Hackaton DCIS 2023

Quickstart guide

September 2023

Contents

1 Overview

This document describes key aspects of the competition regarding hardware and software requirements and guidance, so that participants can set up their machines. Although this document does not reveal more specific aspects of the competition (e.g. the use case of the application, or how participants will be scored), we hope that it will serve to provide an overview of the challenge and help to think about winning strategies.

The general infrastructure scheme for the DCIS 2023 Hackaton challenge is shown in Fig. 1. The organization will set up a server running the ThingsBoard IoT platform [1], which allows participants to create modern dashboards in the browser to display MQTT telemetry data sent using their Python SDK [2]. Using a simple script, participants can manage to read Bluetooth data and send it to the ThingsBoard server using the SDK.

During the competition, the majority of the effort will be probably invested in programming and debugging the MCU part. Participants have to use a STM32 NUCLEO-64 development board provided by the origanization, together with some sensors, actuators and displays. Moreover, in order to work on the development of the proposed system efficiently, **it is highly recommended to use at least two laptops per team**. This will allow to develop the C++ code for the development board (laptop 1) while verifying the Bluetooth and MQTT communication (laptop 2). On the one hand, for laptop 1, it is strongly recommended to use Windows OS[1] and Arduino IDE 1.8+. On the other hand, laptop 2 must have Bluetooth connectivity and participants might use any OS with Python 3 installed.

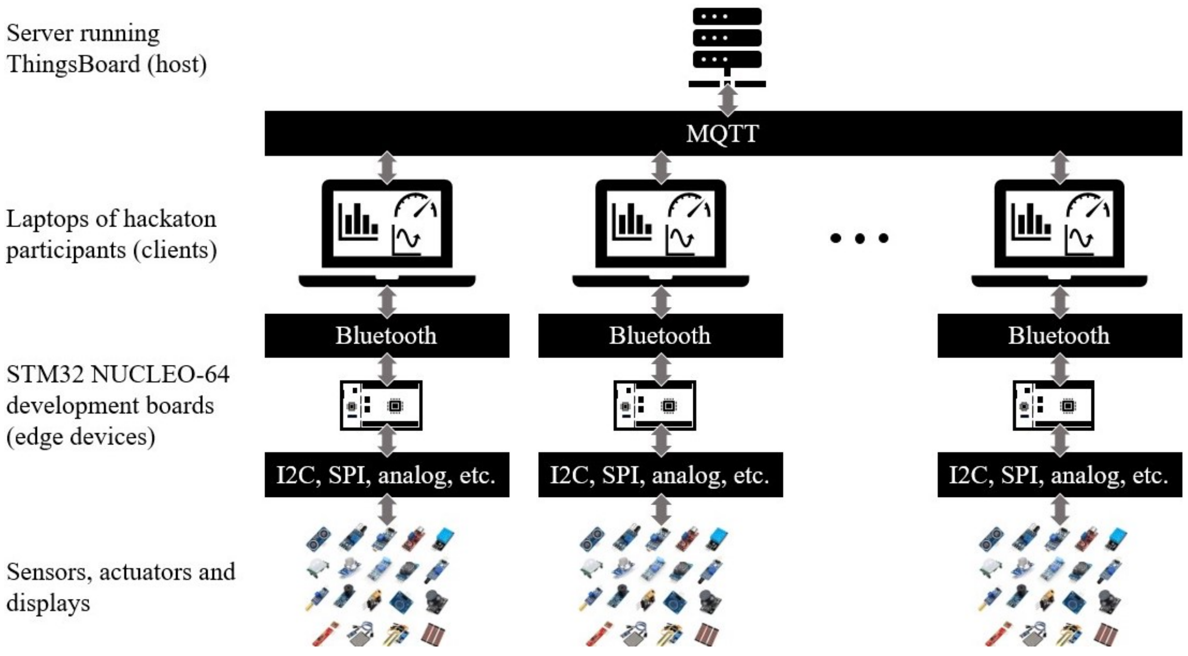

Server running ThingsBoard (host)
MQTT
Laptops of hackaton participants (clients)
Bluetooth
Bluetooth
Bluetooth
STM32 NUCLEO-64 development boards (edge devices)
I2C, SPI, analog, etc.
I2C, SPI, analog, etc.
I2C, SPI, analog, etc.
Sensors, actuators and displays

Figure 1: Diagram indicating the communication between each IoT node and the server.

2 Setting up a minimal working example

A minimal working example is provided in `https://github.com/amc122/dcis2023-hackaton-demo/`, so that participants can build on top of it. Please, follow the steps below to get the example up and running. We encourage participants to try steps 1 to 8 on their own, before the competition, so that the organization can give feedback to solve any potential issue.

[1]Plase contact the organizers if you need or want to use another OS for laptop 1.

1. Connect the HC-05 Bluetooth module and potentiometer to the STM32 board as shown in Fig. 2.

2. Setup your Arduino IDE to work with STM32 boards. See Section 3.

3. Clone the repository containing a minimal working example:

   ```
   git clone https://github.com/amc122/dcis2023-hackaton-demo/
   ```

4. Open `arduino/quickstart/quickstart.ino` with the Arduino IDE and flash the STM32 board.

5. Now the Bluetooth device should be visible, follow the steps in Section 4 to discover Bluetooth devices and pair it to your laptop.

6. Install Python requirements:

   ```
   pip install -r requirements.txt
   ```

7. Modify `SERIAL_PORT` with the one for the HC-05 module in the script `scripts/read_serial.py`. Then run the script to verify potentiometer data is being received:

   ```
   python scripts/read_serial.py
   ```

8. Once Bluetooth communication has been checked, press `Ctrl+C` to stop it.

9. Modify `SERIAL_PORT`, `HOST` and `DEVICE_TOKEN` in `scripts/serial2mqtt.py`. `SERIAL_PORT` is the same in previous step, `HOST` is the server IP or `DNS`, and `DEVICE_TOKEN` is the device access token. Section 5.1 describes the steps to get this access token, which allows participants to transmit data through a ThingsBoard's *Device* entity. Once these constants are set, run the script to start sending MQTT telemetry[2]:

   ```
   python scripts/serial2mqtt.py
   ```

10. Use the ThingsBoard WEB UI to create a dashboard and visualize potentiometer data in real time, as described in Section 5.3. Fig. 3 shows an example *Timeseries Line Chart* widget displaying potentiometer data in real time.

3 Using the development board

3.1 Setting up the Arduino IDE

By default, the Arduino IDE does not provide support for STM32 boards. However, `STM32duino` provides the required dependencies for a fairly large list of STM32 boards [3], including the NUCLEO-64 F401RE. In order to add support for these boards, please, follow the steps in Fig. 4. When adding STM32 boards manager URL (step 4 in Fig. 4a), copy and paste this URL: `https://raw.githubusercontent.com/stm32duino/BoardManagerFiles/main/package_stmicroelectronics_index.json`.

3.2 Programming the STM32 board

In order to flash the board, select the right board and port. On the one hand, figures 5a and 5b show how to select the right *Board* and *Board part number*. Since the target board is a NUCLEO-64 F401RE, select *Nucleo-64* for *Board*, and *Nucleo-64 F401RE* for *Board part number*. On the other hand, Fig. 5c shows how to select the *Port*. In Windows, you will probably have two options, and one of them is *COM1*. If this is the case[3], select the other option, e.g. COM3 in Fig. 5c. Then, participants should be able verify/upload by clicking on the corresponding button in the Arduino IDE as usual.

[2]If this script is not working for you, then try `scripts/serial2mqtt.py` first to verify the MQTT client is is working.

[3]If there are more than two options available, either try the options different from *COM1*, or use Windows *Control Panel* to find it, i.e. go to *Control Panel > All Control Panel Items > Devices and Printers*, then double click *STM32 STLink* and find the COM port under the *Hardware* tab.

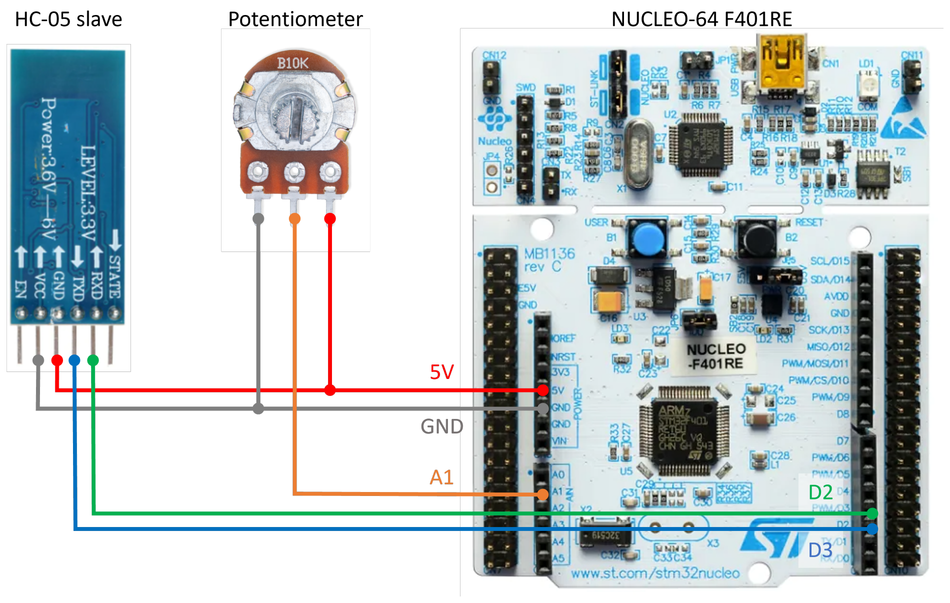


Figure 2: Board and HC-05 module connections for the minimal working example.

3.3 Tips

3.3.1 Set up an additional serial port for Bluetooth communication

It is highly recommended to not use the main hardware serial port (pins RX/D0 and TX/D1) with the Bluetooth module to avoid unplugging and plugging the module everytime the board needs to be programmed and lose the ability to debug using the serial monitor. The following C++ code illustrates how to use the `SoftwareSerial` library to accomplish that.

```cpp
#include <SoftwareSerial.h>

#define USB_BAUDRATE 9600
#define BT_BAUDRATE 9600
#define BT_RX D3 // software serial Tx
#define BT_TX D2 // software serial Rx

SoftwareSerial btSerial(BT_TX, BT_RX);

void setup(){
  Serial.begin(USB_BAUDRATE);
  btSerial.begin(BT_BAUDRATE);
}

// Now use btSerial just like the default Serial object,
// here is an example:
void loop(){

  // Recieve bluetooth data if available,
  // then write this data to the default serial port
  if (btSerial.available())
    Serial.write(btSerial.read());

  // Receive data from default serial port if available,
  // then write this data to the bluetooth serial port
  if (Serial.available())
    btSerial.write(Serial.read());

}
```

Notice the `Serial` object corresponds to the default (hardware) serial port, and `btSerial` to the

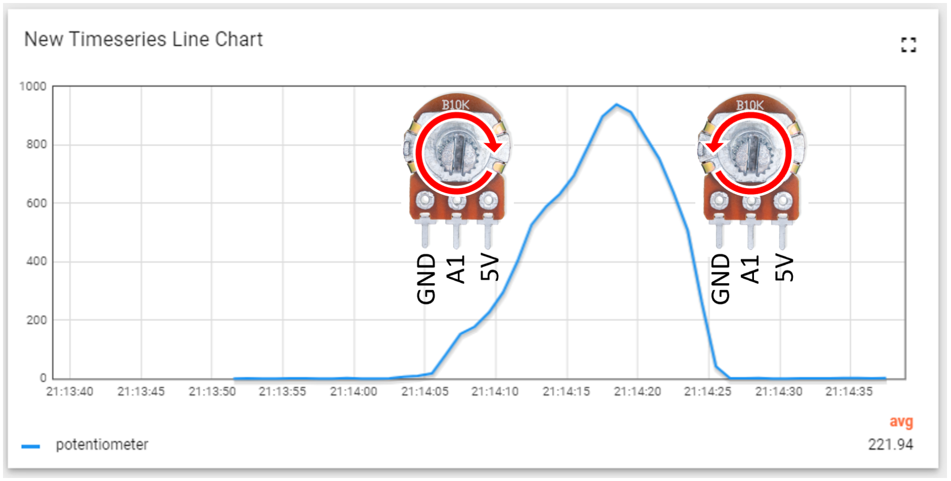


Figure 3: Example chart widget displaying potentiometer data.

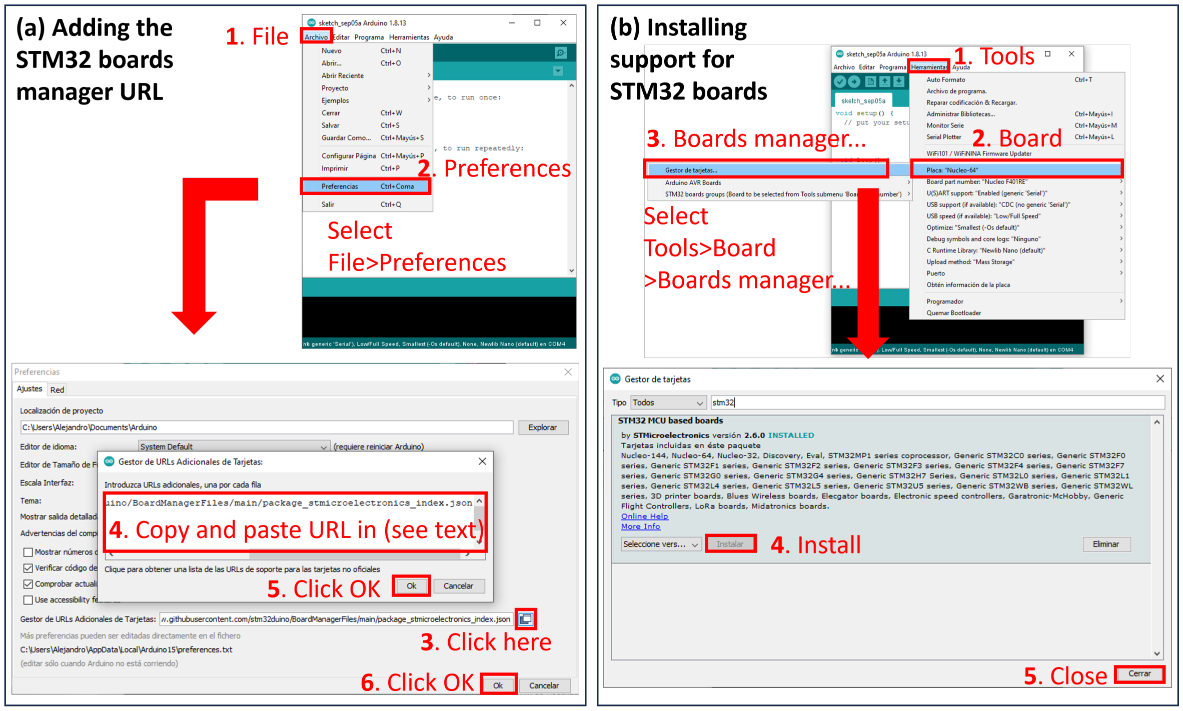
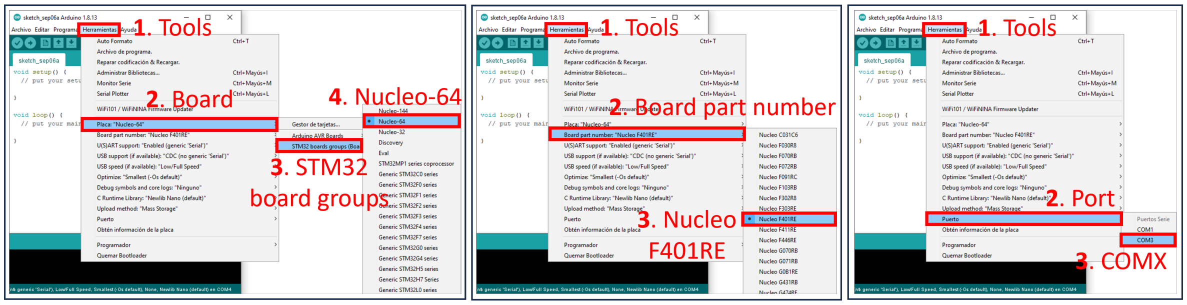


Figure 4: Steps to add the STM32 boards manager URL (a) and install support for STM32 boards (b) using Arduino IDE. When adding STM32 boards manager URL (step 4), copy and paste this URL: `https://raw.githubusercontent.com/stm32duino/BoardManagerFiles/main/package_stmicroelectronics_index.json`.

software defined serial port connected to the Bluetooth module.

3.3.2 Send Bluetooth data in JSON format

Sensor data captured by the development board should be sent to the laptop via Bluetooth, and from the laptop to the server via MQTT, as depicted in Fig. 1. Since the MQTT client requires telemetry to be in JSON format, e.g. `{"temperature": 42.1, "humidity": 70}`, the easiest approach is to



(a) Choosing the *Board* **(b) Choosing the *Board part number*** **(c) Choosing the *Port***

Figure 5: Steps to choose the right *Board*, *Board part number* and *Port* for STM32 boards.

send sensor data through Bluetooth directly in this format to avoid additional manipulations on the client side. Here is a C++ example using a serial port called `btSerial`:

```
[...]
btSerial.write("{\"temperature\":");
btSerial.print(dhtTemperature);
btSerial.write(",\"humidity\":");
btSerial.print(dhtHumidity);
btSerial.write("}\r\n");
[...]
```

4 Setting up the Bluetooth communication

4.1 Disconvering Bluetooth devices

In order to start the Bluetooth communication, we need to pair the laptop and the Bluetooth module. Depending on the operative system, this procedure might change, so we provide guidance for Windows and MAC.
NOTE: The Bluetooth device name, password and baud rate will be provided by the organization together with the HC-05 Bluetooth module.

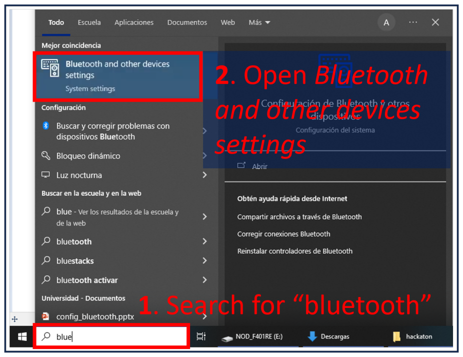
4.1.1 Windows

For Windows users, the process is illustrated in Fig. 6. First, open the Bluetooth settings (Fig. 6a). Then, make sure the Bluetooth is activated and add a new Bluetooth device (Fig. 6b and 6c). Finally, select your HC-05 device, enter the provided password, and pair the device (6d). Moreover, in order to establish communication through the serial port, participants need to know the COM port ID, e.g. COM3 or COM4. So, use the Windows *Control Panel* to navigate to *Control Panel > All Control Panel Items > Devices and Printers*, then double click *HC-05* and find the COM port under the *Hardware* tab.

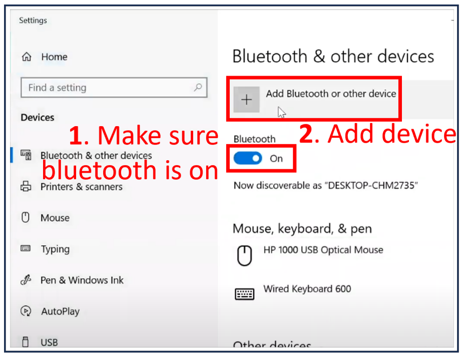
4.1.2 MAC

For MAC users, the process is illustrated in Fig. 7. First, press `cmd+space`, type *bluetooth*, and select the first option (step 1). Then, click on *Activate Bluetooth* (step 2), search and select your Bluetooth HC-05 module by name. Once the right module is selected, click on *connect* (step 3) and use the provided password (step 4). Finally, in order to use the use the serial port, the corresponding path is needed, so discover it by typing `ls /dev/tty.*` (step 5)[4].

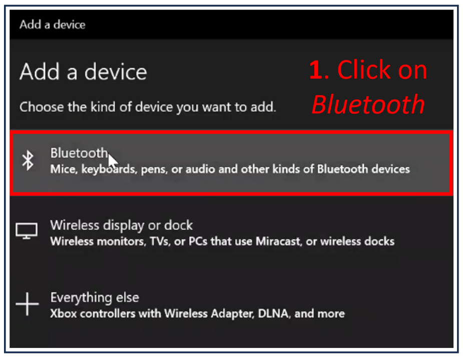
[4]If the device is not found, try `ls /dev/cu.*` instead.


2. Open Bluetooth and other devices settings
Bluetooth and other devices settings
System settings
1. Search for "bluetooth"
blue

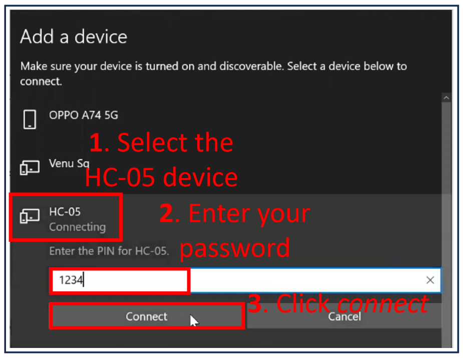
(a) Open Bluetooth settings


Bluetooth & other devices
Add Bluetooth or other device
2. Add device
1. Make sure bluetooth is on
Bluetooth
On
Now discoverable as "DESKTOP-CHM2735"

(b) Add a new bluetooth device

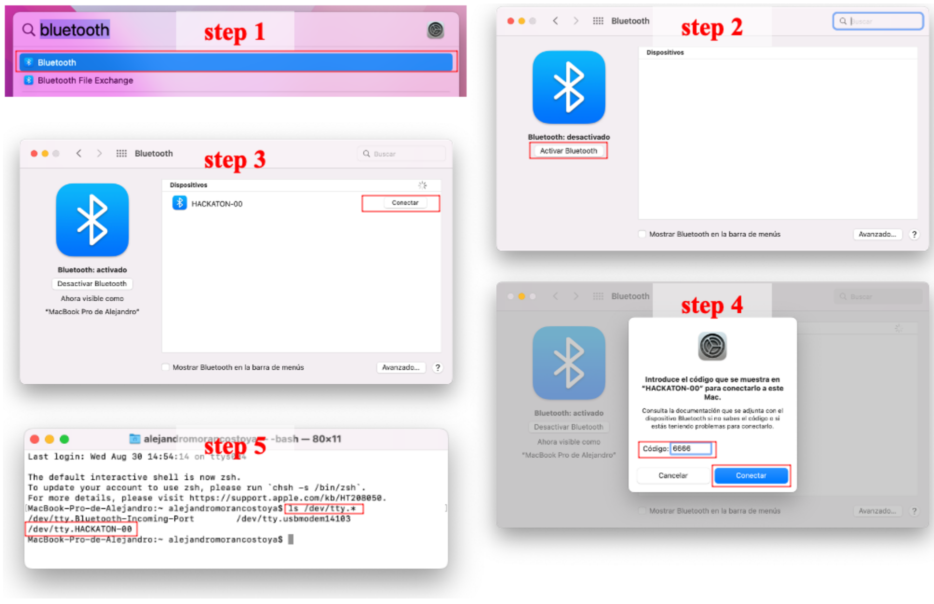

Add a device
Choose the kind of device you want to add.
1. Click on Bluetooth
Bluetooth
Mice, keyboards, pens, or audio and other kinds of Bluetooth devices
Wireless display or dock
Wireless monitors, TVs, or PCs that use Miracast, or wireless docks
Everything else
Xbox controllers with Wireless Adapter, DLNA, and more

(c) Select *Bluetooth*



(d) Pair the HC-05

Figure 6: Steps to pair blueooth device on Windows.



```
Last login: Wed Aug 30 14:54:14 on ttys004

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
MacBook-Pro-de-Alejandro:~ alejandromorancostoya$ ls /dev/tty.*
/dev/tty.Bluetooth-Incoming-Port        /dev/tty.usbmodem14103
/dev/tty.HACKATON-00
MacBook-Pro-de-Alejandro:~ alejandromorancostoya$
```

Figure 7: Steps to pair Bluetooth device on MAC.

4.2 Bluetooth communication with PySerial

Since participants will be using the ThingsBoard Python SDK to send telemetry through MQTT, it is highly recommended to use Python for Bluetooth communication too, using the PySerial module. This approach will allow participants to get Bluetooth data through a serial port and send it via MQTT to the server in a single script. The following sample script shows how to print Bluetooth serial port data every time a new line is received.

```
import time
import json
import serial

SERIAL_PORT = '<YOUR_SERIAL_PORT>'  # e.g. 'COMX' for Windows, or '/dev/tty.HACKATON-
    XX' for MAC
BAUDRATE = 9600

def time_ms():
    """
    Returns timestamp in ms
    """
    return int(round(time.time()*1000))

def str2telemetry(serial_str):
    """
    Convert serial string (assuming JSON format) to python dictionary
    """
    recv_dict = json.loads(recv_str)
    telemetry = {
        "ts": time_ms(),
        "values": recv_dict
    }
    return telemetry

ser = serial.Serial(
    port=SERIAL_PORT,
    baudrate=BAUDRATE,
    parity=serial.PARITY_ODD,
    stopbits=serial.STOPBITS_TWO,
    bytesize=serial.SEVENBITS
)

ser.isOpen()

while True:
    try:
        recv = ser.readline()
        if recv != '':
            recv_str = str(recv, 'utf-8')
            try:
                telemetry = str2telemetry(recv_str)
                print(telemetry)
            except:
                pass

    except KeyboardInterrupt: # i.e. Ctrl+C
        print('\nKeyboard Interrupt')
        ser.close()
        exit()
```

5 Using the ThingsBoard IoT platform

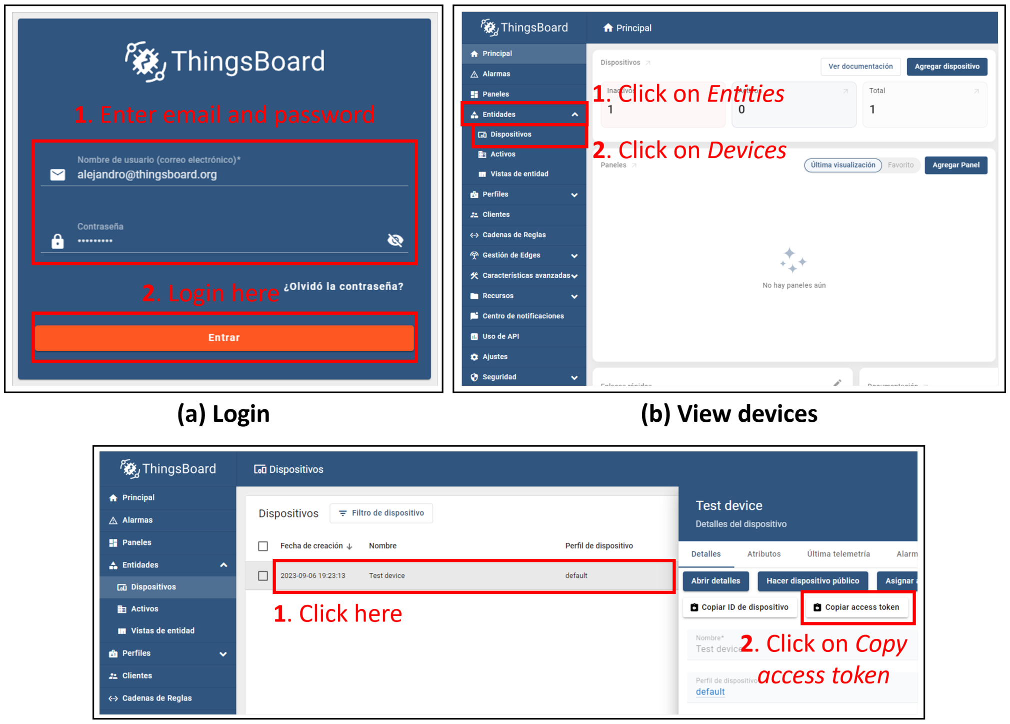
ThingsBoard is an open-source IoT platform for device management, data collection, processing and visualization for IoT solutions [1]. During the Hackaton, the organization will be hosting the service, so that participants will only need to worry about the client side. Those interested in running the application locally might follow the instructions in `https://thingsboard.io/installations/`.

In this section, it is explained how participants will be using the platform in two different ways: using Python scripts to send MQTT telemery, and creating a dashboard to visualize this data. However, the very first step is to sign in to get a device token.

5.1 Logging in and getting a device token

During the competition, there will be a server running the ThingsBoard service with a WEB UI, which participants can use to login using the browser. To get the device token follow the steps in Fig 8:

- Navigate to the ThingsBoard UI URL and login using the provided credentials as shown in Fig. 8a.
- Click on *Entities* > *Devices*, as illustrated in Fig. 8b
- Select the device and copy the device token as shown in Fig. 8c.



(a) Login (b) View devices

(c) Select device and copy the access token

Figure 8: Steps to get the device token.

5.2 Sending telemetry using the Python Client SDK

The device token can be used to send telemetry via MQTT with the ThingsBoard Python Client SDK. Participants can install the SDK using pip:

```
pip install tb-mqtt-client
```

Once installed, the simplest way of sending telemetry is as follows:

```python
from tb_device_mqtt import TBDeviceMqttClient, TBPublishInfo

HOST = '<THINGSBOARD_HOST>' # IP provided by the organization
DEVICE_TOKEN = '<THINGSBOARD_DEVICE_TOKEN>' # sign in in the browswer and copy device
    token

telemetry = {'temperature': 40.0, 'humidity': 90.0}
client = TBDeviceMqttClient(HOST, username=DEVICE_TOKEN)
# Connect to ThingsBoard
client.connect()
```

```
# Sending telemetry without checking the delivery status
# client.send_telemetry(telemetry)
# Sending telemetry and checking the delivery status (QoS = 1 by default)
result = client.send_telemetry(telemetry)
# get is a blocking call that awaits delivery status
success = result.get() == TBPublishInfo.TB_ERR_SUCCESS
# Disconnect from ThingsBoard
client.disconnect()
```

To check the last telemetry values, participants can follow the steps in Fig. 9. If telemetry values are correctly transmitted, everything should be ready to create a simple dashboard (Section 5.3).

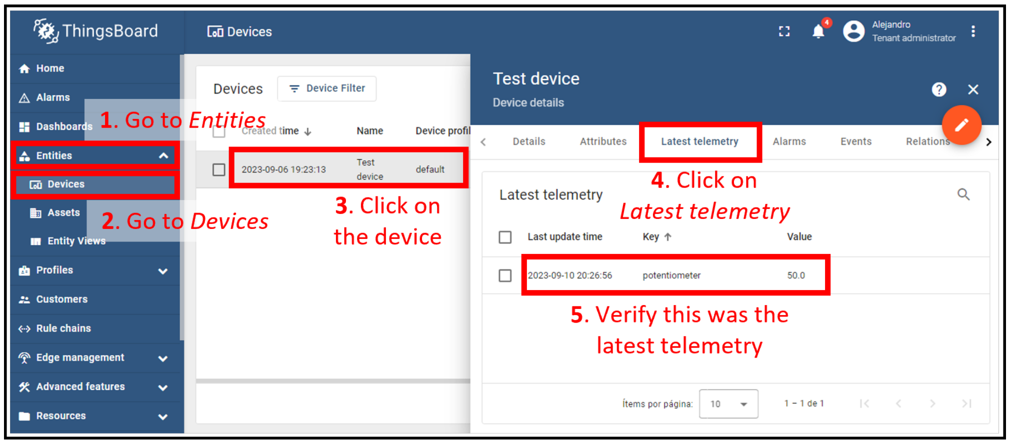
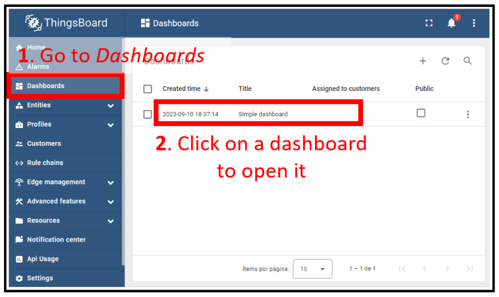


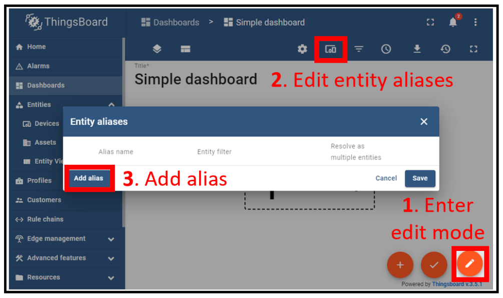
Figure 9: Steps to view and verify the latest telemetry to a *Device* entity in the ThingsBoard WEB UI.

5.3 Creating a simple Dashboard

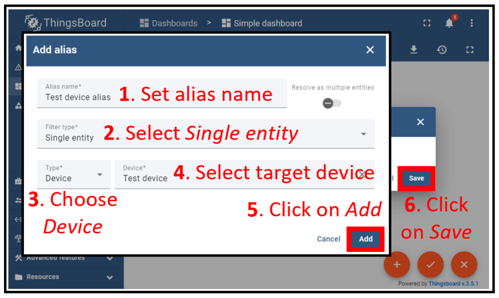
As described in Fig. 10, creating and editing a dashboard is a simple process using the ThingsBoard software. However, before following the steps in Fig. 10f, verify your *Device* entity is receiving MQTT telemetry, otherwise the *Timeseries data keys* options would be empty (step 3 in Fig. 10f). To check the last telemetry values before proceeding to create the dashboard, participants can follow the steps in Fig. 9.



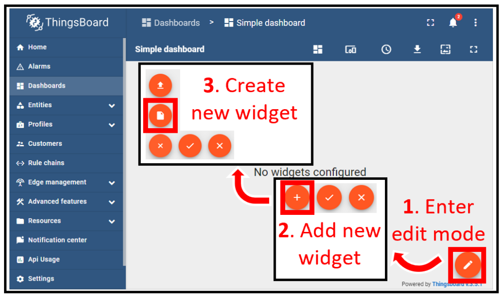
(a) Open a dashboard



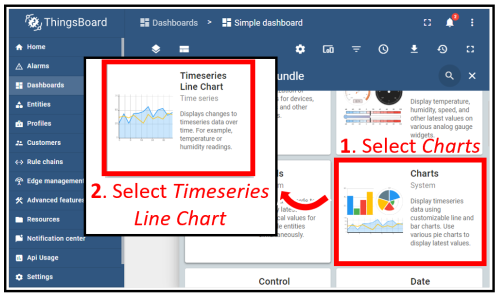
(b) Open the *Add alias* menu



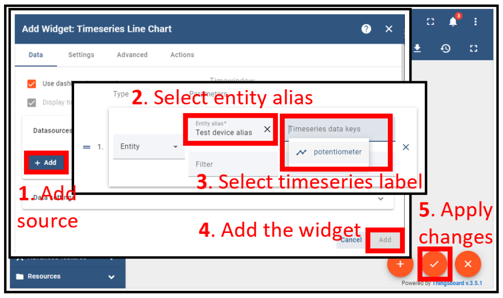
(c) Save alias



(d) Use the ***Create new widget*** option in edit mode



(e) Select e.g. a *Timeseries Line chart*



(f) Add a data source to the widget

Figure 10: Steps to create and edit a simple dashboard, including a chart widget recording telemetry from an existing device as data source.

References

[1] “ThingsBoard,” accessed August 28, 2023. [Online]. Available: https://thingsboard.io/

[2] “ThingsBoard Python SDK,” accessed August 28, 2023. [Online]. Available: https://github.com/thingsboard/thingsboard-python-client-sdk/

[3] “Arduino core support for STM32 based boards,” accessed August 28, 2023. [Online]. Available: https://github.com/stm32duino/Arduino_Core_STM32/