# LAB ASSIGNMENT Nº3

*NB-IOT-BASED ULTRASONIC DETECTOR*

*(QUECTEL BC68 TE-B, STM32L476RG, MEO NB-IOT, FIREBASE, ANDROID STUDIO, NODE.JS)*

## 1 INTRODUCTION

This assignment will introduce the students to the Narrowband IoT (NB-IoT) technology, which is a 4G/5G extension to support applications requiring very low data rates and low power consumption. Like Lab Assignment Nº 2, it builds upon the Lab Assignment Nº 1 in terms of cloud storage and user interface, requiring no further changes to the Firebase project and Android app. Besides, NB-IoT supports IP stack protocols, namely UDP and TCP, which allows the devices to communicate directly with any machine in the Internet.

The instructions in this assignment are abridged so that, at each step, the students will have to investigate how to implement the requested functionality. At the end of this document, a list of useful references is provided.

## 2 SETTING-UP, DEPLOYING AND TESTING THE NB-IOT DEVICE AND APPLICATION

The device used in this assignment will perform the same tasks as the one in Lab Assignment Nº 1, but with a different MCU and communication technology. In fact, the sensor and actuator will be the same. The MCU will now consist of a modified[1] STM32L476RG node, which supports Arduino connectivity. Since this MCU operates at 3.2 V, it is the one recommended for connecting with the Quectel BC68 TE-B board[2]. The pinout of STM32L476RG is depicted in Figure 1.

[1] The STM32 Nucleo boards include the ST-Link programmer and debugger onboard, avoiding the need to employ an external ST-Link board. This allows the use of the ST-Link Utility application running on the PC to monitor the status of the MCU board and receive debugging text messages through a virtual COM port supported by the USB interface. However, the STM32 board serial port is needed to communicate with the Quectel BC68 TE-B, which requires changing the position (desoldering and re-soldering) two shunt resistors. The downside of this operation is that it destroys the possibility of using COM based debugging.

[2] Tests made with the Arduino Leonardo have shown that the Quectel BC68 TE-B does not operate correctly in some situations, besides the risk of malfunction due to mismatched voltage.

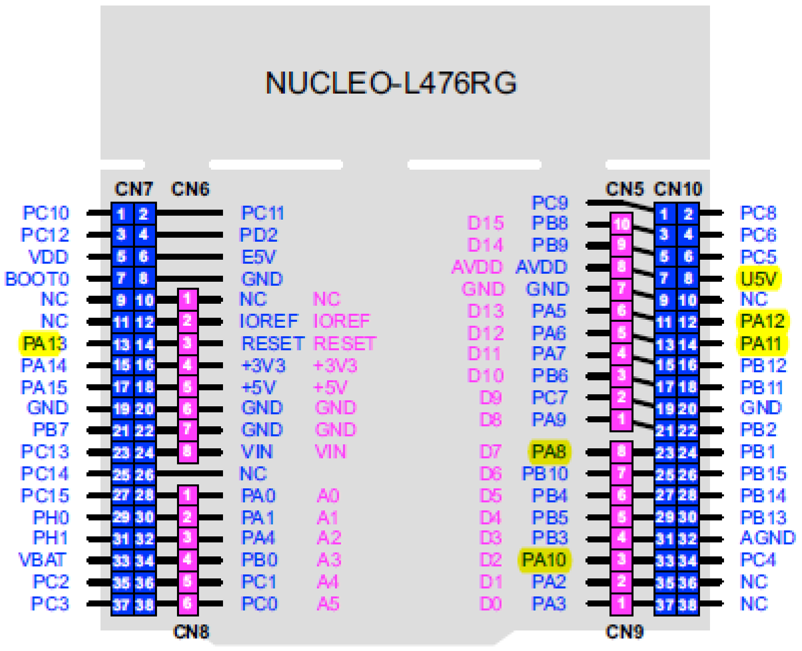


Figure 1: Pinout of STM32L476RG.

As written before, NB-IoT connectivity is provided by the Quectel BC68 TE-B board, which can be used both standalone (in this case, it is fed and controlled directly through the USB interface), or as an Arduino shield (see Figure 2). This board includes a SIM socket (1). A SIM card is required to connect to the mobile network. It also comes with an RF SMA connector (2) for antenna coupling. A two position switch (3) selects whether the board is being used standalone ("Main UART to USB") or as an MCU shield ("Main UART to MCU"). This later configuration will be the one used in this project.

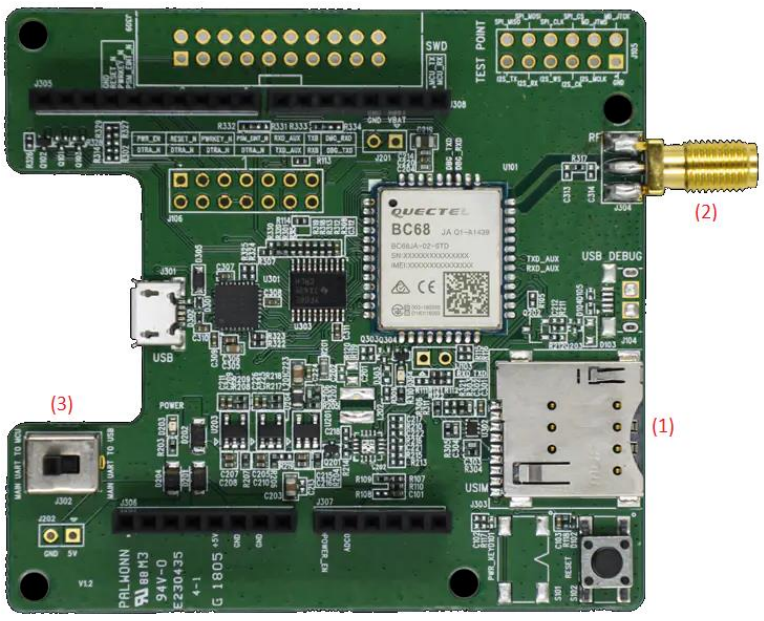


Figure 2: Quectel BC68 TE-B.

This board presents Arduino type connectors, making it easier to interface with sensors and actuators in the same way as would be done for an Arduino Uno or Arduino Leonardo. This will be used to connect to the HC-SR04 ultrasonic sensor and the piezo buzzer (actuator) in the very same way followed in Assignment nº 2.

1) Mount the circuit as depicted in Figure 3.

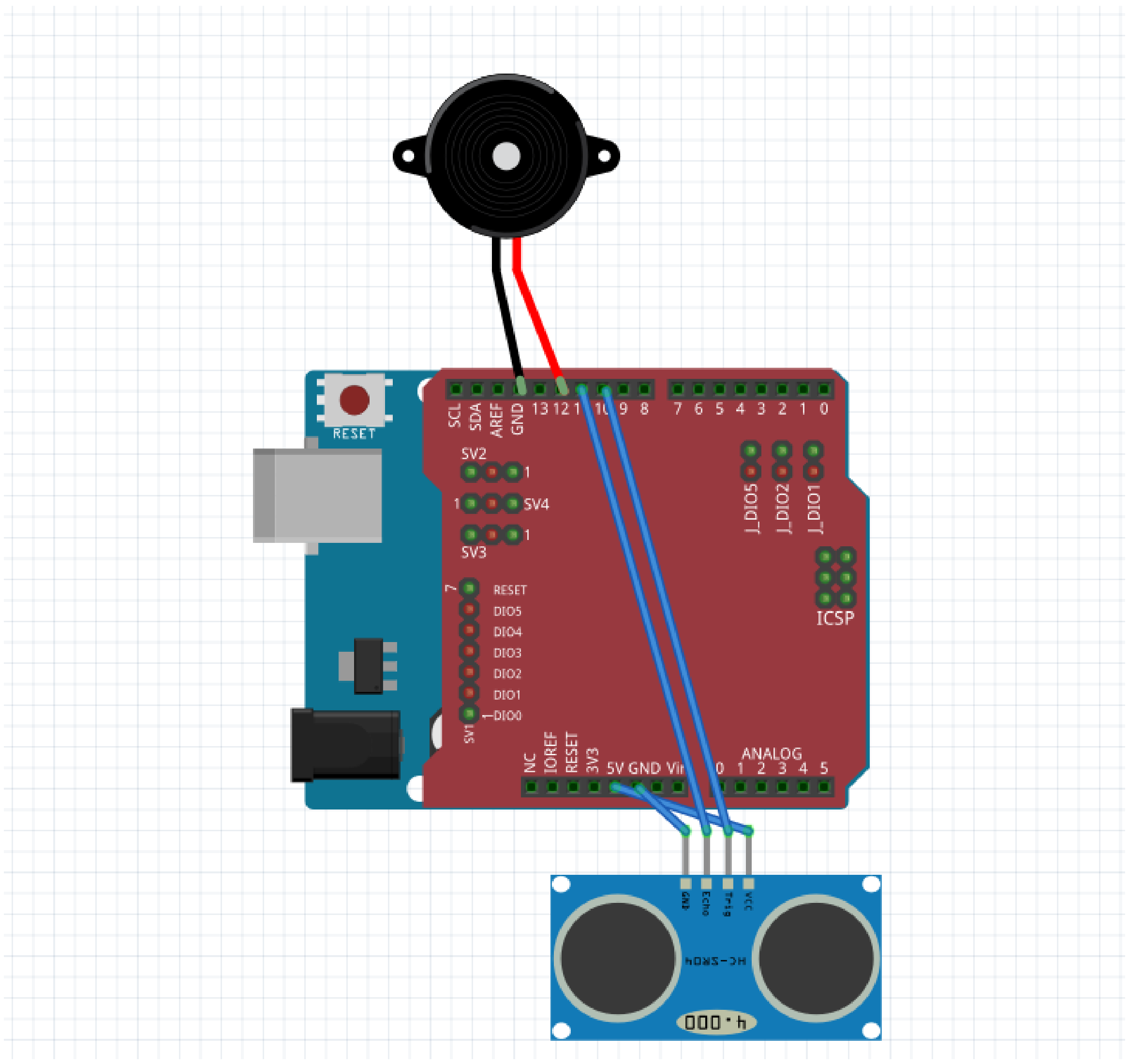


Figure 3: Ultrasonic detection and alarm circuit using Quectel BC68 TE-B shield on top of STM32L476RG, and HC-SR04.

2) In order to install the STM32L476RG board in the Arduino IDE, follow the instructions in https://github.com/stm32duino/wiki/wiki/Getting-Started .
3) Take the *nbiot-arduino-master.zip* archive provided in the webpage of the course and expand it in the *libraries* folder of your Arduino IDE installation[3].

[3] By default, in MS Windows 10, it should correspond to *C:\Users\xxx\Documents\Arduino\libraries\*, where "xxx" is the user's folder.

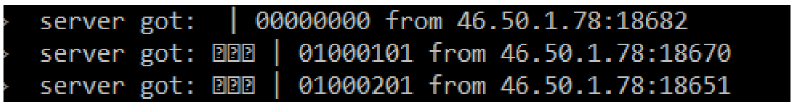
4) Take the *SR04_udp_send_receive_STM* Arduino sketch. **Change the *serverIP* and *serverPort*, so that they match the ones where the Node.js server is listening (see below)**. Check that it compiles and uploads correctly.
5) Like in Assignment nº 2, in order to be able to communicate with Google Firebase, a Node.js script must perform the bridging.

## 3 INTEGRATING WITH A NODE.JS FIREBASE HTTP SERVER

The NB-IoT network will only provide support for TCP/IP or UDP/IP connectivity between user application modules, not it will not provide persistent storage for the data. Consequently, an approach similar to the one followed in LAB2 will be followed: the uplink data will be sent to Google Firebase. In this project, the NB-IoT node will send the data directly to a Node.js UDP/IP server located at student's account in the **sigma.tecnico.ulisboa.pt** cluster. This server will receive the data from its UDP/IP socket and send it to the Google Firebase database. On the other hand, it will receive event notifications from Google Firebase, which are sent to the IoT node through the UDP/IP socket. In order to obtain this functionality, take the following steps[4]:

1) These instructions assume that Node.js and the respective firebase tools were already installed during LAB2.
2) Create the directory that will hold the webserver project, e.g.: **"mkdir ~/iot-alarm-app-nbiot; cd ~/iot-alarm-app-nbiot"**.
3) Execute **"firebase login --no-localhost"**. Copy/paste the login weblink into a browser and login to your Google Firebase account. Copy/paste the login code into the command line prompt. You are now logged-in.
4) Execute **"firebase init"**. Select "Functions" from the menu: select with SPACE, then press Enter. Select "Use and existing project", then select the project implemented in Lab Assignment Nº 1. Select "Typescript", then "Yes" when prompted to have ESLint debug support. Respond "Yes" when prompted to install dependencies with **npm**.
5) Execute **"cd functions"**.
6) In file *tsconfig.json*, add the following compiler option: *""noImplicitAny": false"*. In file *package.json*, replace the *serve* script call definition with the following: *""serve": "npm run build && firebase serve -o **xxx.xxx.xxx.xxx** -p 55**XX** --only functions",*", where ***xxx.xxx.xxx.xxx*** is the public IPv4 address of the sigma cluster computer[5], and ***XX*** is the number of the group (this is done to avoid port conflict in Sigma).
7) Replace the file ./src/index.ts with the one provided in the course's webpage. Edit the file and change the line server.bind(55**XX**, "193.136.128.**XXX**"), so that the port and IP address matches those used in file *package.json* emulator.
8) Compile and run the webserver**: "npm run serve"**. Solve any bugs that are reported. When the emulator weblink appears in the Sigma console, just copy/paste it into a web browser and press the ENTER key once. The script will then continue execution as expected, performing the *server.bind* operation, and then will start to listen for incoming messages (Figure 4).

[4] The described procedure repeats much of the steps used in LAB2 to install the Node.js server, so the students may build upon the installation already done in LAB2.
[5] In order to obtain the latter, you may run the command **"ip addr"**.



```
  server got:     | 00000000 from 46.50.1.78:18682
  server got: ??? | 01000101 from 46.50.1.78:18670
  server got: ??? | 01000201 from 46.50.1.78:18651
```

Figure 4: Messages from the device received by the Firebase emulator in Sigma.

9) Open your Firebase project and access the database. Check that the database variables are updated according to the messages issued by the device.
10) Run the Android app developed in Lab Assignment Nº 1, and check that everything is running smoothly, with the app being able to interact with the NB-IoT device.
11) **Optional:** Cloud Functions for Firebase is a serverless framework that lets you automatically run backend code in response to events triggered by Firebase features and HTTPS requests. The code that you have placed in the **sigma.tecnico.ulisboa.pt** cluster may also be deployed directly in Firebase. In order to do it, you need to create a billing account in Google Cloud and associate it with your project. **Creating a billing account entails providing a credit card number**. However, Google offers $300 in credit, which means that, in practice, during some time placement of the functions in Google Cloud will be free of charge. In order to place your processing function in Google Cloud, you just have to execute the deploy command: **"firebase deploy --only functions"**.

## 4 REFERENCES

[1] Android developer guides, https://developer.android.com/guide.
[2] John Horton, "Android Programming for Beginners", Packt, https://www.packtpub.com/product/android-programming-for-beginners/9781785883262#:~:text=Android%20Programming%20for%20Beginners%20will%20be%20your%20companion,or%20are%20just%20looking%20to%20program%20for%20fun.
[3] Bill Phillips, Chris Stewart, Kristin, Marsicano, "Android Programming – The Big Nerd Ranch Guide", 3rd Edition, Big Nerd Ranch, 2017.
[4] Quectel, "BC68-TE-B User Guide", https://www.tekmodul.de/wp-content/uploads/2018/04/Quectel_BC68_TE-B_User_Guide_V1.1.pdf .
[5] STM32L476RG Wiki, https://github.com/stm32duino/wiki/wiki/Getting-Started .
[6] Node.js documentation, https://nodejs.org/docs/latest-v14.x/api/ .