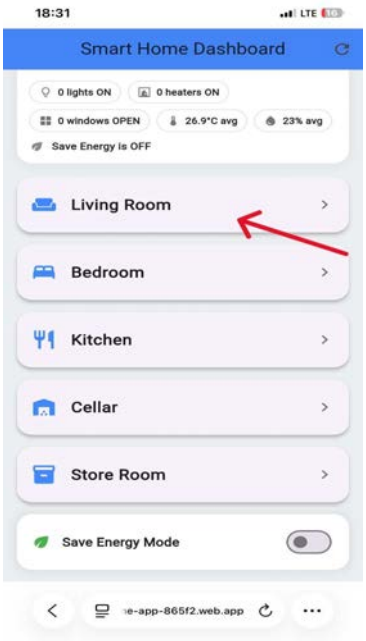
The test was successful. Temperature and humidity values measured by the DHT22 sensor were correctly displayed on both the M5Stack device and the Smart Home application. Changes in the environment were detected and reflected in real time, confirming proper sensor operation and data synchronization.

Use Case 3 – Security Status Monitoring

This use case confirms that the RGB LED indicates the current security status and that the frontend dashboard(mobile app) remains synchronized with the M5Stack in real time.

Step 1 – Simulate a Closed or Locked State

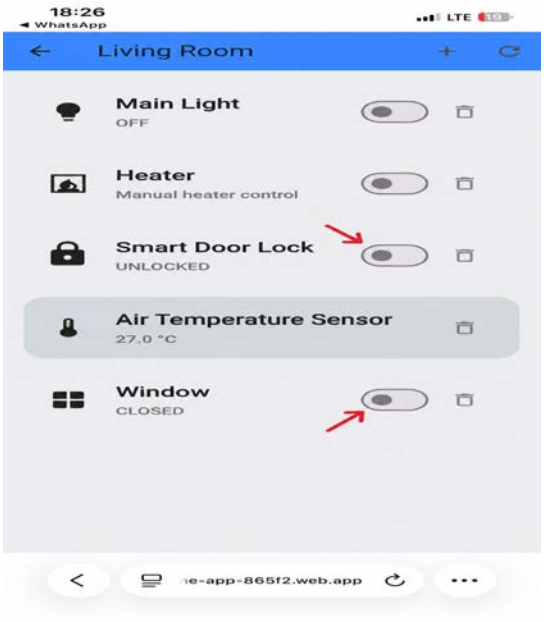
A. In the Smart Home application, click on any room icon, e.g Living room, Bedroom, Kitchen etc.
Expected results : In the image shown below i click on the icon Living room.

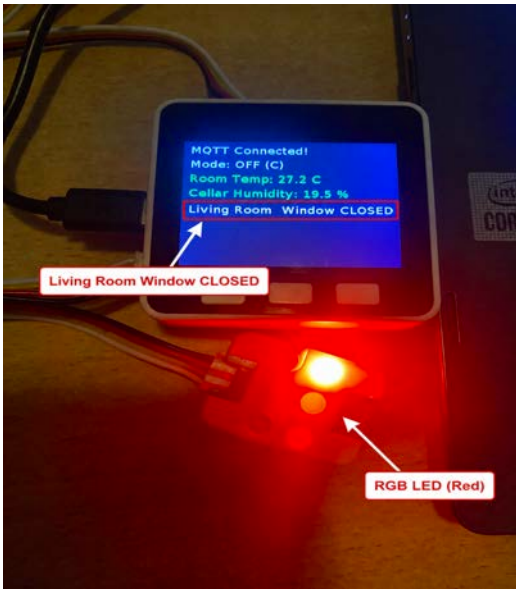


After clicking on the living room, you will view all the things available in the Living Room as illustrated below and slide the control to set the window or smart door lock to a closed or locked state.

B. The M5Stack device screen displays the current status of the **window** or **smart door lock** on the screen.

- Expected Results: When the window is **closed** or the door is **locked**, the RGB LED changes to **red**, indicating a secured state as shown below.

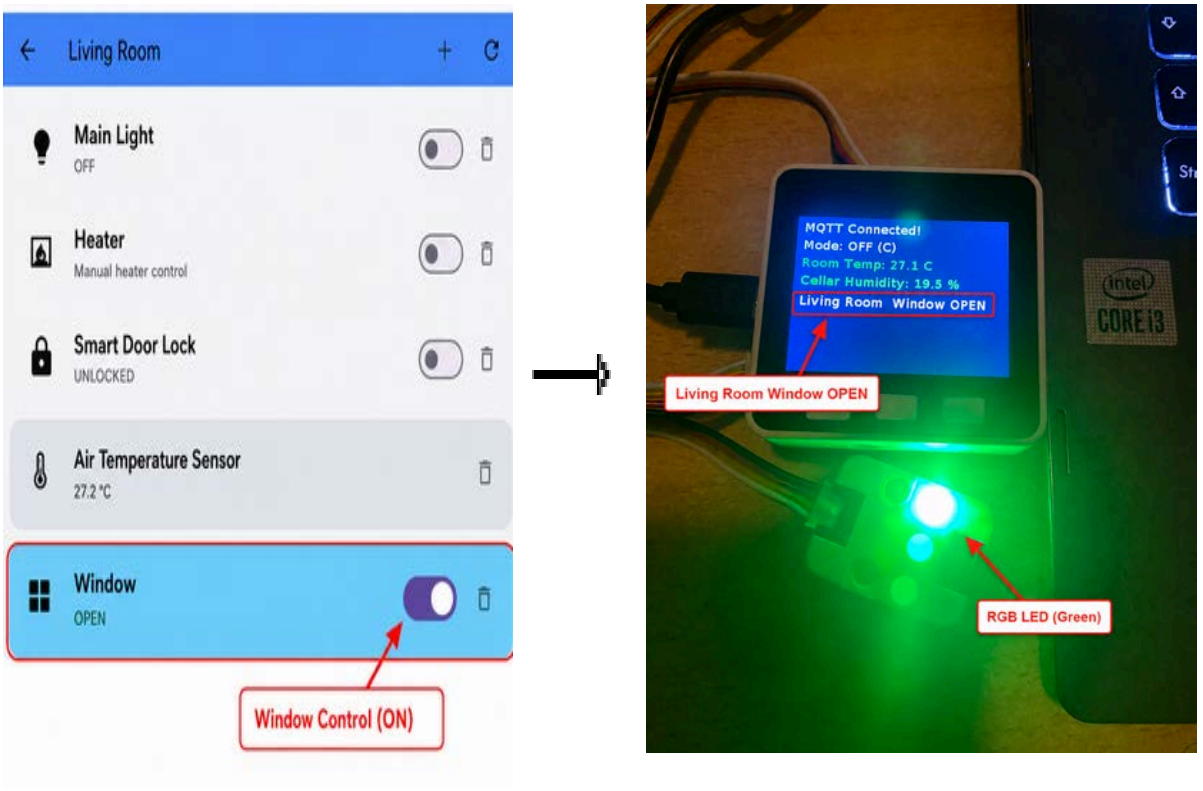




.

Then slide the Window control to the ON position to open the window, as shown below.

- Expected Results : The M5Stack receives the status message and displays **"Living Room Window OPEN"**. The RGB LED changes to **green**, indicating that the window is open as shown below.



If all tests are successful, the Security Status Monitoring use case is considered fully implemented. The results confirm that the Smart Home application correctly sends security status updates to the M5Stack device. The M5Stack accurately displays the current status of the window or smart door lock, while the RGB LED provides a visual indication of the security state (red for CLOSED/LOCKED and green for OPEN/UNLOCKED).

The same testing procedure can also be applied to other rooms in the Smart Home system, allowing verification of the corresponding windows and smart door locks throughout the application.

Option B – Recreate the System Step by Step

If you would like to understand how the Smart Home system was developed or recreate the project yourself using UIFlow, continue with Section 3 – System Implementation and Use Cases.

3 Use Case 1 – Motion Detection and Automatic Lighting

The system detects movement in monitored rooms and automatically triggers actions such as lighting to turn ON/OFF. The room can be selected manually using the M5Stack buttons. The steps shown in the instructions allow you to detect motion in the Keller(basement) or store room by press A or B bottom. If a motion is detected lights in the keller or store room will turn on automatically.

3.1 Overview

The system performs the following actions:

Movement detection in the cellar

● The PIR sensor detects movement in the cellar area.This allows the lights to activate

Movement detection in the storage room

● The PIR sensor detects movement in the storage room. This allows the system to automatically activate lights or notify the system.

Note : The prototype uses only one M5Stack and one PIR sensor. The two rooms are simulated as different operating modes. Button A selects the cellar and Button B selects the storage room, while the same PIR sensor is used for motion detection in both scenarios.

Automatic shutdown after inactivity

● If no motion is detected for 10 seconds, the system sends:The system allows lights or devices to turn off automatically.

3.2 Components Required

● M5Stack Core / Core2

- PIR Motion Sensor

- Grove Hub needed to connect multiple units at the same time(motion sensor + RGB Led)

- RGB LED

- Grove Cables

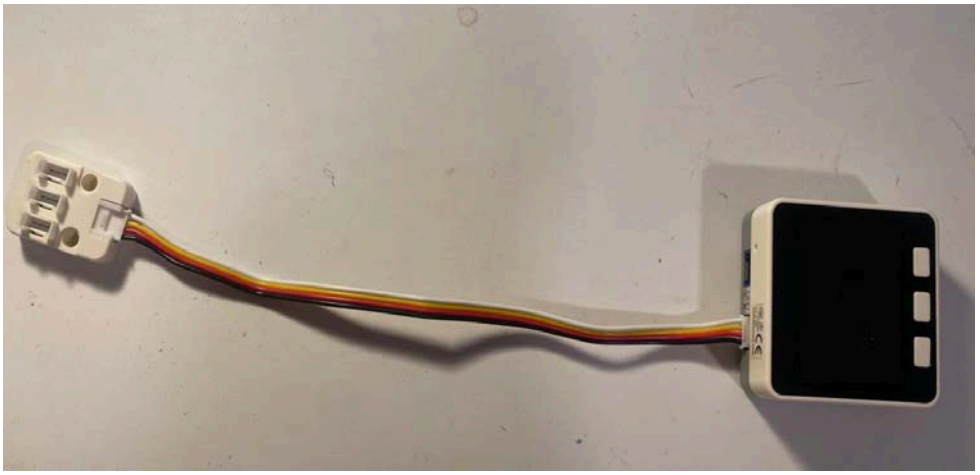
- USB Type-A to USB Type-C cable

3.3 Hardware Setup

This section shows how to connect the m5stack and its components mentioned in section 3.2 . Follow these steps carefully:
Below is a step by step instruction for the task use case 1(Motion Detection and Automatic Lighting)

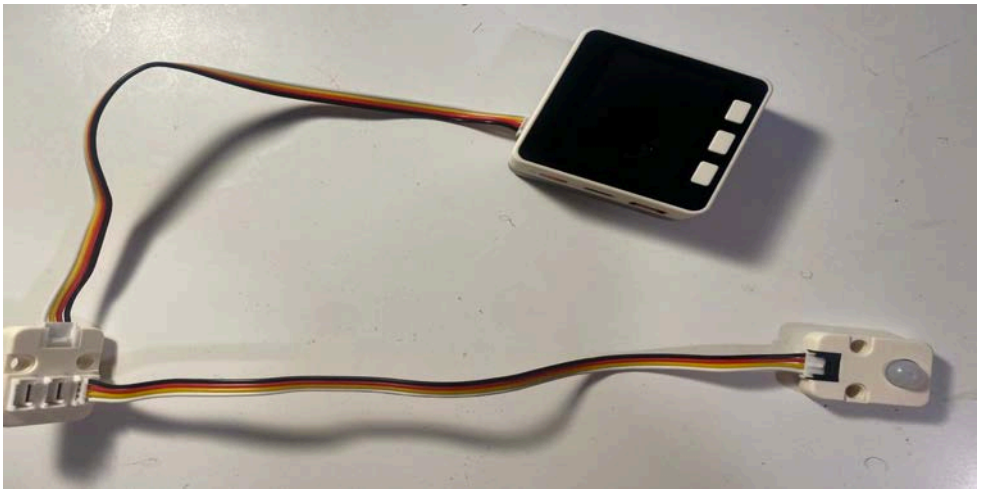
Step 1

Connect the Grove Hub to Port B of the M5Stack using a Grove cable.This enables other components to be connected together.



Step 2

Connect the PIR motion sensor to Port B via the Grove Hub using a Grove cable.



3.4 Software Setup in UIFlow

This section will now show how to add units in UIFlow-Online so that they are recognized by the program.
Go to: https://flow.m5stack.com and connect the M5Stack to your PC using the included USB Type-C to Type-A cable and make sure that your m5stack device is successfully burned by following the steps explained in section 2.

3.5 Implementation

3.5.1 Add Required Units Instructions for use case 1

After a successful connection is established and the M5Stack and UIFlow are ready to use follow the following steps.

Step 1

Click on the **"+ (Plus) icon"** in the Units section to display the list of available components that can be added to the project as shown in figure 9 .

Figure 9: UIFlow Units (Plus icon)

Step 2

From this list, select the **PIR Sensor** then confirm by clicking **OK**. The PIR sensor is required to detect motion in the selected room.

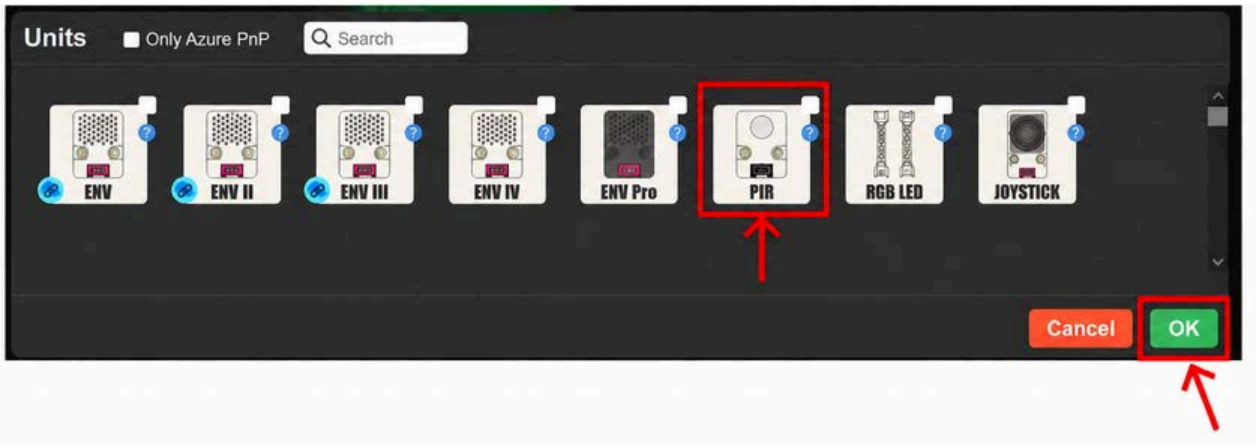


Figure 10: Adding units in UIFlow

After adding the unit by pressing ok, go to the right side category section and click on Units and the units added will appear. In the case PIR unit appeared since it was the only unit added as shown below.

Note: The PIR sensor will be used in at section 3.5.6 Create the Motion Detection Loop. For now continue with the section below to create the User interface on the M5stack screen.

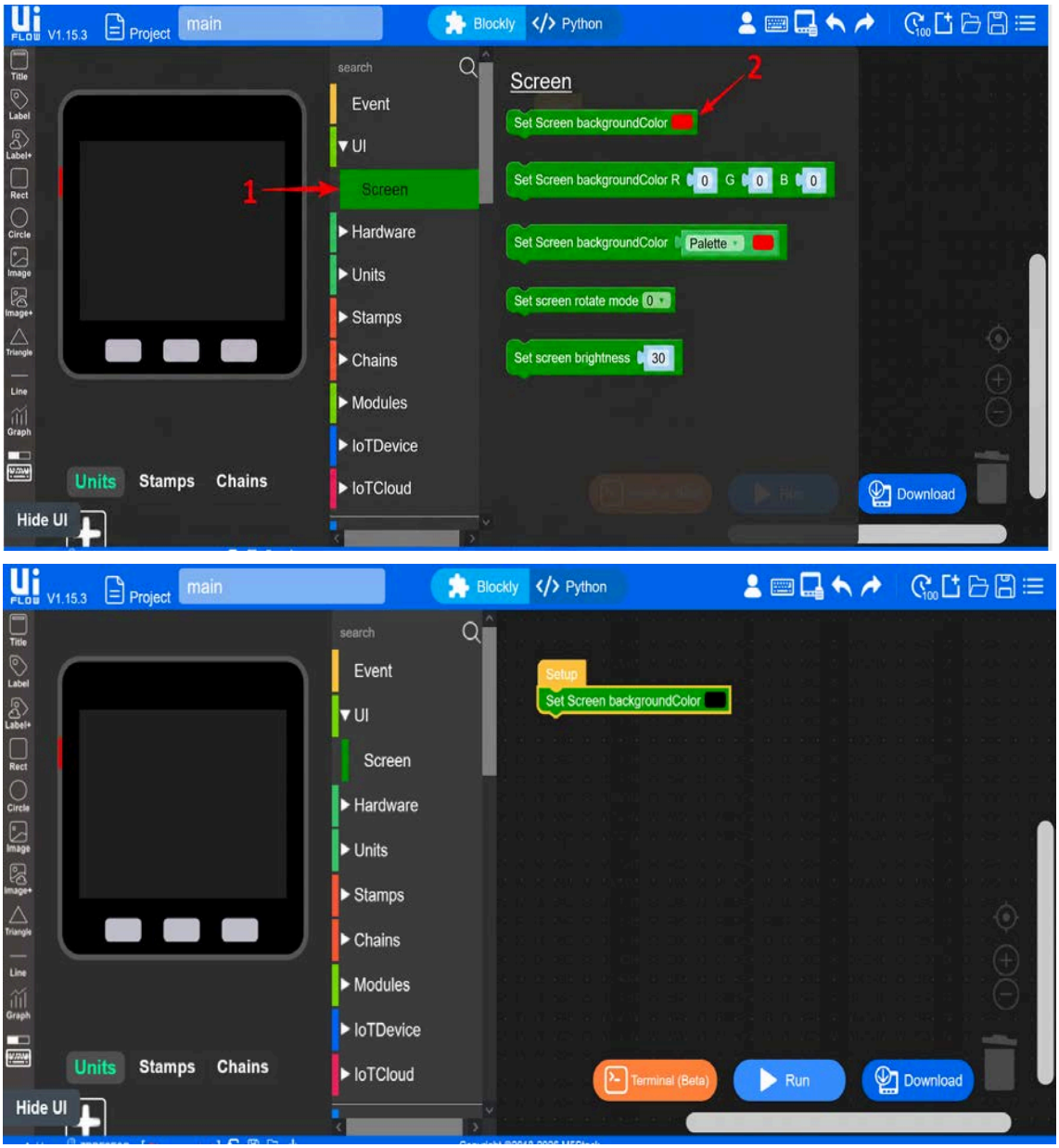
3.5.2 Create the User Interface on the M5Stack screen.

In this step, the user interface is created on the M5Stack screen using UIFlow. The following instructions describe how to add and position label elements for displaying information.

1. In UIFlow, locate and open the UI category at the right side . From there, click on the screen as shown below. The aim is to set the background color of the M5stack. and drag the label element onto the M5Stack screen. This will place a text field on the display where information can be shown.

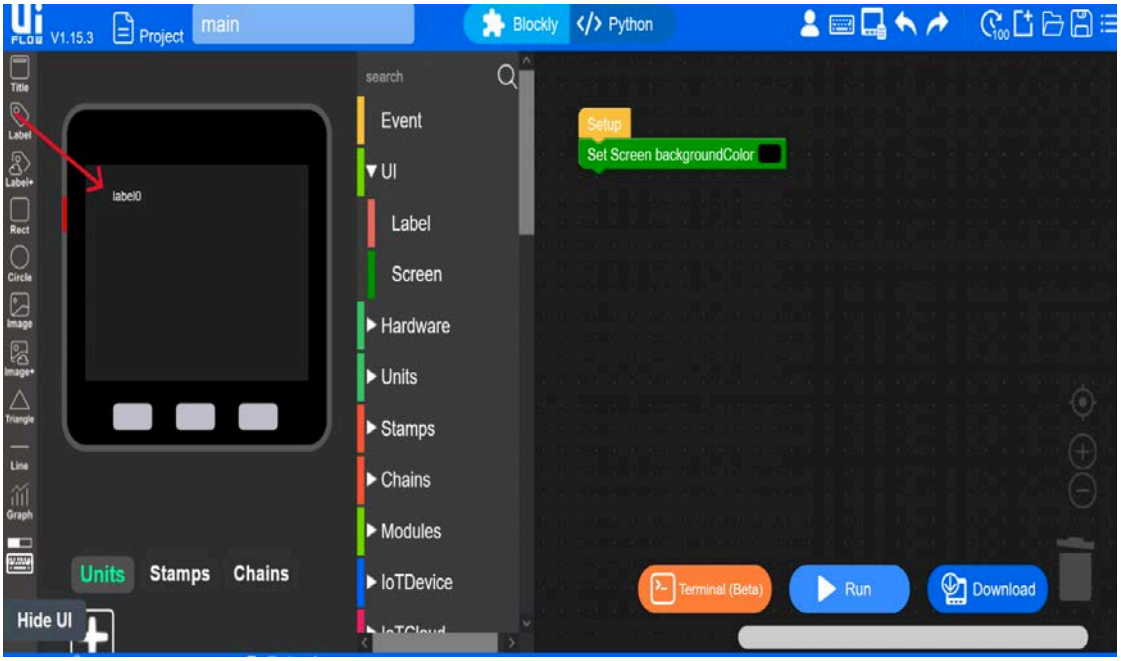
Figure 11: Create the User Interface on the M5Stack screen.

2. After clicking the screen, drag the set screen background color and place it onto the blocky area under set as demonstrated below.

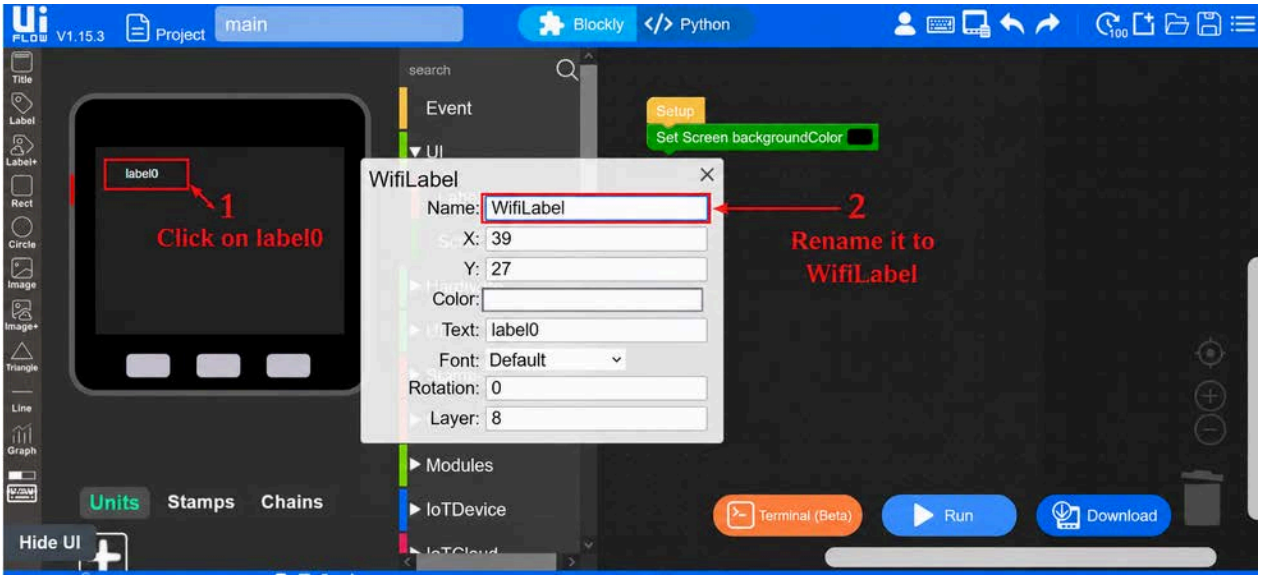


Note: In this case i set my M5stack screen background to be black

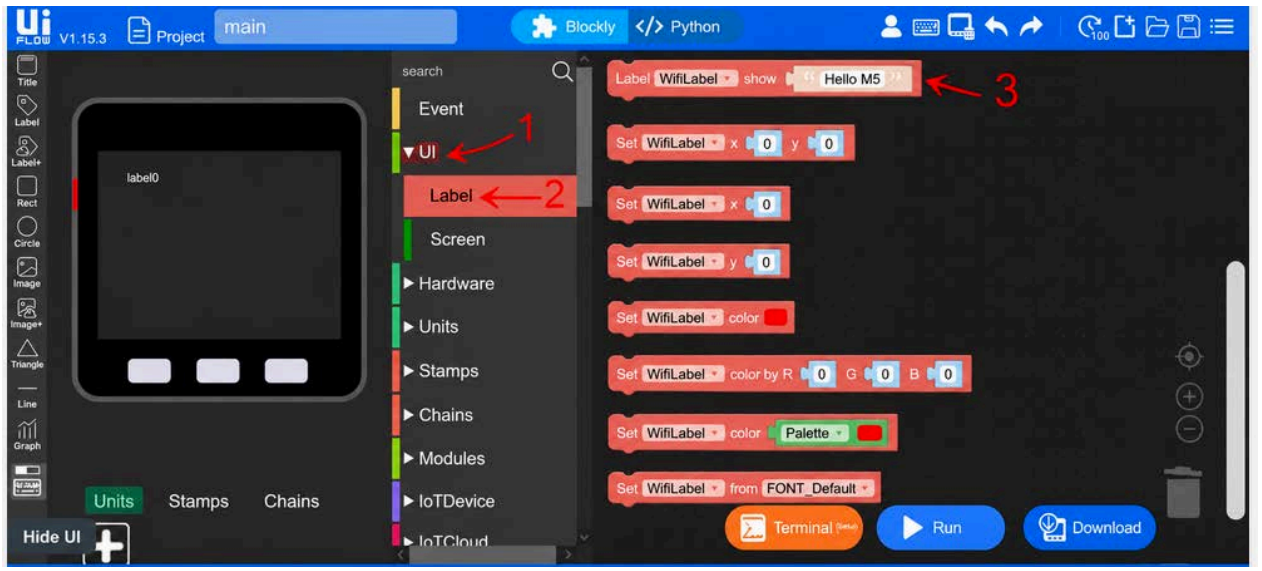
3. Next, go to the top left of the screen and drag the label element onto the M5Stack screen as shown below. The aim is to create the user interface onto the M5stack screen.

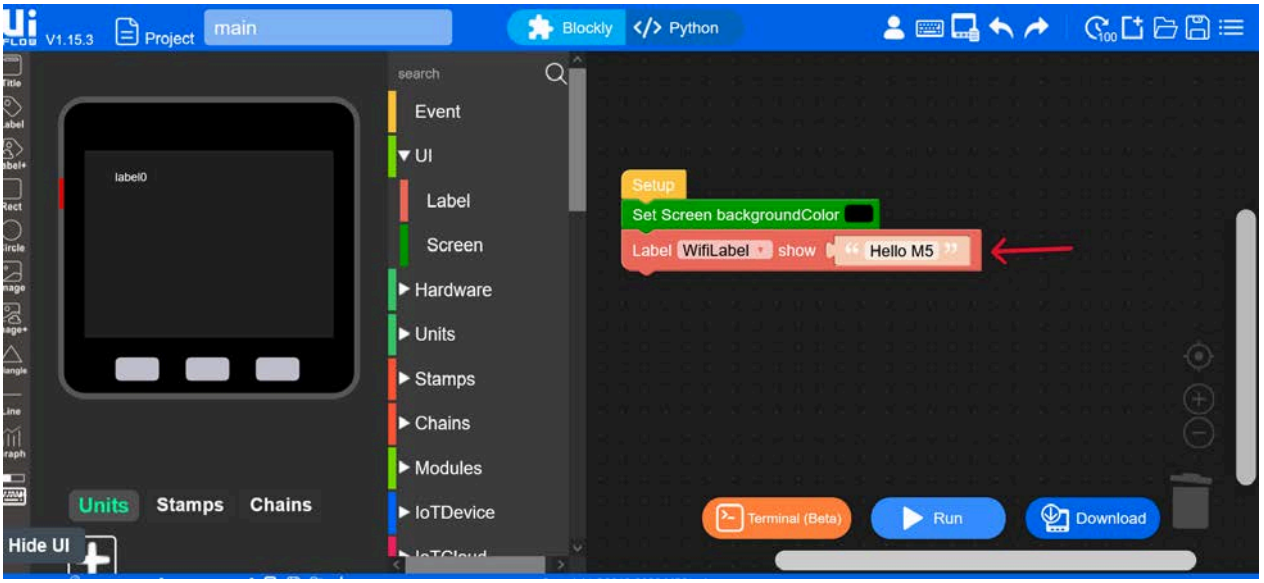


4. Next click on the label0 and rename it WifiLabel as shown below.

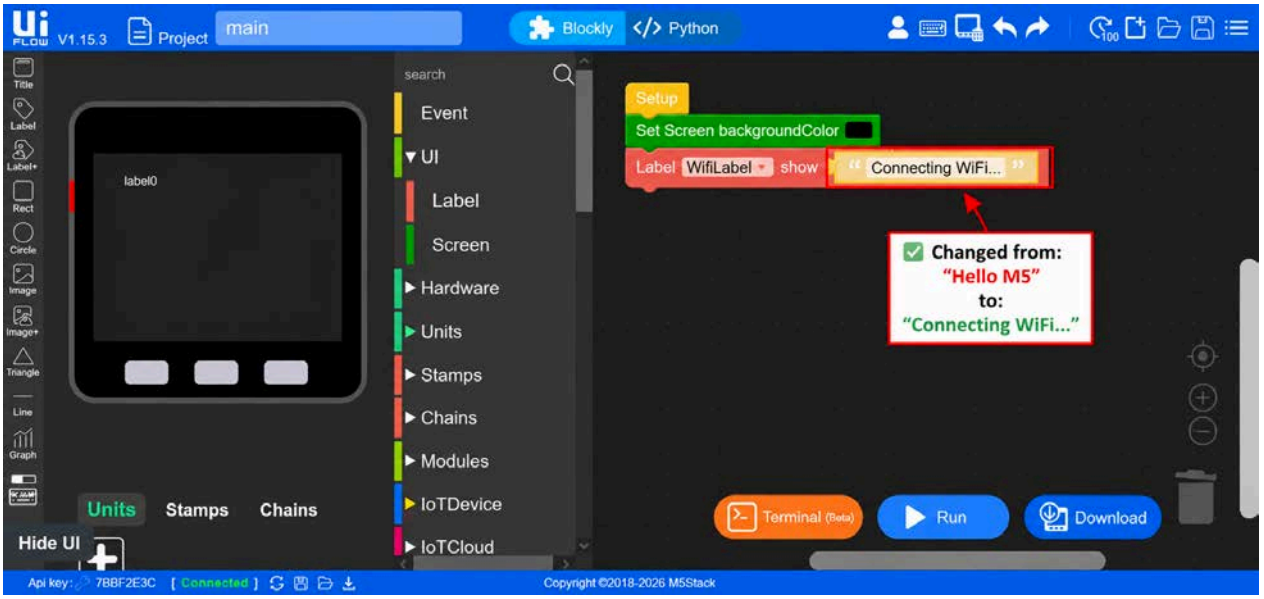


5. Afterwards, under the UI category click Label and drag the block Label WifiLabel Hello m5 on to the blocky area and place it under the set screen background color as shown below.



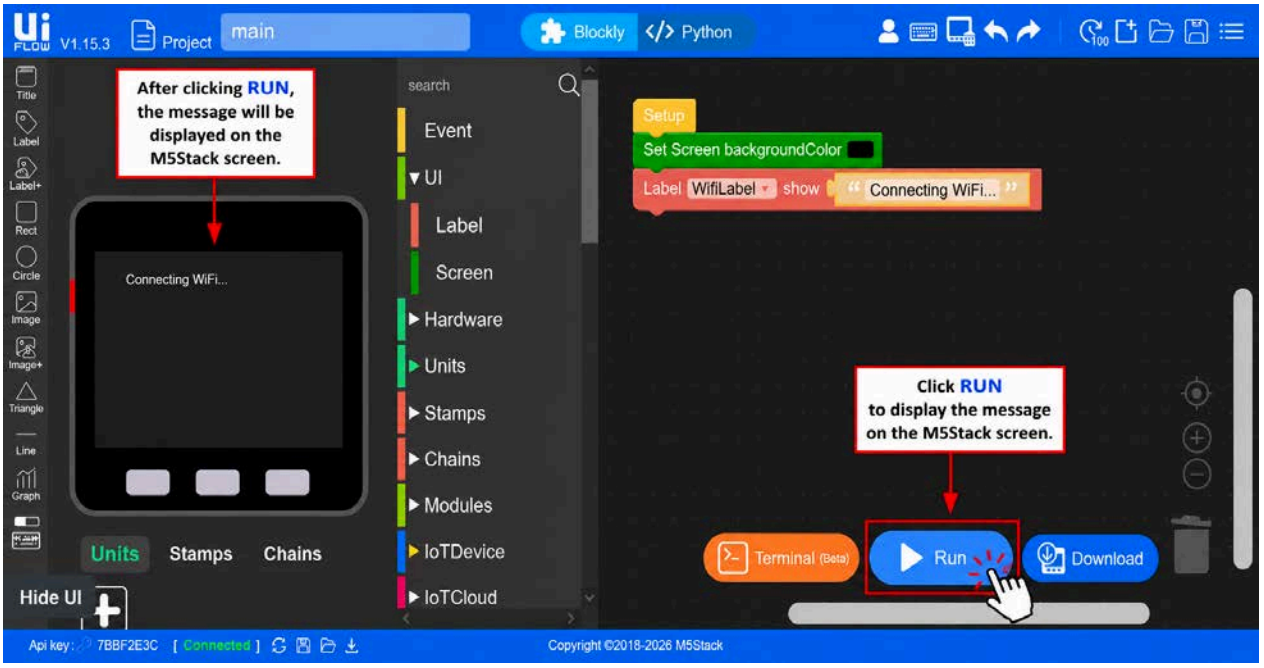


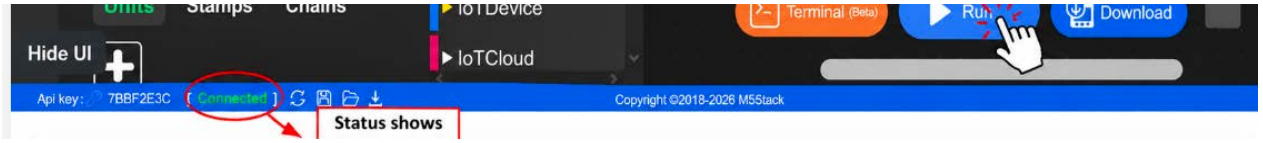
6. Inside this block, click on the text field which says Hello M5 and enter: Connecting WiFi…This block is responsible for displaying text on the M5Stack screen as illustrated below. When the program starts, the message *"Connecting WiFi…"* will appear, informing the user that the device is currently attempting to connect to a wireless network.



Note :This label plays an important role in user interaction, as it provides immediate feedback about the system's current state. It will be updated later in **(Wi-Fi Connection)** to display *"WiFi Connected!"*, indicating that the connection has been successfully established.

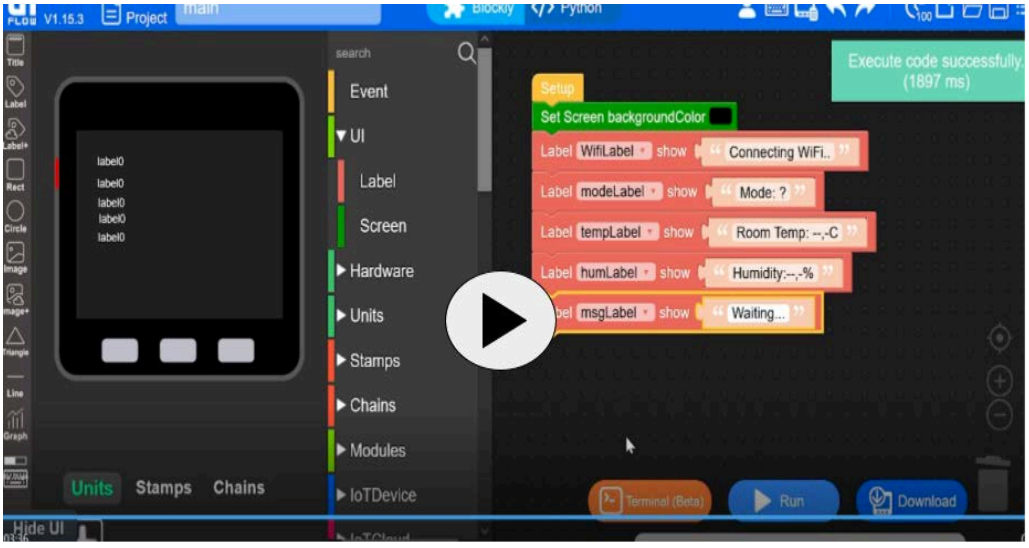
6.1 Then click run to display the message on your M5stack screen as shown below.





Note: Status must show connected before it shows successfully on the m5 stack screen.

7. After creating the first label, Below is a video showing how to create the remaining screen labels using same process. Go to the UI section, drag another Label element onto the screen, and then in the block area select “Label → label → show”. Enter the corresponding text for each label as follows:
Mode: ?
Room Temp: --.- C
Humidity: --.- %
Waiting…



Video tutorial demonstrating the creation of the remaining labels.Click the image to watch the complete video.

Note: At this point, the labels display placeholder values (`?` and `--.-C,--%`) and no real values because no ENV Unit (DHT12 Temperature + Humidity sensor) and other components has been read yet. These values are automatically replaced with real-time information once the program connects to WiFi, receives sensor readings, and begins execution.

These labels are used to display different types of information during program execution. For example, the Mode label shows the selected room mode, the temperature and humidity labels display environmental data, the message label shows system updates, and the PIR label indicates whether motion is detected.

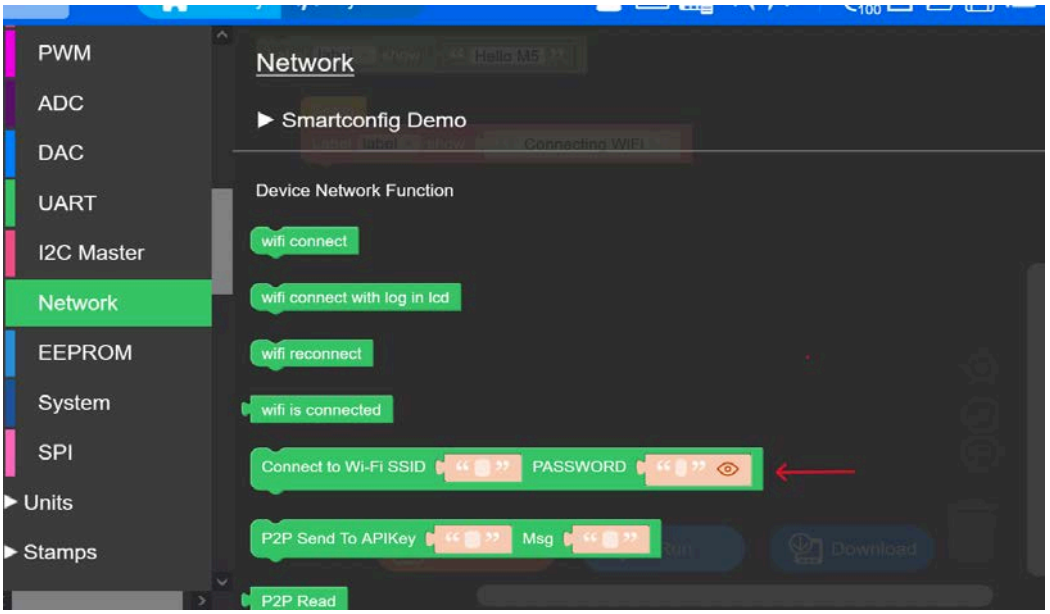
3.5.3 Establish Wi-Fi Connection

This part enables us to establish the WiFi connection in UiFlow so the M5Stack can communicate with the backend via MQTT.

1. In UIFlow, the Network (WiFi) section is located in the block menu on the right side of the screen. Look at the **right panel**, where different categories are listed and click on Hardware.

Establish Wi-Fi Connection

2. Scroll through the list of hardware categories and select Network. After clicking it, you will see blocks related to Wi-Fi, including the WiFi connect block, which is used to connect your device to the internet.

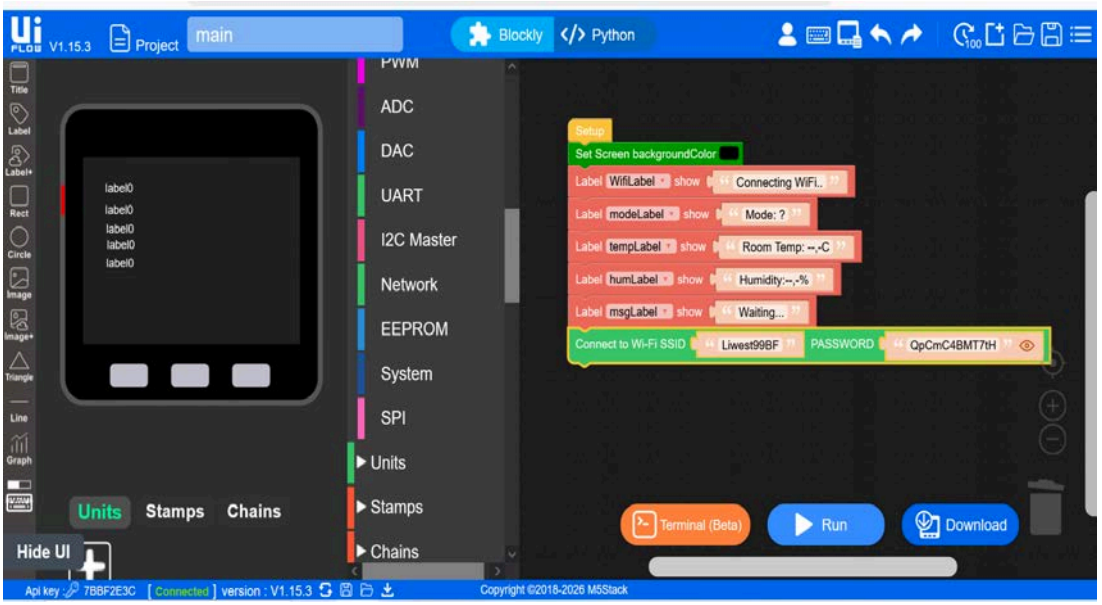


This image shows WiFi setting

3. Then click or add the **WiFi connect block** from the **Network section** and place it under the msgLabel in the setup area. Enter your **Wi-Fi name (SSID)** and **password** into the corresponding fields.

⬇⬇

Below is image showing after adding my WiFi and password



4. After adding connect to Wifi block, add a label block to display the message confirming that the M5stack has successfully been connected to the Wi-Fi network as shown below.

Note: If the message **"WiFi connected"** does not appear, it means that the device has not successfully connected to the Wi-Fi network. In this case, verify that the SSID and password are entered correctly. Then, reset the M5Stack by unplugging it from your computer's USB port, plugging it back in, and clicking **Run** again to restart the program.

3.5.4 Configure MQTT Communication

After the M5Stack is connected to Wi-Fi, the next step is to configure MQTT communication. MQTT allows the M5Stack device to send and receive messages from the Smart Home application.

In this project, the public MQTT broker **broker.hivemq.com** is used. The M5Stack connects to this broker and uses different MQTT topics to exchange information with the frontend application.

Step 1

After the Wi-Fi connection is established, go to the block menu on the right side, scroll down and open the “Advanced” section as shown below.

Step 2

In the block menu, open the MQTT section and drag the Init block into the setup area. Enter the following configuration as demonstrated below:

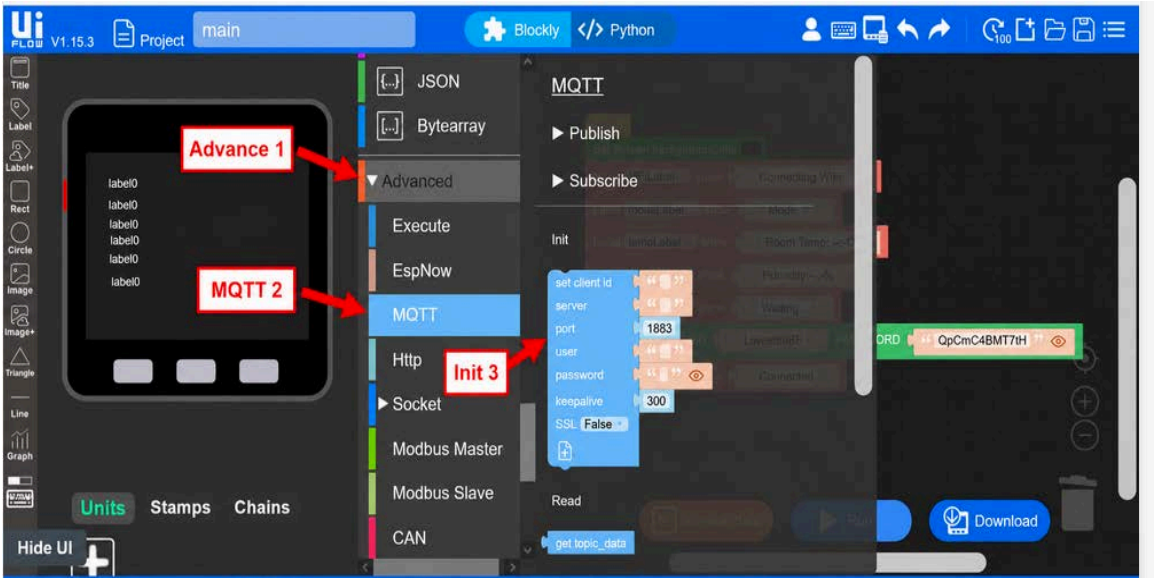
- Client ID: *m5stackClient*
- Server: *broker.hivemq.com*
- Port: *1883*

  User:(leave blank)

  Password:(leave blank)

  Keepalive: 300

  SSL: False



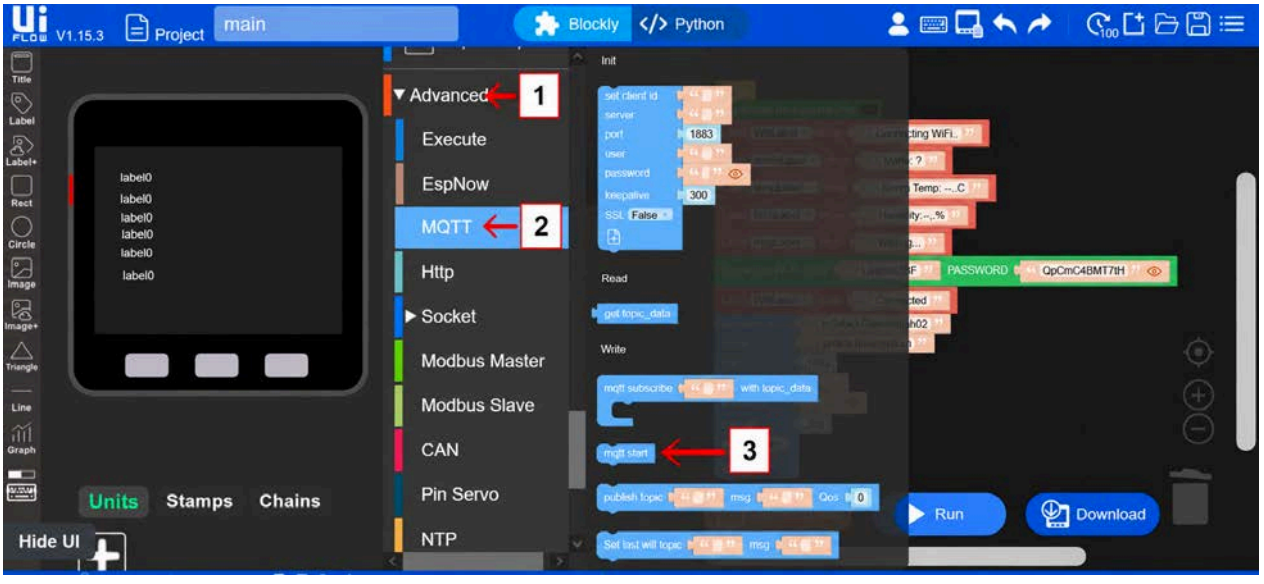
⬇⬇

Below is image showing after adding the init block

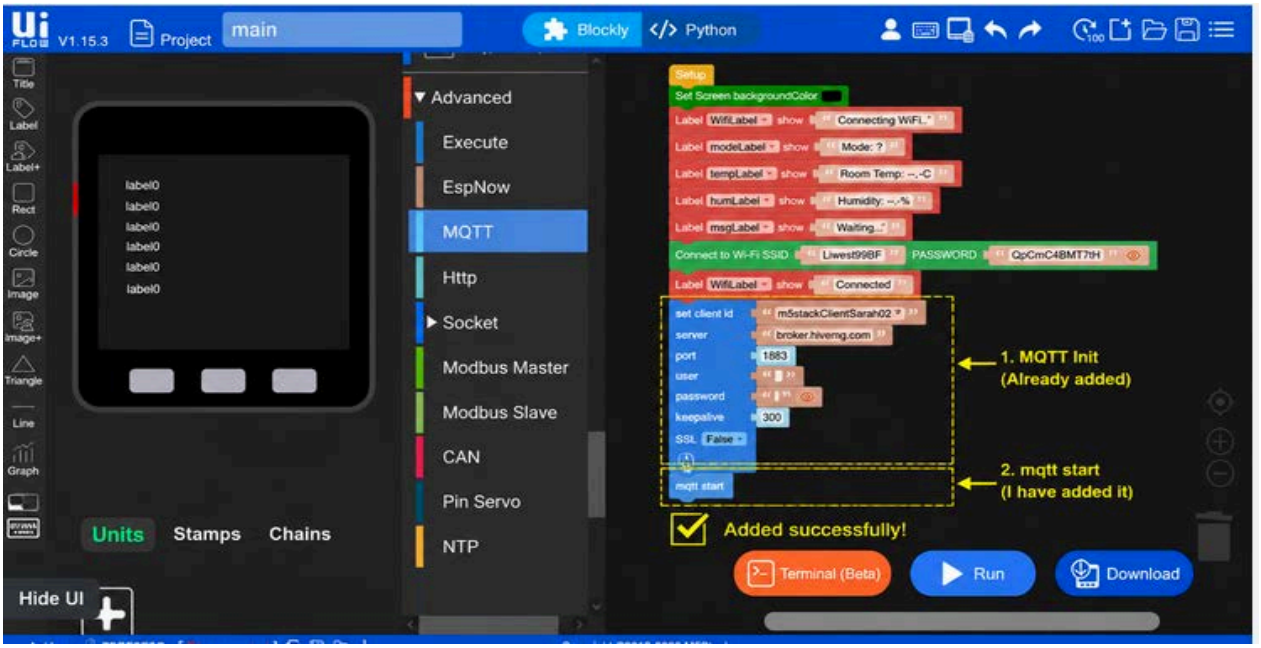
Step 3

After configuring the **MQTT Init** block, the next step is to start the MQTT connection. In UIFlow v1.15.3, this is done using the **mqtt start** block.

3.1 Go to Advanced → MQTT and Select the mqtt start block as shown below.



3.2 Drag the **mqtt start** block into the workspace and place it directly below the **MQTT Init** block as illustrated below.



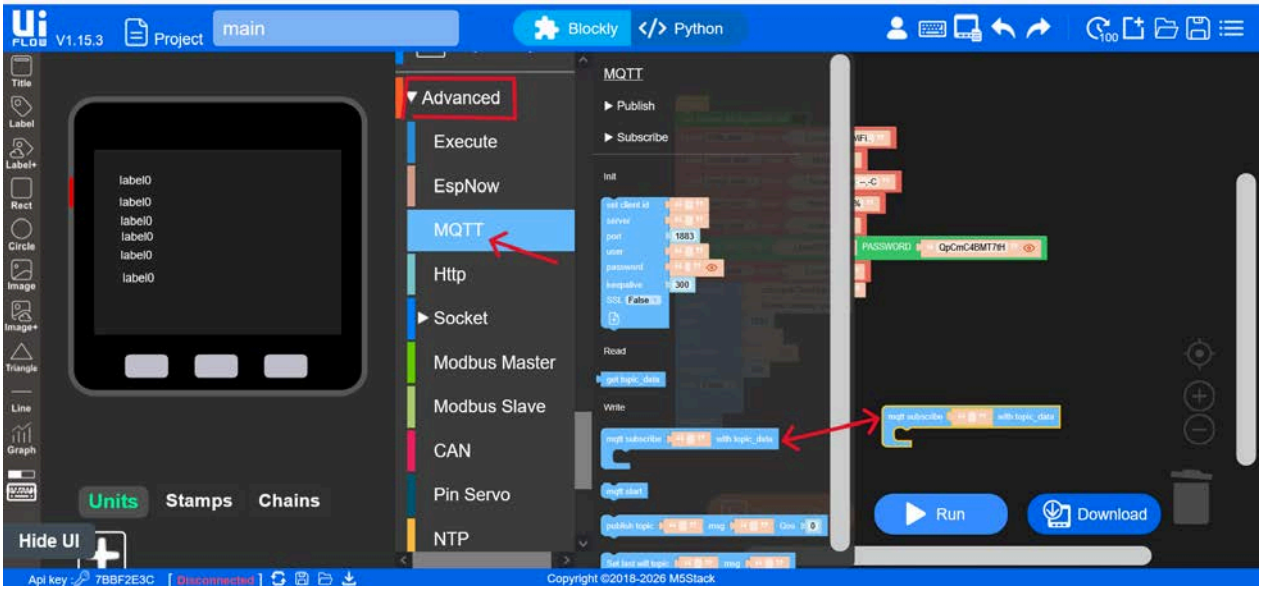
Note: The **MQTT Init** block only stores the MQTT connection settings. The actual connection to the broker starts when the **mqtt start** block is executed. Without this block, the M5Stack cannot publish or receive MQTT messages.

.

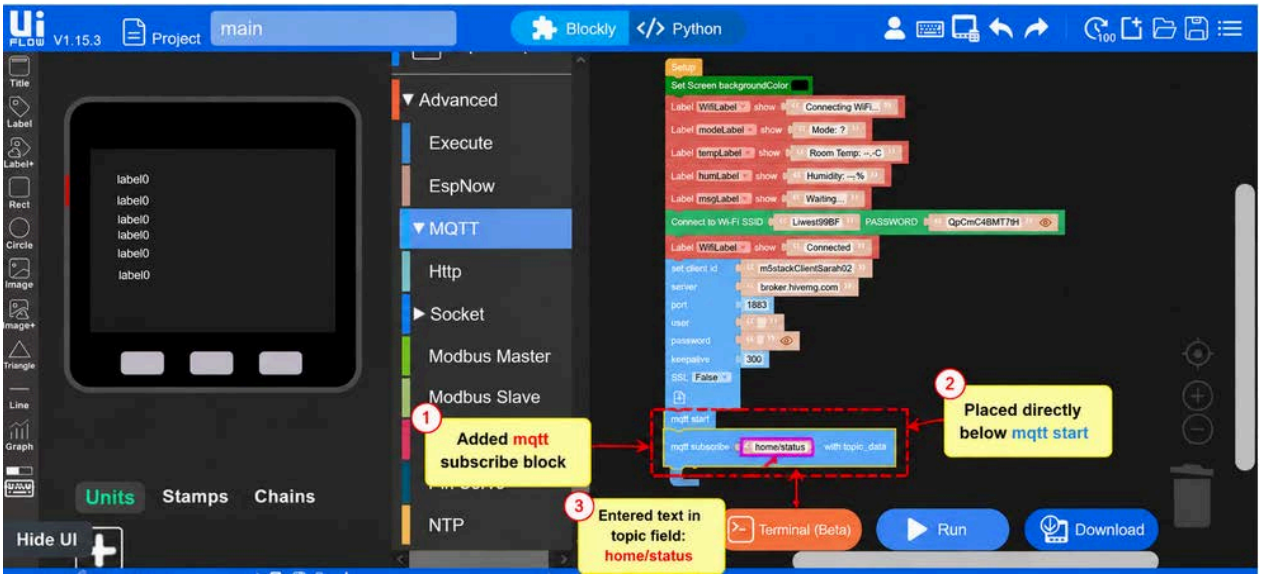
Step 4

After starting the MQTT connection, the M5Stack must subscribe to an MQTT topic. Subscribing enables the device to listen for messages published to that topic, allowing it to receive updates from other devices in the smart home system.

4.1 Under **Advanced**, select **MQTT** , in the **Write** section, drag the **mqtt subscribe** block into the workspace blocky as shown below .



4.2 Place the **mqtt subscribe** block directly below the **mqtt start** block and in the **topic** field, enter: home/status



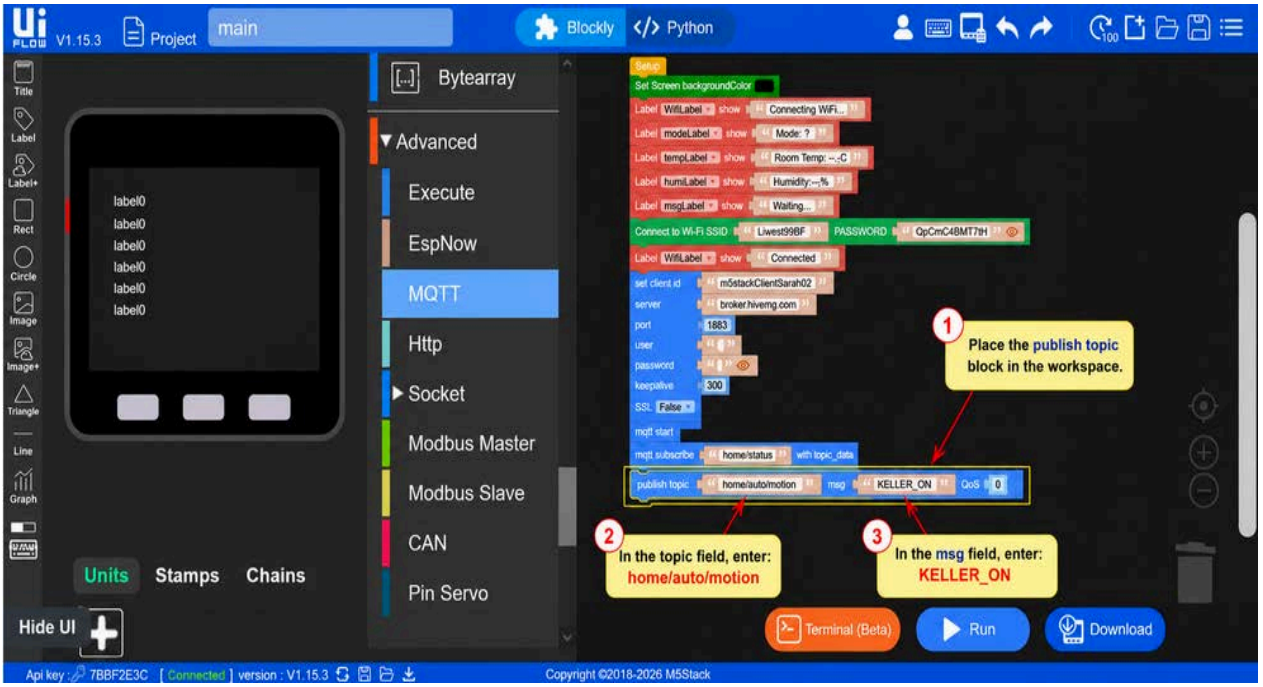
Step 5

After subscribing to the **home/status** topic, the next step is to publish messages from the M5Stack to the MQTT broker. Publishing allows the M5Stack to send information to the Smart Home application.

Note: In this use case, the M5Stack publishes motion detection messages to the topic: home/auto/motion

5.1 Go to Advanced → MQTT and in the Write section, drag the publish topic block into the workspace as shown below.

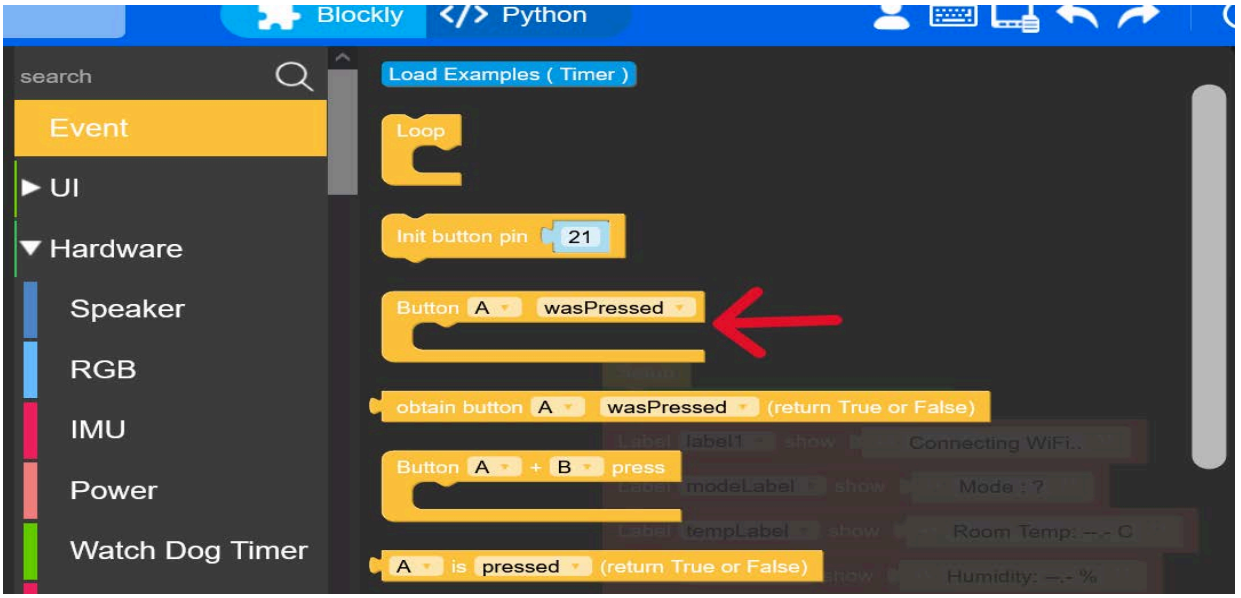
5.2 Add the **publish topic** block in the workspace inside the **mqtt subscribe** block and In the **topic** field, enter:home/auto/motion also in the **msg** field, enter the message that should be sent: KELLER_ON as shown below



3.5.5 Use Buttons to Select the Mode

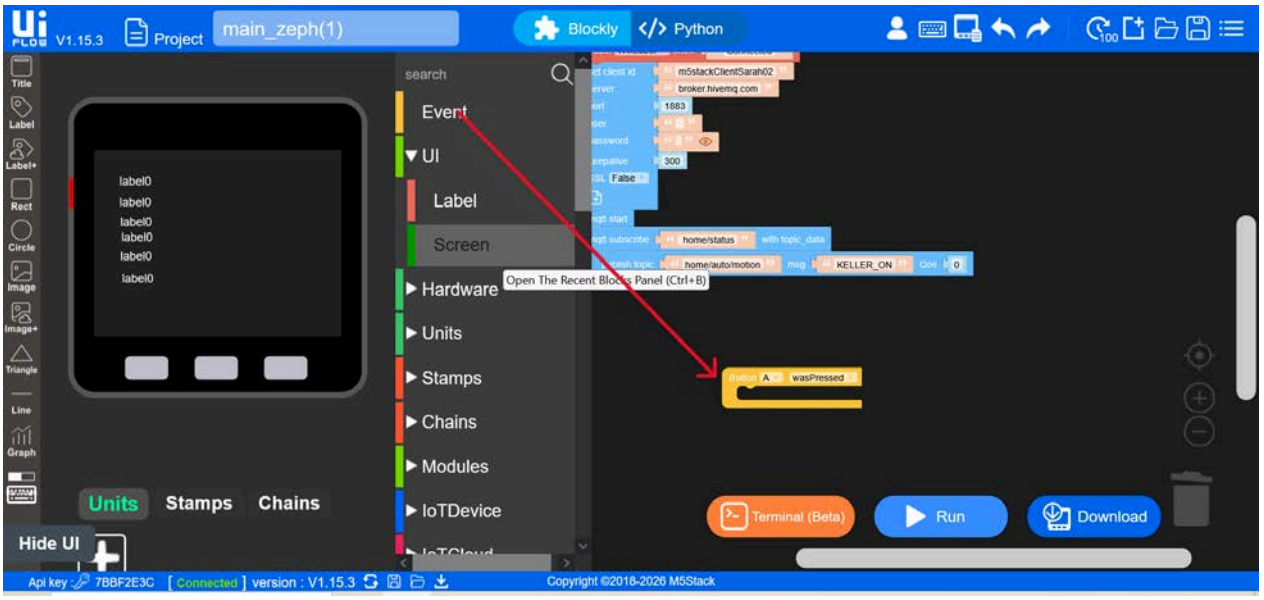
This step allows the user to control the system by selecting which room should be monitored.The selected mode is also shown on the M5Stack screen so the user can clearly see the current system state.

Step 1
In UIFlow, look at the block menu on the right side. Click **Event**. Drag the **Button A wasPressed** block, Button B, and Button C into your program. These buttons will be used to allow the user to select which room should be monitored.
For example:
Button A → Mode: CELLAR
Button B → Mode: STORE ROOM
Button C → Mode: OFF



⬇⬇

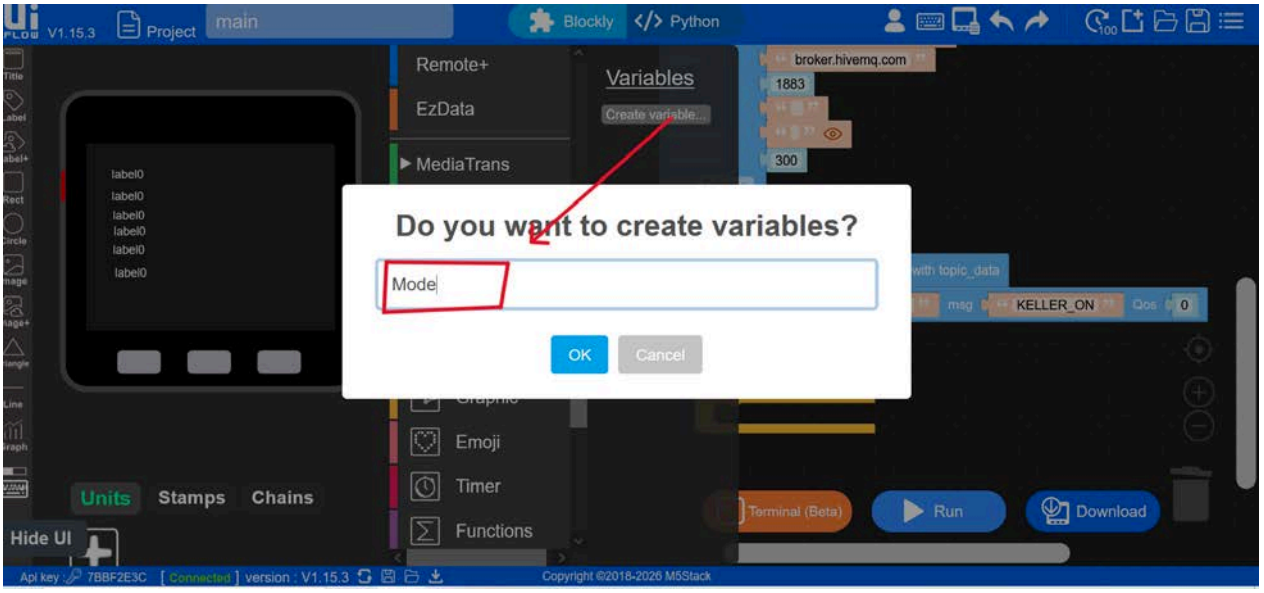
Below is the image after dragging the Button A block onto the blocky area

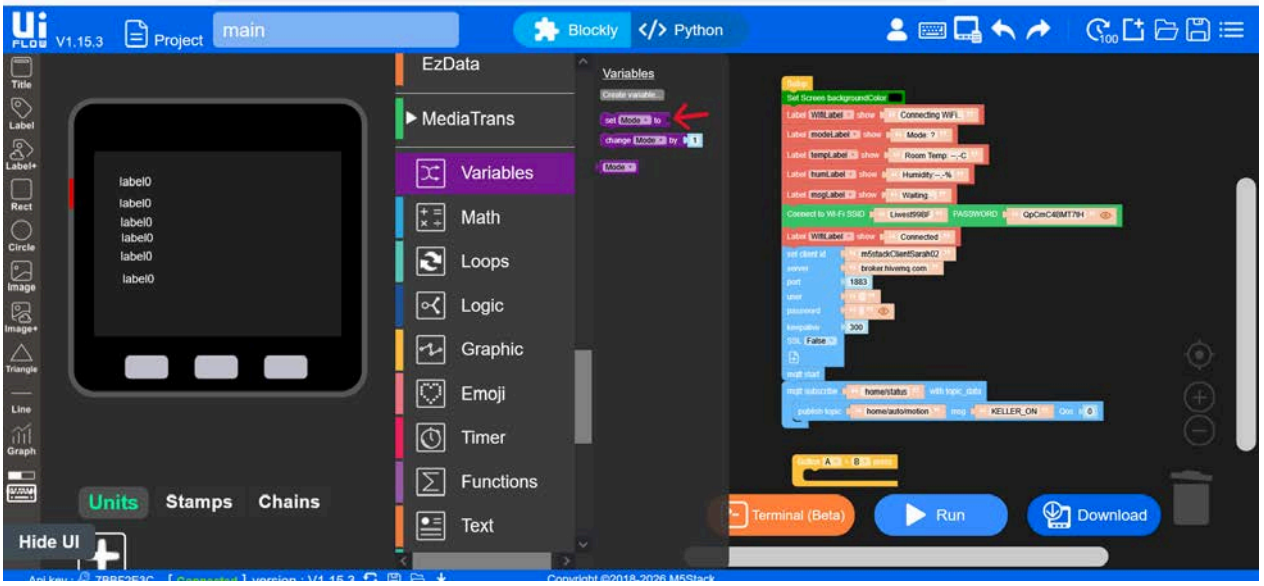


Note: This button will be used for: CELLAR mode

Step 2

Go to variables located on the block menu at the right side and create a new variable called: Mode and Set its starting value to: 1 as shown below



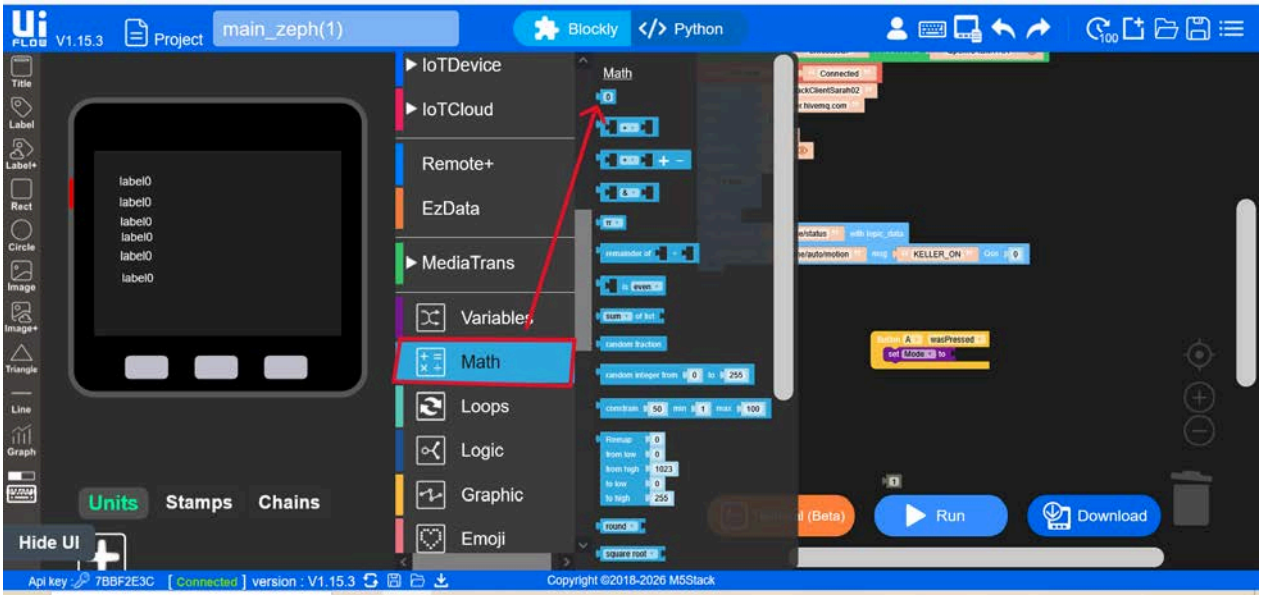


⬇⬇

Drag the set mode variable and insert it inside the **Button A wasPressed** block(Cellar)

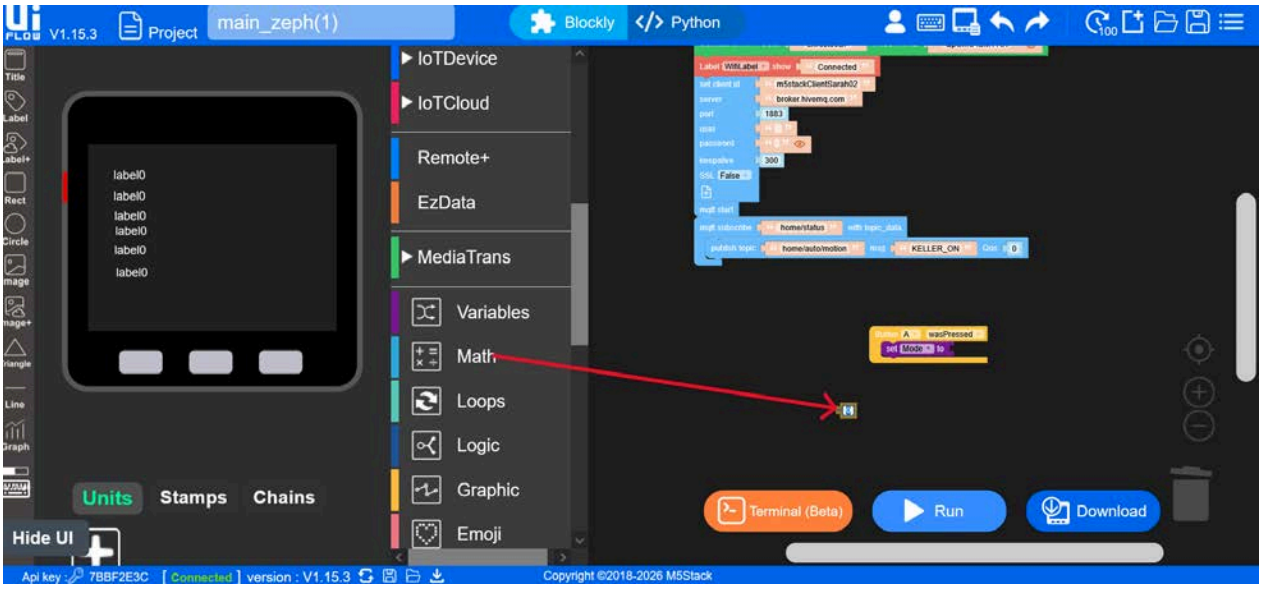
⬇⬇

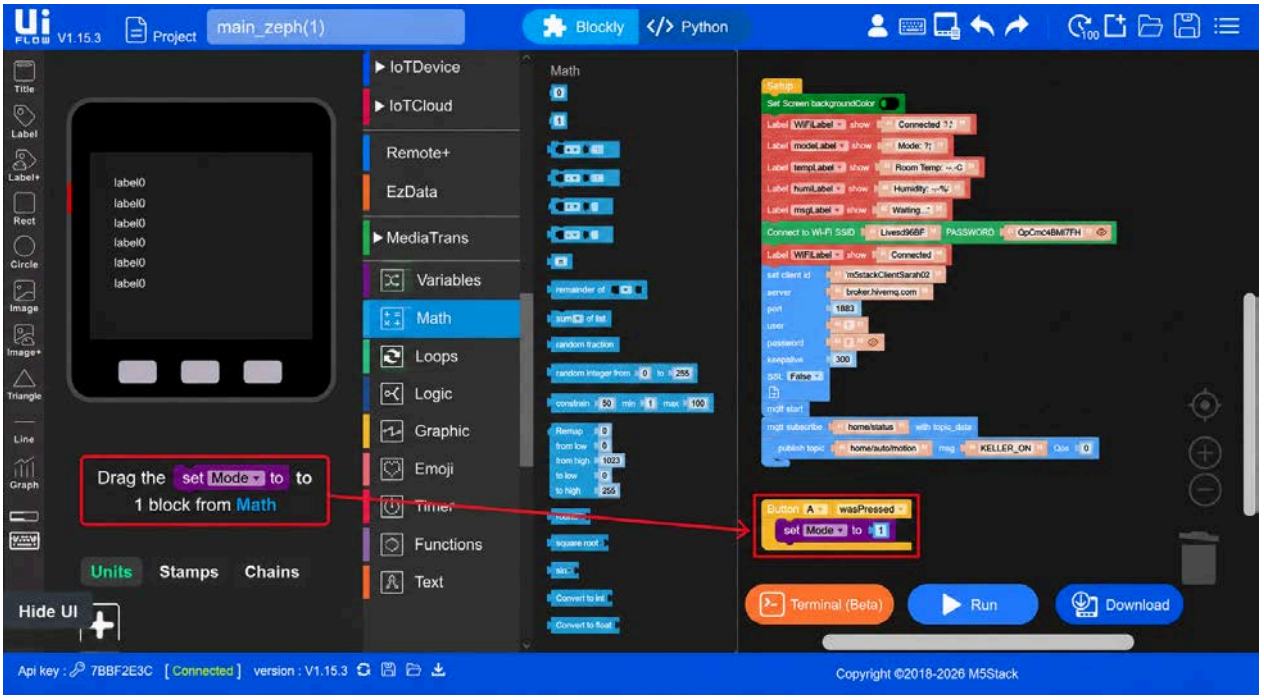
Note: to set the value to 1 go to math under Variables and drag the block as shown below on to your blocky area



Step 3

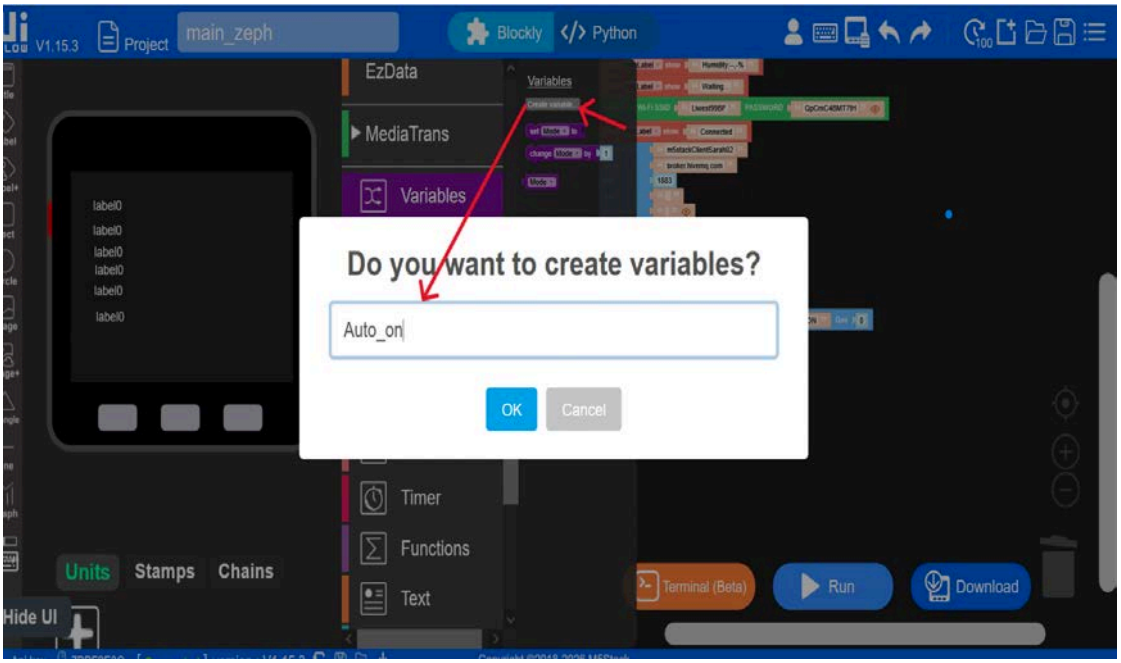
Inside the **Button A wasPressed** block, drag the set t variable *mode* to **1** to activate cellar mode as shown below.

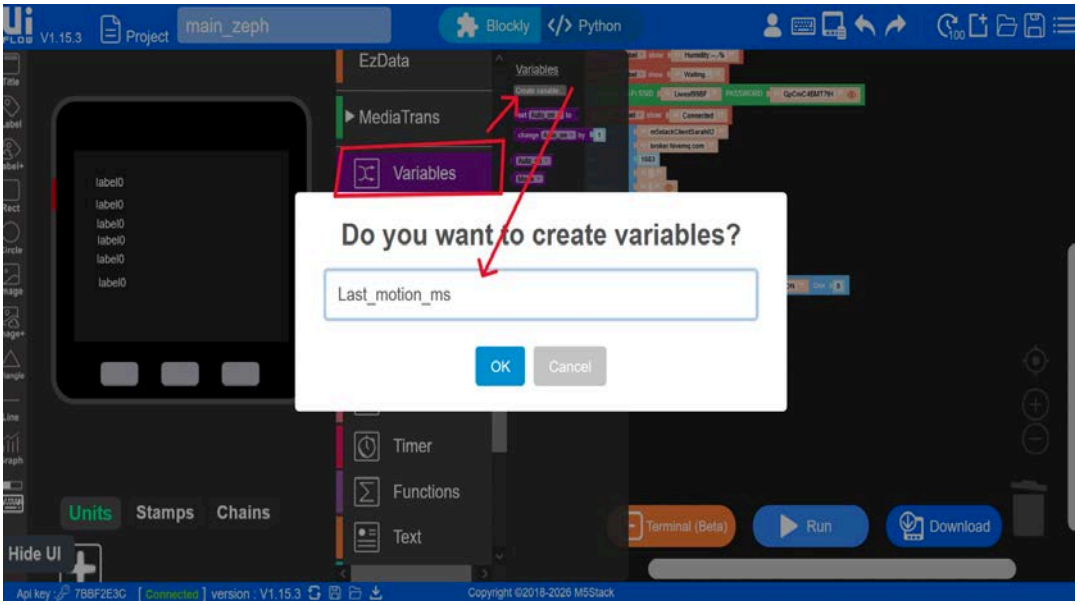


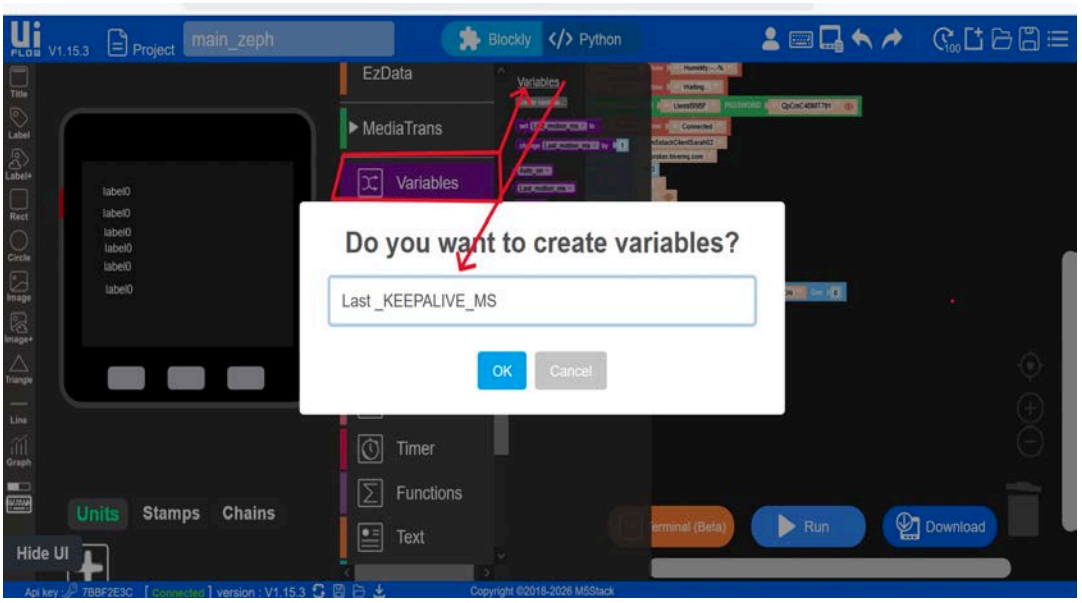


Step 4

Go to variables located on the block menu at the right side again and click create variables to set the following variables namely, Auto_on, Last_motion_ms and Last _KEEPALIVE_MS which will be used inside the Button A wasPressed block.







Note: These variables are created because they allow the system to:

- `auto_on` → remember whether motion is already active.
- `last_motion_ms` → determine when no motion has occurred for a specified timeout.
- `LAST_KEEPALIVE_MS` → periodically send keep-alive MQTT messages while motion continues.

Step 5

After creating the variables, it should appear in the variables list as shown below

Step 6

Drag and place the created variables namely auto_on , last_motion_ms and LAST_KEEPALIVE_MS into the Press A block(cellar) as illustrated in the video link below.

Step 7

The next step is to add the assigned values created to its respective variables created on the blocky area.

For example : auto_on = False

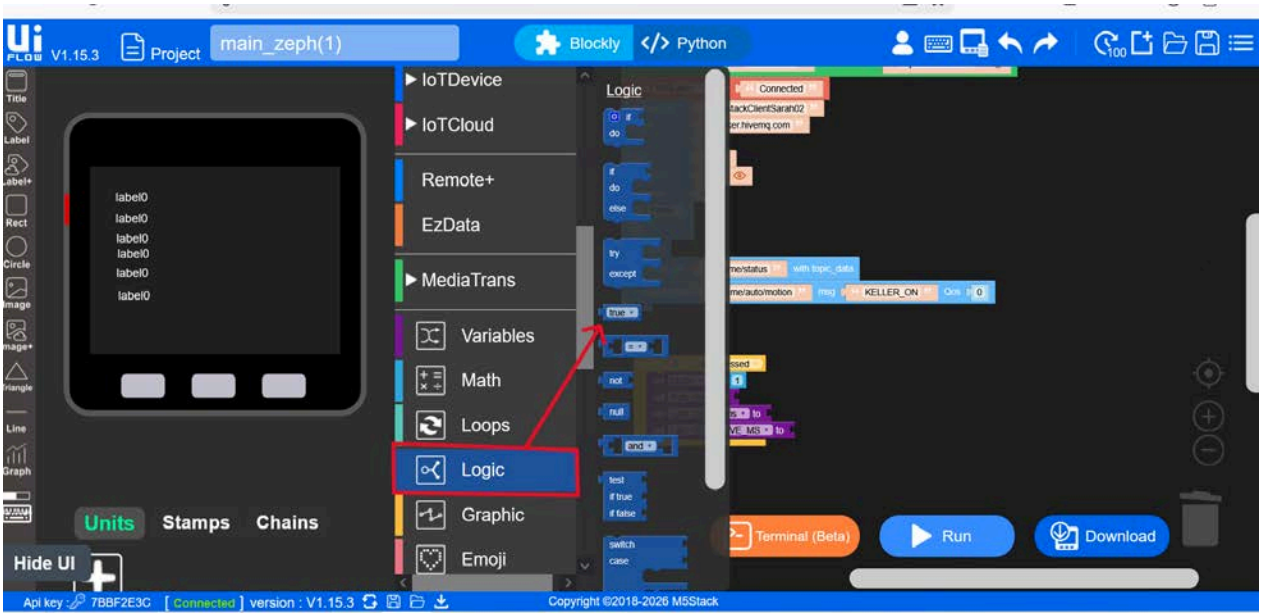
last_motion_ms = 0

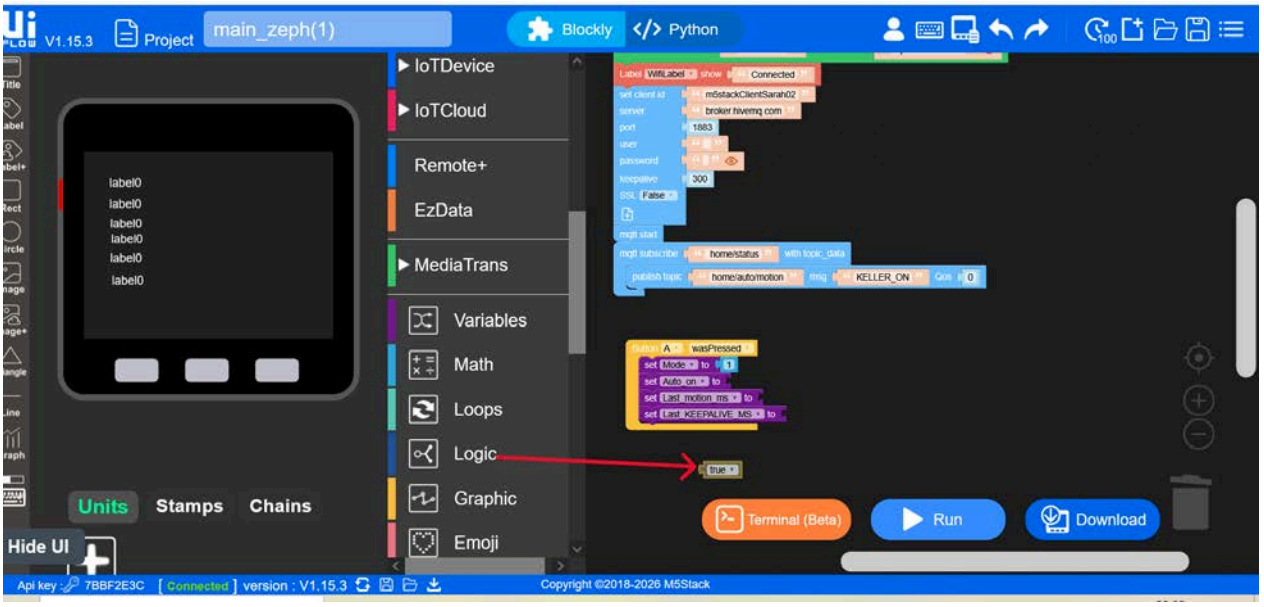
LAST_KEEPALIVE_MS = 0

This resets the motion detection logic so Cellar Mode starts with a clean state.

To add the auto_on = False, go to the right section category and search for Logic and click on it as indicated with the red arrow below.

After clicking on the Logic, the logic blocks will appear, then search for the true block and drag it onto your blocky working space as shown below.





Click on the block and change it from true to false.

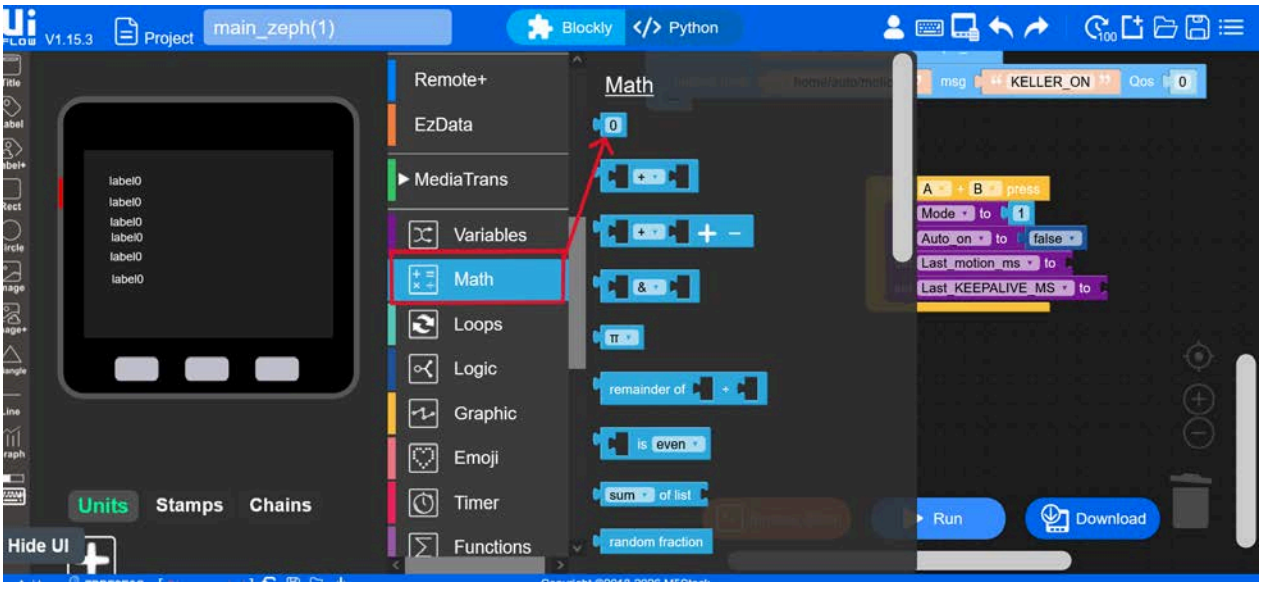
Then insert the logic block to the auto_on block as indicated below.

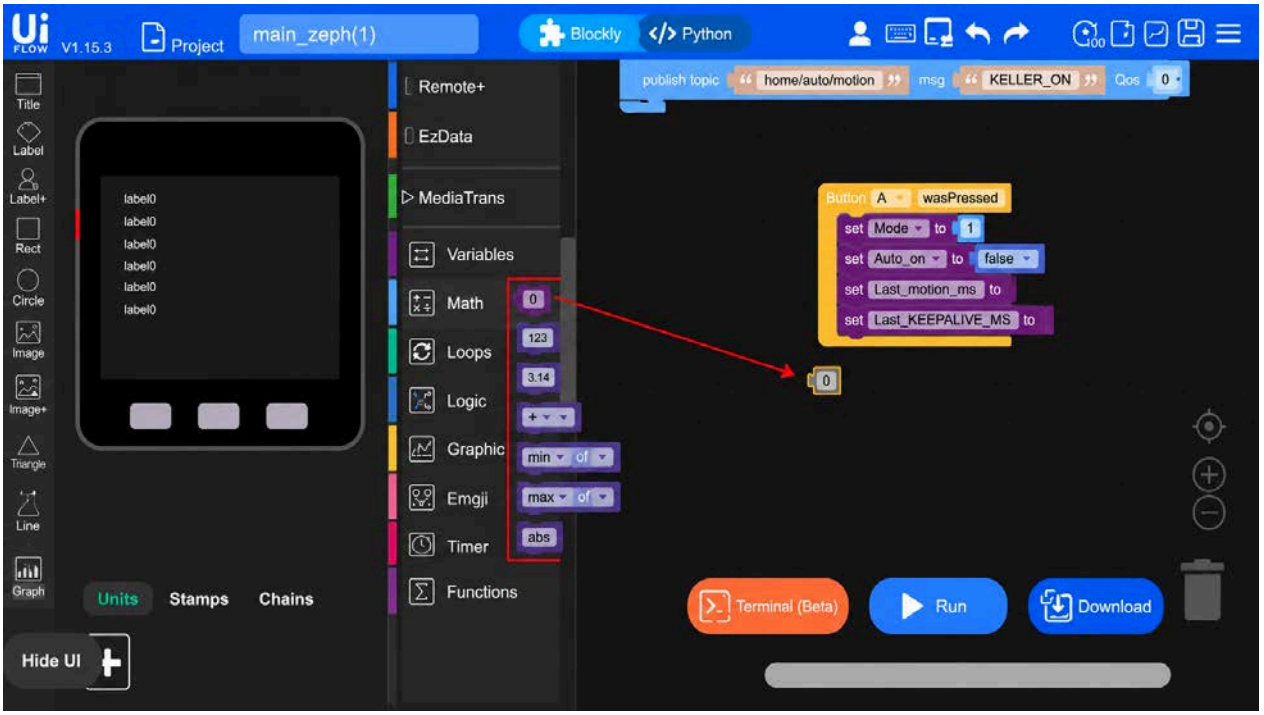
Note: The variable `auto_on` is set to **false** to reset the motion detection status whenever a new mode is selected. This ensures that the next detected movement is treated as a new motion event.

Step 8

The next step is to add values = 0 to the Last_motion_sm and Last_KEEPALIVE_MS variables.

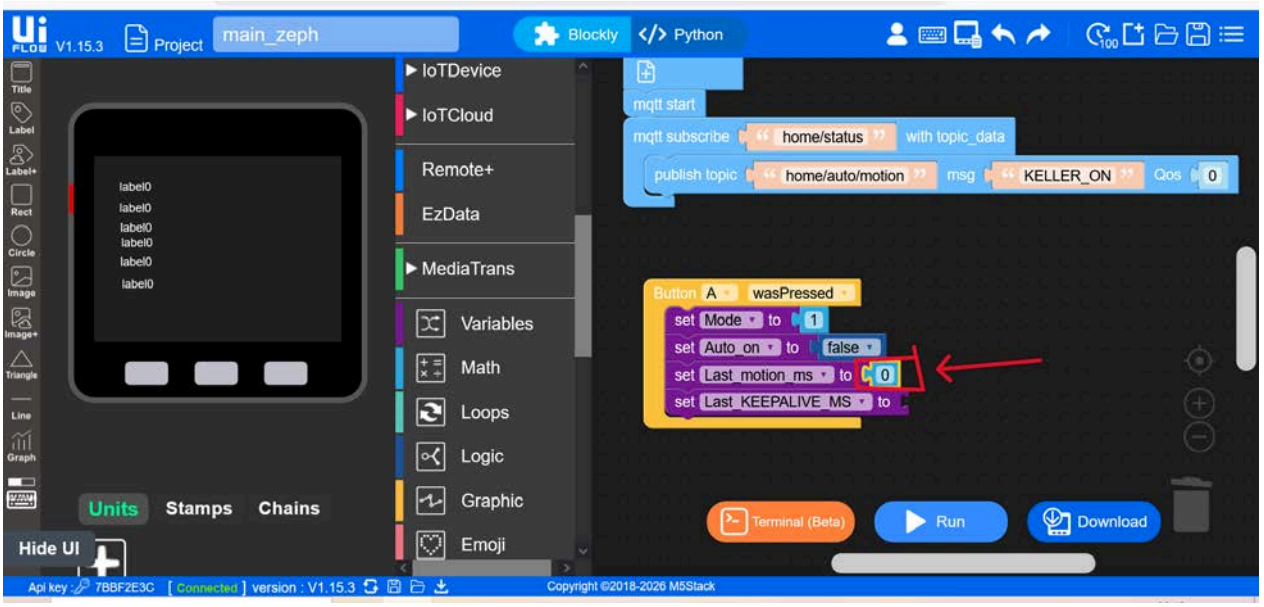
To set the value go to math under Variables and drag the block 0 as shown below on to your blocky area



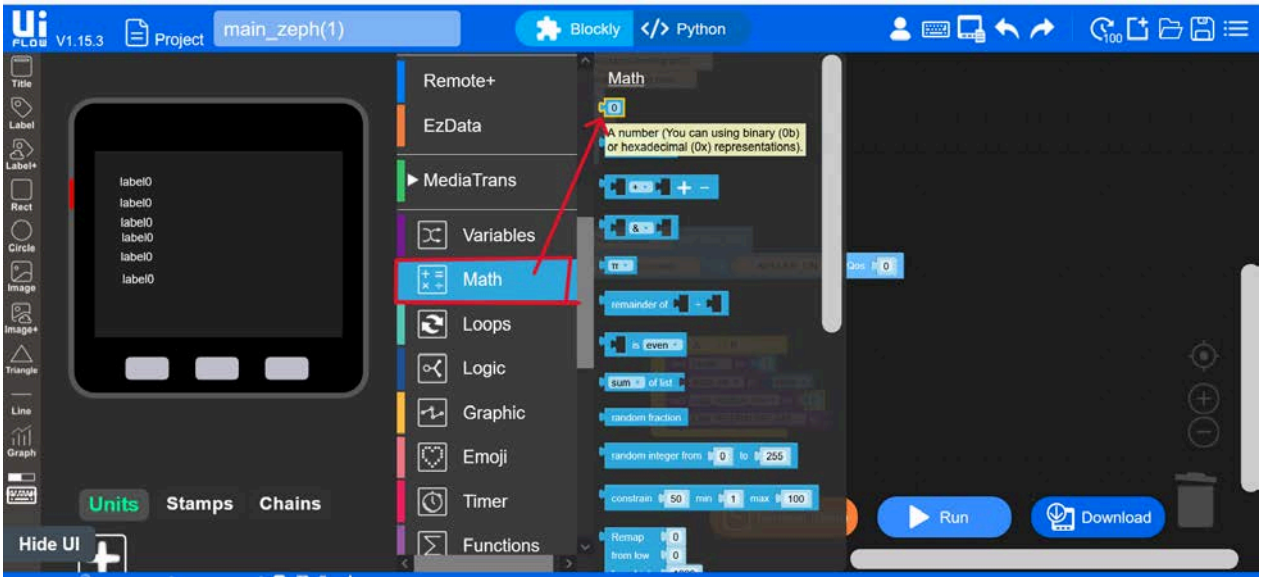


The next step is to drag the math block 0 and insert it to the Last_motion_ms block as shown below.

.



To assign value 0 to Last_KEEPALIVE_MS variables, go to math and drag the block 0 onto the blocky area and then insert it to the Last_KEEPALIVE_MS variables block as shown below.



Note: `last_motion_ms` and `LAST_KEEPALIVE_MS` are initialized to **0** to clear any previous timing information. A value of **0** indicates that no motion has been detected and no keep-alive message has been recorded yet, ensuring that Cellar Mode starts from a clean state.

Step 9

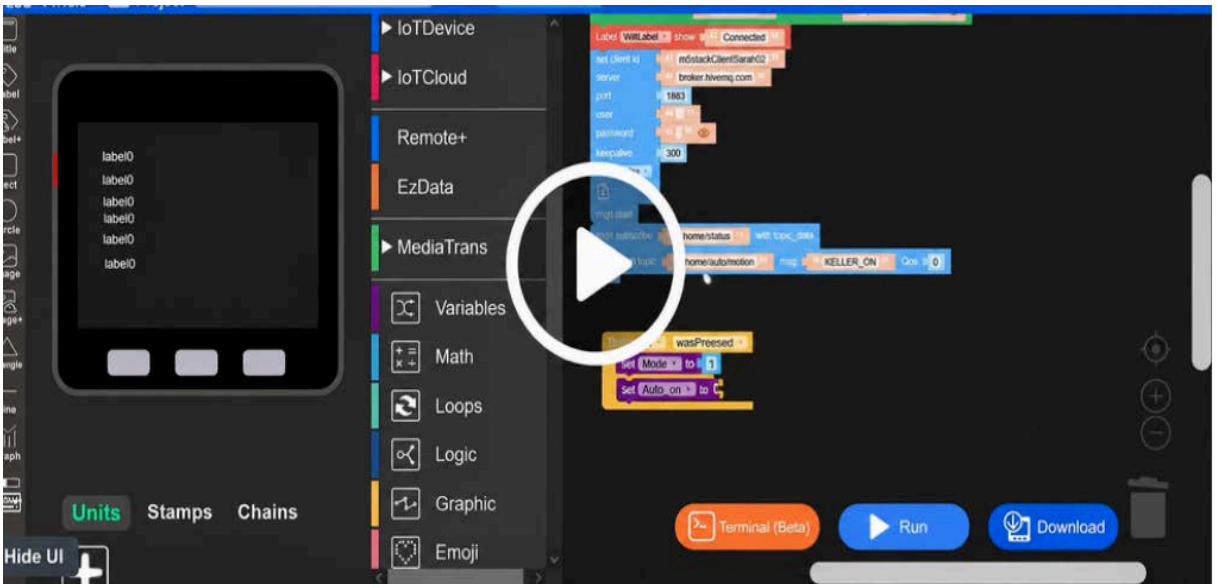
The next step is to **finish Button A(cellar mode)** by adding the screen updates.
Now add the screen labels inside the **Button A wasPressed** block.

1. Go to **UI**.
2. Open **Label**.
3. Drag the **Label show** block into the **Button A wasPressed** block.
4. Select **modeLabel**.
5. Enter this text: CELLAR (A)

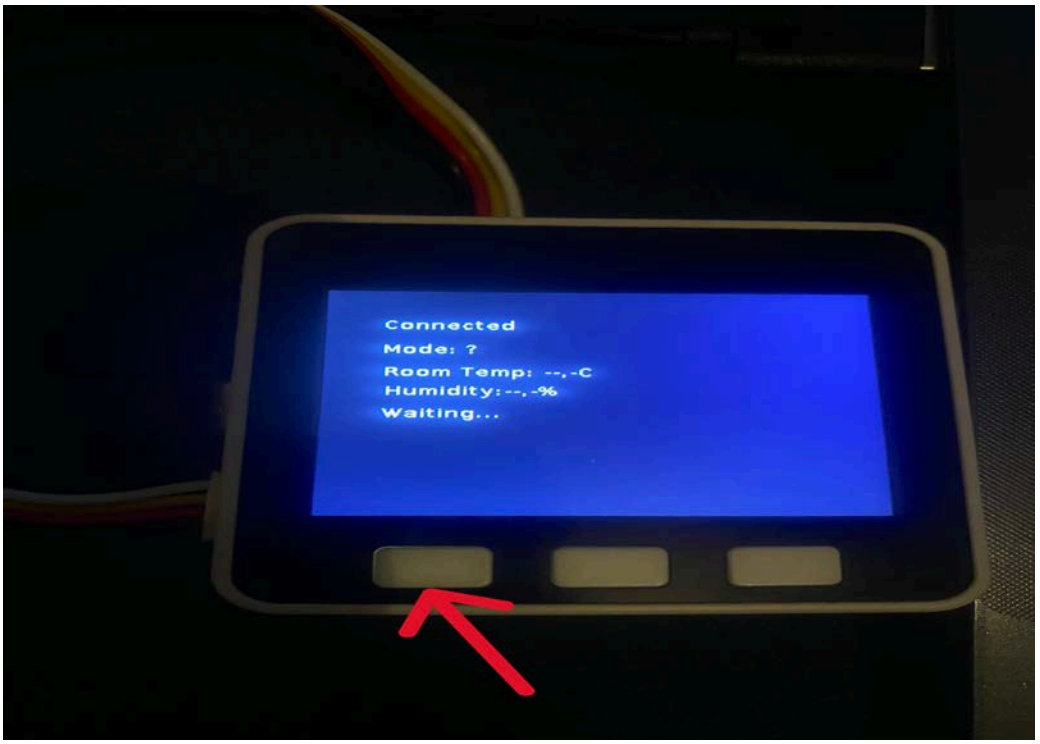
Then add another **Label show** block:

1. Select **msgLabel**.
2. Enter this text: Mode changed

Watch the video below to see how to add the label blocks inside the **Button A wasPressed** block.



After pressing the run on the uiflow the m5stack screen will display the image below. Then press the A button as indicated with the red arrow

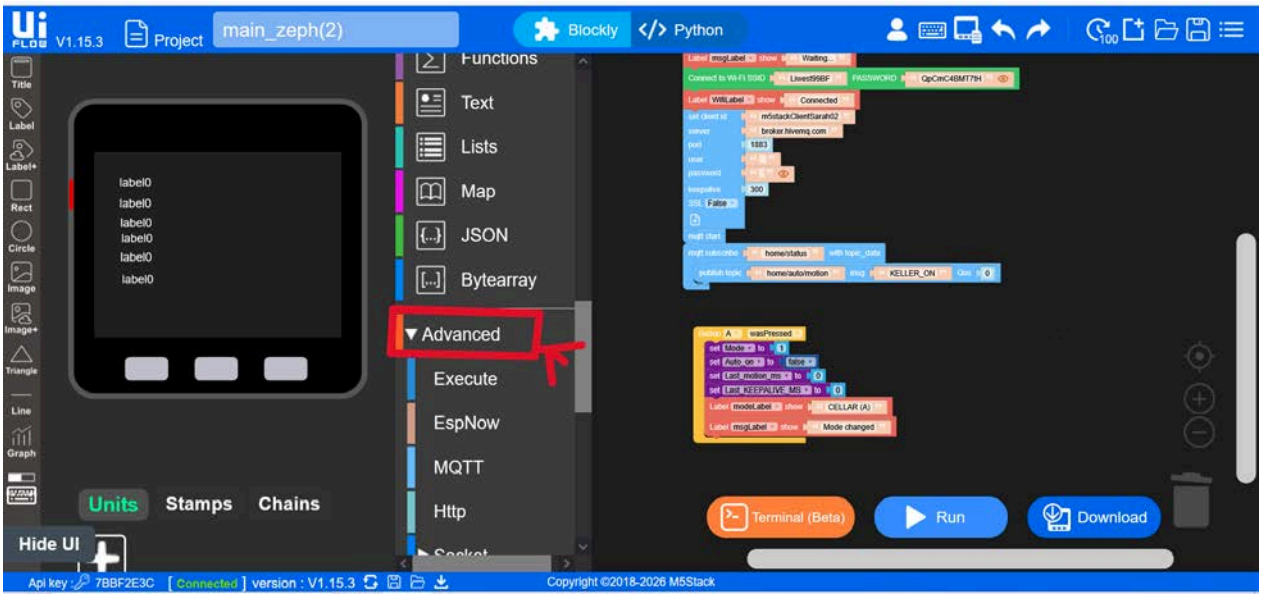


After pressing the A button, the M5Stack switches to Cellar Mode. The display is updated to indicate that Cellar Mode is active, as shown in the image below.

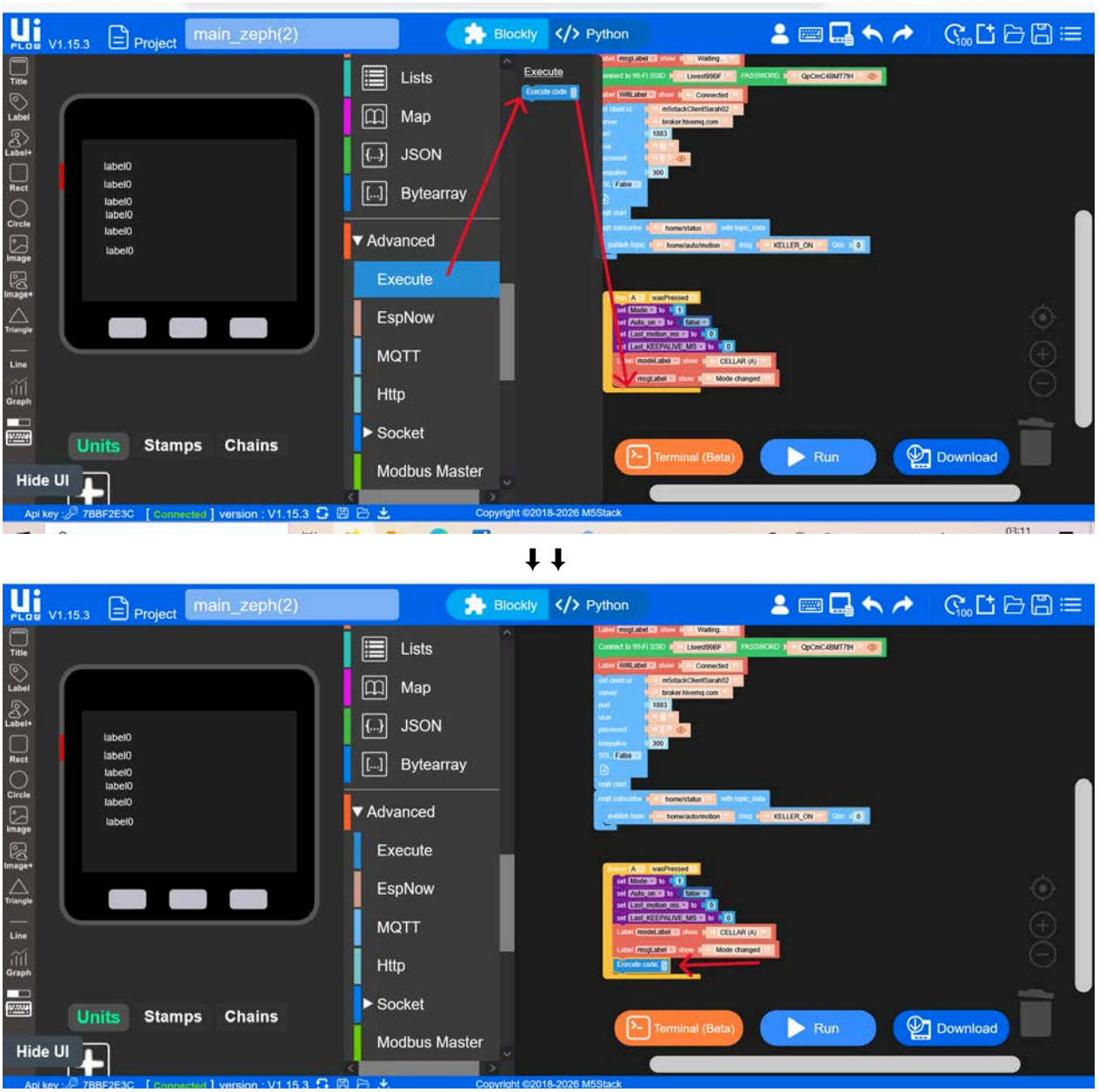
Step 10 – Publish the `STORE_OFF` Message

The last step is to publish the `STORE_OFF` message when **Button A** is pressed. This ensures that **Store Room Mode** is deactivated before **Cellar Mode** becomes active.
To enable this follow these steps below:

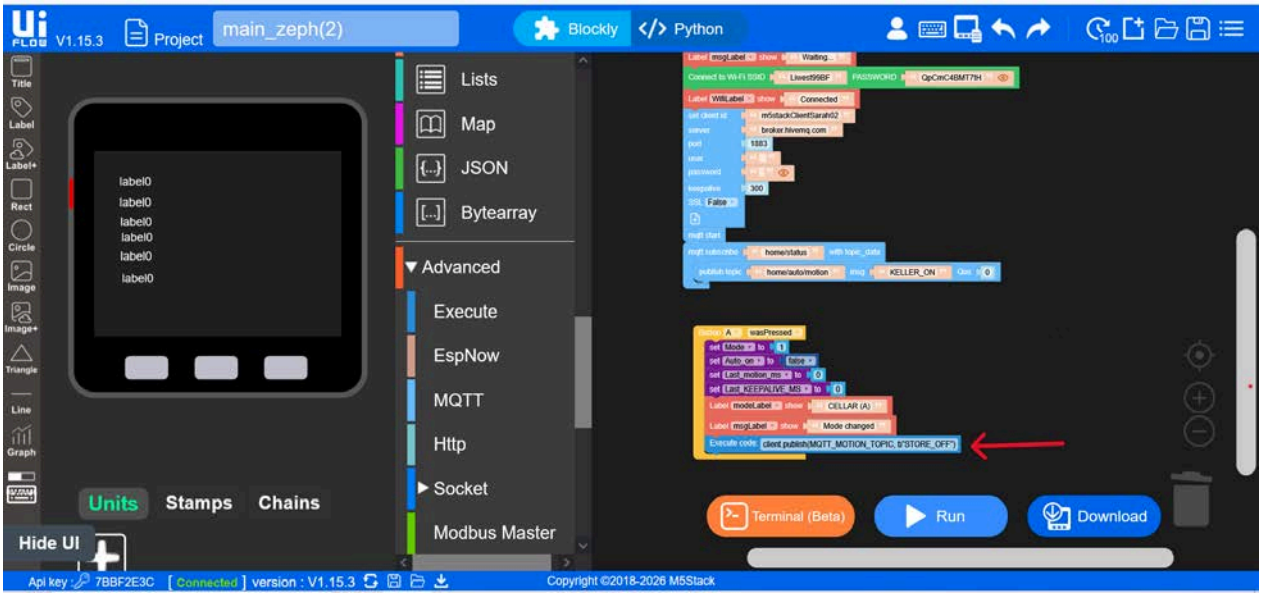
1. Go to the right hand side and search for Advance category as indicated below.



2. Click on execute and drag the **Execute code** block into the **Button A wasPressed** below the **Mode changed** label block as shown below.



3. Enter the following code: client.publish(MQTT_MOTION_TOPIC, b"STORE_OFF") as demonstrated below.



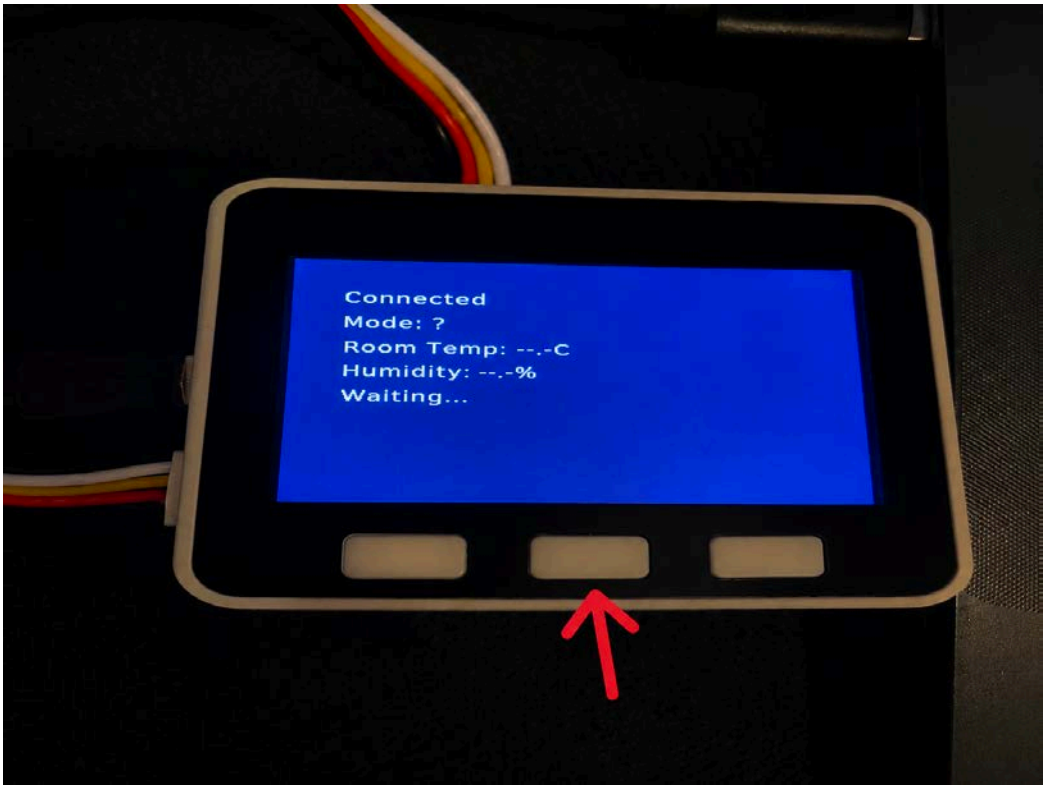
Note: This command publishes the `STORE_OFF` message to the MQTT topic. As a result, Store Room Mode is turned off whenever Cellar Mode is selected, ensuring that only one room is monitored at a time.

Step 11 – Create the Button B Logic (Store Room Mode)

The next step is to configure **Button B** to activate **Store Room Mode**. Similar to Button A, pressing Button B resets the motion detection variables, updates the display, and deactivates Cellar Mode.

Follow the video below to create the **Button B wasPressed** block.

After pressing the run on the uiflow the m5stack screen will display the image below. Then press the B button as indicated with the red arrow



After pressing the b button, the M5Stack switches to Store Room Mode. The display is updated to indicate that Store Room Mode is active, as shown in the image below.

Note: When **Button B** is pressed, the following actions are performed:

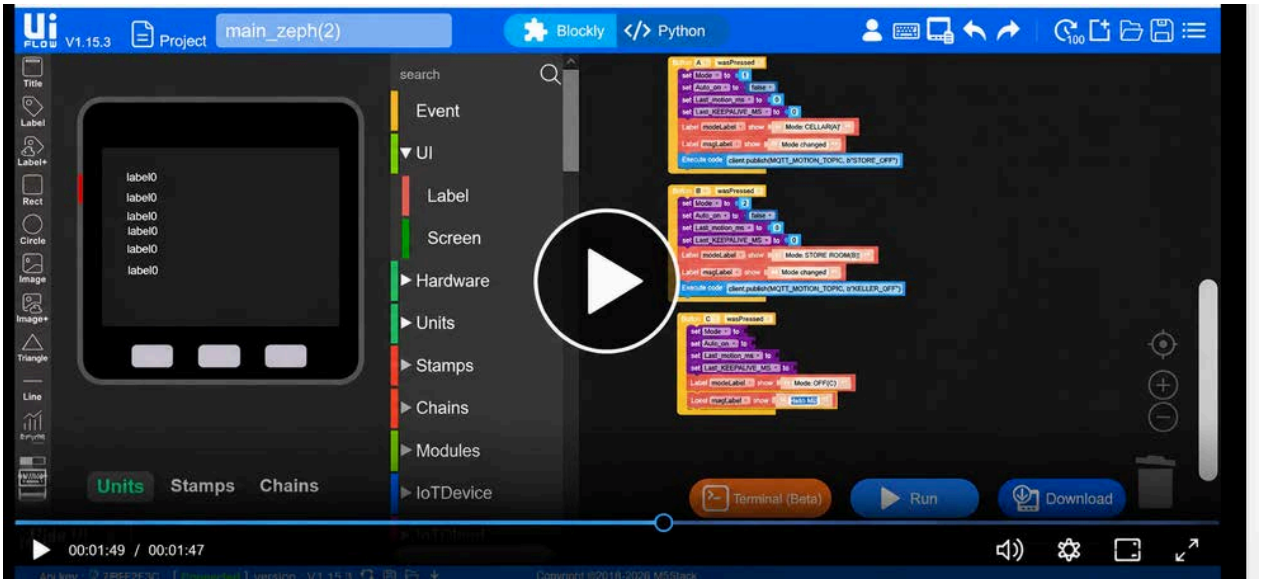
- **Mode** is set to **2** to activate **Store Room Mode**.
- **Auto_on** is reset to `false` to disable any previous motion detection state.
- **Last_motion_ms** and **Last_KEEPALIVE_MS** are reset to `0` so motion detection starts from a clean state.
- The **M5Stack display** is updated to show **Store Room (B)** and **"Mode changed"**.
- The **`KELLER_OFF`** message is published to the MQTT topic, automatically deactivating Cellar Mode so that only one room is monitored at a time.

Explanation : When **Button B** is pressed, the system switches to **Store Room Mode**. The variable **Mode** is set to **2** because each operating mode is represented by a unique value (`1` = Cellar Mode, `2` = Store Room Mode). The variable **Auto_on** is reset to `false` to stop any previous motion detection state before the new mode starts. The variables **Last_motion_ms** and **Last_KEEPALIVE_MS** are reset to `0` to clear any previous motion timestamps, ensuring that motion detection starts with a clean state. The M5Stack display is then updated to show **Store Room (B)** and the message **"Mode changed"**, confirming that the mode has changed successfully. Finally, the following Execute Code block is used to publish the **KELLER_OFF** message to the MQTT topic:*client.publish(MQTT_MOTION_TOPIC, b"KELLER_OFF").* This command notifies the MQTT broker that **Cellar Mode** has been turned off. As a result, only **Store Room Mode** remains active, ensuring that the application monitors only one room at a time and preventing both monitoring modes from running simultaneously.

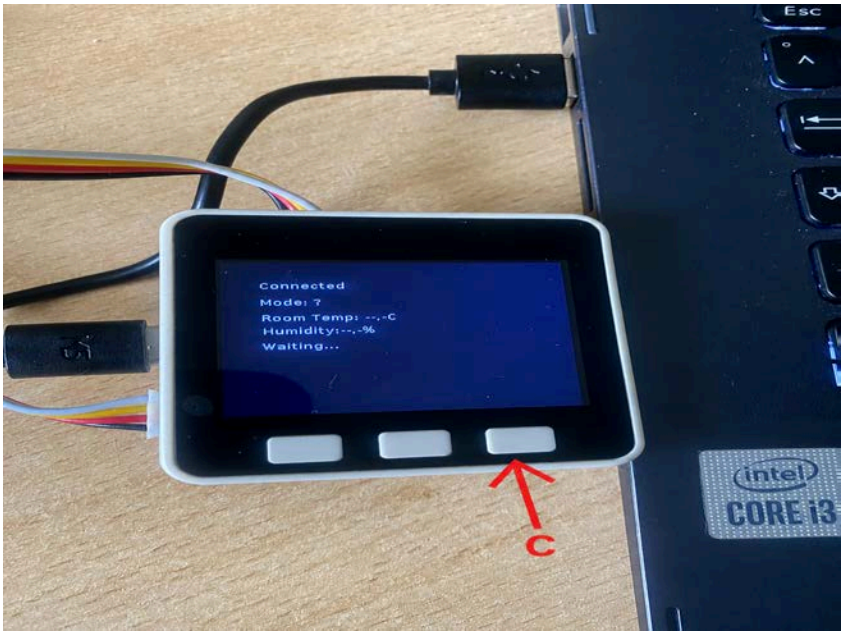
Step 12 – Create the Button C Logic (OFF Mode)

The next step is to configure **Button C** to deactivate the motion detection system. Unlike Buttons A and B, Button C disables monitoring for both rooms by setting the system to **OFF Mode**. It also resets the motion detection variables, updates the display, and publishes MQTT messages to ensure that both Cellar Mode and Store Room Mode are turned off.

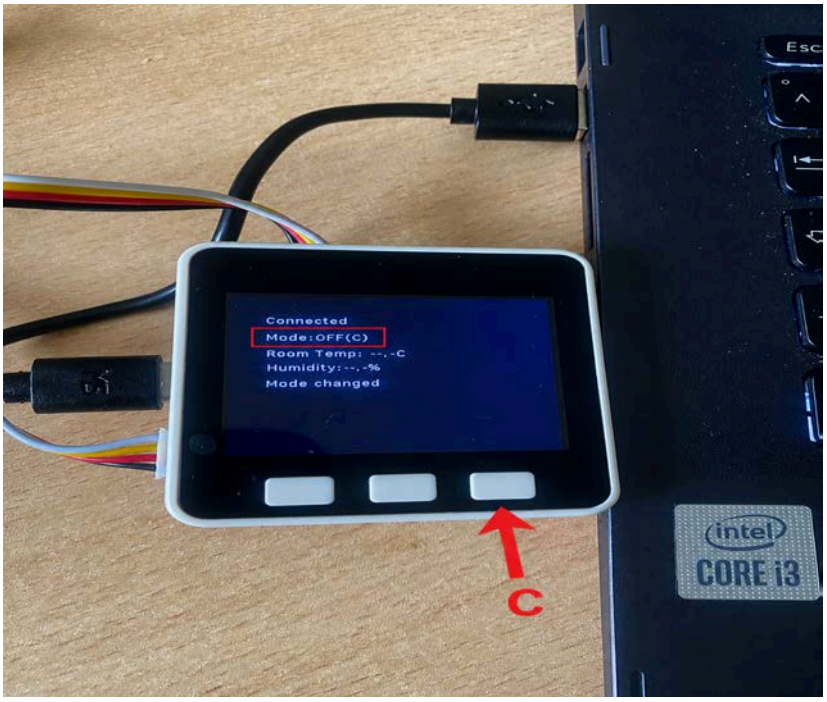
Follow the video below to create the **Button C wasPressed** block.



After pressing the run on the uiflow the m5stack screen will display the image below. Then press the C button as indicated with the red arrow



After pressing the C button, the M5Stack switches to OFF Mode. The display is updated to indicate that OFF Mode is active, as shown in the image below.



Note: When **Button C** is pressed, the following actions are performed:

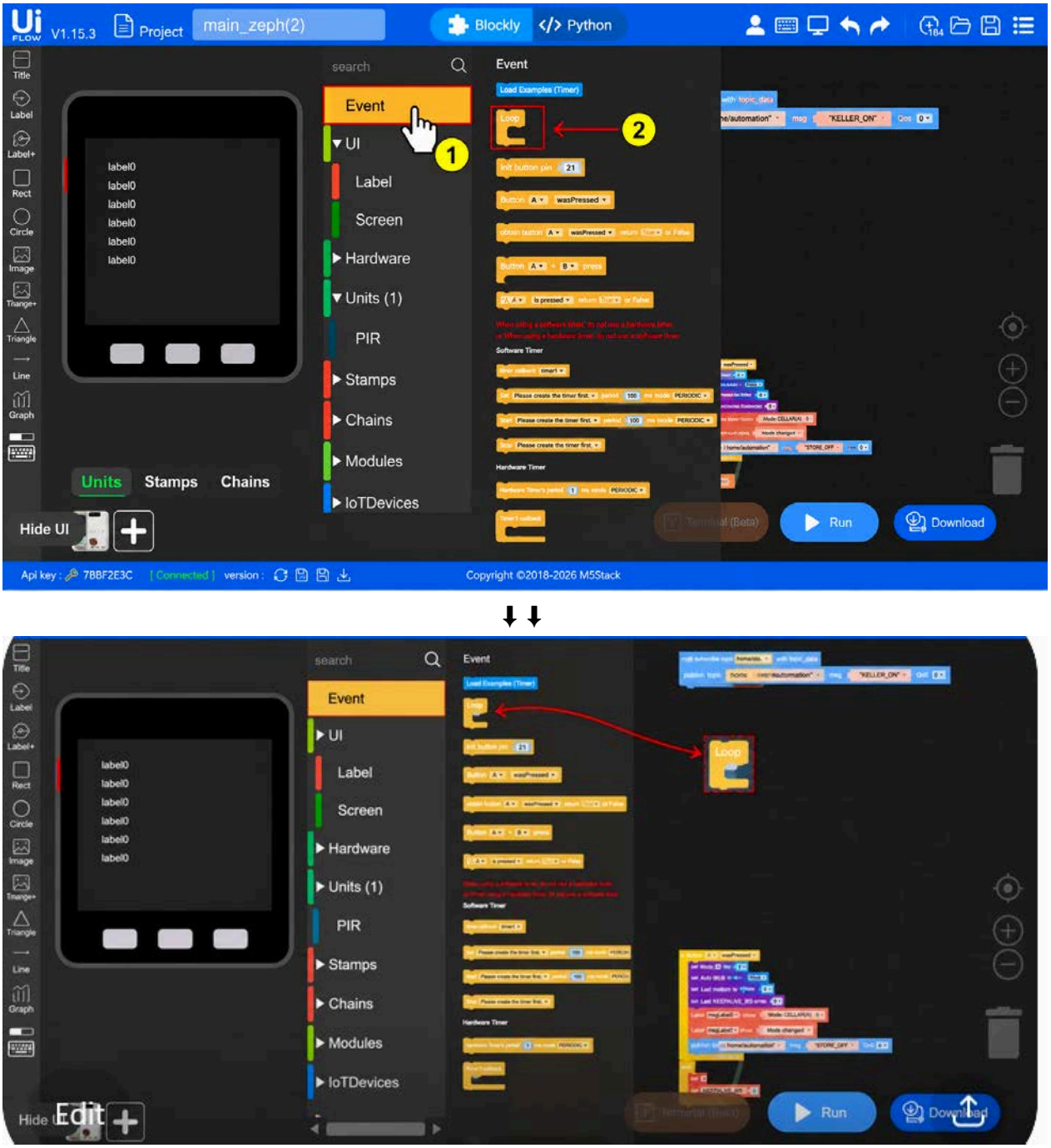
- **Mode** is set to **0**, which disables automatic motion monitoring.
- **Auto_on** is reset to **false** to ensure that motion detection is inactive.
- **Last_motion_ms** and **Last_KEEPALIVE_MS** are reset to **0** so that the next monitoring session starts with a clean state.
- The M5Stack display is updated to show **Mode: OFF (C)** and **"Mode changed"**.
- The commands
  client.publish(MQTT_MOTION_TOPIC, b"KELLER_OFF")
  client.publish(MQTT_MOTION_TOPIC, b"STORE_OFF") are executed to publish **KELLER_OFF** and **STORE_OFF** to the MQTT topic. This ensures that both Cellar Mode and Store Room Mode are deactivated, leaving the system in an idle state until another mode is selected.

3.5.6 Create the Motion Detection Loop

The next stage explains how to implement the **PIR motion sensor** to automatically control the light based on detected movement. The motion detection loop continuously monitors the PIR sensor and determines whether the light should be turned **ON** or **OFF**.

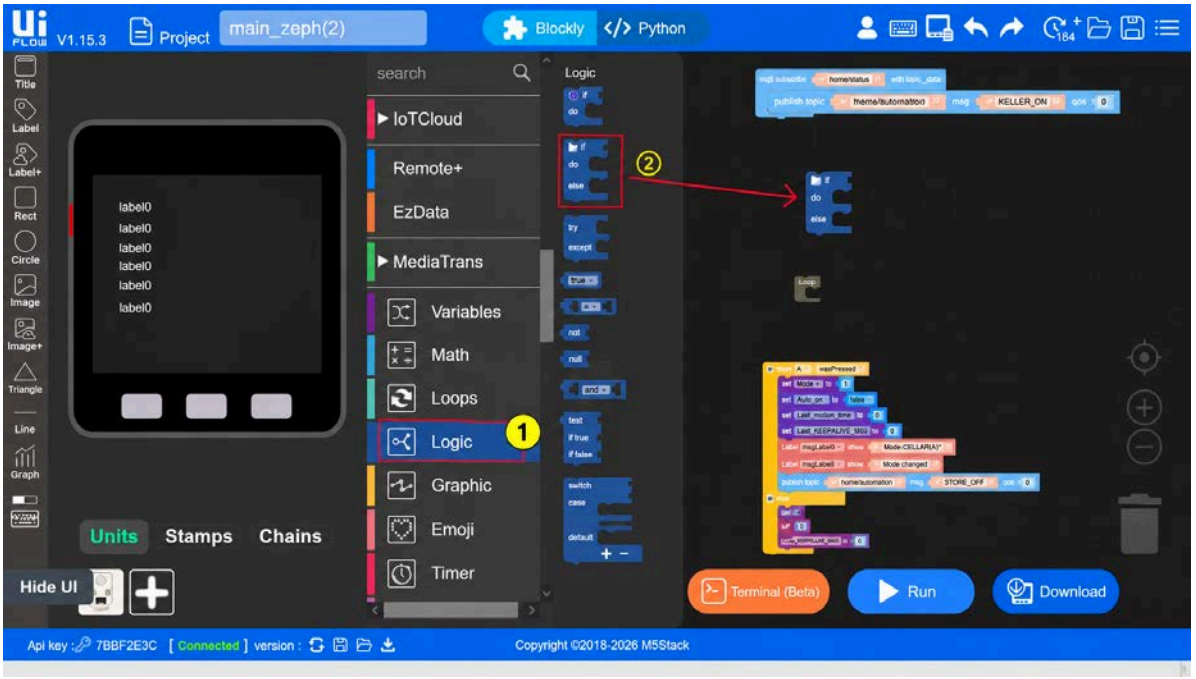
Step 1

In UIFlow, go to the **Event section again** and drag the **“Loop” (repeat while true)** block into your program. This block ensures that the program runs continuously and checks for changes in real time as indicated below.



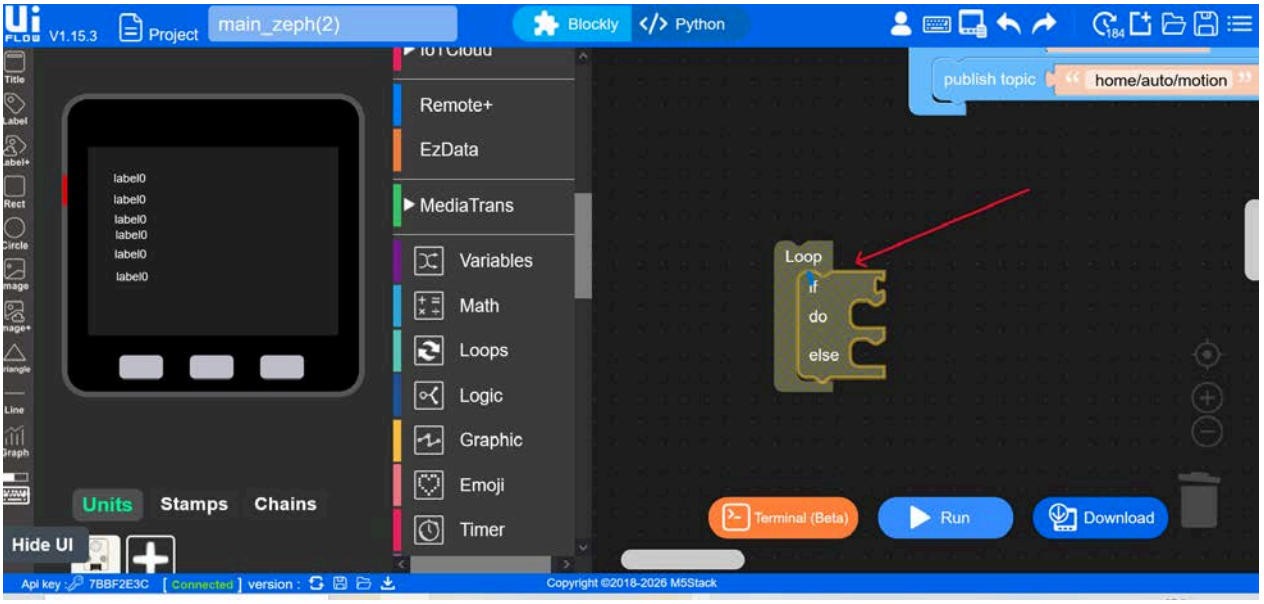
Step 2

Go to logic and drag the if do else onto the blocky area as shown below.



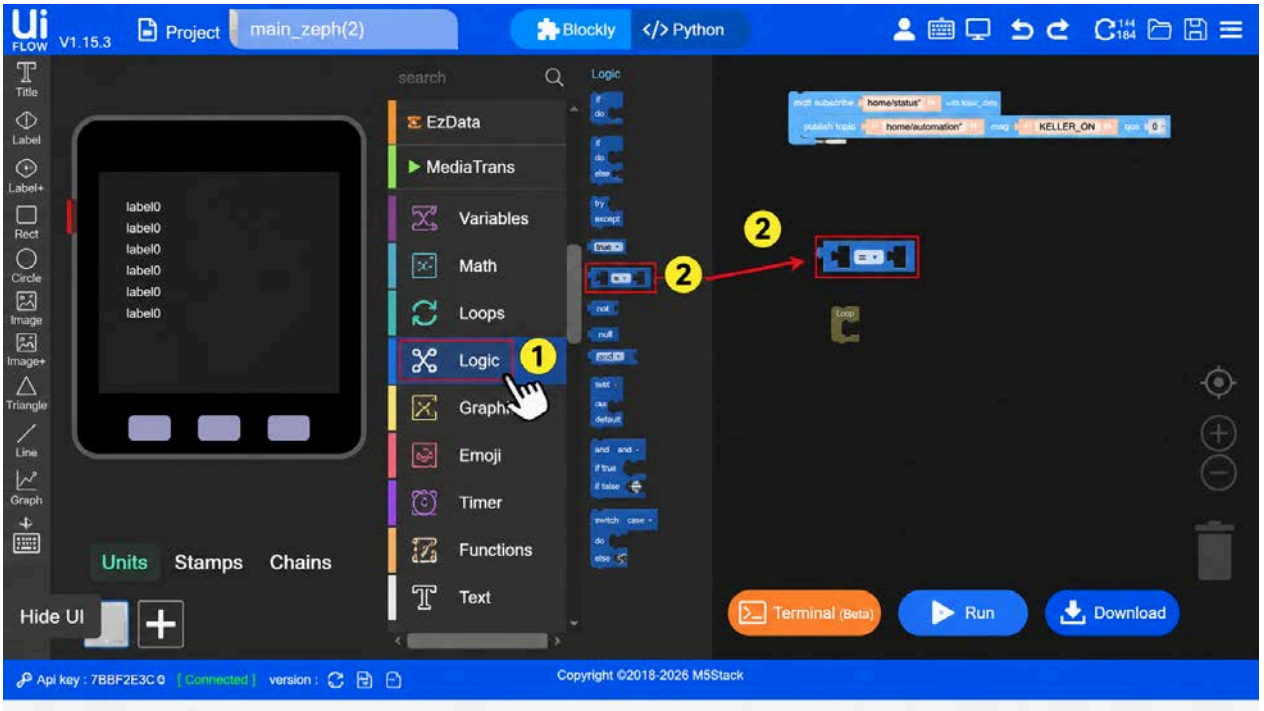
Step 3

Afterwards drag the if do loop and place it inside the loop block as shown below.



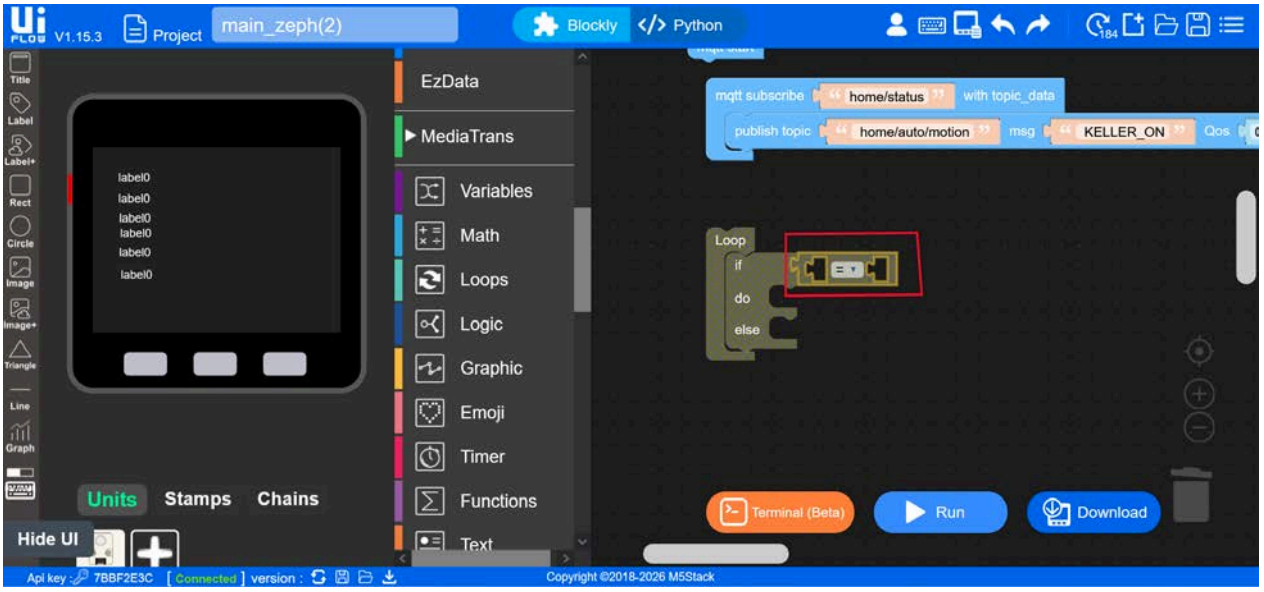
Step 4

Go to the logic section and drag the **comparison (equals) block** shown with the red arrow onto the blocky area as shown below.



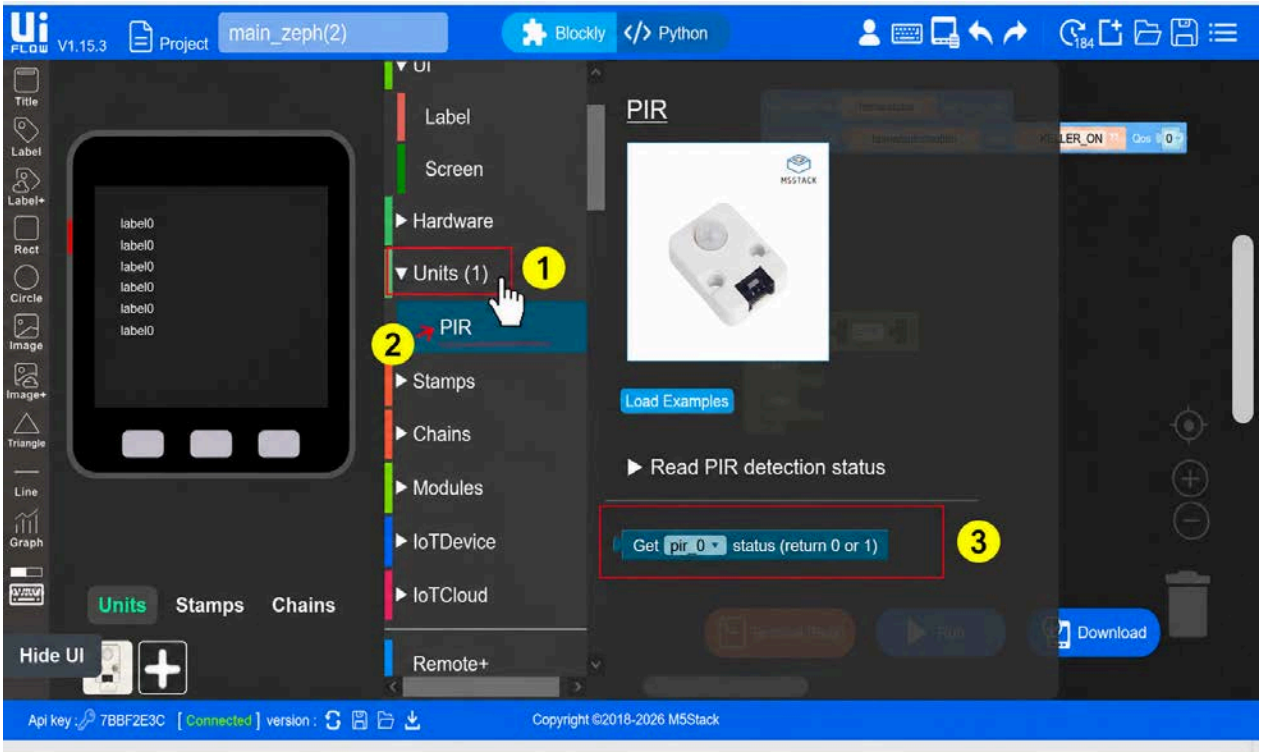
Step 5

Afterwards insert the **comparison (equals) block** into the if do else block as shown below.



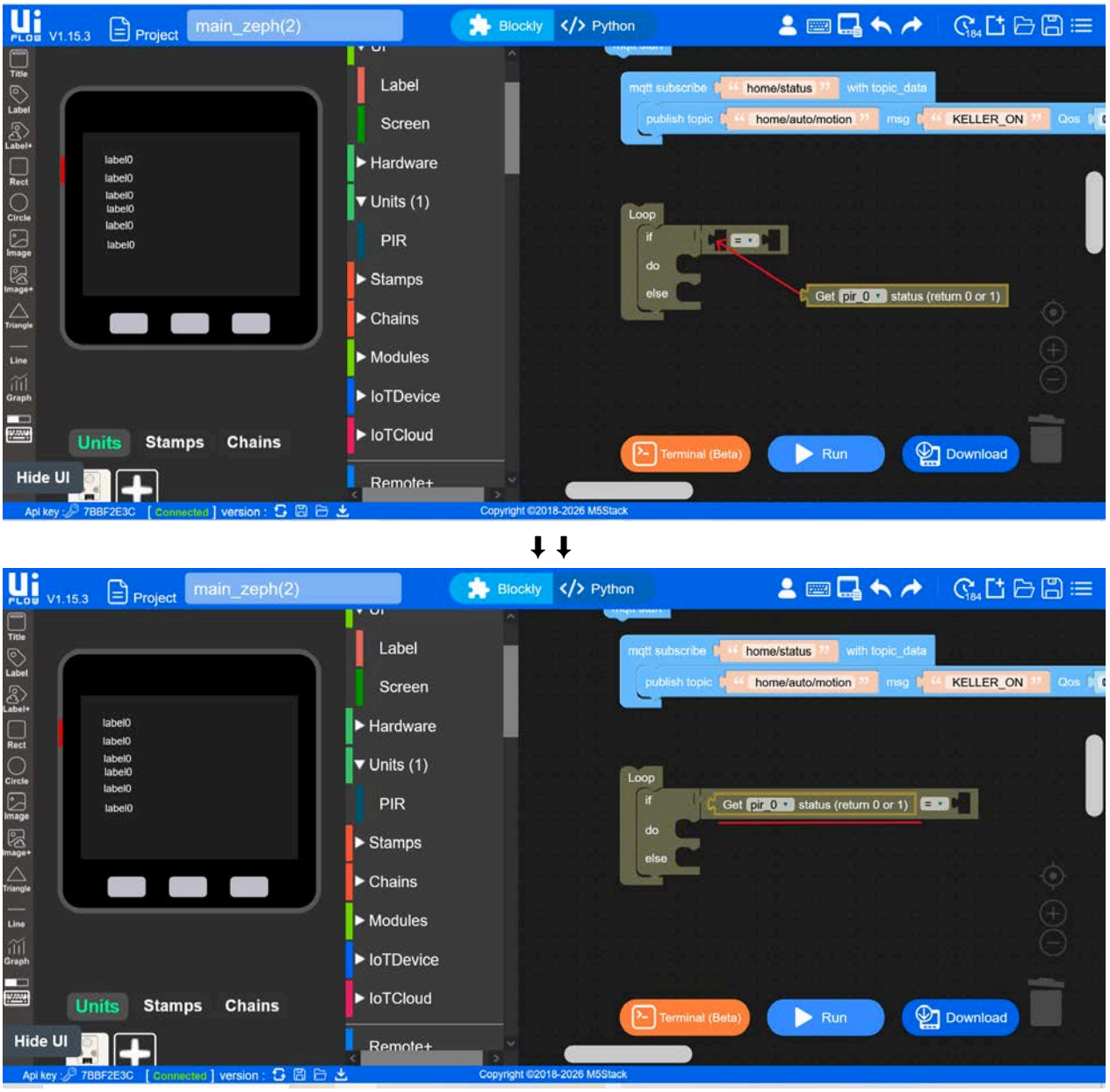
Step 6

Go to the **Units section**,then click on the PIR and drag the **PIR get block** and place it inside **the comparison (equals) block**. This block is used to detect whether motion is present. The PIR sensor returns the value **0** when no motion is detected and **1** when motion is detected.



⬇⬇

Next, drag the **PIR get status (return 0 or 1)** block into the **left input** of the **comparison (equals)** block, as shown below. This block reads the PIR sensor and returns **1** when motion is detected and **0** when no motion is detected.

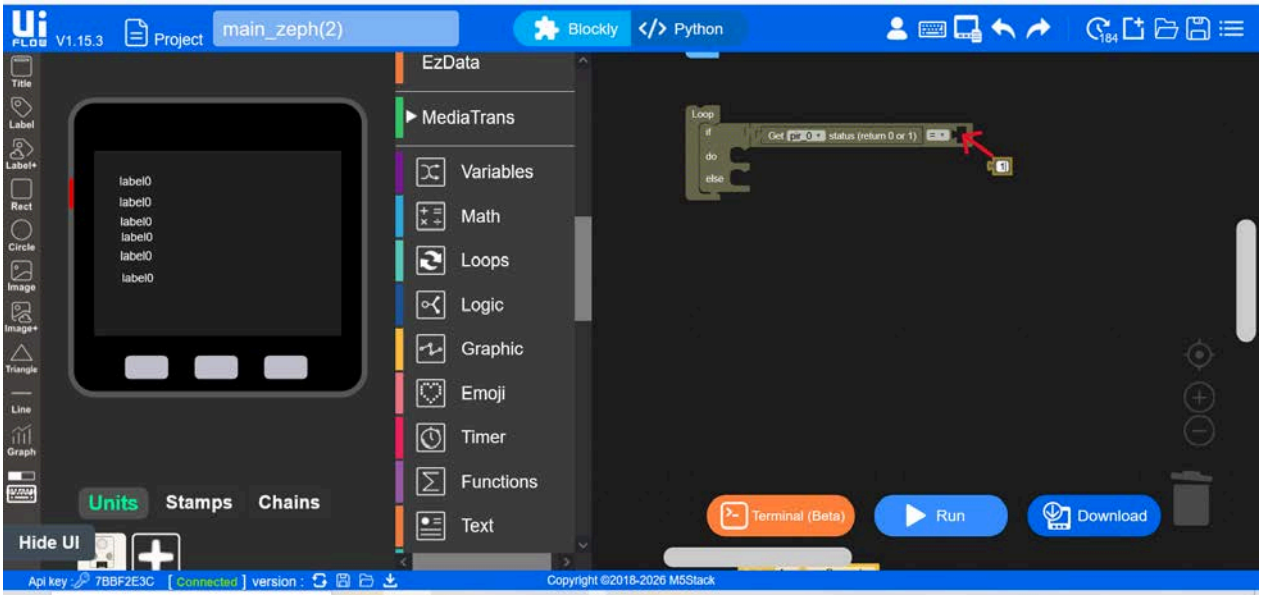


Step 7

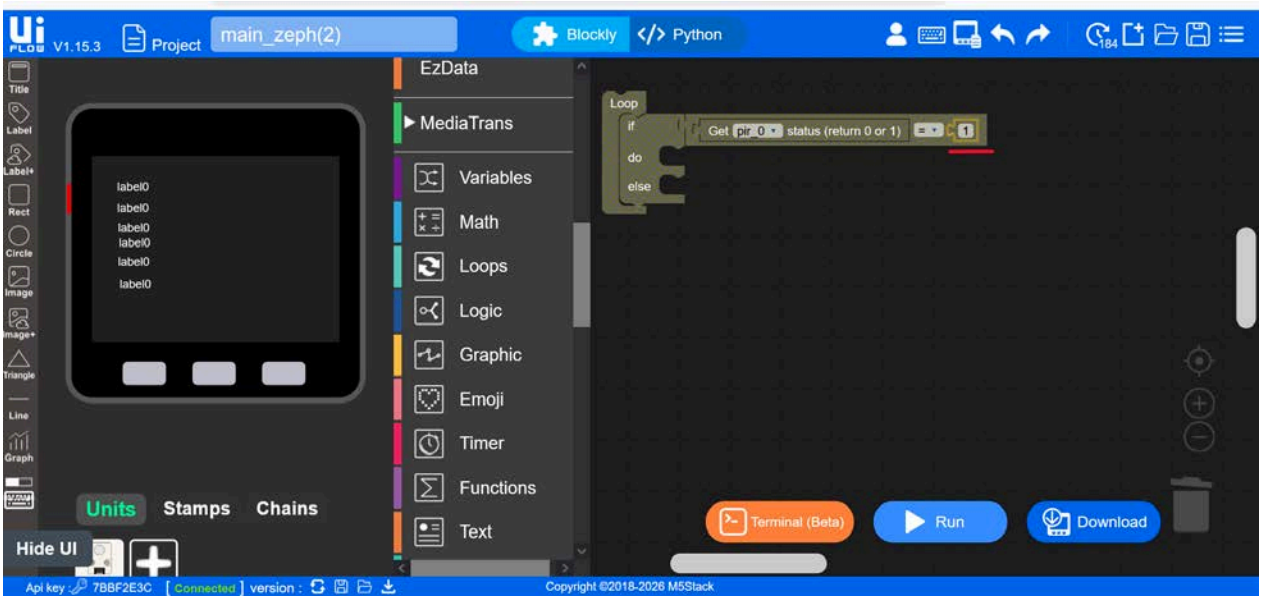
From the **Math** section, drag a **number** block onto the Blockly workspace. This block represents the value that the PIR sensor output will be compared against.

Step 8

Change the value of the **number** block from 0 to **1**, then insert it into the **right input** of the **comparison (equals)** block. This configures the condition to check whether the PIR sensor has detected motion (**PIR status = 1**).



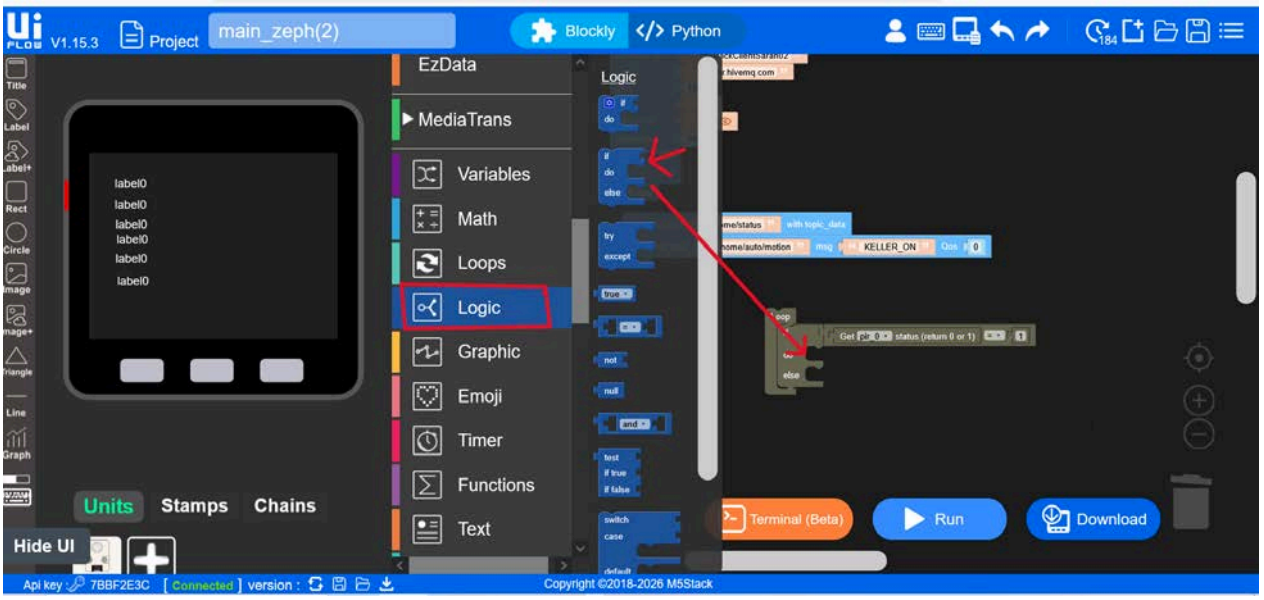
⬇⬇

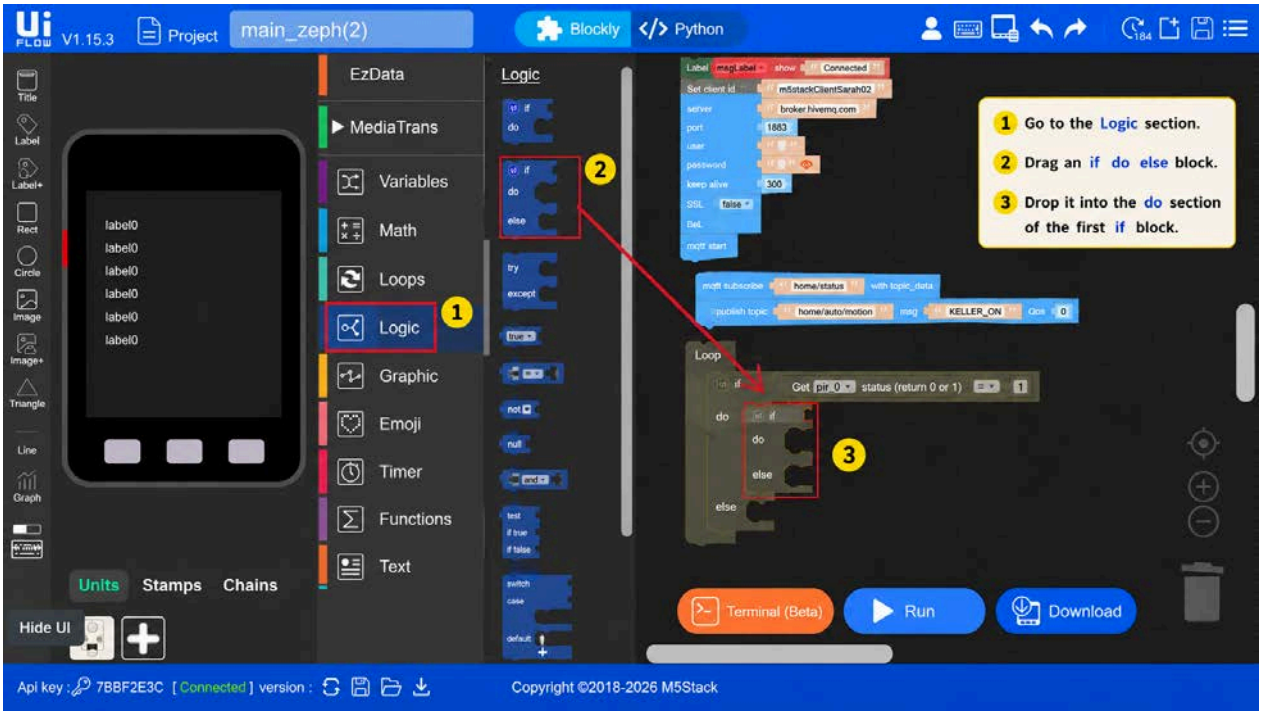


Note:This will evaluate the condition continuously inside the **Loop** block. Whenever the PIR sensor detects motion, it returns **1**, causing the condition to evaluate to **true** and allowing the blocks inside the **if** section to execute.

Step 9

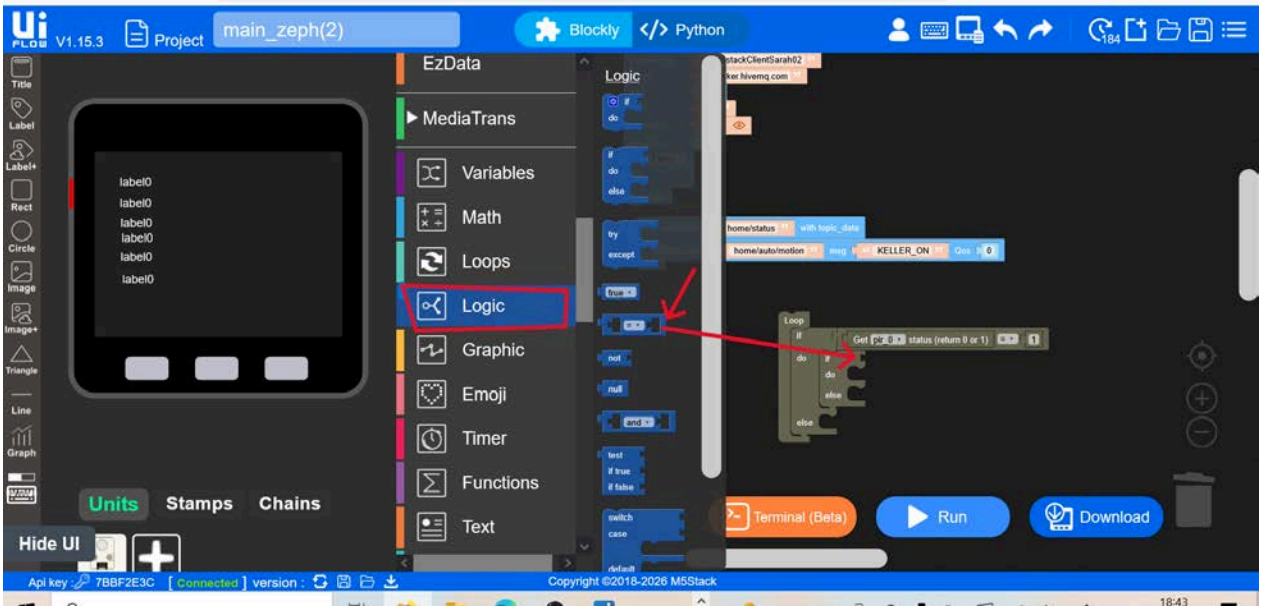
From the **Logic** section, drag another **if do else** block into the **do** section of the first **if** block. This nested condition will determine which room is currently selected when motion is detected.

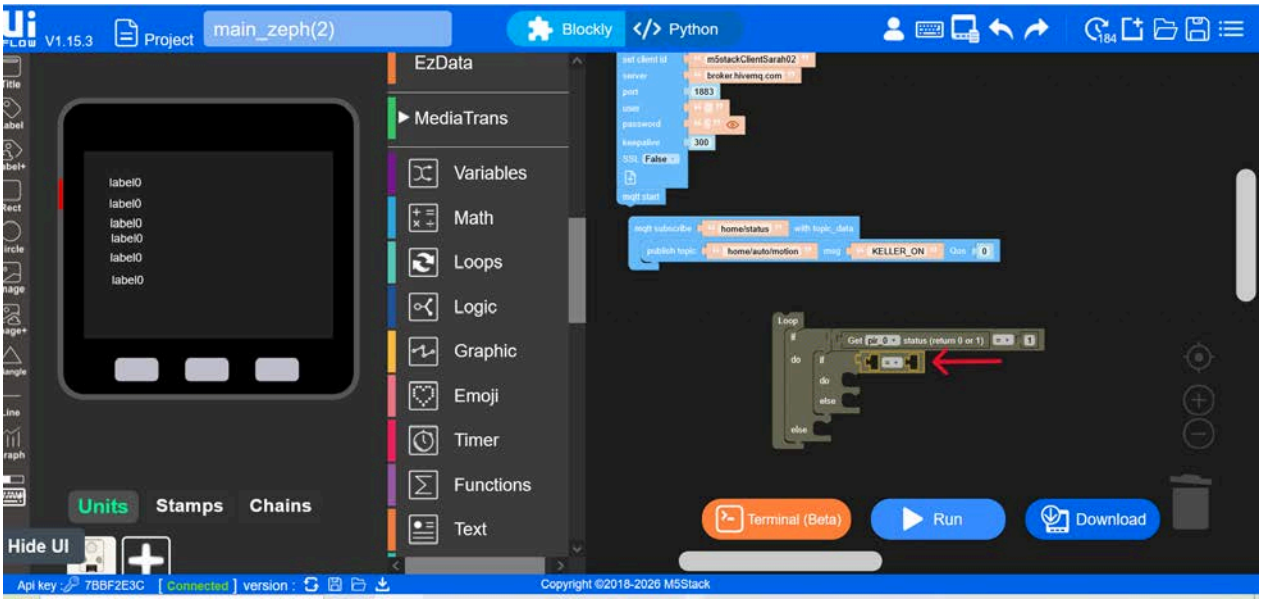




Step 10

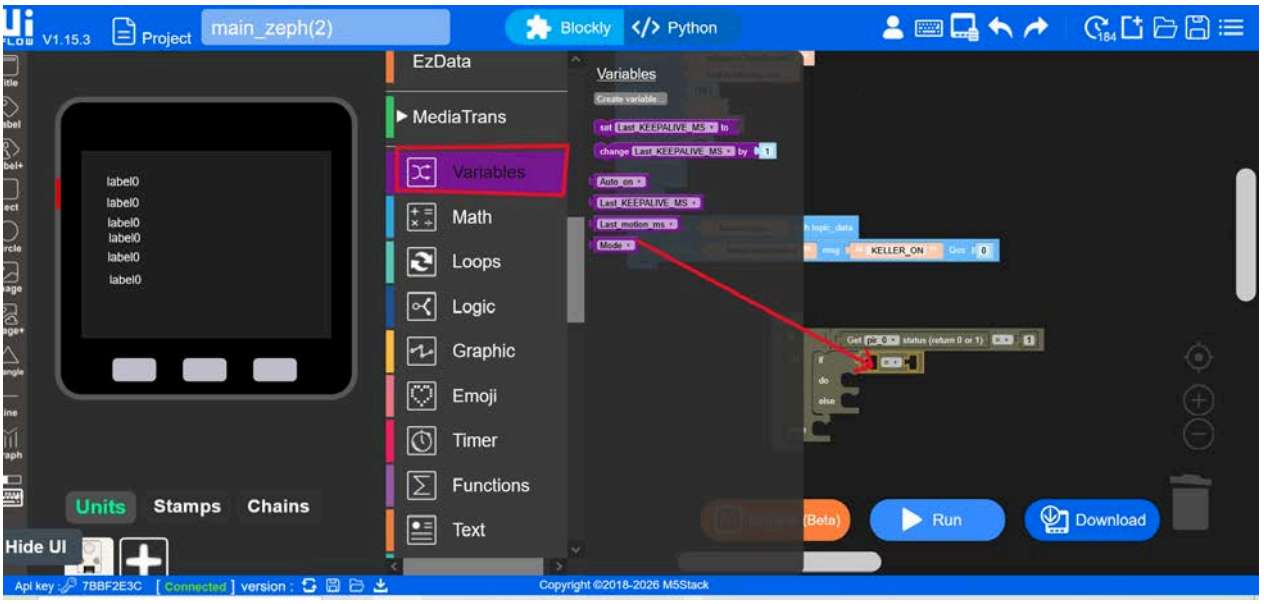
Go to the logic section again and drag the **comparison (equals) block** shown with the red arrow onto the if block as shown below.





Step 11

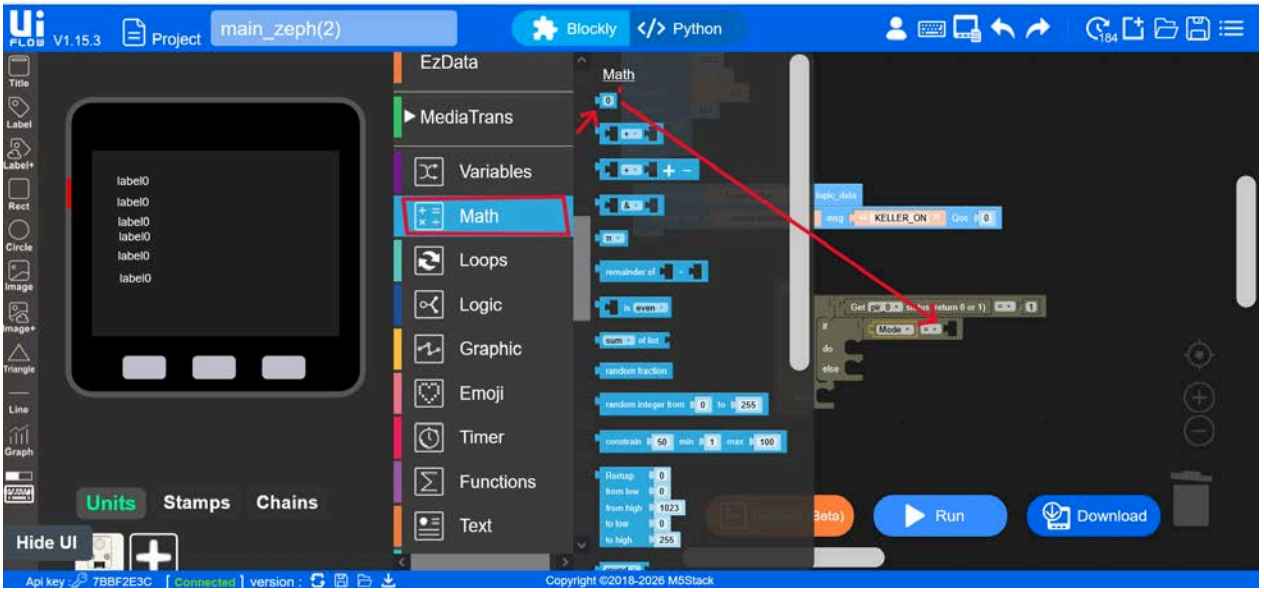
Afterwards **go to the Variables section and drag the `Mode` variable into the left input of the comparison (equals) block inside the nested `if` block.** The `Mode` variable stores the currently selected room. By comparing its value, the program can determine whether the user has selected **Cellar (Mode = 1)** or **Store Room (Mode = 2)**. This allows the program to execute the correct action when motion is detected.



⬇⬇

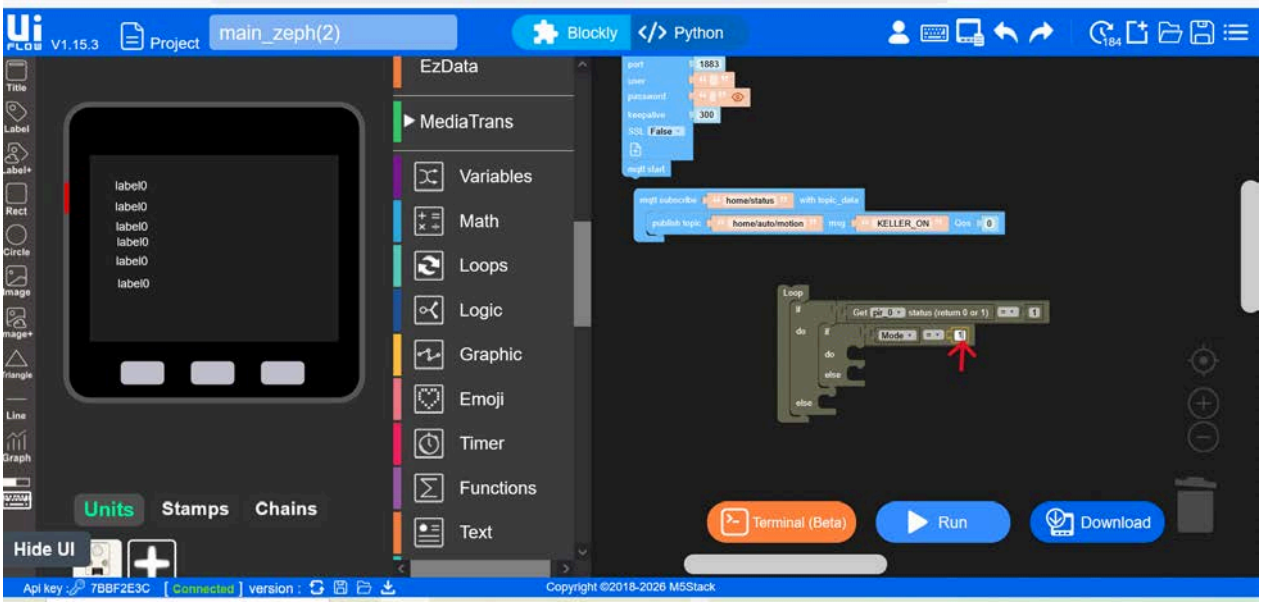
Step 12

From the **Math** section, drag a **number** block into the **right input** of the comparison block and change its value to **1**. This checks whether the selected mode is **Cellar Mode (Mode = 1)**.



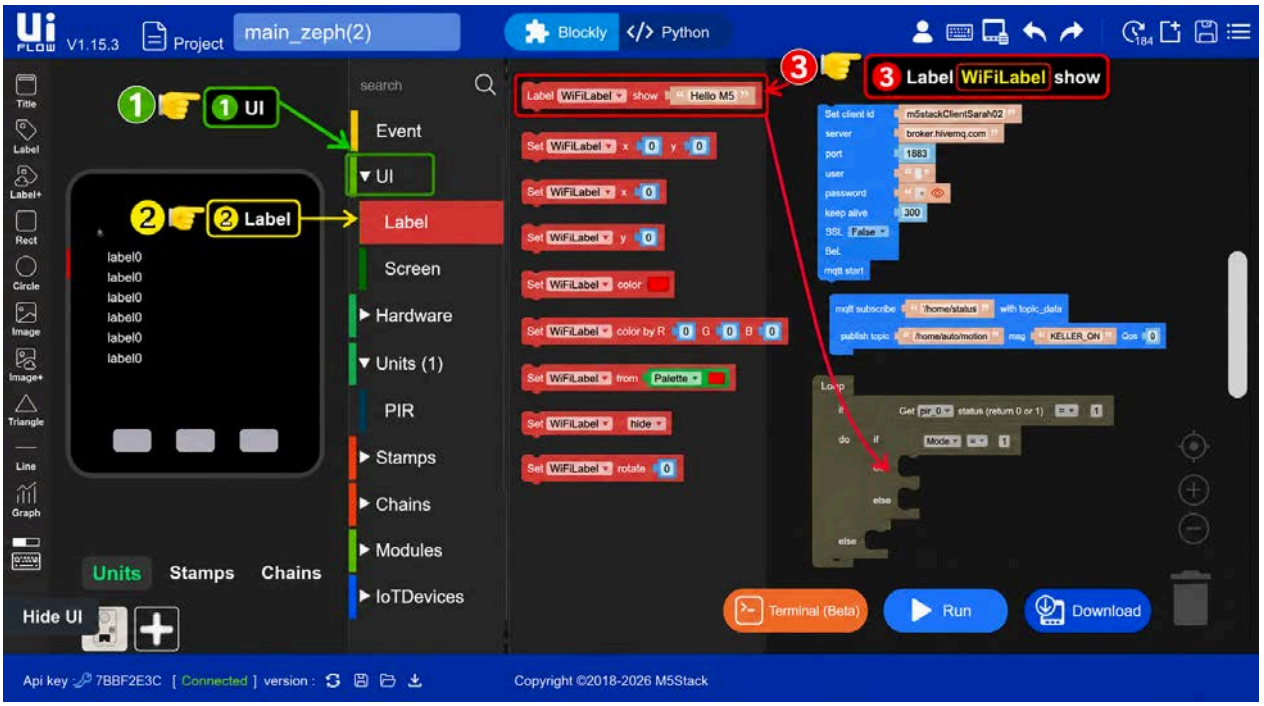
⬇⬇

Note: make sure to change the number block to 1 to indicate cellar mode as illustrated below.

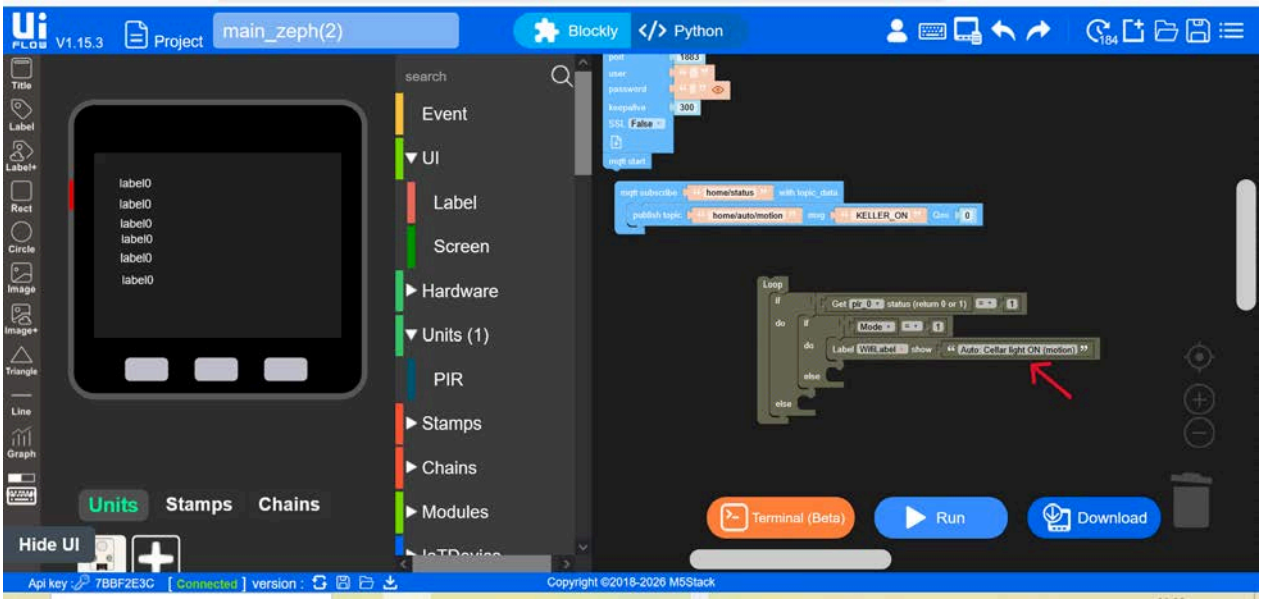


Step 13

Afterwards go to the UI section and click on Label then first drag the WifiLabel show and add it inside the **do** section of the nested **if** block, and set the text to: *Auto: Cellar light ON (motion).* This message will be displayed on the M5Stack screen whenever motion is detected while **Cellar Mode** is active.

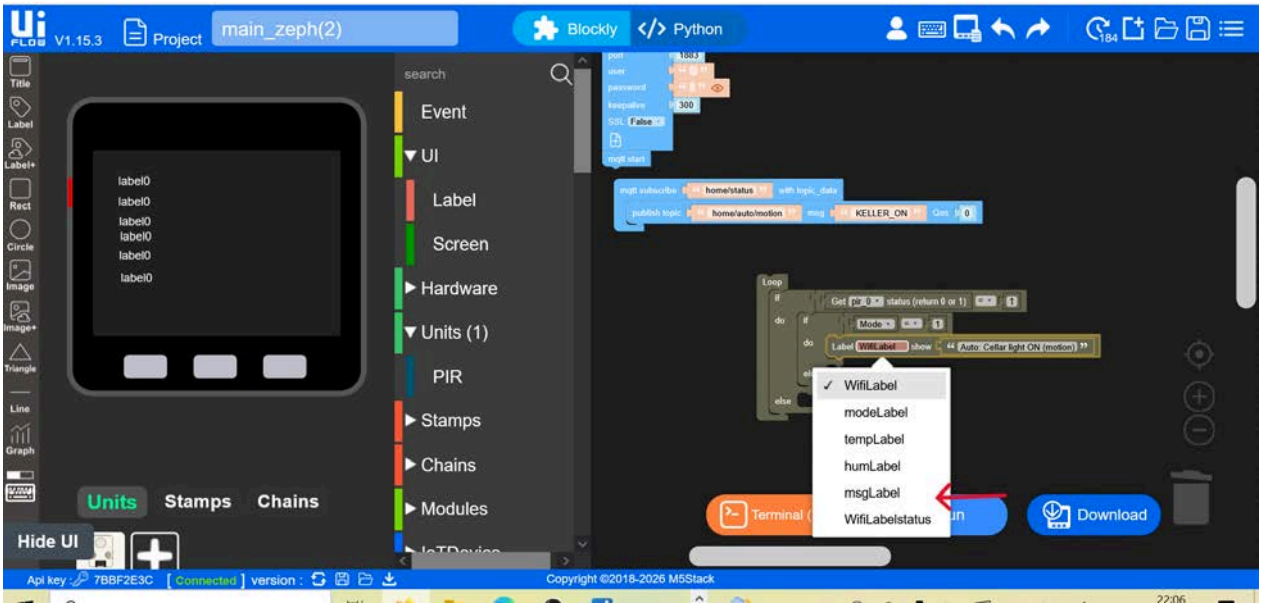


After adding it to the do block set the text to: *Auto: Cellar light ON (motion) as shown below*



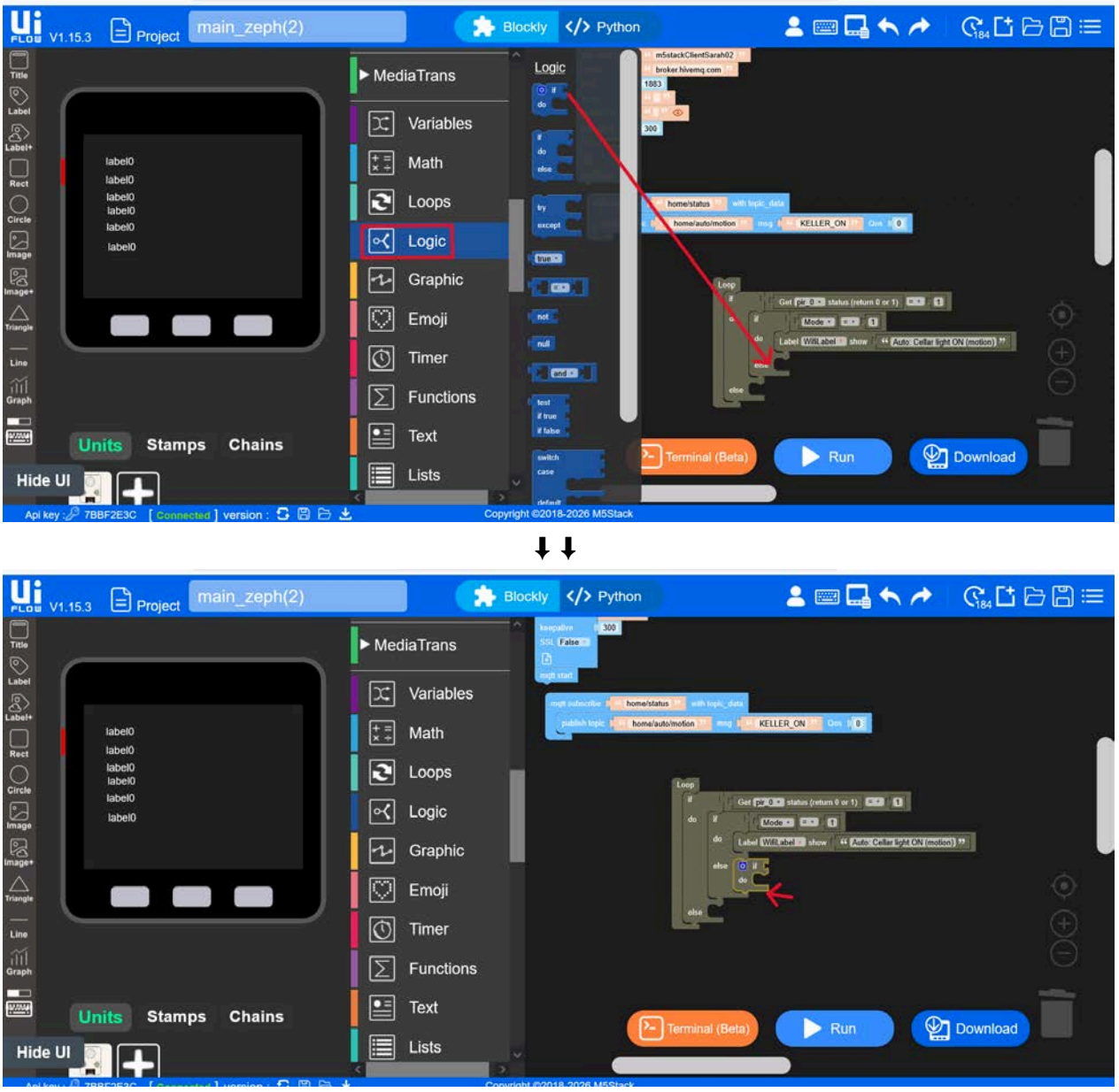
⬇⬇

Then click the `WifiLabel` dropdown and select `msgLabel`, as shown below.
Note: This ensures that all motion status messages are displayed in the correct label on the M5Stack screen.



Step 14

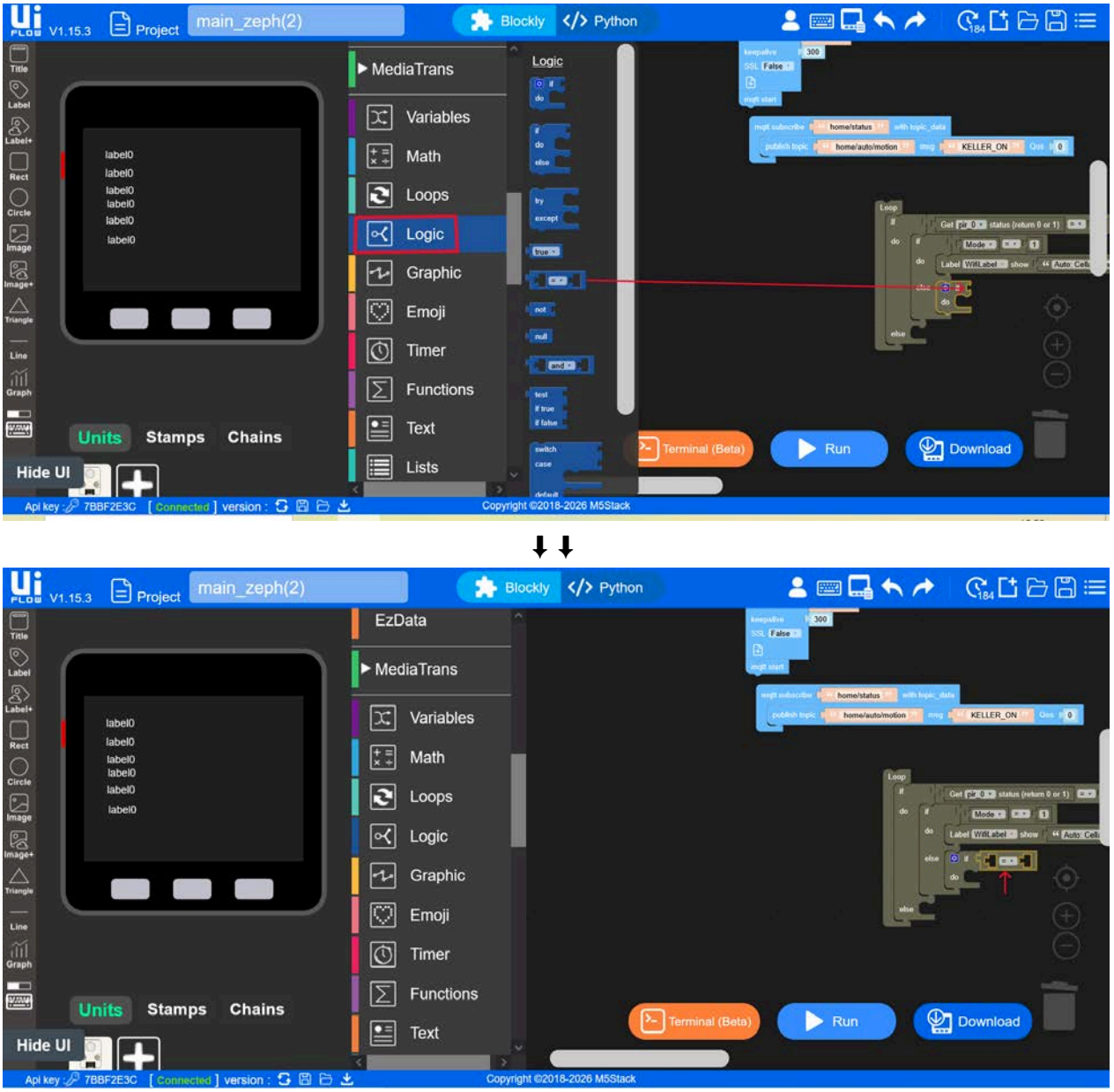
To create the motion detection for the second room(STORE ROOM) block, go to the logic category and drag the first nested **if do** block and add it to the **else** branch shown below.



Step 15

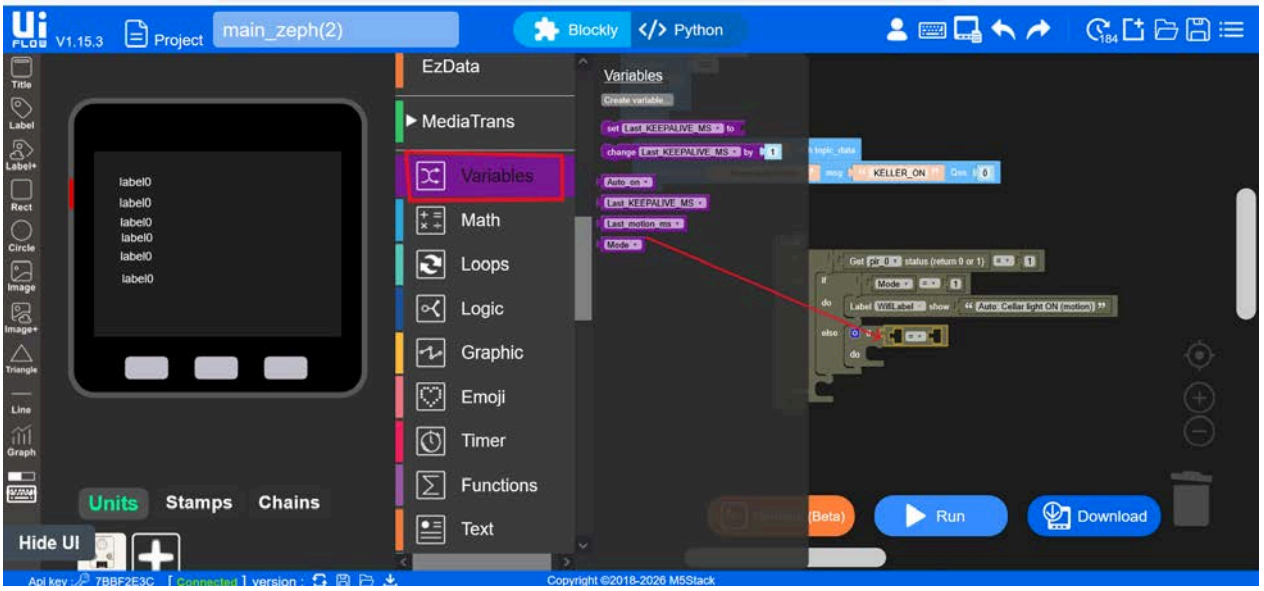
Go to the **Logic** section and drag another **comparison (equals)** block and add it to the if do condition shown below.

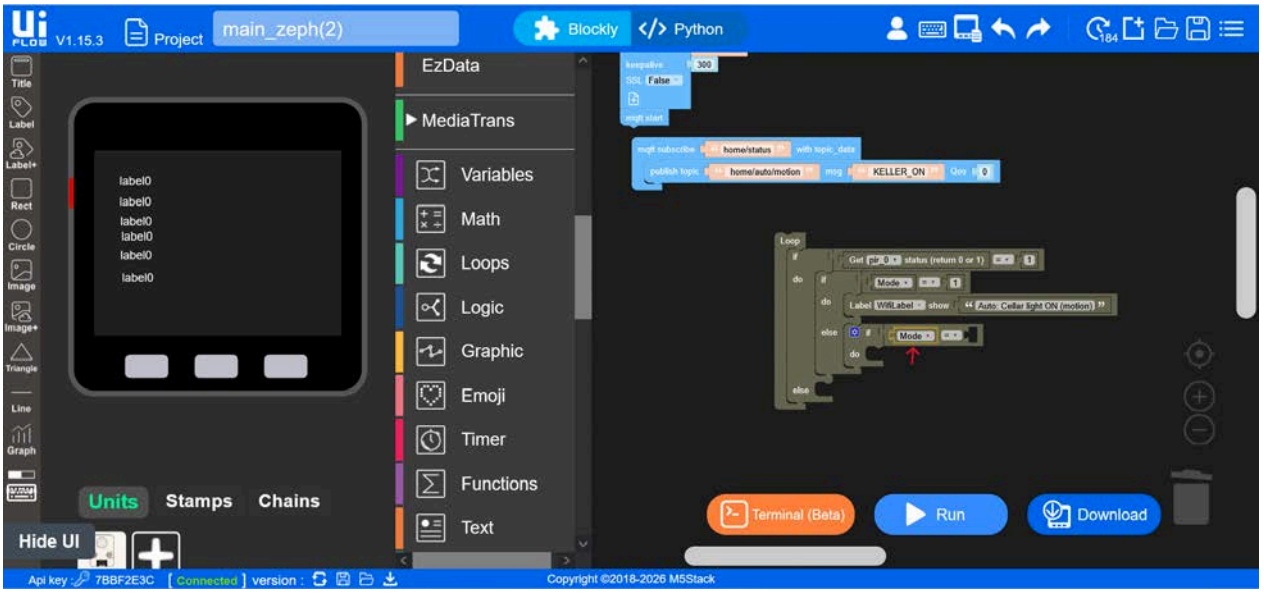
.



Step 16

Afterwards go to variables, drag the Mode variable into the left side of the comparison block and set the value on the right side to **2**. This condition checks whether **Store Room Mode (Mode = 2)** is currently selected.

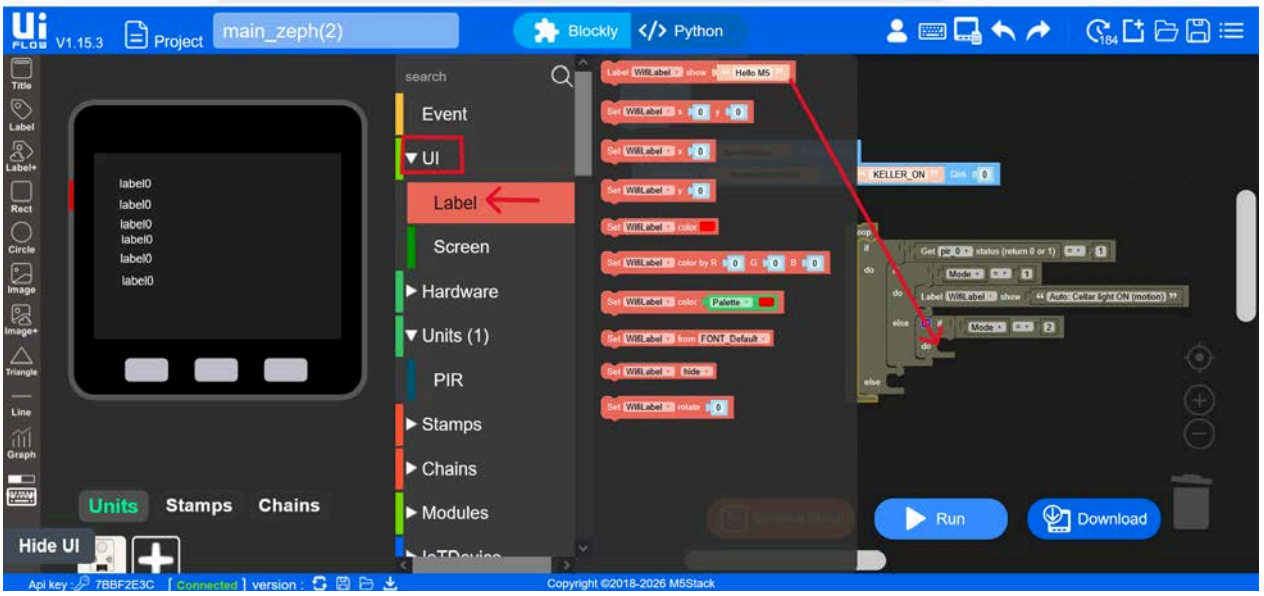

V1.15.3
Project
main_zeph(2)
Blockly
Python
EzData
MediaTrans
Variables
Math
Loops
Logic
Graphic
Emoji
Timer
Functions
Text
Units
Stamps
Chains
Hide UI
Run
Download
Copyright ©2018-2026 M5Stack


V1.15.3
Project
main_zeph(2)
Blockly
Python
EzData
MediaTrans
Variables
Math
Loops
Logic
Graphic
Emoji
Timer
Functions
Text
Units
Stamps
Chains
Hide UI
Terminal (Beta)
Run
Download
Copyright ©2018-2026 M5Stack

To set the variable mode to 2 go to math category, drag the 0 block and insert it to the right hand side of the comparison block and change the value to 2 to represent the store room mode shown below..

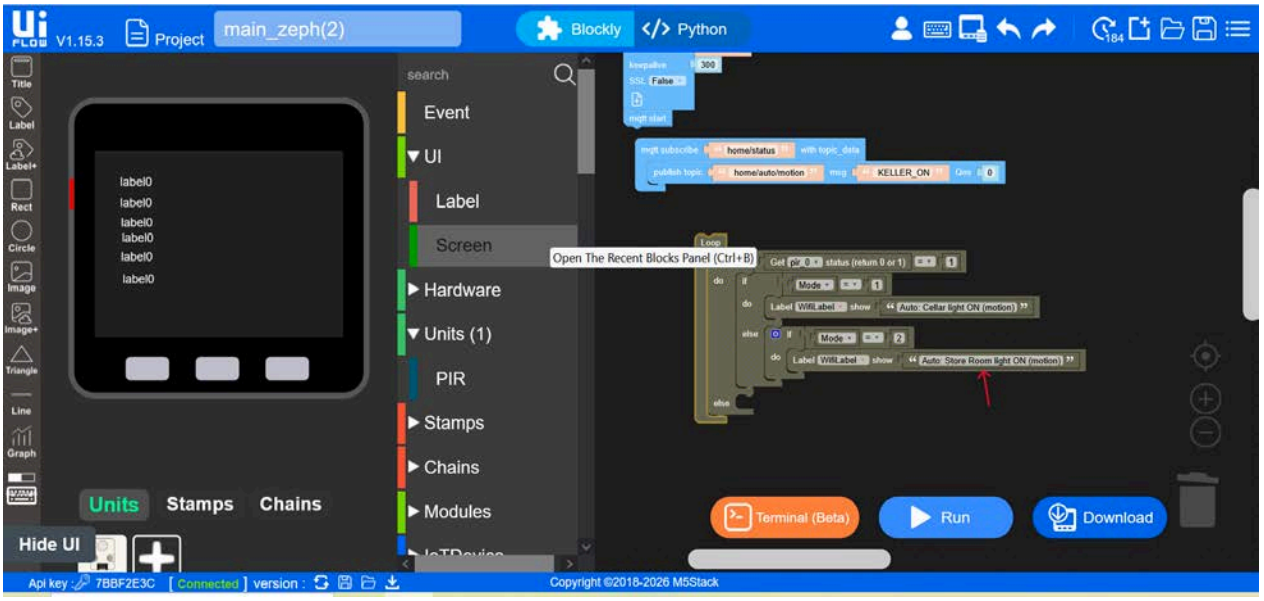
Step 17

Afterwards go to the UI section and click on Label then first drag the WifiLabel show and add it inside the **do** section of the nested **if** block as shown below



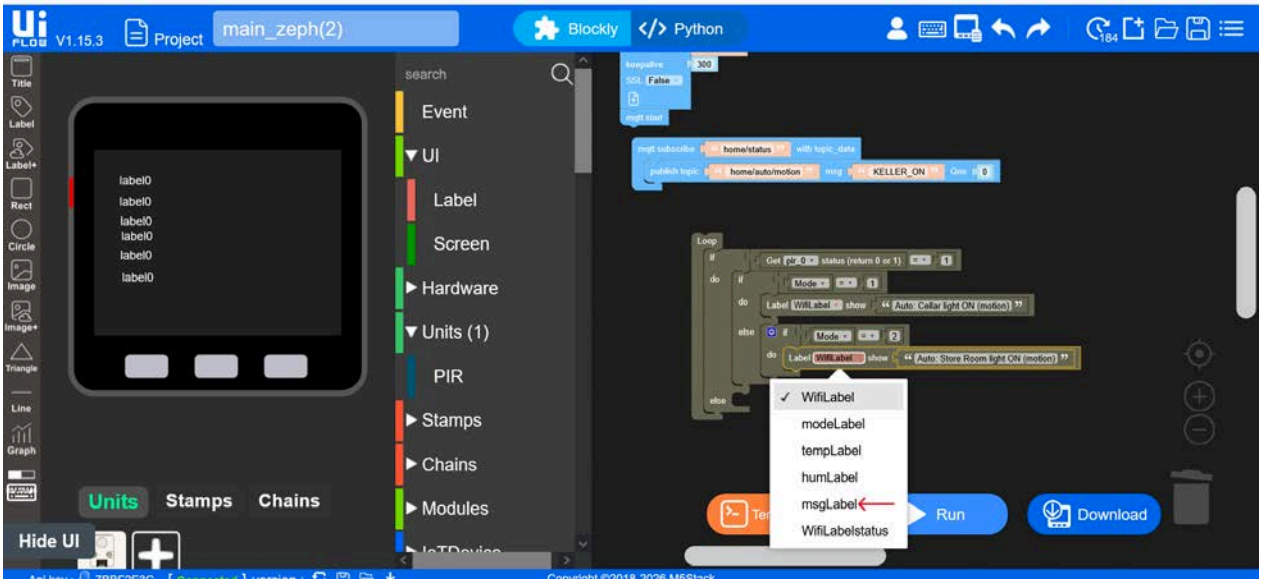
⬇⬇

After adding it to the do block set the text to *Auto: Store Room light ON (motion)*. This message will be displayed on the M5Stack screen whenever motion is detected while **Store room Mode** is active shown below.



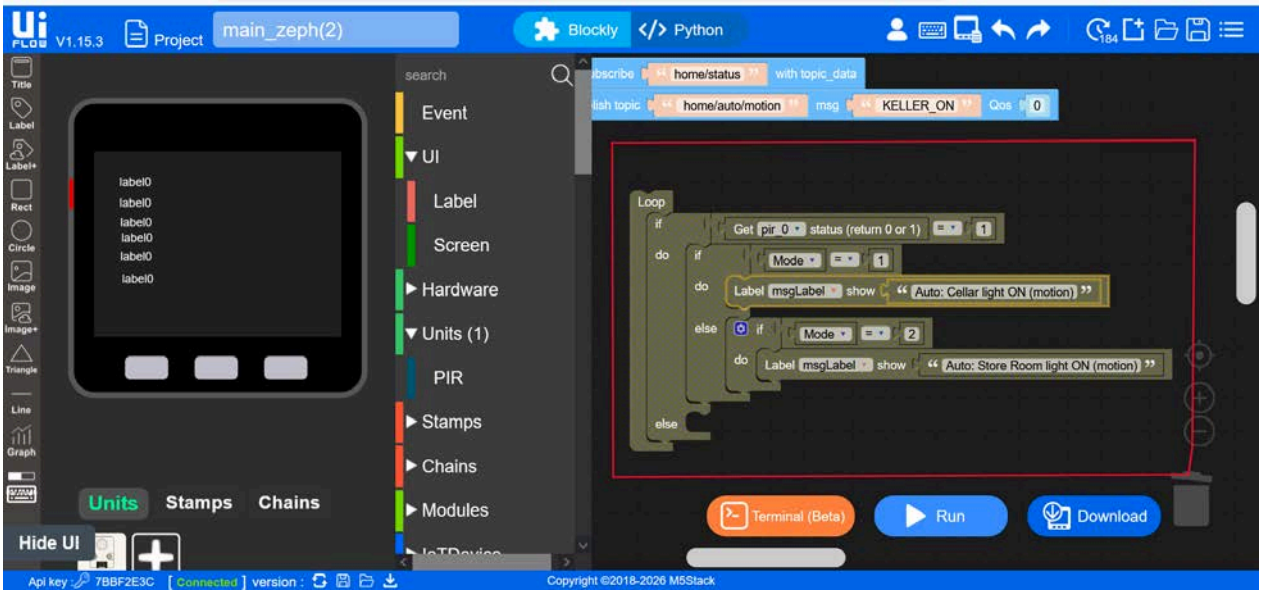
⬇⬇

Then click the `WifiLabel` dropdown and select `msgLabel`, as shown below.
Note: This ensures that all motion status messages are displayed in the correct label on the M5Stack screen.



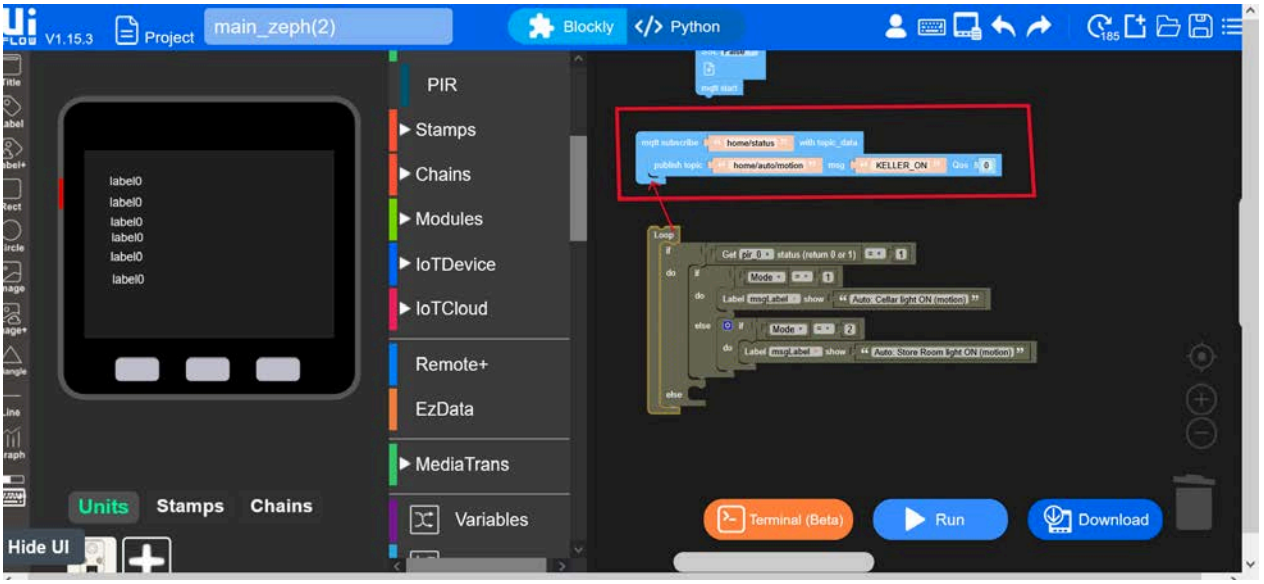
⬇⬇

After the blocks of light ON for mode 1 and mode 2 your block code must look similar to the block below



Step 18

After completing the Light ON logic for Mode 1 and Mode 2, connect the entire motion detection logic to the `mqtt subscribe` block as shown below. . This ensures that the motion detection code is executed whenever a new MQTT message is received. As a result, the M5Stack continuously processes the latest mode selection and updates both the lighting state and the user interface in real time.



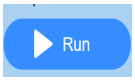
Step 19

After the automatic light-ON logic for mode 1(cellar mode) and mode 2(store room mode) is implemented as shown above, the automatic light-OFF logic is implemented in a similar manner to the light-ON logic described in the previous steps.

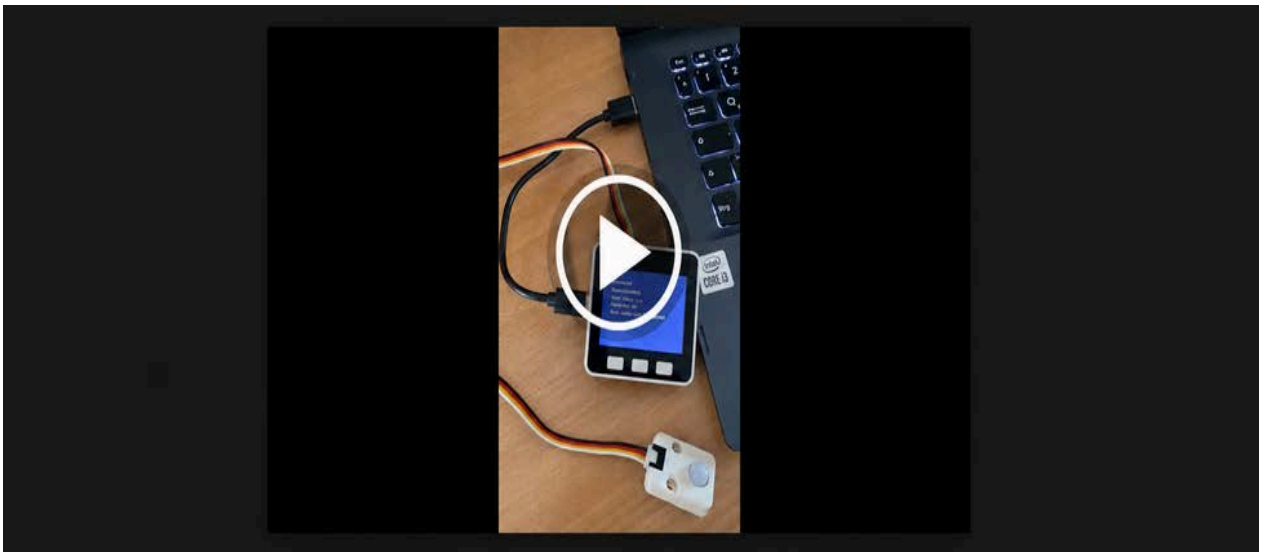
The video below demonstrates how to configure the logic for both **Mode 1 (Cellar)** and **Mode 2 (Store Room)** automatic light-OFF logic.

The video shows how to:

- configure the automatic light-OFF logic for **Mode 1 (Cellar)** and **Mode 2 (Store Room)**,
- add the **5-second delay block.** This block switches the light off when no motion is detected for 5 seconds and,
- update the `msgLabel` to display the appropriate light-OFF status message on the M5Stack screen.

- Then click the Run  button on the Uiflow to test the use case 1

Afterwards, click the Run button in UIFlow, test the automatic light control using the PIR motion sensor with the m5stack device. The following video demonstrates how to test the automatic light control using the PIR motion sensor.



The video demonstrates that, pressing the first button (Button A) switches the M5Stack to Cellar Mode. Moving a hand in front of the PIR motion sensor automatically turns the Cellar light ON. After no motion is detected for 5 seconds, the light turns OFF automatically.

Next, pressing the second button (Button B) switches the M5Stack to Store Room Mode. The PIR motion sensor is then used to automatically turn the Store Room light ON and OFF. Finally, pressing the third button (Button C) switches the system to OFF Mode, disabling automatic light control.

3.6 Save Progress

After implementing Use Case 1 in UIFlow,

1. **Enter the project name** (e.g., `M5stackConfig`) in the project name field.
2. **Click the Save icon** (floppy disk) in the top-right corner to save the project , as illustrated below. Note remember where you saved your project on your computer for later reference.

Note: The project name is user-defined and may be changed to any name of your choice. For consistency, this guide uses **`main_zeph`** as the example project name.

4. Use Case 2 Environmental Monitoring (Temperature & Humidity)

This use case describes how the system continuously monitors the environmental conditions inside a room using one physical ENV Unit (DHT12 sensor) connected to the M5Stack device.

The same ENV Unit (DHT12 environmental sensor) measures both temperature and humidity values in real time, meaning that separate sensors are not required for these two measurements. These values are processed by the system and displayed directly on the M5Stack screen, allowing the user to observe current room conditions at any time.

4.1 Overview

The measured values are:

- displayed on the M5Stack screen and on the mobile app so users can monitor the room conditions remotely..

Temperature monitoring

The system performs the following actions:

- The DHT12 sensor measures the room temperature.

  Example output on the display: Room Temp: 22.4 C

  This allows the user to monitor the indoor temperature.

Humidity monitoring

- The sensor also measures the air humidity.

Example output : cellar Humidity: 45 %

This allows the system to track cellar air quality.

4.2 Components Required

- M5Stack Core / Core2

- ENV Unit (DHT12 Temperature + Humidity sensor)

- Grove cables

- USB Type-A to USB Type-C cable

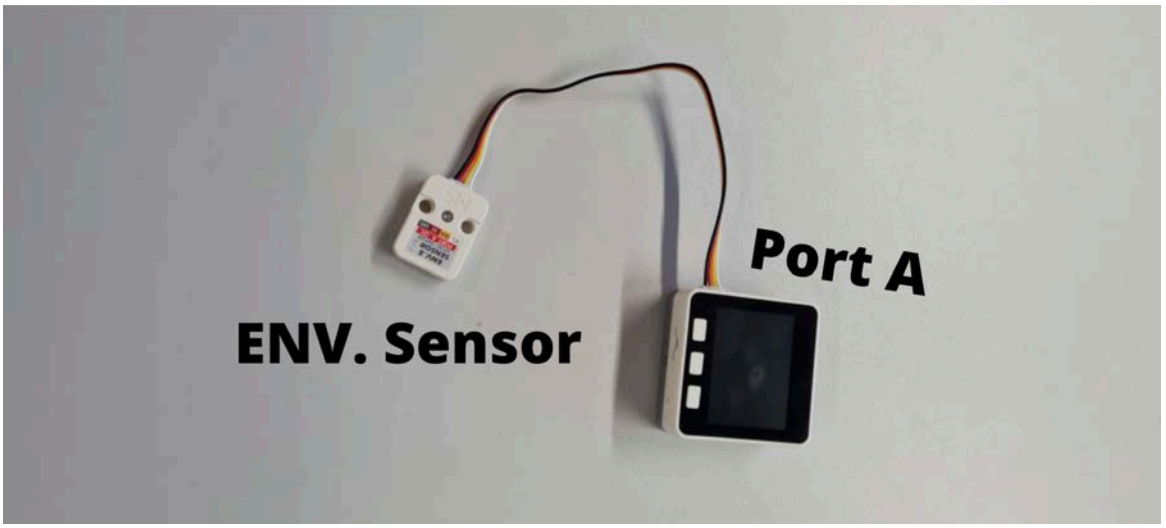
4.3 Hardware and Software Setup in UIFlow

This section shows how to connect the m5stack and its components.

Below is a step by step instruction for the task use case 2(Environmental Monitoring (Temperature & Humidity)

Step 1 (hardware Setup)

Connect the ENV sensor (DHT12) to Port A of the M5Stack using a Grove cable. This sensor is required to measure the room temperature and the air humidity. It will be used in the following steps to read environmental values and display them on the screen.



4.4 Implementation

Before implementing **Use Case 2: Environmental Monitoring**, open the project created in **Use Case 1**. This allows you to continue developing the existing application without recreating the previous configuration.

1. Launch **UIFlow** https://flow.m5stack.com and connect your M5Stack device to your computer via the USB Type-A to USB Type-C cable . Note: Here we assume that you have followed the steps explained in section 2 .(M5stack device flashing process)
   .
2. Click the **Open Project** icon located in the top-right corner of the UIFlow interface.
3. Select the previously saved project (e.g, `M5stackConfig`) on your computer.
4. Click **Open** to load the project to continue implementing Use Case 2

.

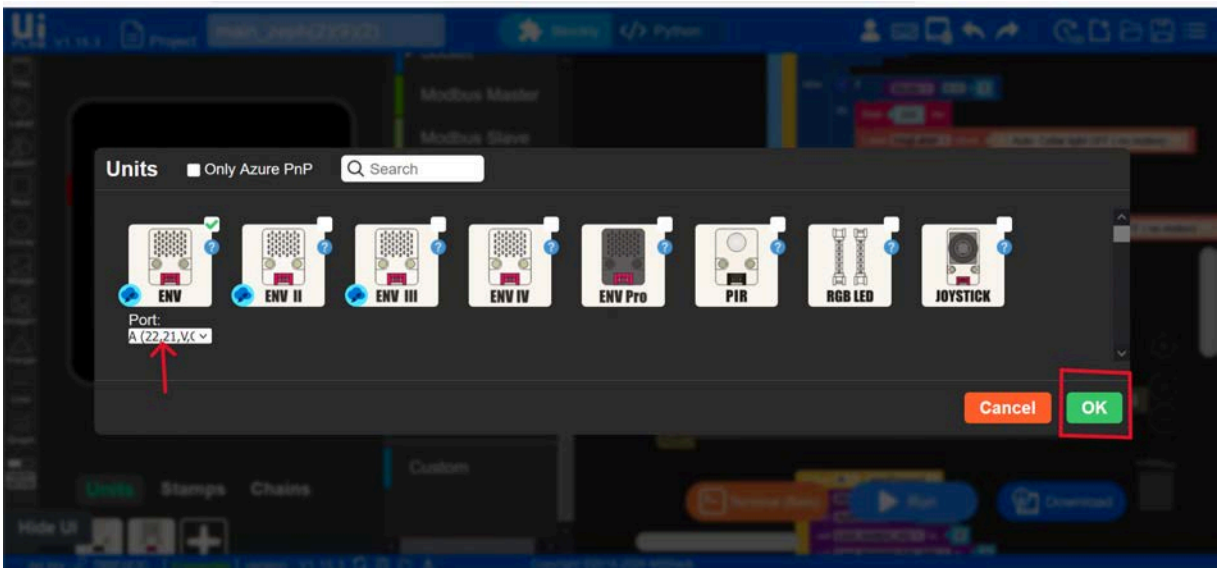
Note :In this use case we want to display the temperature and humidity value from the display screen variable created in use case 1.

4.5.1 Add the Environmental Monitoring Functionality (ENV Sensor)

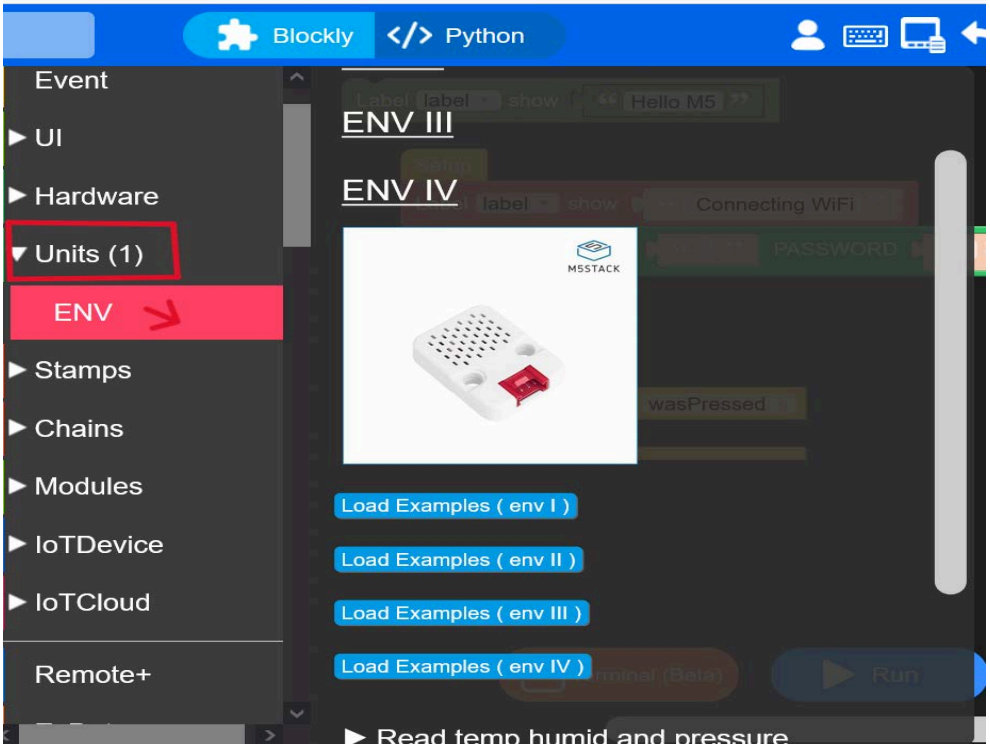
The project from Use Case 1 will now be extended by adding support for temperature and humidity monitoring using the ENV IV sensor. The following steps explain how to read the sensor values, display them on the M5Stack screen, and publish them to the web application via MQTT.

1. Open UIFlow and go to the **Units** in the lower-left corner. . Click the “+ (Plus) icon”

2. add the ENV II sensor to your project and press ok



3. After adding it, the sensor will appear in UIFlow under Units as shown below.

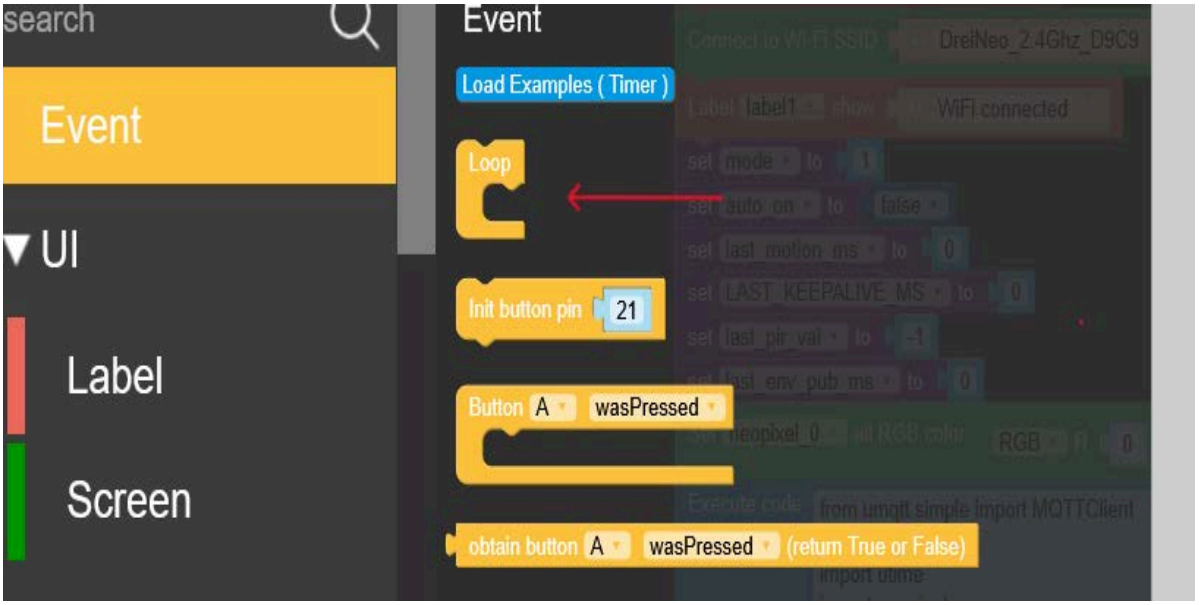


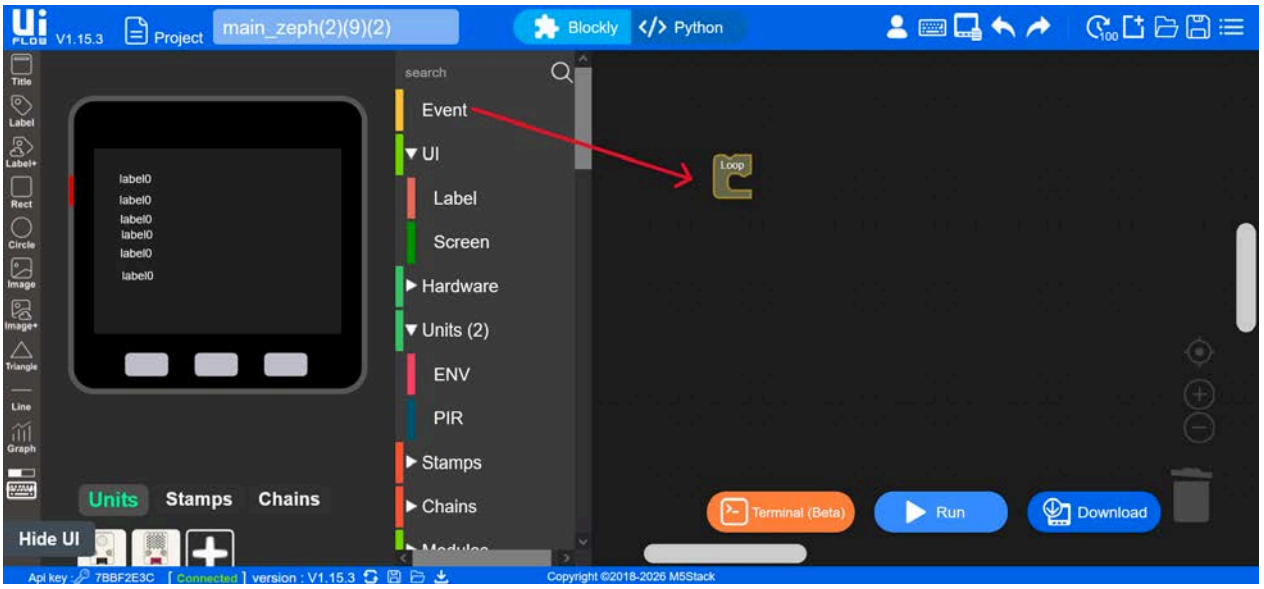
Note: This is important because the program needs to know which sensor should be used when reading the temperature and humidity values.

4.5.2 Create a Continuous Reading Loop

Before implementing the sensor logic, it is necessary to create a loop that allows the system to run continuously. This is important because the temperature and humidity values change over time, and the system must constantly update these values to provide real-time monitoring.

1. Go to the **Event section** in UIFlow and drag the **Loop** block into your program..





This block ensures that the program runs continuously while the device is powered on.

4.5.3 Create variables and Read the sensor values

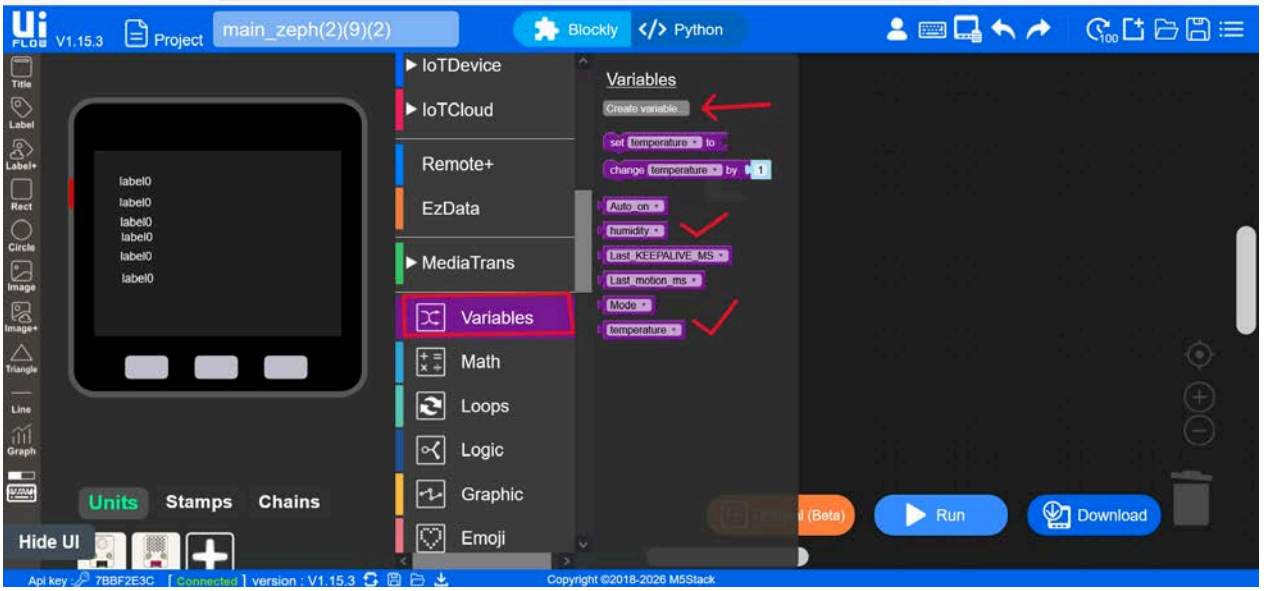
Create two variables to store the sensor readings.

1. Open **Variables**.
2. Click **Create Variable**.
3. Create a variable named:

- `temperature`

4. Create another variable named:

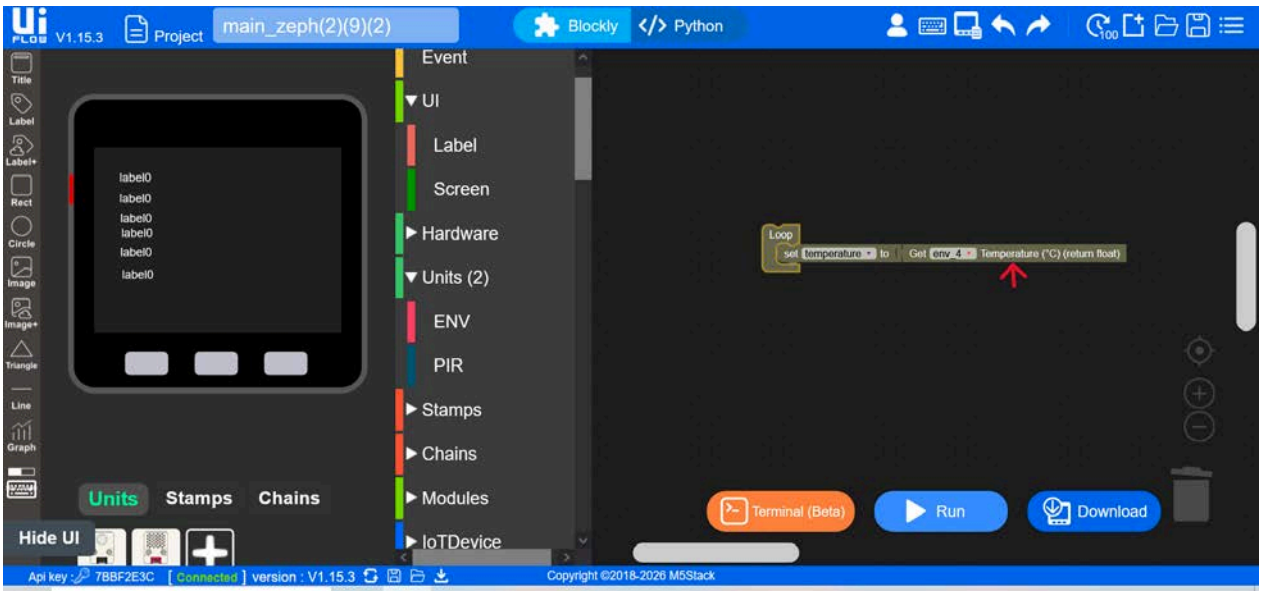
- `Humidity`



5. Drag a **Set temperature variable** block inside the **Loop** block.

6. Afterwards go to the Units **category and click on ENV** ,then drag the get **Temperature (°C)** block into the set temperature

.



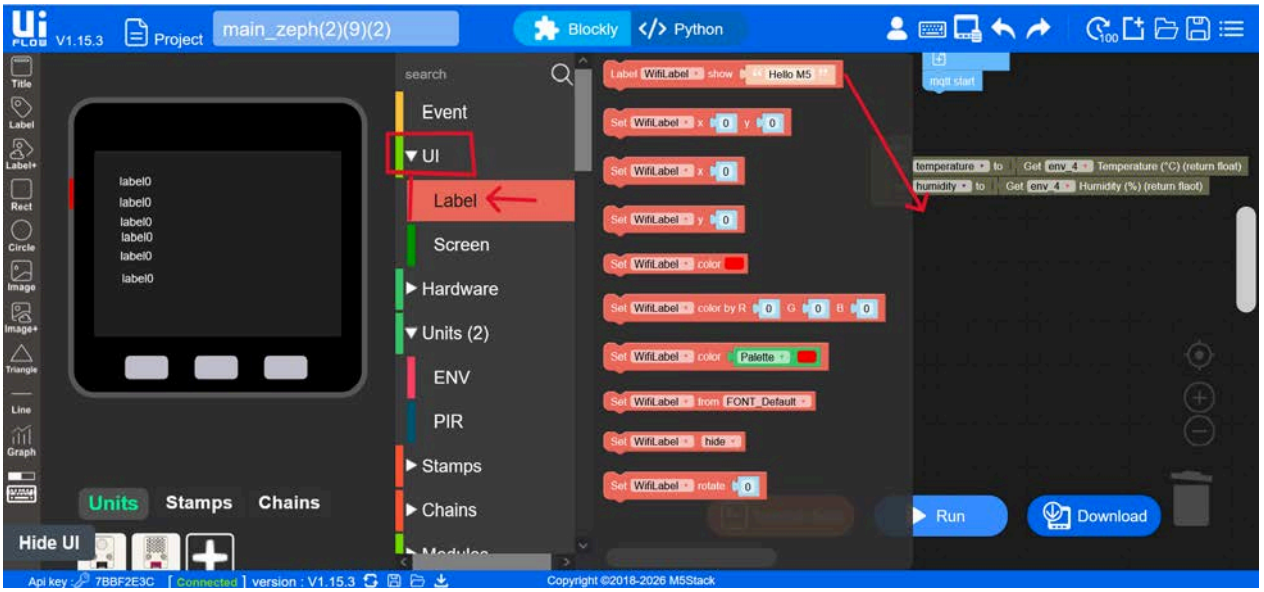
7. Afterwards repeat the same process for humidity.Set humidity then Go to the Units **category and click on ENV** ,then drag the get **humidity (°%)** block into the set humidity as shown in the video below

4.5.4 Display Temperature and Humidity on the Screen

After reading the temperature and humidity values, the next step is to display them on the M5Stack screen so the user can see the current environmental conditions.

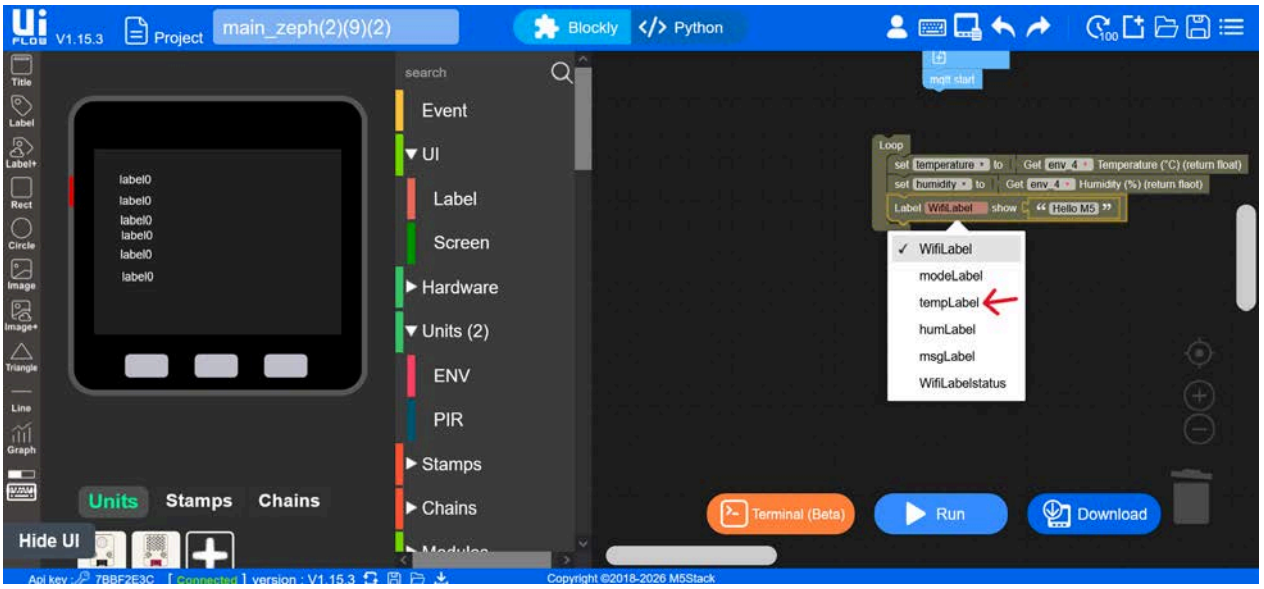
Step 1

Go to the UI category and click on Label, then drag the **Label show** block into the **Loop** block.



Step 2

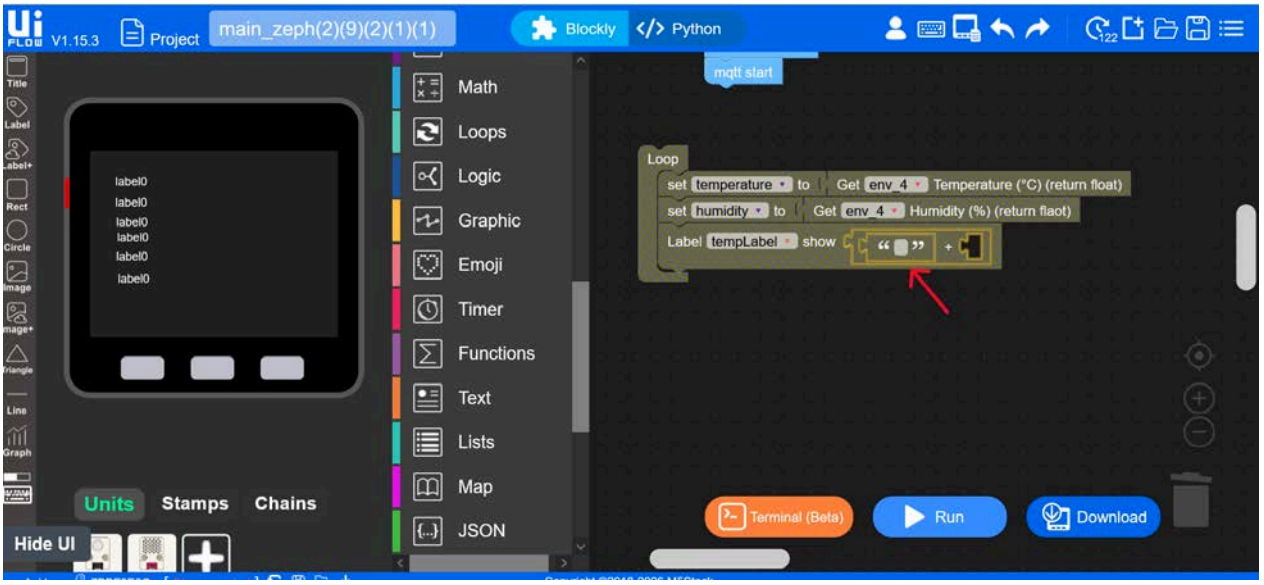
Then from the drop-down menu, select **tempLabe** as shown below.



Step 3

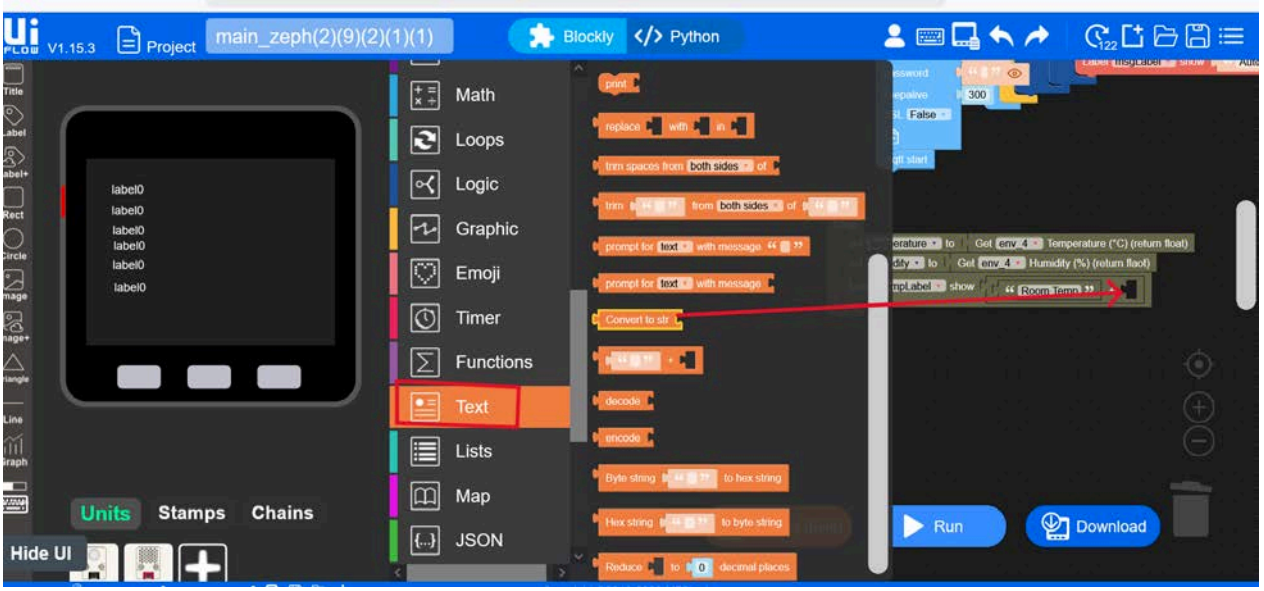
To display the current room temperature on the M5Stack screen, the sensor value must first be converted into text. This allows it to be combined with a descriptive label before being displayed.

1. Click **Text** in the left Blockly menu and drag the **Join (+)** block into the **Label show** block (for **tempLabel**) as shown below.

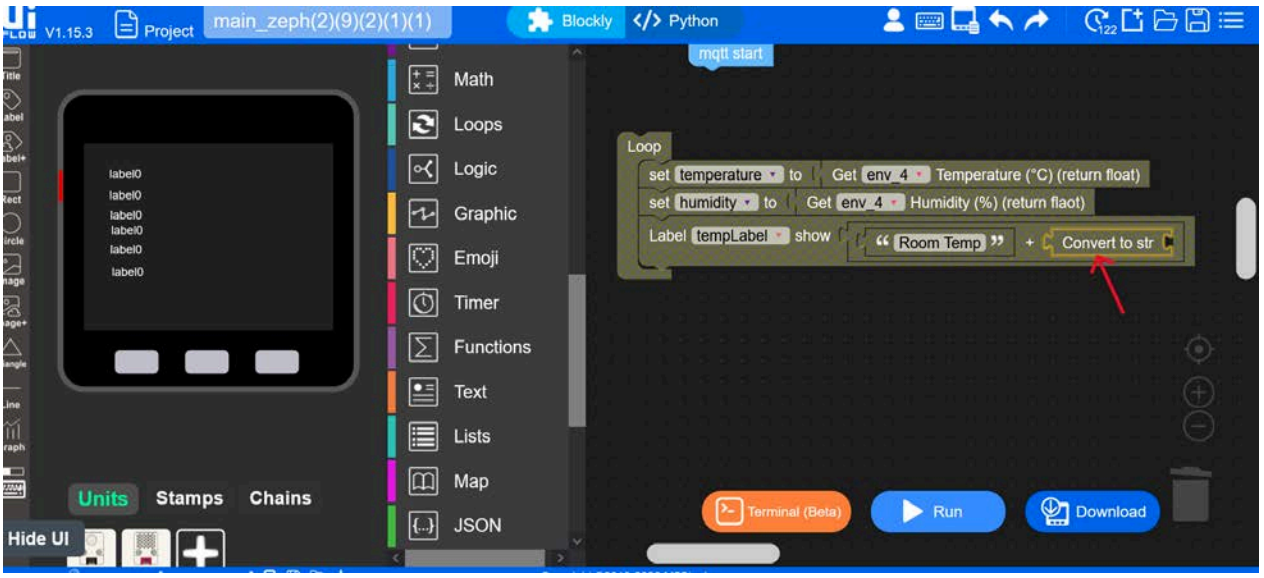


2. In the left input of the **Join (+)** block, enter the text:Room Temp

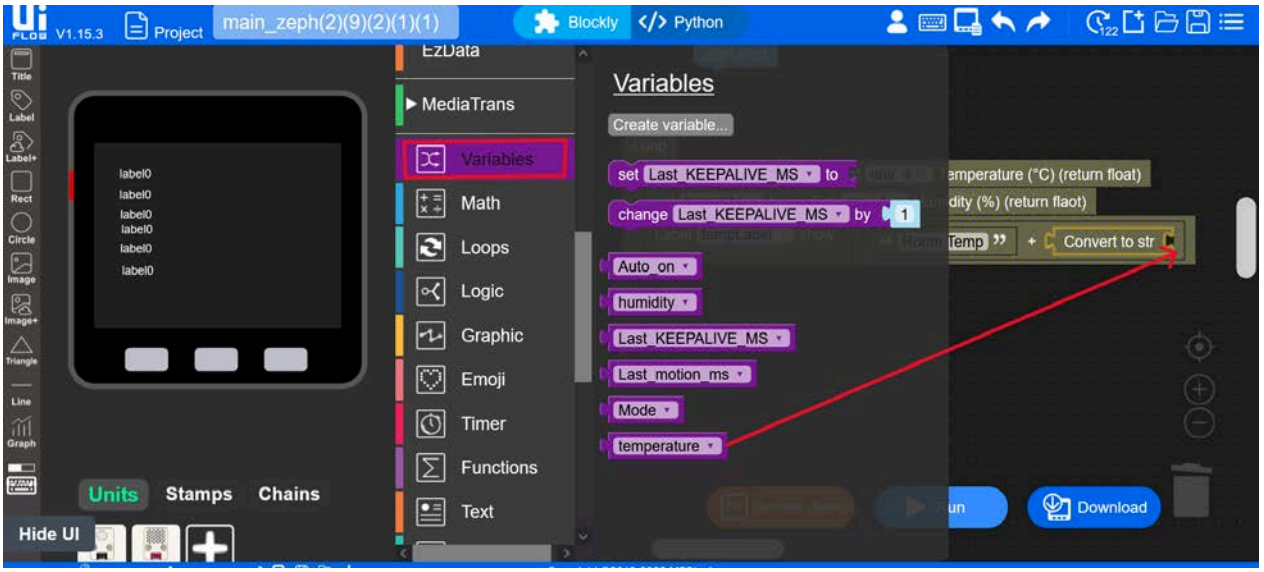
3. From the **Text** category, scroll down and locate and drag the **Convert to str** block into the right input of the **Join (+)** block.



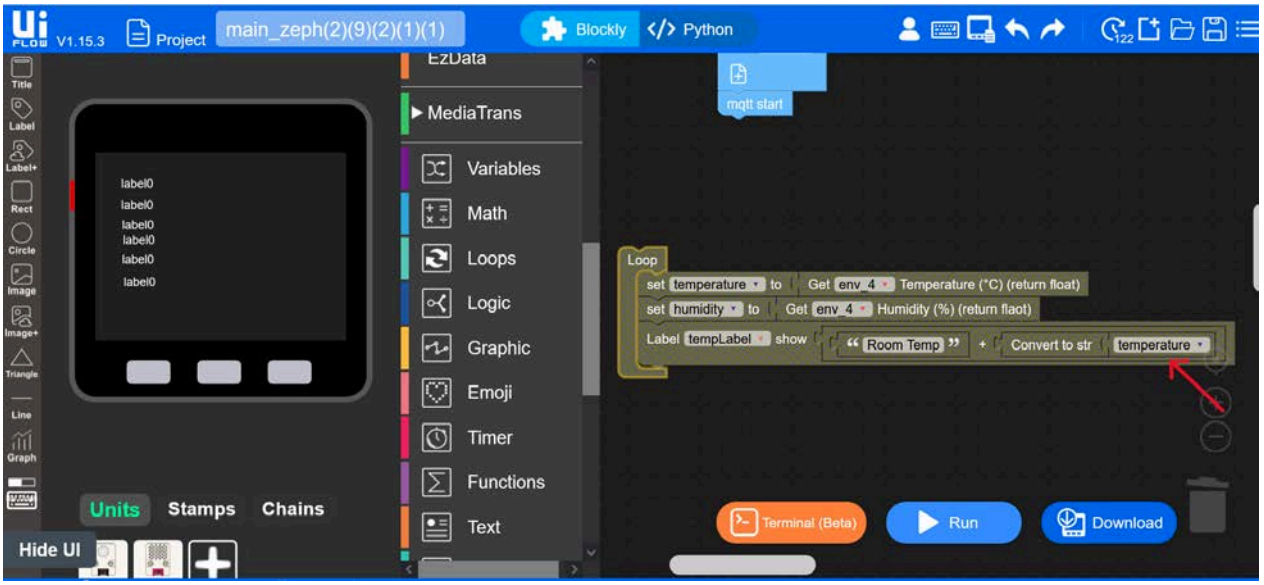
⬇⬇



4. Click **Variables** and drag the **temperature** variable into the **Convert to str** block.



⬇⬇

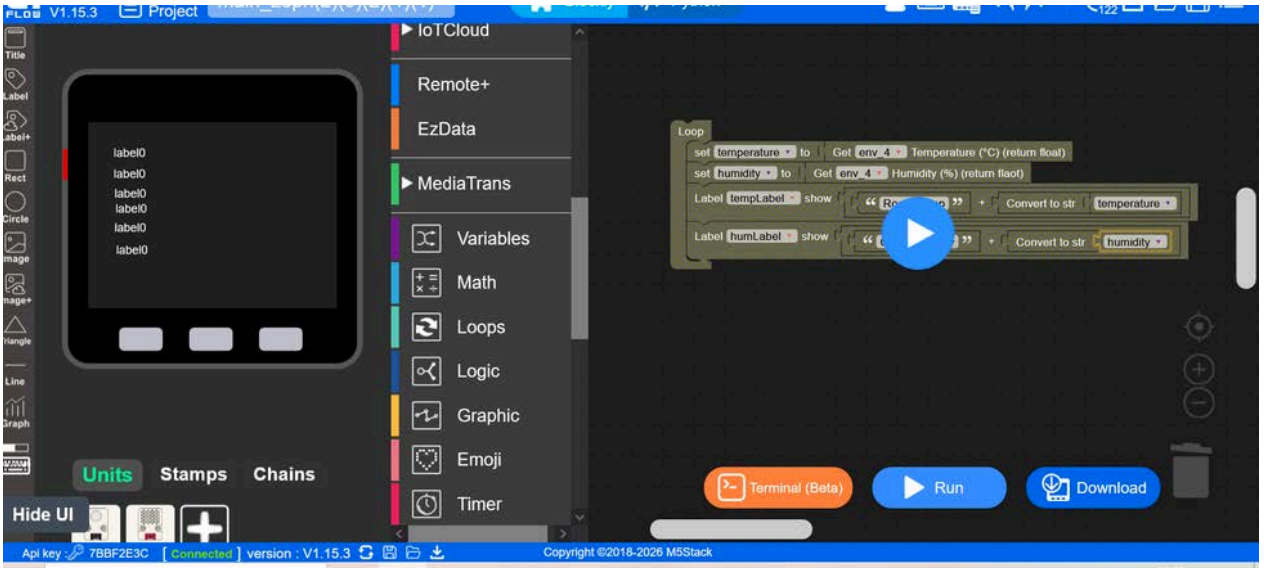


Note: The completed block combines the text **"Room Temp:"** with the current temperature value and displays it on the M5Stack screen. The output appears similar to: *Room Temp: 27.3.*

Step 4

The cellar humidity value is displayed using the same approach as shown above under step 3. The value is converted to text and combined with a descriptive label before being shown on the M5Stack screen.

The video below demonstrates how to configure the humidity display on the M5Stack screen. First click on units category and click on label to add a second `Label show` block , followed by from the text category click the `Join (+)` block to combine the label text with the humidity value. The `Convert to str` block is then used to convert the `humidity` variable into text before it is displayed using `humLabel`.



4.5.5 Publish the temperature value and humidity value

To enable real-time monitoring in the web application, the temperature and humidity values must be published to the MQTT broker. Each sensor value is assigned to a separate MQTT topic so that the frontend can receive and display the latest readings. Please follow the steps below to enable this.

1. First,click on the Advanced category from the middle right search section as shown below

2.After clicking on the advanced locate and click MQTT and Drag **two Publish Topic** blocks and place them below the humidity display block.

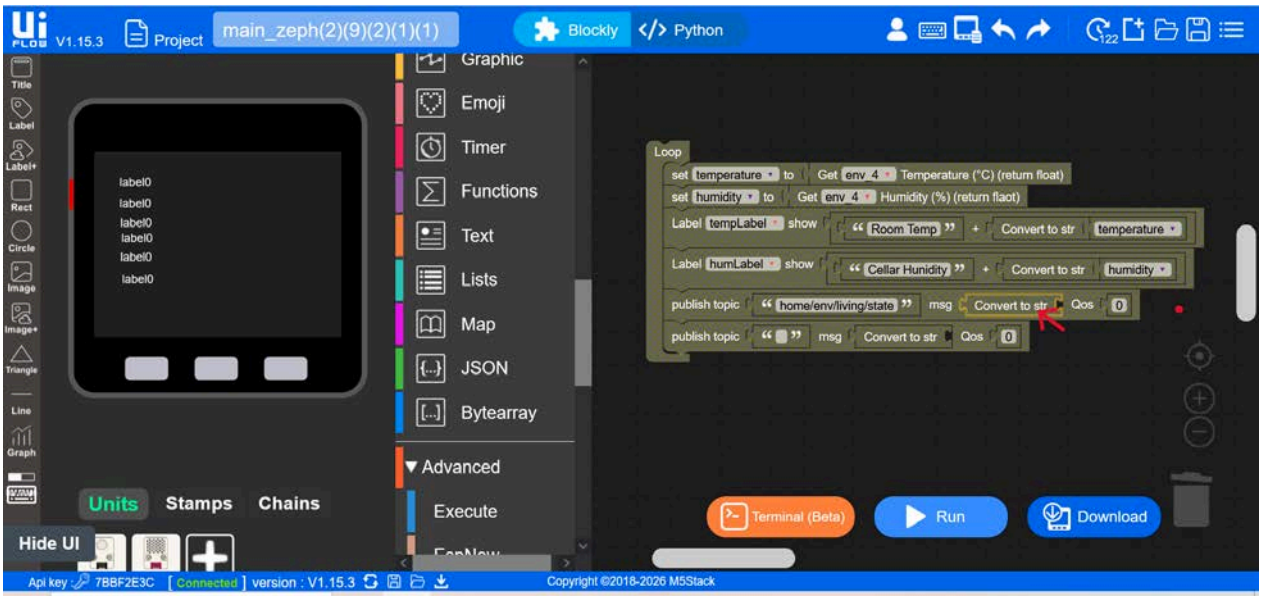
Note: These blocks are used to publish the temperature and humidity sensor readings to the MQTT broker.

3. Configure the first **Publish Topic** block as follows:

- **Topic:** `home/env/living/state`

Note:This MQTT topic is used to publish the temperature data so that the web application can receive and display the latest temperature reading in real time.

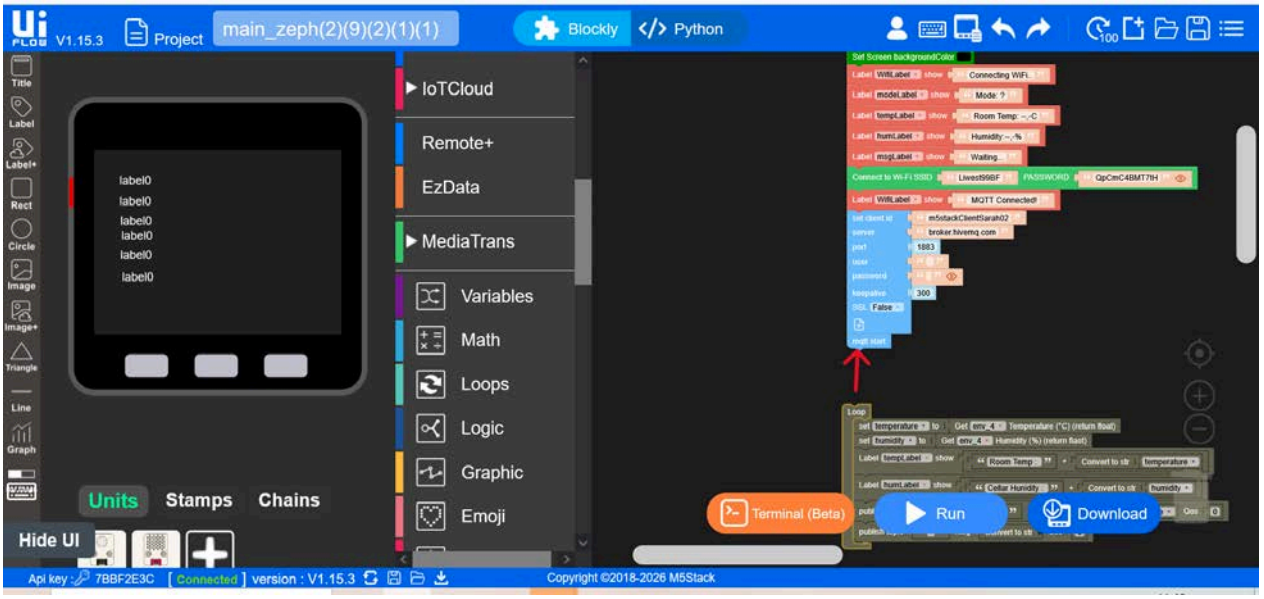
4. Afterwards click on the text category from the middle right section, then drag a **Convert to str** block into the **Message** field.

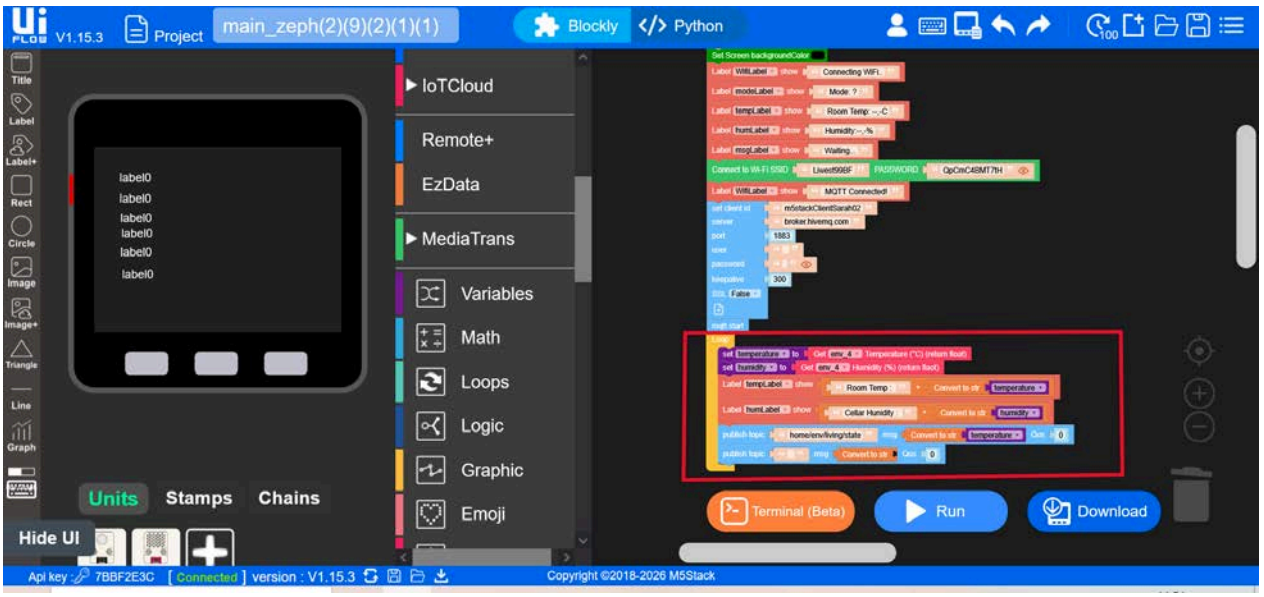


Note:This block converts the temperature value from a numeric data type into a text string, allowing it to be transmitted correctly through the MQTT *Publish Topic* block.

5 Then click the variable category and insert the **temperature** variable to the **Convert to str** block.

6. After configuring the Publish the temperature block, connect the Loop block directly below the `mqtt start` block which was created in use case 1. This ensures that the MQTT connection is established before the program begins reading, displaying, and publishing the temperature and humidity values. Without this connection, the Loop may not execute as part of the main program flow, preventing the sensor values from being published correctly.





7. The next step is to configure the Publish Topic block for humidity.

The video below demonstrates how to:

- configure the second MQTT **Publish Topic** block;
- set the MQTT topic to `home/env/keller/state`;
- insert the **Convert to str** block into the **Message** field;
- add the **humidity** variable; and
- click the **Run** button in UIFlow and observe the temperature and humidity values displayed on the M5Stack screen.

After clicking the **Run** button in UIFlow, the M5Stack begins reading the sensor data. The current room temperature and cellar humidity are then displayed on the M5Stack screen, as shown below.

This image confirms that the ENV sensor is connected correctly and the values are being read successfully.

4.6 Save Progress

After implementing Use Case 2 in UIFlow, Enter the project name (e.g., M5stackConfig) in the project name field as demonstrated in 3.6 Save Progress to save your progress.

5 Use Case 3 – Security Status Monitoring

The goal of this use case is to monitor the status of doors, windows, and locks and show the current security state on the M5Stack screen. In addition, the RGB LED is used as a local visual indicator so that the user can quickly see whether something is open, closed, locked, or unlocked.

5.1 Overview

This use case allows the user to immediately see whether the monitored room is secure.

The system performs the following actions:

Door status monitoring
The system receives the current door status.

For example:

- door open
- door closed

This status is shown on the M5Stack screen.

Window status monitoring
The system receives the window status.
For example:

- window open
- window closed

This helps the user check whether the room is secured.

Local RGB status indication
The RGB LED is used as a local indicator:

- If the RGB led turns green, it indicates that a window or door is open / unlocked
- If the RGB led turns red, it indicates that a window or door is closed / locked

5.2 Components needed

- M5Stack Core / Core2

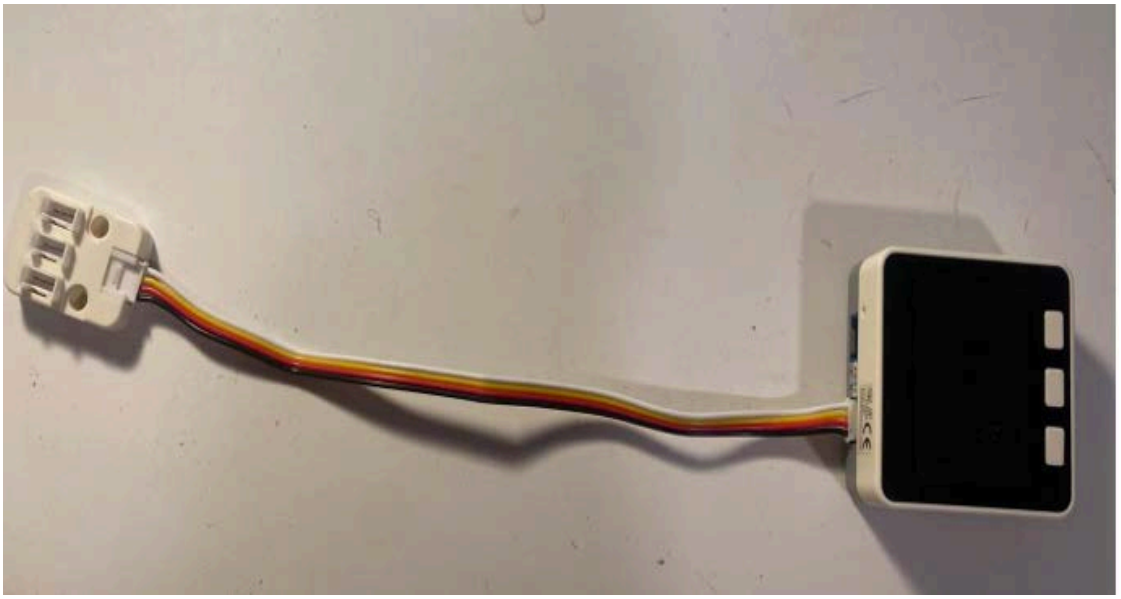
- RGB LED

- Grove Hub needed to connect multiple units at the same time(motion sensor + RGB Led) in port B

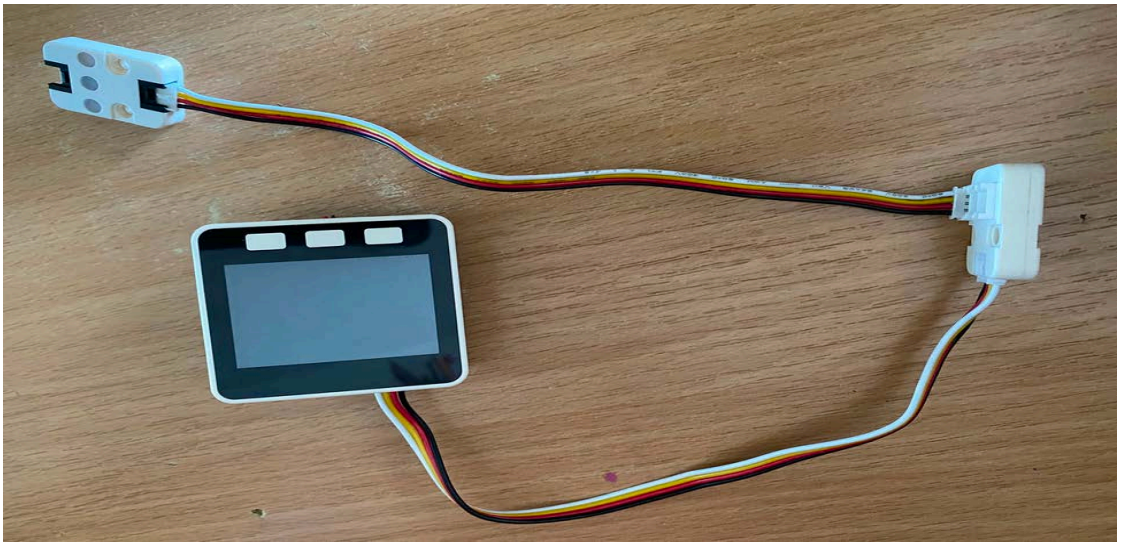
- Grove cable

5.3 (hardware Setup)

Step 1
Connect the Grove Hub to Port B of the M5Stack using a Grove cable.This enables other components to be connected together.



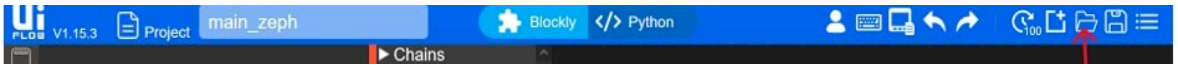
Step 2
Also connect the RGB LED to Port B via the Grove Hub free port using a Grove cable.



5.4 Implementation

Use Case 3 continues from the project created in the previous use cases.

1. Launch UIFlow https://flow.m5stack.com and make sure that the m5stack and its components are connected via usb cable.
2. Click the **Open Project** icon in the top-right corner.



3. Select the previously saved project.
4. In this guide, the project is named: *main_zeph*

5. *Click **Open** to load the project.*

After a successful load, your UIflow will display the last save project as shown in the video below

***Note:** The project name is user-defined. Select the project name that you used when saving the previous implementation.*

Below is a step by step instruction for the task use case 3(Security Status Monitoring)

5.5 Execute Code Blocks

For this use case two Execute Code blocks are required:

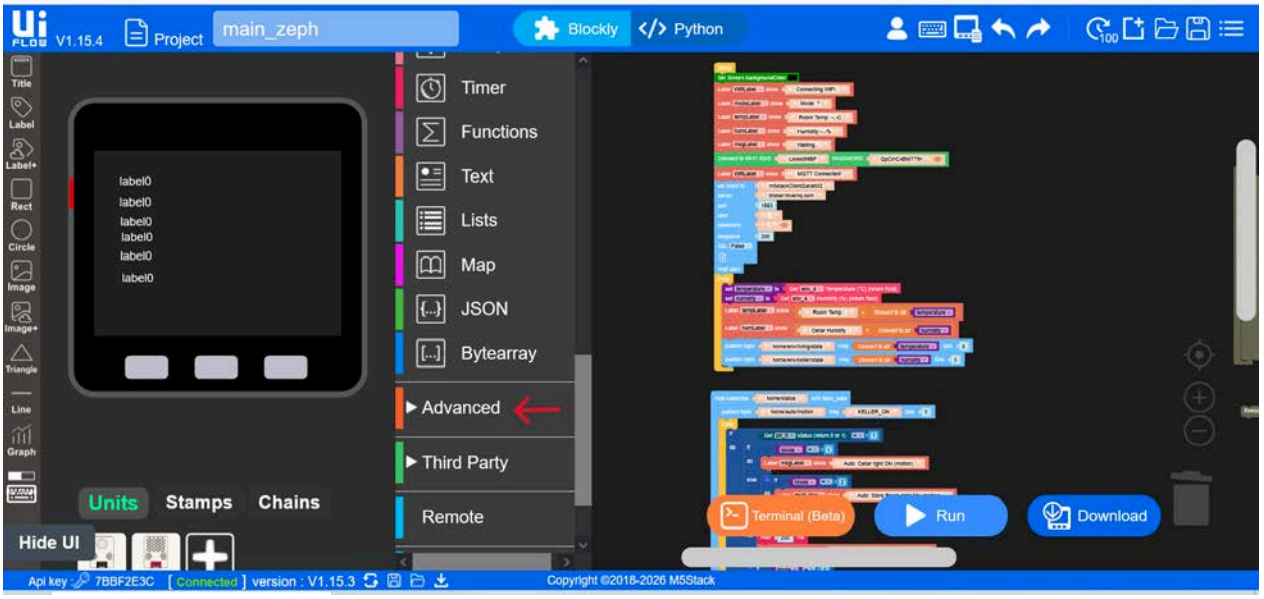
1. The first block configures the security MQTT connection, subscribes to the `home/status` topic, processes incoming messages, updates the M5Stack screen, and controls the RGB LED.
2. The second block continuously checks whether a new security message has been received from the frontend.

Note: Using Execute Code blocks reduces the number of complex Blockly conditions required for processing door, window, and lock messages.

5.6 Add the Security Setup Execute Code Block for first block

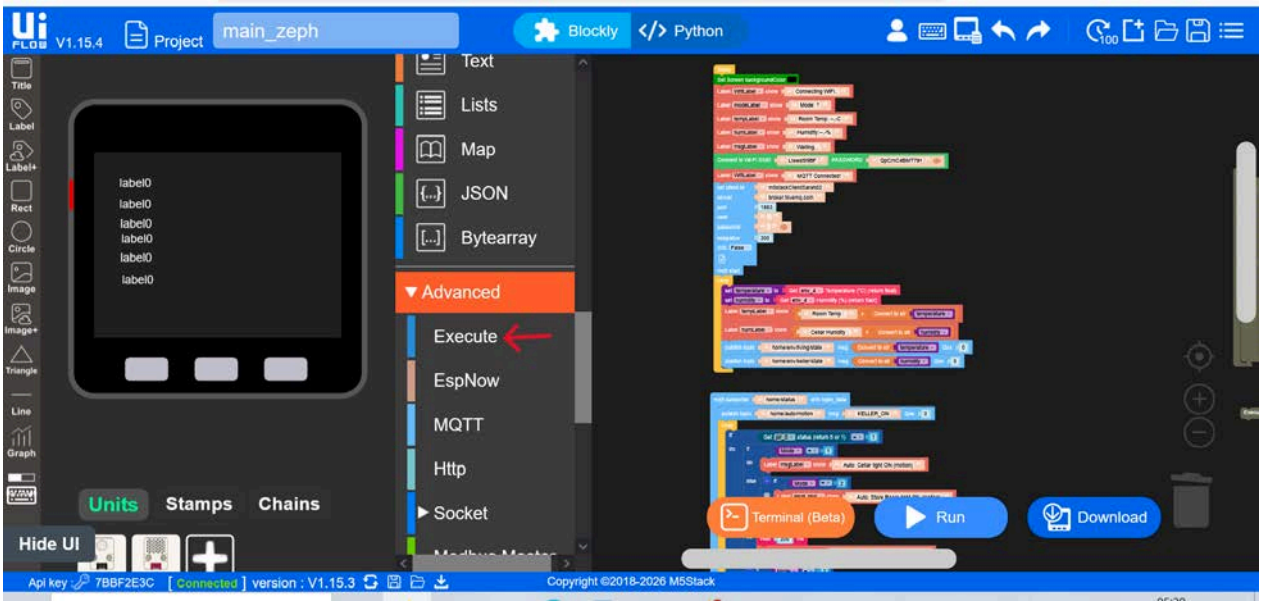
Step 1

In the Blockly menu, click **Advanced** as shown below in the red arrow.



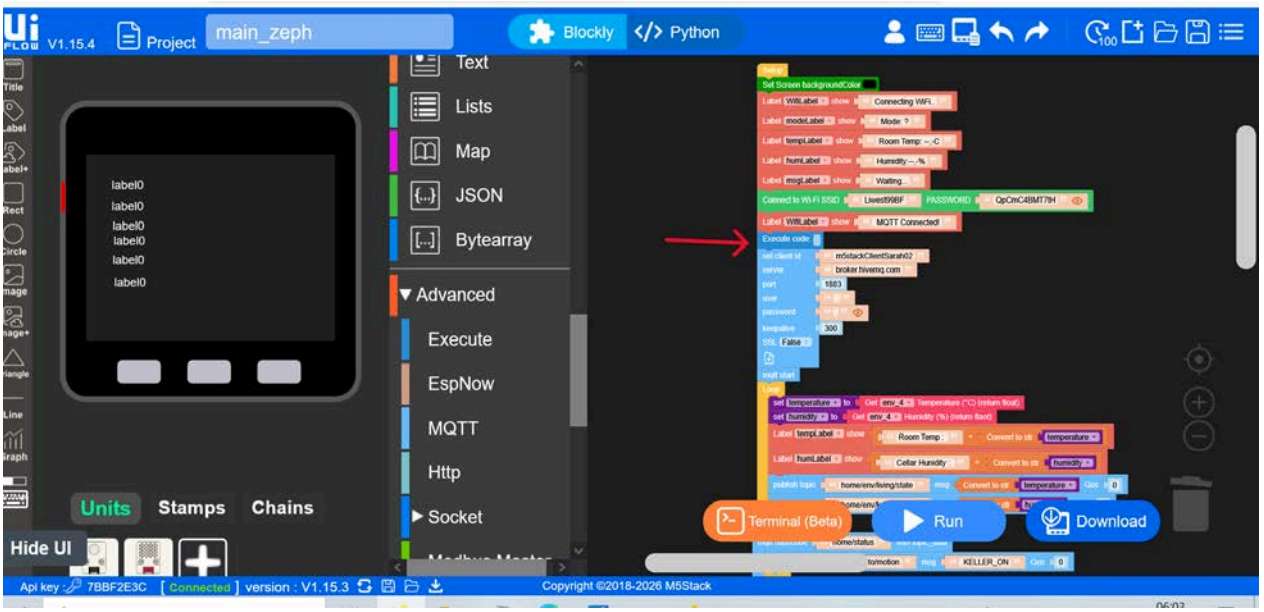
Step 2

Afterwards locate and click **Execute**.



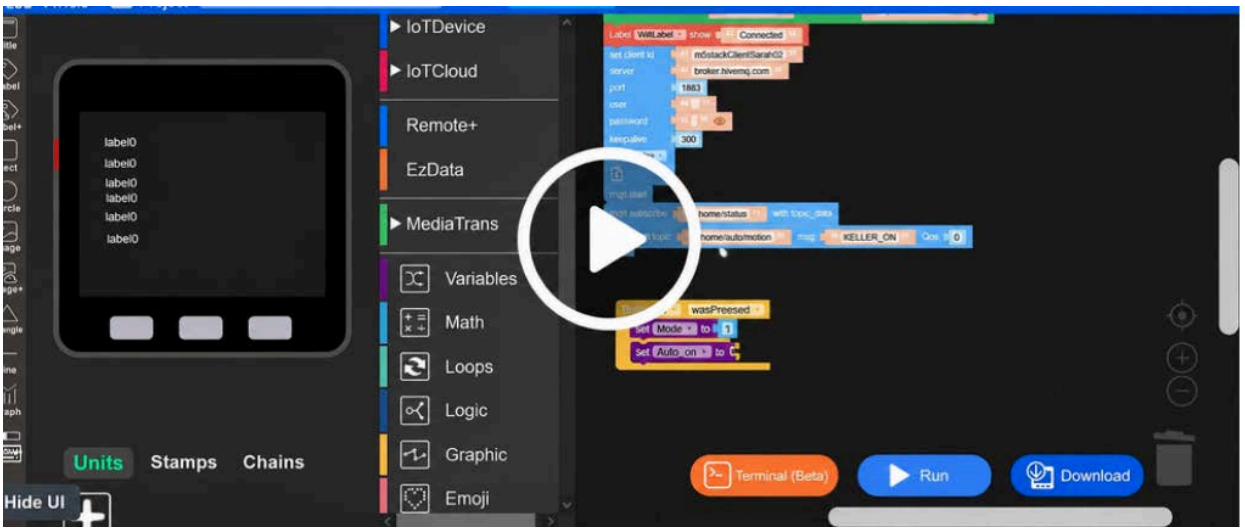
Step 3

Drag an **Execute Code** block into the main setup section. Place it after the Wi-Fi connection has been established and before the main Loop begins.



Step 4

Download and insert this [python code] into the Execute Code block as demonstrated in the video below.



5.6 Why the First Execute Code Block Is Required

The first Execute Code block provided above performs several tasks.

1 MQTT configuration

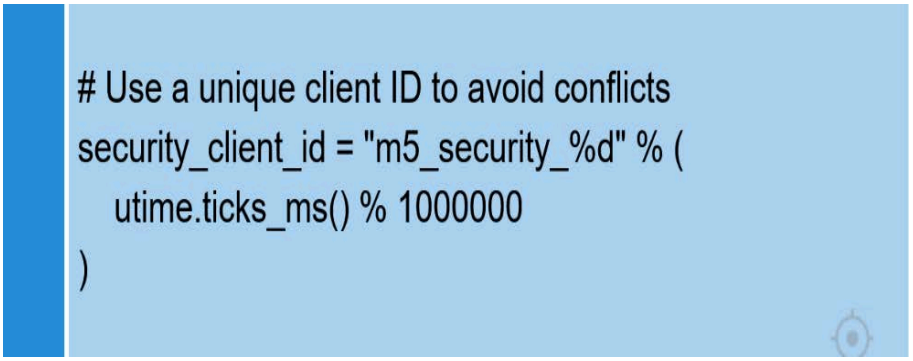
The following values configure the connection:

```
# ========================================
# USE CASE 3: SECURITY STATUS MONITORING
# ========================================

SECURITY_BROKER = "broker.hivemq.com"
SECURITY_PORT = 1883
SECURITY_TOPIC = b"home/status"
```

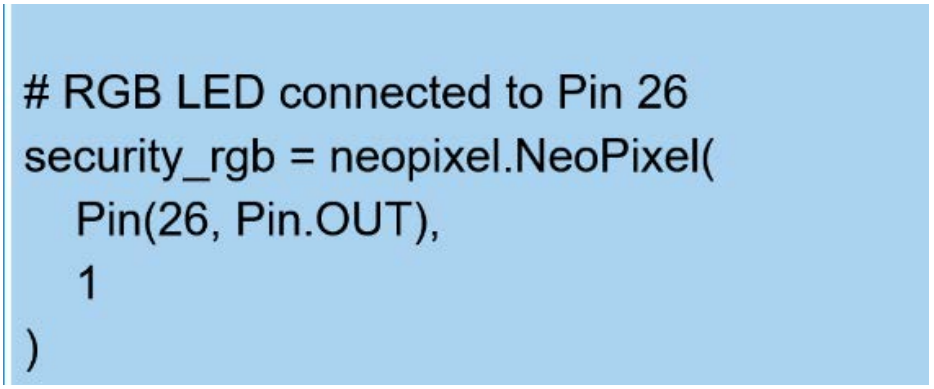
Note: The M5Stack subscribes to `home/status` because the frontend publishes door, window, and lock status messages to this topic.

2 Unique client ID



Note: A unique client ID prevents the MQTT broker from disconnecting the M5Stack because another device is using the same identifier.

3 RGB LED configuration



Note: This configures the RGB LED connected to Pin 26.

5.7 Set RGB LED Color Based on Message

Next, define the Python logic that changes the RGB LED color depending on the message content.

</>python:

```
if "open" in t or "unlocked" in t:

rgb_green()

elif "closed" in t or "locked" in t:

rgb_red()
```

In this use case:

- **Green** means open or unlocked
- **Red** means closed or locked

This allows the user to immediately understand the system state without reading the full message.

This topic is used by the backend to send security-related messages to the M5Stack.

Examples of incoming MQTT messages:

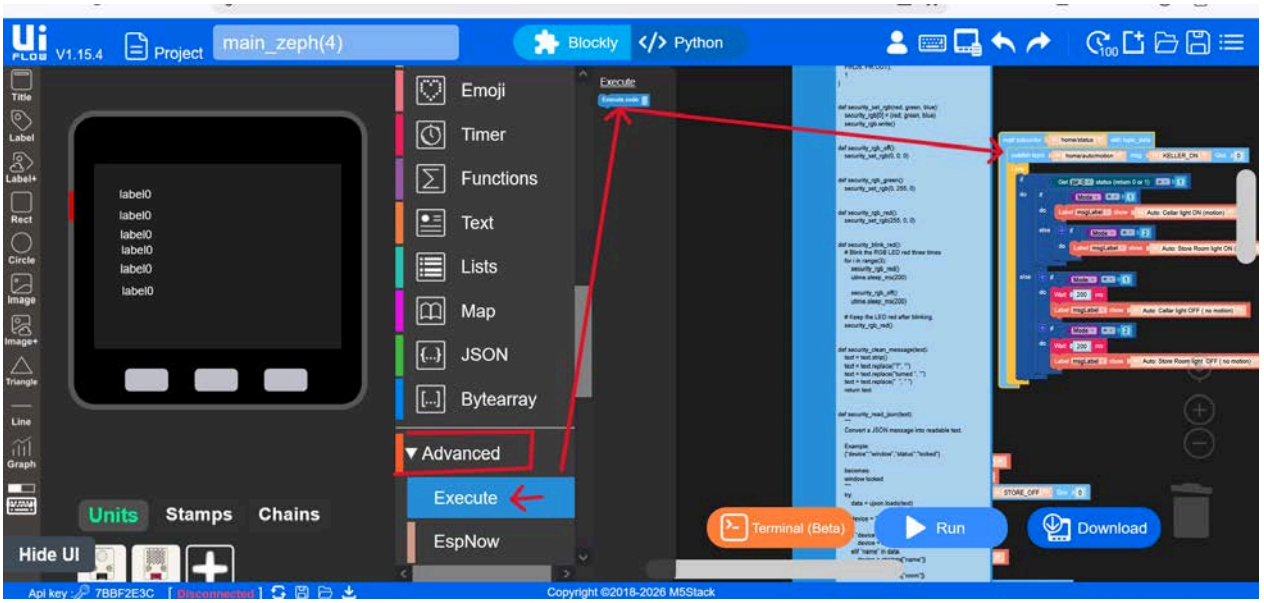
- door open
- door closed
- window open
- window closed
- lock unlocked
- lock locked

5.8 Add the Message-Checking 2nd Execute Code Block

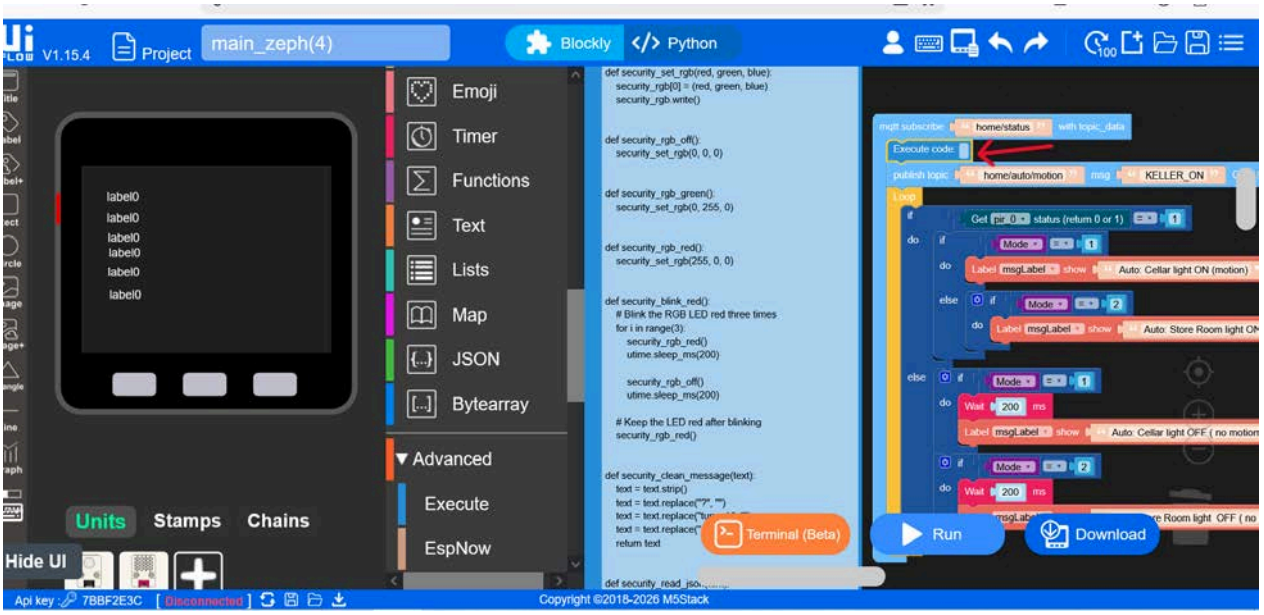
The first block creates the MQTT connection, but the M5Stack must also check continuously for new messages.

Step 1

Click **Advanced**, open **Execute**, and drag another **Execute Code** block into the beginning of the mqtt subscribe block..

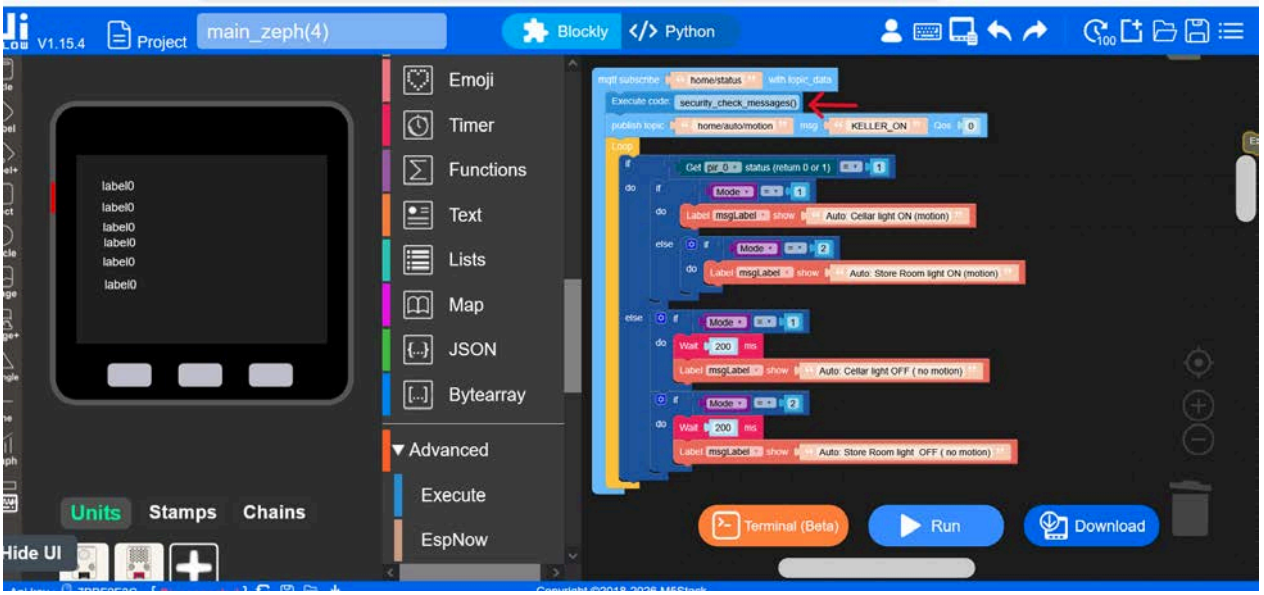


⬇⬇



Step 2

Insert the following command: *security_check_messages() into the code block.*



Note : This checks whether the frontend has published a new message to `home/status`.Without this command, the M5Stack may connect and subscribe successfully, but it will not continuously process new status updates.

The function also reconnects automatically if the MQTT connection is interrupted.

5.9 Run and test the Program

After completing the configuration, Click the **Run** button in UIFlow.

- the RGB LED changes color according to the current state
- the frontend dashboard reflects the same status as the M5Stack
- the system correctly distinguishes between:
    - open / unlocked → green RGB
    - closed / locked → red RGB

The following screenshots show the successful communication between the M5Stack device, RGB LED and the frontend dashboard .

If you enter living room and slide the close window on the mobile App via the link :https://smarthome-app-865f2.web.app the m5stack receives the message and displays the status on the m5stackscreen and the rgb led indicates Red.

On the other hand If you slide the open window on the mobile App the m5stack receives the message and displays the status on the stackscreen and the rgb led indicates green.

This test confirms that the system correctly receives, processes, and displays security-related information in real time.

6. Deployment and Usage Instructions

This section describes how the system is deployed. The system is composed of a frontend application, a backend service and an MQTT broker for communication that is fully cloud-based.

For this project, the following platforms were used:

- Frontend (Flutter – Firebase Hosting)
- Backend(Spring Boot) were deployed using *Railway*

- The MQTT broker used is *HiveMQ (public broker)*

This setup allows the system to be accessed remotely via a mobile web browser without requiring local installation of the frontend or backend.