Table of Contents

1. Bluetooth Low Energy for STM32 - peripheral 2
1.1. Installing the library and importing the example 2
1.2. Description of the most important elements of the code 3

1. Bluetooth Low Energy for STM32 - peripheral

This laboratory shows the details of the example provided as part of the STM32duinoBLE library by Arduino.

1.1. Installing the library and importing the example

As the first step, we need to install the library to the current project (how to download and import the first STM32WB55 project is described here. We can choose the simplest blink project as the base for our BLE peripheral).

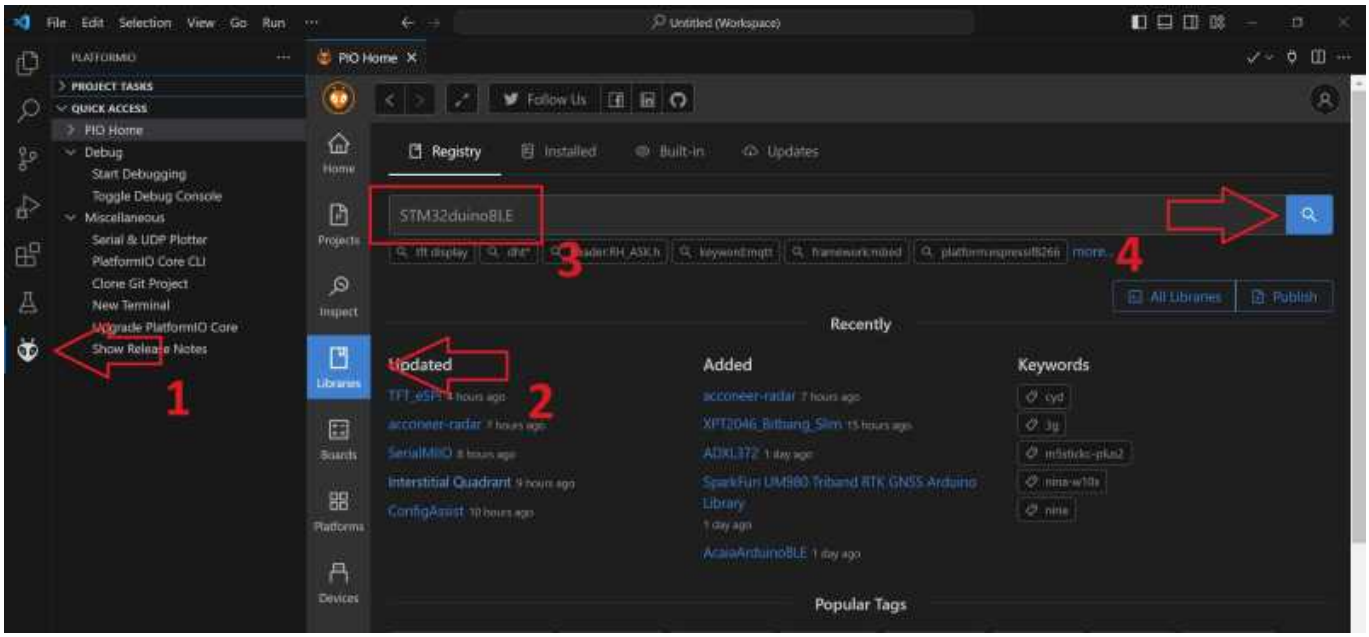


Fig. 1: Search for the BLE library for STM32 in VSCode/PlatformIO

After finding the library in the repository we can add it to our project. By clicking the “Add to Project” button you can select the already existing “arduino blink” project to add the library entry to the platformio.ini file automatically. It will create the dependency of your project build with the specific version of the STP32duinoBLE library. To use the code provided as the example, select the code of the “LED” example and paste it as the content of the “Blink.cpp” program chosen from the project tree.

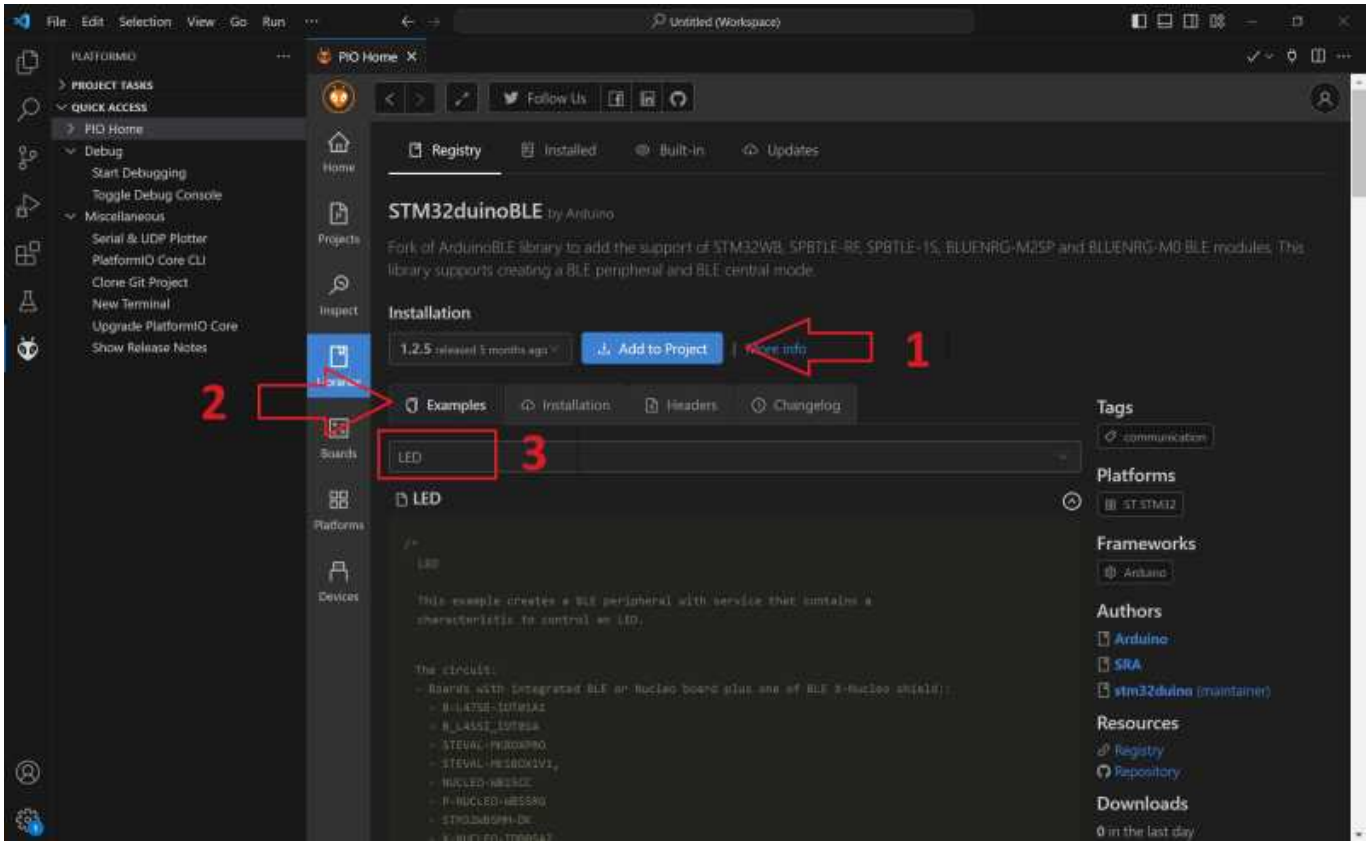


Fig. 2: Install the BLE library in the existing project

1.2. Description of the most important elements of the code

To use the BLE device you need to create the object of the BLELocalDevice class. This object can work using the SPI-connected Bluetooth module or the built-in unit in SoC. In the case of STM32WB55, the internal unit is used, so the HCISharedMemTransportClass is chosen as the connection to the BLE module.

```
HCISharedMemTransportClass HCISharedMemTransport;
BLELocalDevice BLEObj(&HCISharedMemTransport);
BLELocalDevice& BLE = BLEObj;
```

After creating the BLE object, we can configure services, characteristics and advertising process. Let's define the service first. To use the service you define, you specify the 128-bit version of the UUID. You can also use predefined 16-bit services as listed in Bluetooth SIG documents.

```
BLEService ledService("19B10000-E8F2-537E-4F6C-D104768A1214"); // BLE LED
Service
```

Services usually contain characteristics, also identified by 128-bit or 16-bit UUIDs.

```
BLEByteCharacteristic switchCharacteristic("19B10001-E8F2-537E-4F6C-
D104768A1214", BLERead | BLEWrite);
```

The number of characteristics in one service is limited to 512 by BLE specification. You can experiment on how many characteristics can be used in Arduino implementation.
The BLE subsystem must be initialized before use. It is possible to check if initialisation is successful.

```
// begin initialization
 if (!BLE.begin()) {
   Serial.println("starting BLE failed!");

   while (1);
 }
```

After successful initialisation, you can specify the name of your device which appears while scanning the BLE neighbourhood. It is included in the advertising frames. Also, the service you defined can be added to the advertising information.

```
// set advertised local name and service UUID:
 BLE.setLocalName("LED");
 BLE.setAdvertisedService(ledService);
```



Arduino BLE implementation limits the number of advertised services to one.

The next step is to create a list of services and characteristics.

```
 // add the characteristic to the service
 ledService.addCharacteristic(switchCharacteristic);

 // add service
 BLE.addService(ledService);
```

The order is important, first, you need to add characteristic (or more characteristics) to the service, next add the service to the list of services.



The number and list of services available after establishing the connection can differ from those advertised.

From:
https://dokuwiki.zmitac.aei.polsl.pl/ - **RAU6 Dokuwiki**

Permanent link:
https://dokuwiki.zmitac.aei.polsl.pl/en:edu:zmitac:s2:iotn:ble1?rev=1702731145

Last update: **2023/12/16 13:52**