Instituto Superior Técnico

Production Management

Laboratory Guide

Marta Batalha and Ricardo Febra

March 2022

Contents

1 Introduction

To develop your group project, within Production Management course framework, a set of devices (sensors, boards, receivers, etc.) is provided. In order to have a proper functioning of these devices and to be able to gather data and analyse it, some code files to run in Arduino IDE will be necessary. A platform called ThingsBoard will be used to store and visualize the collected data.

During your first visit to Sustainable Production 4.0 Laboratory, you will have the opportunity to see 4 different demonstrations where the devices will be used with different purposes. Each demonstration aims to enlighten how the different devices can be helpful to gather relevant data in production management context.

In this document you will find important information and guidelines that will help you during your project development.

2 Devices

In the Sustainable Production 4.0 Laboratory there is a set of available devices that can be used to develop your projects. In this section of the guide, they are presented with a brief description.

2.1 Light Dependent Resistor (LDR)

A Light Dependent Resistor (LDR) (Figure 1) is an electronic component that is used to detect the presence or the level of light. As the light intensity increases, their resistance decreases (Figure 2). For example, in the dark, at low light levels, the resistance of a LDR is high, meaning there is few current flowing and a high voltage.

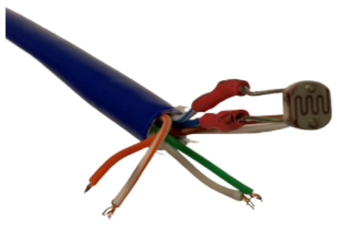


Figure 1: LDR

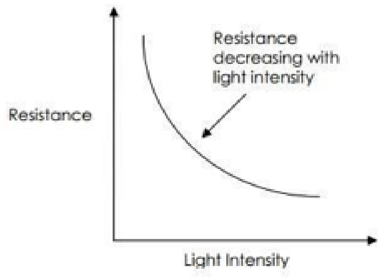


Figure 2: Resistance variation with light intensity

The analogic output of the LDR is a value between 0 and 1023. In the dark the output values will be higher than in brighter environments.

2.2 Buttons

This device (Figure 3) consists on a simple set of 5 buttons (and is also equipped with 3 LEDs). Every time a button is pressed, an output that indicates which of the 5 buttons was pressed is generated. It can be used for a wide variety of purposes, as it will be exemplified in different demonstrations.

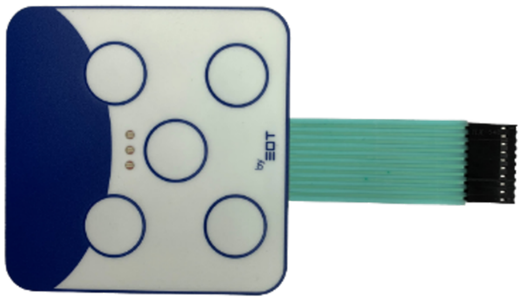


Figure 3: Buttons

2.3 Time of Flight (ToF)

The ToF sensor (Figure 4) allows to get the distance between itself and the closest object. Its working principle is simple (Figure 5). A light (infrared, invisible to human eyes) signal is emitted by the sensor. This signal will hit the first object in front of the sensor and will bounce back to it. Knowing the time it takes for the signal to travel that path, it is possible to know the distance between the sensor and the object.

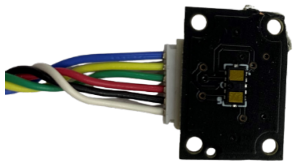


Figure 4: ToF

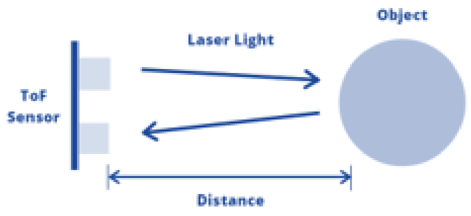


Figure 5: ToF working principle

2.4 Boards

The iStartLab V2.0 boards (Figure 6) were developed by IST iStart-Lab. The communication between the boards and ThingsBoard uses LoRa technology, which requires low power to send data. More information about this can be found in "IoT Technology" provided document. Among several relevant features, it should be noted that the boards are equipped with a buzzer, an USB port and a reset button.

Some of the previously mentioned devices should be connected to the iStartLab V2.0 board so that, after uploading the programs written in Arduino IDE, data from devices can be collected. These devices are: LDRs, buttons and ToF.

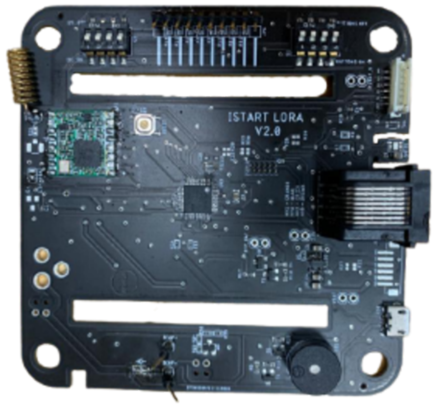


Figure 6: iStartLab V2.0 Board

2.5 Beacon + Receiver

An Arduino ESP32 board works as a receiver (Figure 7). It is placed in an interest point and will detect beacons (Figure 8) signals, making it easy to identify the beacons that were near the interest point. So, if the signal received from beacon A has higher intensity than the signal received from beacon B, you can conclude that beacon A was closer to the interest point.

The beacons use Bluetooth Low Energy (BLE) technology, which requires a considerably lower power consumption than classic Bluetooth. But the receiver must be plugged because it is constantly trying to find beacons and receive their signals, requiring more energy. It communicates via Wi-Fi to send the collected data to ThingsBoard.

A specific program written in Arduino IDE must be uploaded to the receiver board, so data can be collected. For further explanation, see section 3 of this document.

Figure 7: Beacon

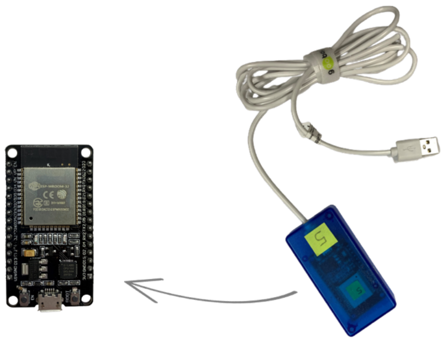


Figure 8: Receiver board (on the left) and receiver inside its box, with cable (on the right)

2.6 Vibration sensor

It is possible to identify a machine state, working or stopped, by detecting its vibration. For that, an Arduino Nano RP2040 Connect (Figure 9) is used, which has an onboard accelerometer. Since vibration in all directions is relevant, accelerometer values in x, y and z axis are considered. Also, to ignore the offset of the acceleration, the variation values are used instead of the acceleration value itself. The obtained output is F, computed according to equation 1, where N represents the number of samples used (N=128).

$$F = \sum_{n=2}^{N} \left[|accX_n - accX_{n-1}| + |accY_n - accY_{n-1}| + |accZ_n - accZ_{n-1}| \right] \quad (1)$$

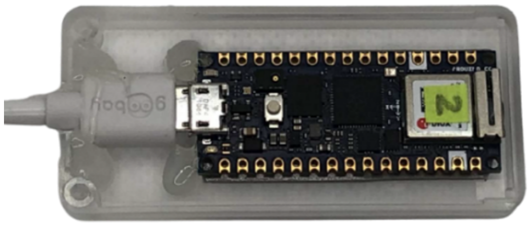


Figure 9: Arduino Nano RP2040 Connect inside its box

A specific program written in Arduino IDE must be uploaded to this board, so data can be collected. For further explanation, see section 3 of this document.

3 Arduino IDE sketches

As explained before, for the devices to properly work, specific sketches must be uploaded to the iStartLab board (in case the devices being used are LDRs, buttons or ToF), vibration sensor board and receiver board.

Table 1 shows the folder where you will find the required sketch to upload when using each device.

Table 1: Organisation of the code folders

Folder name	Upload when using
LDR	LDR
Buttons_Counter	Buttons
ToF	ToF
BLE_receiver	Beacon + Receiver
Vibration_Sensor	Vibration sensor

These sketches will be provided in the beginning of your project development. You are free to adapt and change it according to your project specifications.

4 Tips and Important Guidelines

To develop your group project, different software will be required. Each has its own characteristics and some might need to be changed. In this section you can find tips and important guidelines, with step-by-step explanations.

4.1 General

4.1.1 Wi-Fi

Whenever you are working with any of the devices previously mentioned, it is required to connect to a specific Wi-Fi network:

- **Name**: Production Management
- **Password**: ist_1911

4.1.2 Arduino IDE

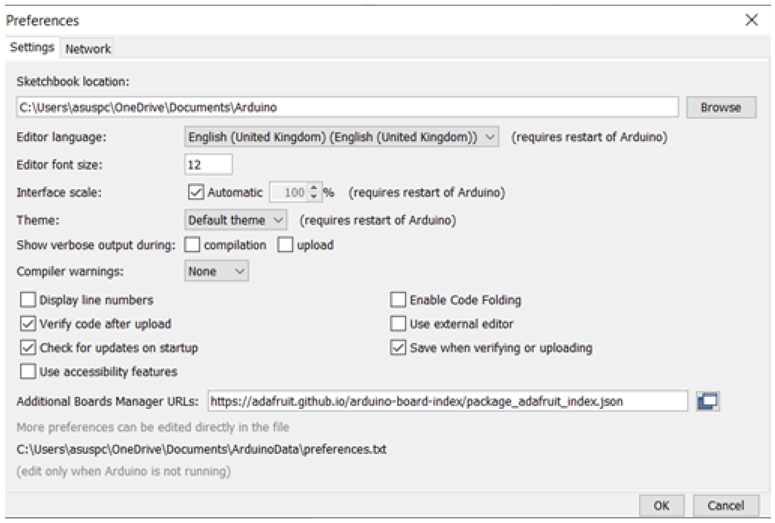
If you don't have it yet, install **Arduino IDE** software. It can be done through this link. Note that you will need to be using **version 1.8 or higher** for this project.

4.2 LDR and Buttons

4.2.1 Choose Board

To be able to upload sketches into iStartLab boards, you must select the board "Adafruit Feather M0", which requires the installation of "Adafruit SAMD Board" in Boards Manager. For that, follow the next steps:

- On the top left **Menu**, select **File** and then **Preferences**.
- In the **Additional Boards Manager URLs** option (Figure 10), write the link: `https://adafruit.github.io/arduino-board-index/package_adafruit_index.json`



Sketchbook location: C:\Users\asuspc\OneDrive\Documents\Arduino

Editor language: English (United Kingdom) (English (United Kingdom)) (requires restart of Arduino)

Editor font size: 12

Interface scale: [✓] Automatic 100 % (requires restart of Arduino)

Theme: Default theme (requires restart of Arduino)

Show verbose output during: [ ] compilation [ ] upload

Compiler warnings: None

[ ] Display line numbers
[✓] Verify code after upload
[✓] Check for updates on startup
[ ] Use accessibility features
[ ] Enable Code Folding
[ ] Use external editor
[✓] Save when verifying or uploading

Additional Boards Manager URLs: https://adafruit.github.io/arduino-board-index/package_adafruit_index.json

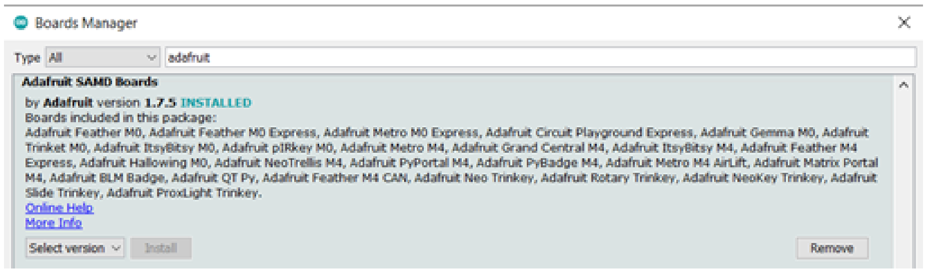
More preferences can be edited directly in the file

C:\Users\asuspc\OneDrive\Documents\ArduinoData\preferences.txt

(edit only when Arduino is not running)

Figure 10: Preferenes tab

- Back to the top left **Menu**, select **Tools**, then **Board** and finally **Boards Manager**.
- Use the search bar to find **"Adafruit SAMD Boards"** and **install** it (Figure 11).


Boards Manager

Type: All | adafruit

Adafruit SAMD Boards
by **Adafruit** version **1.7.5** INSTALLED
Boards included in this package:
Adafruit Feather M0, Adafruit Feather M0 Express, Adafruit Metro M0 Express, Adafruit Circuit Playground Express, Adafruit Gemma M0, Adafruit Trinket M0, Adafruit ItsyBitsy M0, Adafruit pIRkey M0, Adafruit Metro M4, Adafruit Grand Central M4, Adafruit ItsyBitsy M4, Adafruit Feather M4 Express, Adafruit Hallowing M0, Adafruit NeoTrellis M4, Adafruit PyPortal M4, Adafruit PyBadge M4, Adafruit Metro M4 AirLift, Adafruit Matrix Portal M4, Adafruit BLM Badge, Adafruit QT Py, Adafruit Feather M4 CAN, Adafruit Neo Trinkey, Adafruit Rotary Trinkey, Adafruit NeoKey Trinkey, Adafruit Slide Trinkey, Adafruit ProxLight Trinkey.
Online Help
More Info

Select version | Install | Remove

Figure 11: Boards Manager with "Adafruit SAMD Boards" installed

- Having the required package installed, go to **Menu**, **Tools**, **Board** and **select** the board **"Adafruit Feather M0"** (Figure 12).

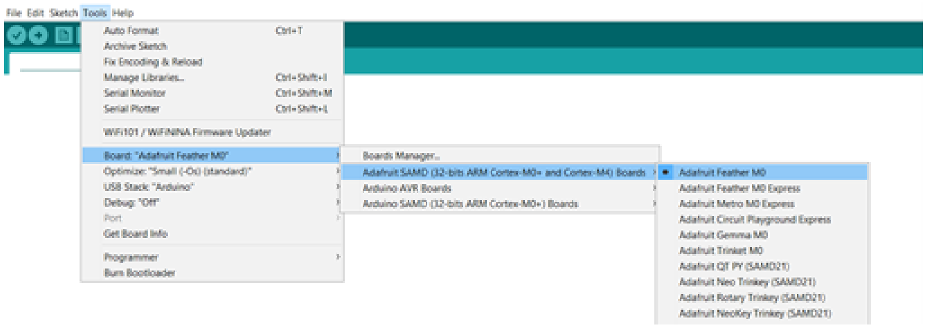
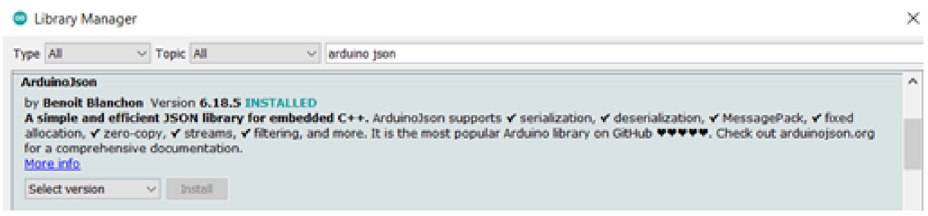

File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Adafruit Feather M0"
Optimize: "Small (-Os) (standard)"
USB Stack: "Arduino"
Debug: "Off"
Port
Get Board Info
Programmer
Burn Bootloader
Boards Manager...
Adafruit SAMD (32-bits ARM Cortex-M0+ and Cortex-M4) Boards
Arduino AVR Boards
Arduino SAMD (32-bits ARM Cortex-M0+) Boards
Adafruit Feather M0
Adafruit Feather M0 Express
Adafruit Metro M0 Express
Adafruit Circuit Playground Express
Adafruit Gemma M0
Adafruit Trinket M0
Adafruit QT PY (SAMD21)
Adafruit Neo Trinkey (SAMD21)
Adafruit Rotary Trinkey (SAMD21)
Adafruit NeoKey Trinkey (SAMD21)

Figure 12: "Adafruit Feather M0" board selected

4.2.2 Install libraries

For the sketch code to be compiled without any error message, some libraries must be installed. That can be done following the next steps:

- On the top left **Menu**, select **Sketch**, then **Include Library** and finally **Manage Libraries**.
- Use the search bar to find **"ArduinoJson"** and **install** it (Figure 13).



arduino json

ArduinoJson

by **Benoit Blanchon** Version **6.18.5** **INSTALLED**
A simple and efficient JSON library for embedded C++. ArduinoJson supports ✔ serialization, ✔ deserialization, ✔ MessagePack, ✔ fixed allocation, ✔ zero-copy, ✔ streams, ✔ filtering, and more. It is the most popular Arduino library on GitHub ♥♥♥♥♥. Check out arduinojson.org for a comprehensive documentation.
More info

Figure 13: “ArduinoJson” library installed

4.3 ToF

4.3.1 Choose Board

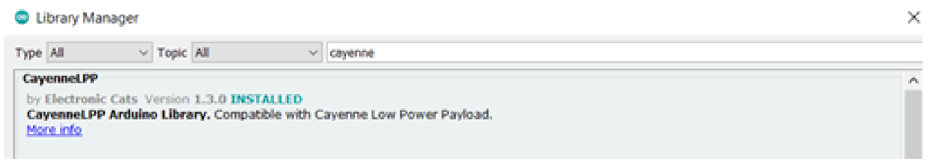
To be able to upload sketches into iStartLab boards, you must select the board **“Adafruit Feather M0”** (Figure 12), which requires the installation of “Adafruit SAMD Board” in Boards Manager. If you haven’t done it yet, follow the steps described in section 4.2.1 of this document.

4.3.2 Install libraries

For the sketch code to be compiled without any error message, some libraries must be installed. That can be done following the next steps:

- On the top left **Menu**, select **Sketch**, then **Include Library** and finally **Manage Libraries**.

- Use the search bar to find **“CayenneLPP.h”** and **“RTCZero.h”** and **install** it (Figures 14 and 15).

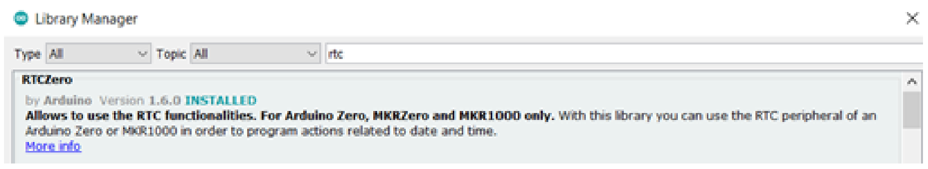


cayenne

CayenneLPP

by Electronic Cats Version 1.3.0 **INSTALLED**
CayenneLPP Arduino Library. Compatible with Cayenne Low Power Payload.
More info

Figure 14: “CayenneLPP” library installed



RTCZero
by Arduino Version 1.6.0 INSTALLED
Allows to use the RTC functionalities. For Arduino Zero, MKRZero and MKR1000 only. With this library you can use the RTC peripheral of an Arduino Zero or MKR1000 in order to program actions related to date and time.
More info

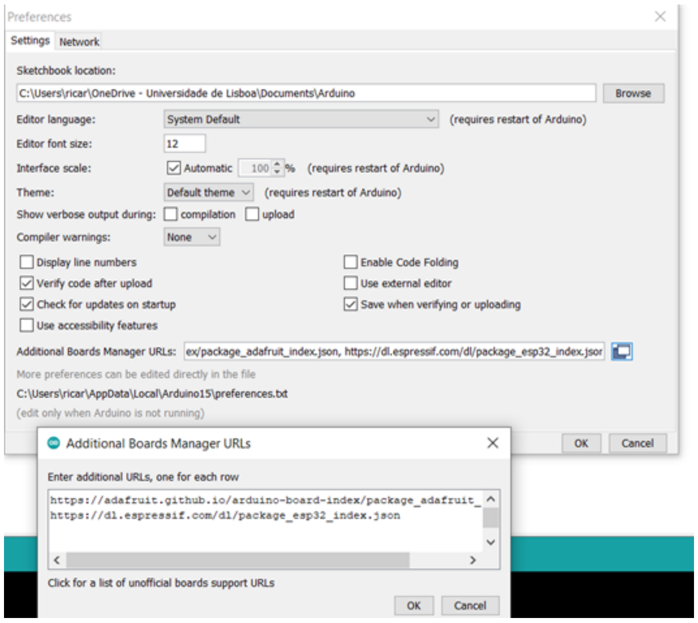
Figure 15: "RTCZero" library installed

4.4 Beacon + Receiver

4.4.1 Choose Board

To be able to upload sketches into receiver boards, you must select the board **"ESP32 Dev Module"**, which requires the installation of **"esp32"** in Boards Manager. The procedure (Figures 16 to 18) is similar to the one described in section 4.2.1, being necessary to do the following changes:

- The link to write in Additional Boards Manager URLs field is:
  `https://dl.espressif.com/dl/package_esp32_index.json`
  Note that, if other links are written in this field, they must be separated by comma.



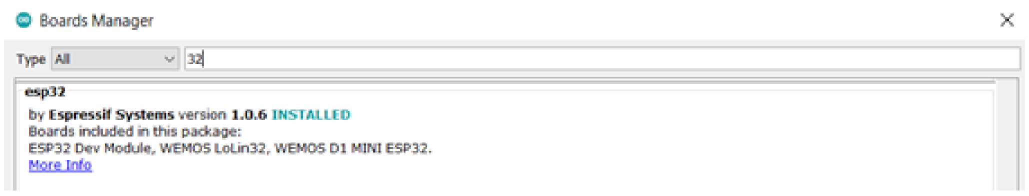
Sketchbook location: C:\Users\ricar\OneDrive - Universidade de Lisboa\Documents\Arduino
Editor language: System Default (requires restart of Arduino)
Editor font size: 12
Interface scale: ☑ Automatic 100 % (requires restart of Arduino)
Theme: Default theme (requires restart of Arduino)
Show verbose output during: ☐ compilation ☐ upload
Compiler warnings: None
☐ Display line numbers
☑ Verify code after upload
☑ Check for updates on startup
☐ Use accessibility features
☐ Enable Code Folding
☐ Use external editor
☑ Save when verifying or uploading
Additional Boards Manager URLs: ex/package_adafruit_index.json, https://dl.espressif.com/dl/package_esp32_index.jsor
More preferences can be edited directly in the file
C:\Users\ricar\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)

Enter additional URLs, one for each row

```
https://adafruit.github.io/arduino-board-index/package_adafruit_
https://dl.espressif.com/dl/package_esp32_index.json
```

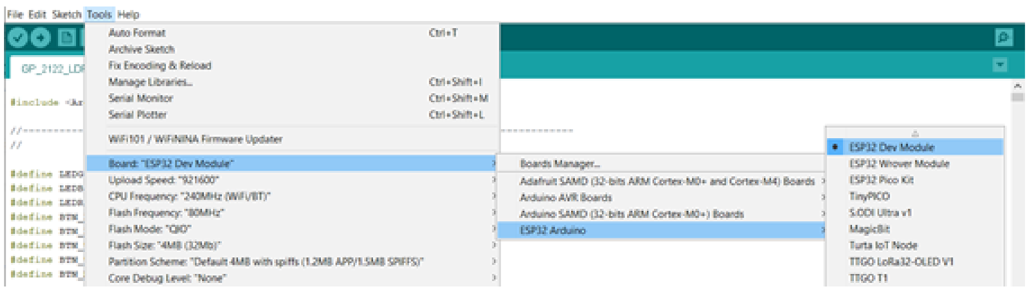
Click for a list of unofficial boards support URLs

Figure 16: Preferences tab



esp32
by **Espressif Systems** version **1.0.6** **INSTALLED**
Boards included in this package:
ESP32 Dev Module, WEMOS LoLin32, WEMOS D1 MINI ESP32.
More Info

Figure 17: Boards Manager with “esp32” installed


File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Boards Manager...
Adafruit SAMD (32-bits ARM Cortex-M0+ and Cortex-M4) Boards
Arduino AVR Boards
Arduino SAMD (32-bits ARM Cortex-M0+) Boards
ESP32 Arduino
ESP32 Dev Module
ESP32 Wrover Module
ESP32 Pico Kit
TinyPICO
S.ODI Ultra v1
MagicBit
Turta IoT Node
TTGO LoRa32-OLED V1
TTGO T1

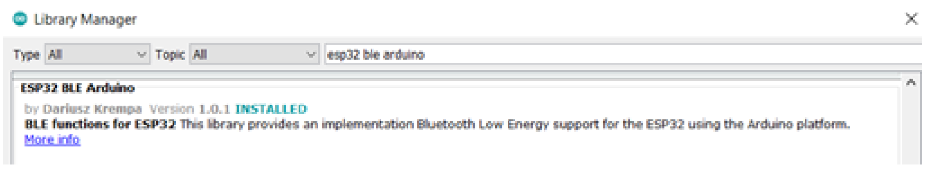
Board: "ESP32 Dev Module"
Upload Speed: "921600"
CPU Frequency: "240MHz (WiFi/BT)"
Flash Frequency: "80MHz"
Flash Mode: "QIO"
Flash Size: "4MB (32Mb)"
Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"
Core Debug Level: "None"

Figure 18: “ESP32 Dev Module” board selected

4.4.2 Install libraries

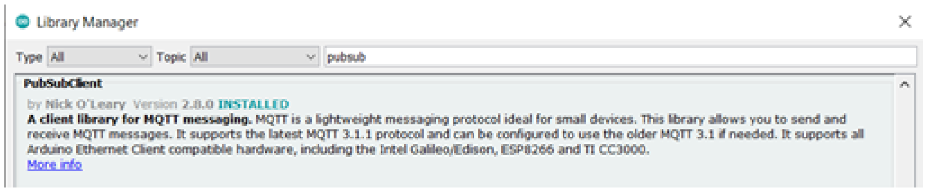
For the sketch code to be compiled without any error message, some libraries must be installed. That can be done following the next steps:

- On the top left **Menu**, select **Sketch**, then **Include Library** and finally **Manage Libraries**.
- Use the search bar to find **“ESP32 BLE Arduino”**, **“PubSub-Client”**, **“ThingsBoard”** and **install** them (Figure 19 to 21).



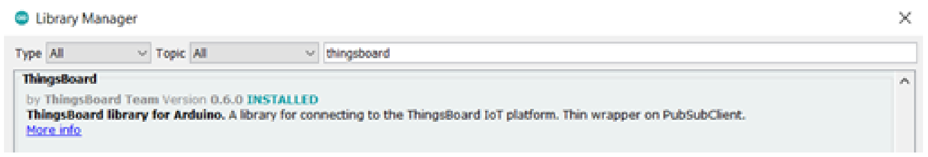
ESP32 BLE Arduino
by Dariusz Krempa Version 1.0.1 **INSTALLED**
BLE functions for ESP32 This library provides an implementation Bluetooth Low Energy support for the ESP32 using the Arduino platform.
More info

Figure 19: “ESP 32 BLE Arduino” library installed



PubSubClient
by Nick O'Leary Version 2.8.0 INSTALLED
A client library for MQTT messaging. MQTT is a lightweight messaging protocol ideal for small devices. This library allows you to send and receive MQTT messages. It supports the latest MQTT 3.1.1 protocol and can be configured to use the older MQTT 3.1 if needed. It supports all Arduino Ethernet Client compatible hardware, including the Intel Galileo/Edison, ESP8266 and TI CC3000.
More info

Figure 20: "PubSubClient" library installed

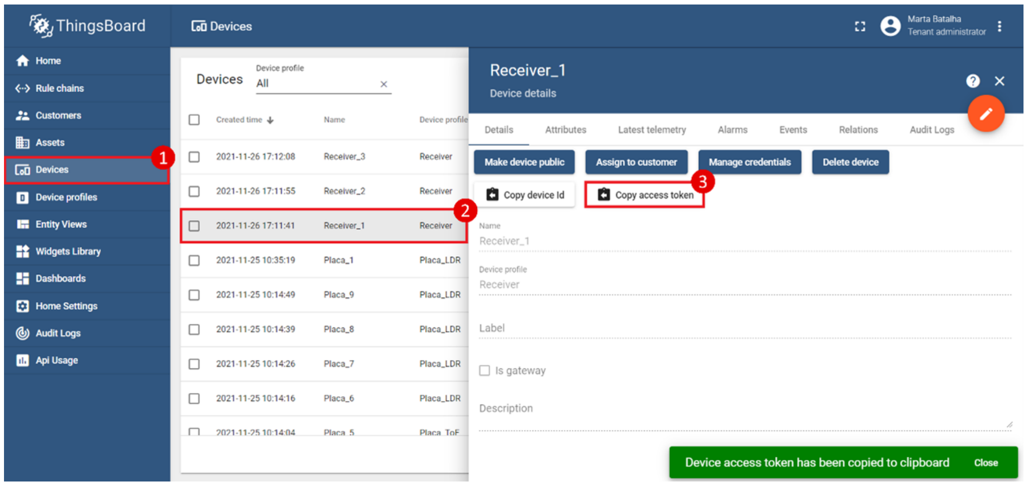


ThingsBoard
by ThingsBoard Team Version 0.6.0 INSTALLED
ThingsBoard library for Arduino. A library for connecting to the ThingsBoard IoT platform. Thin wrapper on PubSubClient.
More info

Figure 21: "ThingsBoard" library installed

4.4.3 Access token

Each receiver has a unique access token that allows to have access and view the collected data in ThingsBoard platform. You can find the access token on ThingsBoard. On the left side menu choose **Devices**, then select the receiver you are working with and click on **Copy access token** (Figure 22).



Devices — Device profile: All

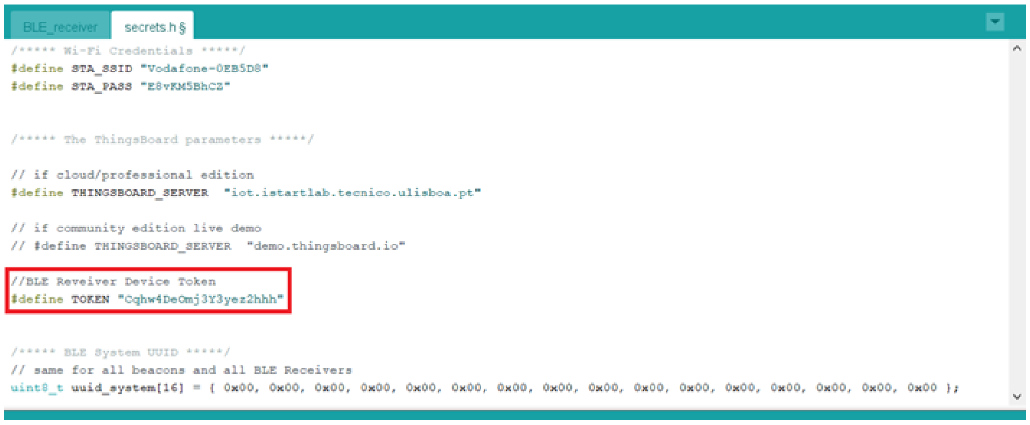
Created time	Name	Device profile
2021-11-26 17:12:08	Receiver_3	Receiver
2021-11-26 17:11:55	Receiver_2	Receiver
2021-11-26 17:11:41	Receiver_1	Receiver
2021-11-25 10:35:19	Placa_1	Placa_LDR
2021-11-25 10:14:49	Placa_9	Placa_LDR
2021-11-25 10:14:39	Placa_8	Placa_LDR
2021-11-25 10:14:26	Placa_7	Placa_LDR
2021-11-25 10:14:16	Placa_6	Placa_LDR
2021-11-25 10:14:04	Placa_5	Placa_ToF

Name: Receiver_1
Device profile: Receiver
Label
Is gateway
Description

Device access token has been copied to clipboard

Figure 22: ThingsBoard interface with steps to get access token

Finally, you need to change the access token in the proper section (Figure 23) of file "secrets.h", which is in the folder "BLE_receiver". The code should be uploaded to the receiver board only after completing this step.



```
/***** Wi-Fi Credentials *****/
#define STA_SSID "Vodafone-0EB5D8"
#define STA_PASS "E8vKM5BhCZ"


/***** The ThingsBoard parameters *****/

// if cloud/professional edition
#define THINGSBOARD_SERVER  "iot.istartlab.tecnico.ulisboa.pt"

// if community edition live demo
// #define THINGSBOARD_SERVER  "demo.thingsboard.io"

//BLE Reveiver Device Token
#define TOKEN "Cqhw4DeOmj3Y3yez2hhh"


/***** BLE System UUID *****/
// same for all beacons and all BLE Receivers
uint8_t uuid_system[16] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };
```

Figure 23: "secrets.h" code for BLE receiver

In this same file it is essential to confirm that in define STA_SSID "name" and define STA_PASS "password" the name and password are those of the Wi-Fi network to which one is connected.

4.4.4 Compiling error troubleshooting

If the error message below shows up when you try to compile the code, you will have to change the Partition Scheme, according to Figure 24.

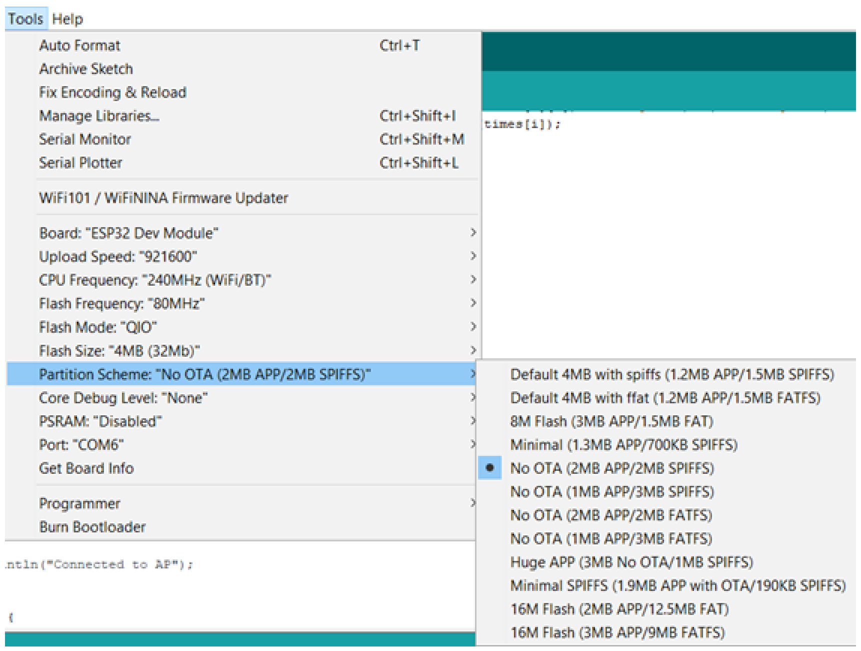
Error Message:

"Sketch uses 1496186 bytes (114%) of program storage space. Maximum is 1310720 bytes.text section exceeds available space in board.
Global variables use 55568 bytes (16%) of dynamic memory, leaving 272112 bytes for local variables. Maximum is 327680 bytes.
Sketch too big; see https://support.arduino.cc/hc/en-us/articles/360013825179 for tips on reducing it.
Error compiling for board ESP32 Dev Module."

4.5 Vibration Sensor

4.5.1 Choose Board

To be able to upload sketches into the vibration sensor, you must select the board **"Arduino Nano RP2040 Connect"**, which requires the instal-



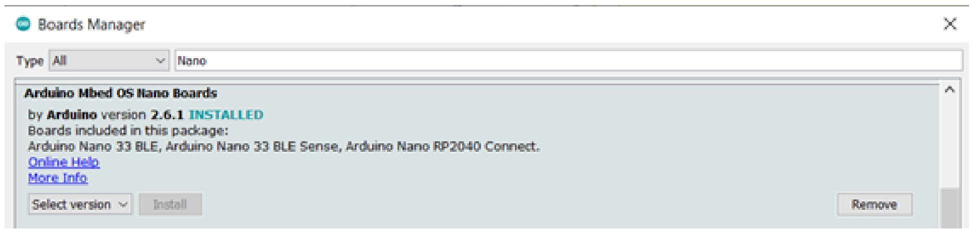
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "ESP32 Dev Module"
Upload Speed: "921600"
CPU Frequency: "240MHz (WiFi/BT)"
Flash Frequency: "80MHz"
Flash Mode: "QIO"
Flash Size: "4MB (32Mb)"
Partition Scheme: "No OTA (2MB APP/2MB SPIFFS)"
Core Debug Level: "None"
PSRAM: "Disabled"
Port: "COM6"
Get Board Info
Programmer
Burn Bootloader

Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)
Default 4MB with ffat (1.2MB APP/1.5MB FATFS)
8M Flash (3MB APP/1.5MB FAT)
Minimal (1.3MB APP/700KB SPIFFS)
● No OTA (2MB APP/2MB SPIFFS)
No OTA (1MB APP/3MB SPIFFS)
No OTA (2MB APP/2MB FATFS)
No OTA (1MB APP/3MB FATFS)
Huge APP (3MB No OTA/1MB SPIFFS)
Minimal SPIFFS (1.9MB APP with OTA/190KB SPIFFS)
16M Flash (2MB APP/12.5MB FAT)
16M Flash (3MB APP/9MB FATFS)

```
times[i]);
ntln("Connected to AP");
```

Figure 24: Partition Scheme selection

lation of **"Arduino Mbed OS Nano Boards"** in Boards Manager. This procedure (Figures 25 and 26) is similar to the one described in section 4.2.1 of this document.



Nano

Arduino Mbed OS Nano Boards
by **Arduino** version **2.6.1** INSTALLED
Boards included in this package:
Arduino Nano 33 BLE, Arduino Nano 33 BLE Sense, Arduino Nano RP2040 Connect.
Online Help
More Info

Figure 25: Boards Manager with "Arduino Mbed OS Nano Boards" installed

4.5.2 Install libraries

For the sketch code to be compiled without any error message, some libraries must be installed. That can be done following the next steps:

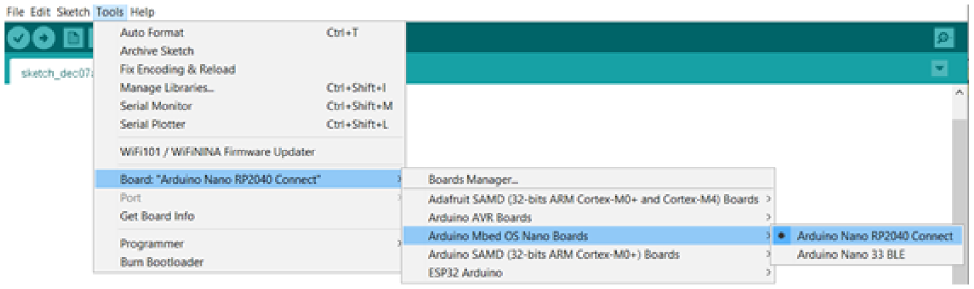
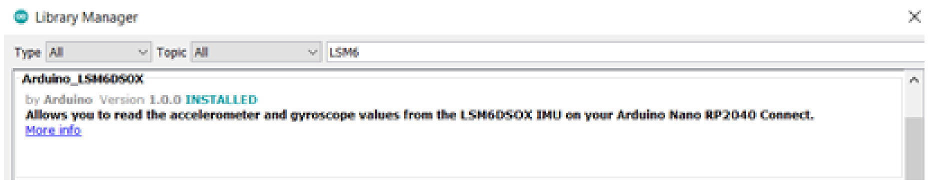

File Edit Sketch Tools Help
sketch_dec07
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Nano RP2040 Connect"
Port
Get Board Info
Programmer
Burn Bootloader
Boards Manager...
Adafruit SAMD (32-bits ARM Cortex-M0+ and Cortex-M4) Boards
Arduino AVR Boards
Arduino Mbed OS Nano Boards
Arduino SAMD (32-bits ARM Cortex-M0+) Boards
ESP32 Arduino
Arduino Nano RP2040 Connect
Arduino Nano 33 BLE

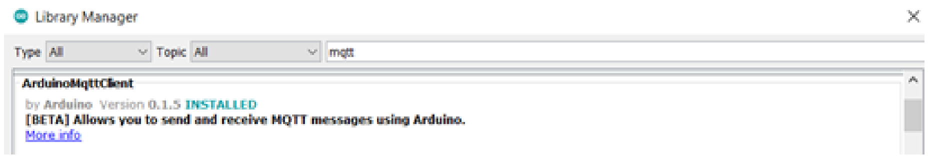
Figure 26: "Arduino Nano RP2040 Connect" board selected

- On the top left **Menu**, select **Sketch**, then **Include Library** and finally **Manage Libraries**.
- Use the search bar to find **"Arduino_LSM6DSOX"**, **"ArduinoMqttClient"** and **"WiFiNINA"** and **install** it (Figures 27 to 29).



Arduino_LSM6DSOX
by **Arduino** Version 1.0.0 INSTALLED
Allows you to read the accelerometer and gyroscope values from the LSM6DSOX IMU on your Arduino Nano RP2040 Connect.
More info

Figure 27: "Arduino_LSM6DSOX" library installed



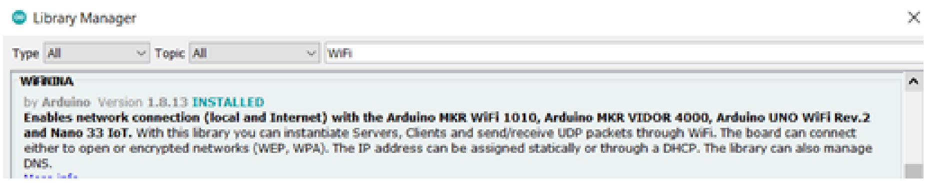
ArduinoMqttClient
by **Arduino** Version 0.1.5 INSTALLED
[BETA] Allows you to send and receive MQTT messages using Arduino.
More info

Figure 28: "ArduinoMqttClient" library installed

4.5.3 Access token

Each vibration sensor has a unique access token that allows to have access and view the collected data in ThingsBoard platform.

You can find the access token on Thingsboard, following a similar procedure to the one explained in section 4.4.3 of this document. On the left

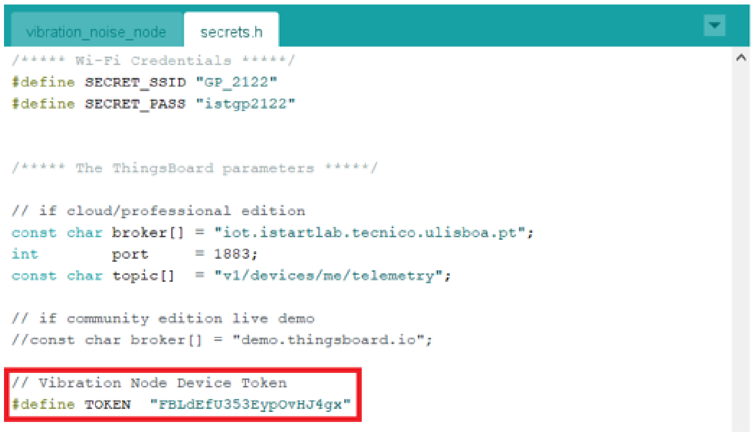


WiFiNINA
by **Arduino** Version **1.8.13** **INSTALLED**
Enables network connection (local and Internet) with the Arduino MKR WiFi 1010, Arduino MKR VIDOR 4000, Arduino UNO WiFi Rev.2 and Nano 33 IoT. With this library you can instantiate Servers, Clients and send/receive UDP packets through WiFi. The board can connect either to open or encrypted networks (WEP, WPA). The IP address can be assigned statically or through a DHCP. The library can also manage DNS.

Figure 29: “WiFiNINA” library installed

side menu choose **Devices**, then select the receiver you are working with and click on **Copy access token**.

Finally, you need to change the access token in the proper section (Figure 30) of file “secrets.h”, which is in the folder “Vibration_Sensor”. The code should be uploaded to the vibration sensor board only after completing this step.



```
/***** Wi-Fi Credentials *****/
#define SECRET_SSID "GP_2122"
#define SECRET_PASS "istgp2122"


/***** The ThingsBoard parameters *****/

// if cloud/professional edition
const char broker[] = "iot.istartlab.tecnico.ulisboa.pt";
int        port     = 1883;
const char topic[]  = "v1/devices/me/telemetry";

// if community edition live demo
//const char broker[] = "demo.thingsboard.io";

// Vibration Node Device Token
#define TOKEN  "FBLdEfU353EypOvHJ4gx"
```

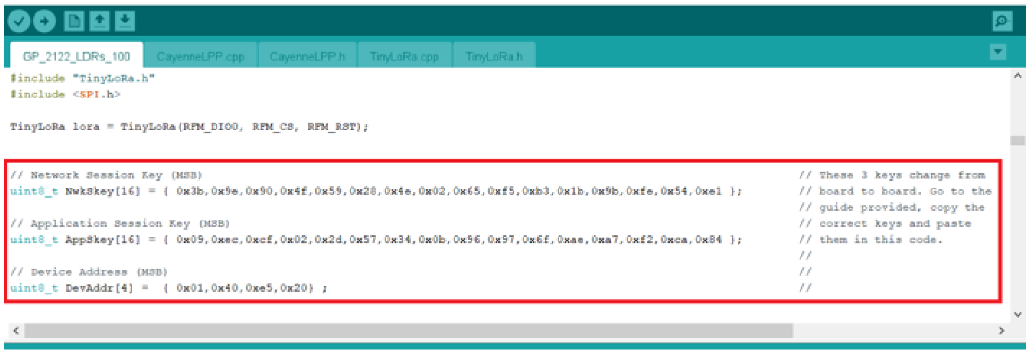
Figure 30: “secrets.h” code for vibration sensor

In this same file it is essential to confirm that in define STA_SSID “name” and define STA_PASS “password” the name and password are those of the Wi-Fi network to which one is connected.

4.6 Changing boards keys

Each iStartLab board has 3 unique keys – network session key (NSK), application session key (ASK) and device address (DA). More than one board can be in use at the same time. However, the platform used to gather data is the same for every board. This means that these keys are essential to identify the board from where a certain message was sent. For this to be possible, it is mandatory to **change the board keys to the ones of the board being used before uploading the sketch to the board**. These keys, in the appropriate format, can be found in Appendix A.

If you are using "LDR", change the keys in the section shown in Figure 31. If you are using "Buttons_Counter", change the keys in the section shown in Figure 32. If you are using "ToF", change the keys in the section shown in Figure 33.

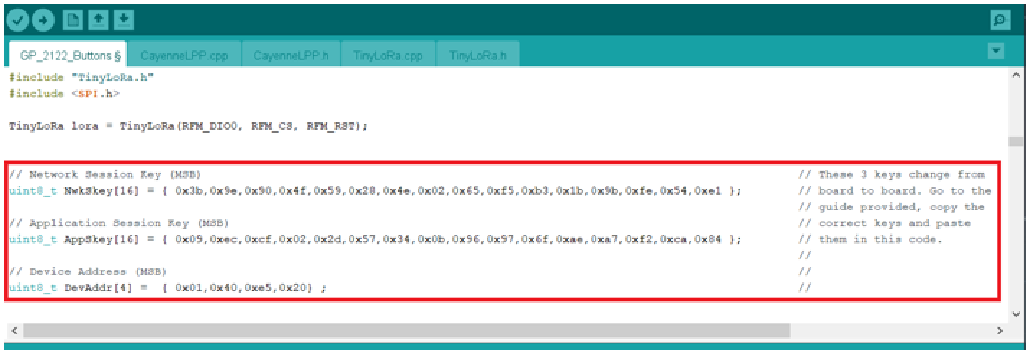


```
#include "TinyLoRa.h"
#include <SPI.h>

TinyLoRa lora = TinyLoRa(RFM_DIO0, RFM_CS, RFM_RST);

// Network Session Key (MSB)                                                                                       // These 3 keys change from
uint8_t NwkSkey[16] = { 0x3b,0x9e,0x90,0x4f,0x59,0x28,0x4e,0x02,0x65,0xf5,0xb3,0x1b,0x9b,0xfe,0x54,0xe1 };        // board to board. Go to the
                                                                                                                  // guide provided, copy the
// Application Session Key (MSB)                                                                                   // correct keys and paste
uint8_t AppSkey[16] = { 0x09,0xec,0xcf,0x02,0x2d,0x57,0x34,0x0b,0x96,0x97,0x6f,0xae,0xa7,0xf2,0xca,0x84 };        // them in this code.
                                                                                                                  //
// Device Address (MSB)                                                                                            //
uint8_t DevAddr[4] =  { 0x01,0x40,0xe5,0x20} ;                                                                     //
```

Figure 31: Keys in LDR code

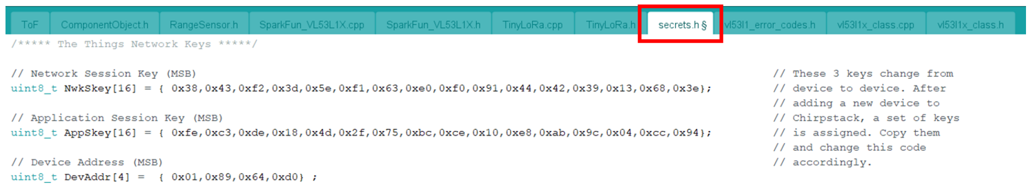


```
#include "TinyLoRa.h"
#include <SPI.h>

TinyLoRa lora = TinyLoRa(RFM_DIO0, RFM_CS, RFM_RST);

// Network Session Key (MSB)                                                                                       // These 3 keys change from
uint8_t NwkSkey[16] = { 0x3b,0x9e,0x90,0x4f,0x59,0x28,0x4e,0x02,0x65,0xf5,0xb3,0x1b,0x9b,0xfe,0x54,0xe1 };        // board to board. Go to the
                                                                                                                  // guide provided, copy the
// Application Session Key (MSB)                                                                                   // correct keys and paste
uint8_t AppSkey[16] = { 0x09,0xec,0xcf,0x02,0x2d,0x57,0x34,0x0b,0x96,0x97,0x6f,0xae,0xa7,0xf2,0xca,0x84 };        // them in this code.
                                                                                                                  //
// Device Address (MSB)                                                                                            //
uint8_t DevAddr[4] =  { 0x01,0x40,0xe5,0x20} ;                                                                     //
```

Figure 32: Keys in buttons code



```
/***** The Things Network Keys *****/

// Network Session Key (MSB)                                                                          // These 3 keys change from
uint8_t NwkSkey[16] = { 0x38,0x43,0xf2,0x3d,0x5e,0xf1,0x63,0xe0,0xf0,0x91,0x44,0x42,0x39,0x13,0x68,0x3e};  // device to device. After
                                                                                                      // adding a new device to
// Application Session Key (MSB)                                                                      // Chirpstack, a set of keys
uint8_t AppSkey[16] = { 0xfe,0xc3,0xde,0x18,0x4d,0x2f,0x75,0xbc,0xce,0x10,0xe8,0xab,0x9c,0x04,0xcc,0x94};  // is assigned. Copy them
                                                                                                      // and change this code
// Device Address (MSB)                                                                               // accordingly.
uint8_t DevAddr[4] =  { 0x01,0x89,0x64,0xd0} ;
```

Figure 33: Keys in ToF code

A Boards keys

Keys for **BOARD 1**

```
// Network Session Key (MSB)
uint8_t NwkSkey[16] =
{ 0x3b,0x9e,0x90,0x4f,0x59,0x28,0x4e,0x02,0x65,0xf5,0xb3,0x1b,0x9b,0xfe,0x54,0xe1};
// Application Session Key (MSB)
uint8_t AppSkey[16] =
{ 0x09,0xec,0xcf,0x02,0x2d,0x57,0x34,0x0b,0x96,0x97,0x6f,0xae,0xa7,0xf2,0xca,0x84};
// Device Address (MSB)
uint8_t DevAddr[4] =
{ 0x01,0x40,0xe5,0x20} ;
```

Keys for **BOARD 2**

```
// Network Session Key (MSB)
uint8_t NwkSkey[16] =
{ 0x6b,0xb4,0x08,0x41,0xe7,0x7c,0x43,0x39,0xf8,0x1d,0xfe,0x36,0x2d,0x10,0xb0,0x43};
// Application Session Key (MSB)
uint8_t AppSkey[16] =
{ 0x89,0x93,0xe2,0x78,0xc3,0xf4,0xb1,0xfc,0xbb,0x16,0xf2,0x63,0x58,0x6c,0x8b,0x35};
// Device Address (MSB)
uint8_t DevAddr[4] =
{ 0x00,0x11,0x31,0xe6} ;
```

Keys for **BOARD 3**

```
// Network Session Key (MSB)
uint8_t NwkSkey[16] =
{ 0x38,0x43,0xf2,0x3d,0x5e,0xf1,0x63,0xe0,0xf0,0x91,0x44,0x42,0x39,0x13,0x68,0x3e};
// Application Session Key (MSB)
uint8_t AppSkey[16] =
{ 0xfe,0xc3,0xde,0x18,0x4d,0x2f,0x75,0xbc,0xce,0x10,0xe8,0xab,0x9c,0x04,0xcc,0x94};
// Device Address (MSB)
uint8_t DevAddr[4] =
{ 0x01,0x89,0x64,0xd0} ;
```

Keys for **BOARD 4**

```
// Network Session Key (MSB)
uint8_t NwkSkey[16] =
{ 0xd1,0xce,0x66,0x65,0x33,0x44,0x74,0x25,0x09,0x34,0x43,0x24,0x58,0x15,0x03,0x06};
// Application Session Key (MSB)
uint8_t AppSkey[16] =
{ 0xfb,0x0b,0x69,0xc5,0x1d,0xea,0x61,0x6c,0x18,0x44,0xfe,0x00,0x0b,0x2c,0xa4,0xde};
// Device Address (MSB)
uint8_t DevAddr[4] =
{ 0x01,0x1e,0x70,0x8d} ;
```

Keys for **BOARD 5**

```
// Network Session Key (MSB)
uint8_t NwkSkey[16] =
{ 0x21,0xc9,0xbe,0x10,0x89,0xa5,0x1f,0x46,0xdc,0x31,0x98,0x74,0x65,0x9f,0x7f,0x95};
// Application Session Key (MSB)
uint8_t AppSkey[16] =
{ 0xdb,0xb7,0x2b,0x42,0xe0,0xe2,0x3b,0xcb,0xb6,0xbf,0x80,0x52,0x52,0x65,0xbe,0x2e};
// Device Address (MSB)
uint8_t DevAddr[4] =
{ 0x00,0xdb,0xc8,0x81} ;
```

Keys for **BOARD 6**

```
// Network Session Key (MSB)
uint8_t NwkSkey[16] =
{ 0xfb,0x71,0x16,0x1c,0xde,0x4a,0x43,0x23,0x63,0x17,0xee,0xf2,0xe7,0xa1,0xf6,0x90};
// Application Session Key (MSB)
uint8_t AppSkey[16] =
{ 0x31,0x71,0x3e,0xfc,0xcc,0x75,0xfb,0xec,0xf7,0x63,0xb8,0x1c,0xbb,0xdf,0x56,0xe0};
// Device Address (MSB)
uint8_t DevAddr[4] =
{ 0x01,0x84,0x60,0xea} ;
```

Keys for **BOARD 7**

```
// Network Session Key (MSB)
uint8_t NwkSkey[16] =
{ 0x3f,0x30,0x64,0x02,0x62,0xe5,0x42,0xb3,0xef,0xe7,0xb4,0x4b,0x7a,0x28,0xd9,0x15};
// Application Session Key (MSB)
uint8_t AppSkey[16] =
{ 0x73,0x72,0x85,0x6d,0x85,0x42,0xad,0x5c,0x01,0x08,0xaa,0xda,0x82,0xf6,0xf5,0xac};
// Device Address (MSB)
uint8_t DevAddr[4] =
{ 0x01,0x7f,0xfb,0x22} ;
```

Keys for **BOARD 8**

```
// Network Session Key (MSB)
uint8_t NwkSkey[16] =
{ 0xc6,0x90,0xbe,0x33,0x65,0xaa,0xe7,0xe8,0x69,0x4b,0xca,0x29,0xc2,0xaa,0xfc,0xa2};
// Application Session Key (MSB)
uint8_t AppSkey[16] =
{ 0xbc,0x6a,0x65,0xf2,0xaa,0xa3,0xf9,0xa7,0x0c,0x63,0x1c,0xf4,0x4d,0xf6,0x0c,0x38};
// Device Address (MSB)
uint8_t DevAddr[4] =
{ 0x01,0x6d,0x0c,0x4d} ;
```

Keys for **BOARD 9**

```
// Network Session Key (MSB)
uint8_t NwkSkey[16] =
{ 0xd5,0x97,0xaa,0xd5,0x30,0x4e,0x31,0xa6,0xa1,0x5f,0xc2,0xfc,0xef,0xb9,0x39,0x87};
// Application Session Key (MSB)
uint8_t AppSkey[16] =
{ 0xb9,0x08,0x6a,0x81,0x93,0x88,0x67,0x0d,0x69,0x69,0x4f,0xaa,0x86,0x2e,0x94,0x3c};
// Device Address (MSB)
uint8_t DevAddr[4] =
{ 0x00,0x2f,0x57,0x35} ;
```