LAB ASSIGNMENT Nº1

WIFI-BASED ULTRASONIC DETECTOR

(FIREBASE, ANDROID APP AND NODE MCU)

1 INTRODUCTION

This assignment is intended to be followed by the students in order to give their first steps in Android IoT application programming. In the course of this tutorial, a simple Android app will be developed and connected to a Google Firebase project. The developed app will be used as a basis for subsequent lab assignments. It is assumed that the students have the Android Studio IDE installed in their machines.

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 CREATING A GOOGLE FIREBASE PROJECT

This sequence steps will guide you through the creation of a Google Firebase project that will store the data of your IoT application in the cloud.

1) Sign-in to Google or register a Google account.
2) Go to Google Firebase.
3) Go to the console.
4) Create a project called "IoT Alarm App" with default settings and location in Portugal.
5) Create a Realtime Database database. Realtime Database is used instead of Cloud Firestore because it is compatible with existing Arduino libraries that will be used with Arduino devices in later assignments. Start the database in **locked mode** for restricted access.
6) Go to the Data separator of the just created **iot-alarm-app** database and add the following items:
    a. *alarm_status*, setting it to Boolean value **true**.
    b. *enable_alert_sound*, setting it to **true**.
    c. *measurement*, setting it to 0.
7) Go to database Rules separator and check that **.read** and **.write** rules are both set to **true**.
8) In the Firebase menu, Go to *Build/Authentication/Sign-in method*. Enable the *Anonymous* method.

3 CREATING A SIMPLE APP

This sequence of steps will guide you through the creation of a basic app layout, which will later be completed with functionality allowing interaction between the user and the Google Firebase database.

1) Create a new project called "Assignment 1 GR00", with project type "Empty Activity" and minimum SDK version 19 (Android 4.4 KitKat). The *GR00* indicates the RMSF lab group number, in this case group 0. The language should be **"Java"**.
2) Verify that the new project has a single activity called *MainActivity*, whose class is defined in file *MainActivity.java*. The layout of *MainActivity* is very simple and has only two widgets: a **ConstraintLayout** with a **TextView** inside.
3) Modify the layout so that it will include two **TextView** gadgets and one **Button** gadget positioned as shown in Fig. 1. The default text in each gadget indicates the database item to which it will be associated. You can check the layout using the Android Emulator.
4) **Customize the app icon to be the IST logo[1]!**

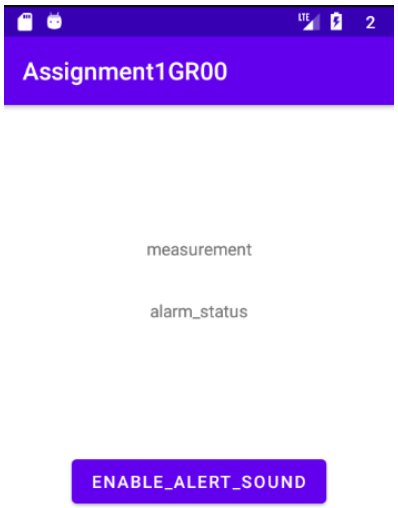


Fig. 1: Default layout of app Assignment 1.

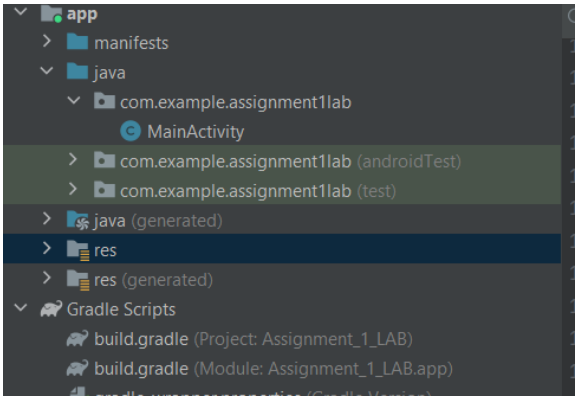
4 CONNECT THE ANDROID APP TO THE GOOGLE FIREBASE PROJECT

We are now in position to link the app to the Google Firebase database. Go to the Google Firebase console, select the "IoT Alarm App" project and then **Add app**. Select the Android platform. Follow the indicated steps until the developed app is successfully connected to the Google Firebase project[2]. The following tips might be useful:

1) When creating the app, one is asked the name of the package, which is the application ID. The latter can be found in project's app folder inside build.gradle.

[1] https://developer.android.com/studio/write/image-asset-studio#create-adaptive
[2] Some instructions and outdated regarding the current version of Gradle and should be adapted. See, for example this link.

2) There are different 2 main build gradle files. To access the build gradle of the project/ app choose one of "Project: …" or "Module: …" within the app folder.



3) In the new versions of build.gradle it's recommended to put the "Build script" given by Firebase tutorial before plugins and " all projects" below plugins and on top of task clean.
4) When configuring *build.gradle* at app level, don't forget to add the following dependency: *implementation 'com.google.firebase:firebase-database'*.
5) This video may be useful: https://www.youtube.com/watch?v=hjKt0Mvauus .

5 BIND THE APP GADGETS TO DATABASE ITEMS

At the end of this sequence of steps, the Android app will allow full interaction between the user and the Google Firebase database.

1) Add functionality to class **MainActivity**, so that the value of the *measurement* database item is shown in the *measurement* **TextView**.
2) Add functionality to class **MainActivity** class, so that:
    a. Whenever the *alarm_status* database item is set to **false** the *alarm_status* **TextView** displays "Nothing to Report".
    b. Whenever the *alarm_status* database item is set to **true** the *alarm_status* **TextView** displays "ALERT! ALERT!" and one notification sound is played.
3) Add functionality to class **MainActivity** class, so that:
    a. The *enable_alert_sound* **Button** is initialized with a string that corresponds to the value of the *enable_alert_sound* database item. In case this is **true**, the button will display "Disable Buzzer"; in case it is **false**, the button will display "Enable Buzzer".
    b. Whenever the *enable_alert_sound* **Button** displays "Disable Buzzer" and the user presses the button, database item *enable_alert_sound* will be set to **false** and the button will display "Enable Buzzer".
    c. Whenever the *enable_alert_sound* **Button** displays "Enable Buzzer" and the user presses the button, database item *enable_alert_sound* will be set to **true** and the button will display "Disable Buzzer".

Tip 1: Add the following imports, which will let you call the Firebase API methods:

- com.google.firebase.database.DataSnapshot;

- com.google.firebase.database.DatabaseError;
- com.google.firebase.database.DatabaseReference;
- com.google.firebase.database.FirebaseDatabase;
- com.google.firebase.database.ValueEventListener;

Tip2: Class DatabaseReference will allow you to get references to the Firebase database resources through calls to *DatabaseReference::getInstance().getReference().child(...)*. Get and set operations on the database items can be accomplished through *DatabaseReference::getValue(...)* and *DatabaseReference::SetValue(...)* methods[3].

Tip3: When configuring *build.gradle* at app level, don't forget to add the following dependency: *implementation 'com.google.firebase:firebase-database'*.

6 ADDING NODEMCU TO THE SYSTEM[4]

The interaction between the user and the cloud database is established. In order to achieve a true IoT application, the loop will now be closed to integrate the "thing", i.e., the sensor/actuator device. In this assignment, NodeMCU will be will be used, whose pinout is depicted in Fig. 2.

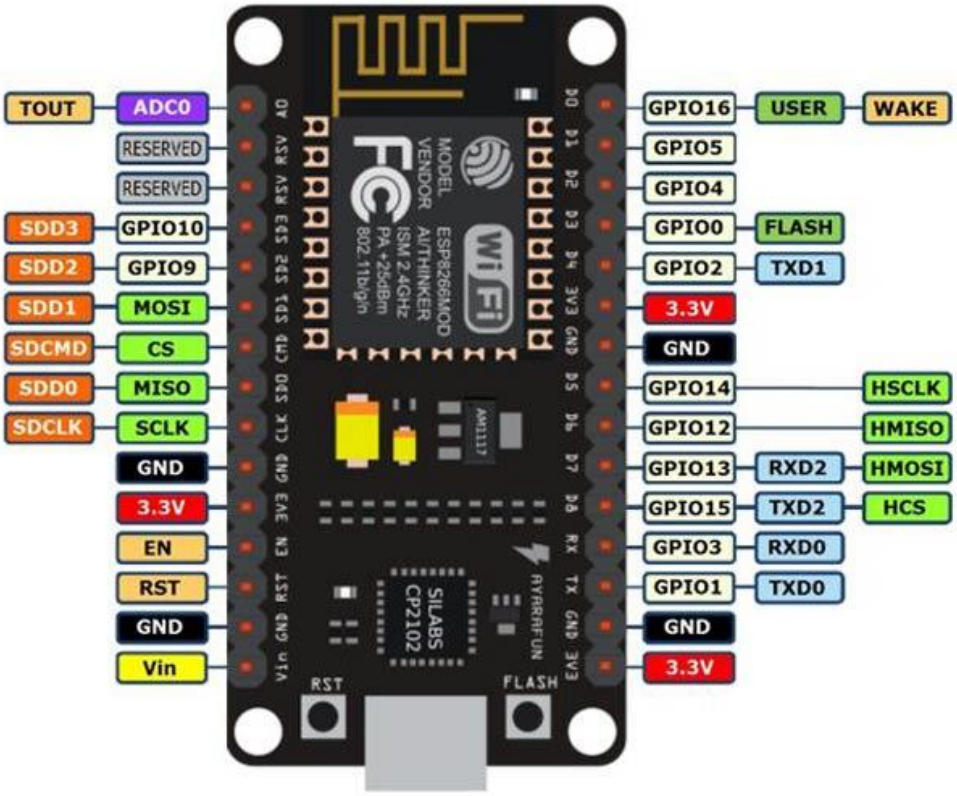


Fig. 2: Pinout of NodeMCU.

[3] More info can be found at https://firebase.google.com/docs/database/android/read-and-write .

[4] Some of the kits include the ESP32 WROOM DevKitC v4 instead of the NodeMCU. Instructions to install this development board are similar and can be found in https://www.iottechtrends.com/getting-started-with-esp32-wroom-devkitc/ .

The node MCU will receive power from the USB interface and will be connected with a HC-SR04 ultrasonic sensor (see Fig. 3) and a piezo buzzer (actuator).

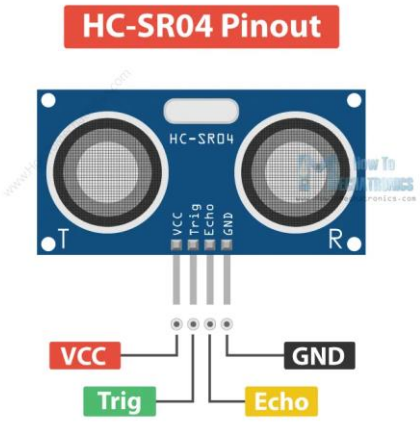


Fig. 3: HC-SR04 ultrasonic sensor.

NodeMCU includes an onboard LED that can be controlled by programs through the GPIO2 pin of the ESP8266 chip.

1) The Node MCU will be programmed using the Arduino IDE. In order for the Node MCU to be recognized in this development environment, configure Arduino IDE for NodeMCU according to the instructions in this link or an equivalent one.
2) Mount the circuit as depicted in Fig. 4.

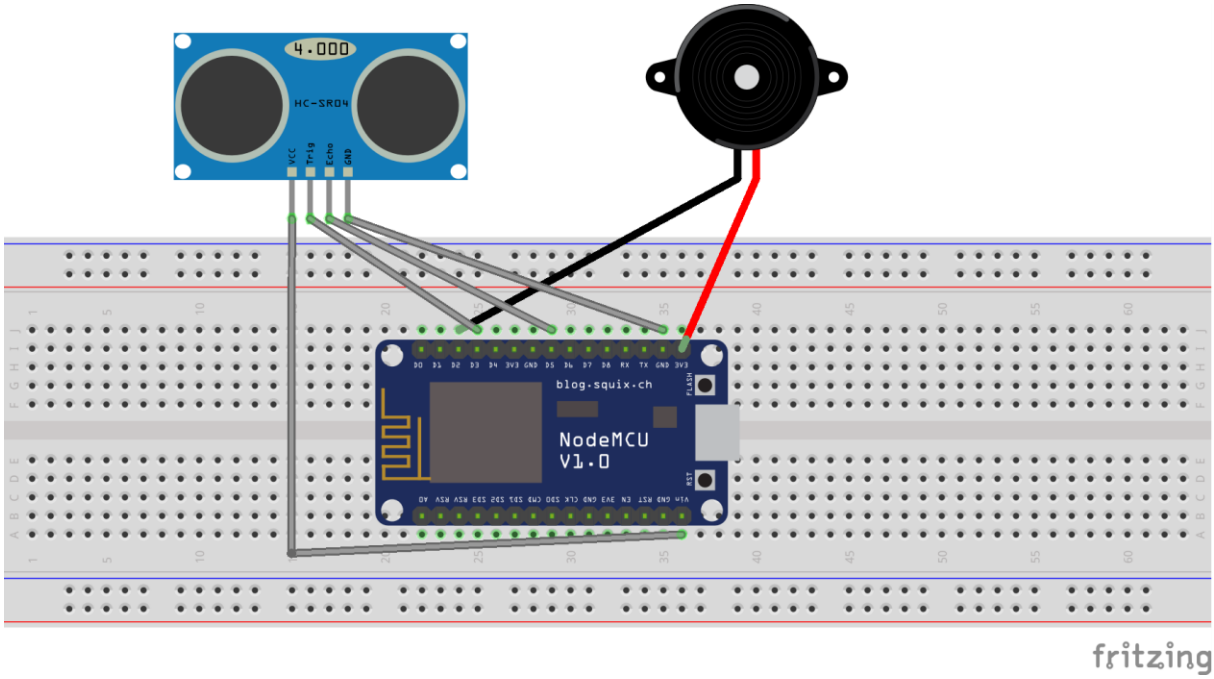


Fig. 4: Ultrasonic detector and alarm circuit using NodeMCU and HC-SR04.

3) Install the Firebase Arduino Client Library for ESP8266 and ESP32 by Mobizt as shown in Fig. 5.

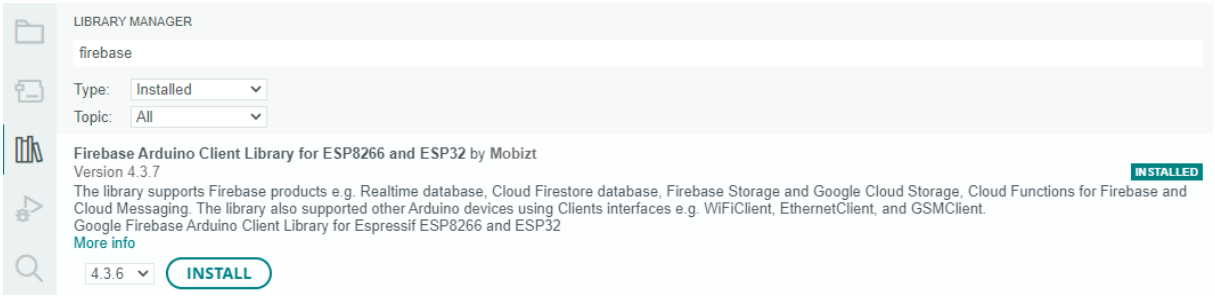


Fig. 5: Library Manager of the Arduino IDE with Firebase ESP8266 Client library installed.

4) Based on a library example (e.g., *Blynk.ino*, or *Basic.ino*), build your own Arduino program to perform the following actions in the main loop (*loop()* function):
   - Read the distance measured by the sensor. If the distance is shorter than 20 cm, consider that an intruder is detected, otherwise, there is no alert situation. Update the *alarm_status* database variable accordingly. In case motion is detected, increment the *measurement* database variable and activate the NodeMCU LED.
   - If the *enable_alert_sound* database variable is **true** and motion was detected, sound the buzzer (suggestion: use the *tone()* function together with pitch constants provided in file *pitch.h*).
   - Wait for 1 second before the end of the *loop()* function.
5) In order to configure the *.ino* project, you will need to obtain the **API_KEY** and **DATABASE_URL** parameters, which can be retrieved from the Google Firebase console. In order to obtain the API Key, go to *Project Overview/Project settings*. The Database URL can be directly retrieved from *Realtime Database*.
6) NodeMCU is based on the ESP8266 chip. This chip is incompatible with *eduroam* authentication procedures. In order to connect to Internet, inside the classroom use the following WiFi network:
   - **SSID:** Labs-LSD
   - **Password:** aulaslsd
7) Once the NodeMCU program is correct, you should observe the expected hardware behavior and the Firebase database variables should change according to the motion detection events. Once the Android App is running, it should also show the current database status, as well as to enable and disable the buzzer when the user presses the button.

7 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.