



WIFI AC REMOTE

by Lior Yaacov
(ID: 201607173)
Electrical Engineering Dept.
Ariel University



Academic supervisors: Dr. Avihai Aharon and Prof. Amir Abramovich

OCTOBER 18, 2018

Table of Contents

1.	Internet of Things (IoT)	5
1.1	What is IoT?	5
2.	IR Communication	7
2.1	What is IR?	7
2.2	Modulation	7
2.3	Pulse Width Modulation (PWM)	8
2.4	The Transmitter Circuit	9
2.5	The Receiver Circuit	10
3	WiFi AC Project	12
3.1	Why we need it?	12
3.2	Main Concept	12
4.	Firebase	13
4.1	What is Firebase?	13
4.1.1	Firebase as a Realtime Database	13
4.1.2	Firebase as a Files Storage	13
4.1.3	Firebase Authentication	13
4.1.4	Firebase Hosting	13
4.2	Firebase as WiFi-AC Database	14
4.2.1	Firebase Authentication	14
4.2.2	Firebase Database	15
5.	MIT App Inventor	17
5.1	What is MIT App Inventor?	17
5.2	Main Screen	18
5.2.1	Set WiFi Credentials Screen	18
5.3	Sign In/Up Screen	18
5.4	Operation Screen	19
5.4.1	Add New AC Screen	19
5.5	Remote Control Screen	20
6.	Arduino & ESP	21
6.1	What is Arduino?	21
6.2	Arduino Nano	21
6.3	ESP8266 & NodeMCU	23

7	3D Printing.....	27
7.1	What is 3D Printing?.....	27
7.2	Anet A8.....	27
7.3	3D Printing in WiFi-AC Project	27
7.3.1	Main Board Package	28
7.3.2	The Transmitter Package	28
8	The Code	29
8.1	Motivation	29
8.2	Libraries	29
8.3	Objects.....	29
8.4	Main Code Functions.....	30
8.5	Setup() function	30
8.6	Loop() Function	31
8.6.1	Case 1: Get a code from Firebase and send it to AC	31
8.6.2	Case 2: Record New Code	34
8.6.3	Case 3: Reset WiFi Credentials.....	35
8.7	The Arduino IDE	36
9	The Board	38
9.1	Motivation	38
9.2	The Eagle	38
9.3	Main Board	39
9.4	Transmitter Board	40
10	Tests	42
10.1	Testing the system	42
10.1.1	Creating an account.....	42
10.1.2	Creating a new AC	42
10.1.3	Record new codes	43
10.1.4	Sending the code	44
10.2	IR LED Range Test.....	45
10.3	RF Range Test.....	46
11	Conclusions.....	47
11.1	IoT In Your Home	47
11.2	Broker Unit	47

12	References.....	48
12.1	Figures List.....	48
12.2	Tables List	49
12.3	References.....	49

Gratitude

I would like to thank Dr. Avihai Aharon for accompanying me for over a year with this project. His support and guidance make this project professional and reliable.

To Eng. Meir Buskila, for having devoted his time to me and helping me plan the boards. His experience and his wide knowledge make all this happen.

I also want to thank my wife and daughter for encouraging me to go on when things didn't go well, and share with me the satisfaction of success.

1. Internet of Things (IoT)

1.1 What is IoT?

IoT systems allow users to achieve deeper automation, analysis, and integration within a system. They improve the reach of these areas and their accuracy. IoT utilizes existing and emerging technology for sensing, networking, and robotics.

IoT exploits recent advances in software, falling hardware prices, and modern attitudes towards technology. Its new and advanced elements bring major changes in the delivery of products, goods, and services; and the social, economic, and political impact of those changes.



Figure 1 - IoT Vision

IoT – Key Features

The most important features of IoT include artificial intelligence, connectivity, sensors, active engagement, and small device use. A brief review of these features is given below:

AI – IoT essentially makes virtually anything “smart”, meaning it enhances every aspect of life with the power of data collection, artificial intelligence algorithms, and networks. This can mean something as simple as enhancing your refrigerator and cabinets to detect when milk and your favorite cereal run low, and to then place an order with your preferred grocer.

Connectivity – New enabling technologies for networking, and specifically IoT networking, mean networks are no longer exclusively tied to major providers. Networks can exist on a much smaller and cheaper scale while still being practical. IoT creates these small networks between its system devices.

Sensors – IoT loses its distinction without sensors. They act as defining instruments which transform IoT from a standard passive network of devices into an active system capable of real-world integration.

Active Engagement – Much of today's interaction with connected technology happens through passive engagement. IoT introduces a new paradigm for active content, product, or service engagement.

Small Devices – Devices, as predicted, have become smaller, cheaper, and more powerful over time. IoT exploits purpose-built small devices to deliver its precision, scalability, and versatility.

2. IR Communication

2.1 What is IR?

IR stands for Infra-Red and is an electromagnetic radiation (EMR) with longer wavelengths than those of visible light (above 700nm), which make it invisible to the human eye. That's one of the reasons why IR is chosen for remote control purposes – we want to use it but not interested in seeing it. Another reason is because IR LEDs are easy to make and therefore can be very cheap.

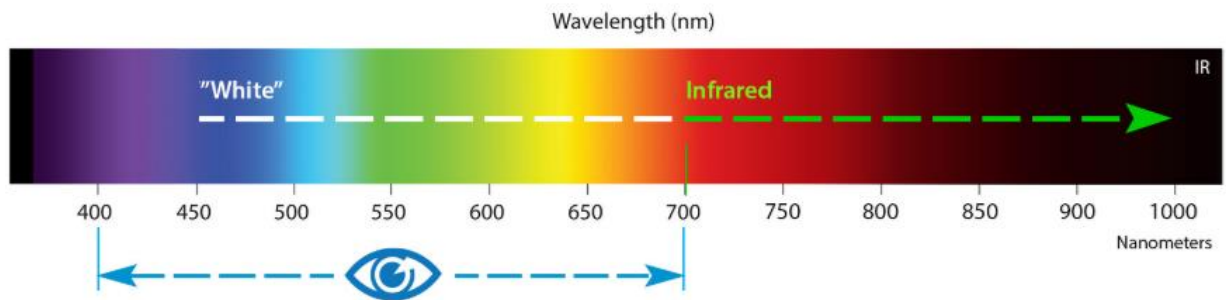


Figure 2 - The Light's Spectrum

There are a lot of sources of Infra-Red light. The sun is the brightest source of all, but there are many others such as light bulbs, candles and even our body radiates Infra-Red light. In fact, everything that radiates heat, also radiates Infra-Red light. Therefore, we must take some precautions to guarantee that our IR message gets across to the receiver without errors.

2.2 Modulation

Modulation of the signal on a carrier frequency is our answer to make our signal stand out above the noise. With modulation the IR LED can blink in a predefined frequency. The IR receiver will be tuned to that frequency, so it can ignore everything else. Each AC command will have its own set of blinking order. For example, to turn on a Tadiran AC on 30 degrees, we need to send that code:

```
8984, 4384, 680, 516, 702, 490, 706, 1572, 706, 486, 706, 490, 676, 1596, 712, 486, 706,
486, 706, 490, 702, 1576, 706, 1572, 712, 1570, 706, 490, 702, 490, 680, 512, 706, 486, 706,
486, 708, 486, 706, 486, 706, 486, 706, 486, 706, 1576, 706, 486, 706, 486, 706, 516, 676,
490, 702, 490, 676, 516, 680, 1596, 708, 490, 702, 1572, 710, 486, 706, 512, 682, 1572, 684,
512, 708, 19770, 706, 516, 676, 486, 706, 486, 706, 490, 680, 512, 682, 512, 680, 512, 708,
486, 706, 486, 708, 486, 706, 486, 708, 486, 706, 486, 708, 486, 706, 486, 708, 486, 706,
486, 708, 486, 706, 486, 706, 486, 706, 490, 676, 516, 682, 512, 680, 512, 682, 512, 706,
486, 706, 486, 706, 486, 706, 486, 706, 486, 706, 1572, 686, 1598, 706
```


The reason that the code is so long is that every time we send a command to the AC, we send all the configuration of the current active status, and only changing the exact parameter we want to change. For example, if we want to increase the temperature by 1 degree, we need to send the new temperature and include all the other parameters such as the AC mode (hot, cold, fan), the fan power, timers and so on. The numbers in the code are representing the amount of time the led will be on (in milliseconds). If we connect the IR receiver to an oscilloscope, we expect to see something like this:

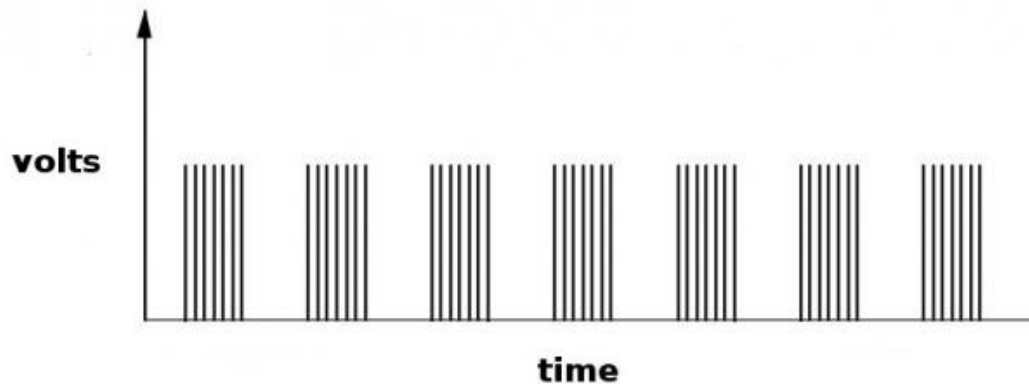


Figure 3 - IR code as a function of volts and time

As we know, the Arduino's I/O pins are generating high (5 volts) or low (app. 0 volts), so basically, we can adjust it to send that kind of code. But there is a problem: we need to switch the LED on and off very fast and the I/O pins are too slow for that. The solution is to use a very important ability of the Arduino – PWM.

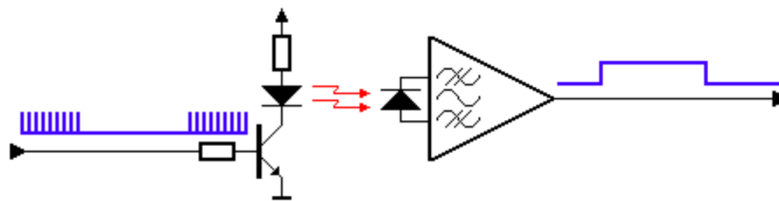


Figure 4 - Modulation Circuit

2.3 Pulse Width Modulation (PWM)

Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means. Digital control is used to create a square wave, a signal switched between on and off. This on-off pattern can simulate voltages in between full on (5 Volts) and off (0 Volts) by changing the portion of the time the signal spends on versus the time that the signal spends off. The duration of "on time" is called the pulse width. To get varying analog values, we change, or modulate, that pulse width. If we repeat this on-off pattern fast enough with an LED for example, the result is as if the signal is a steady voltage between 0 and 5v controlling the brightness of the LED.

In fig # below, the green lines represent a regular time period. This duration or period is the inverse of the PWM frequency. In other words, with Arduino's PWM frequency at about 500Hz, the green lines would measure 2 milliseconds each. A call to `analogWrite()` is on a scale of 0 - 255, such that `analogWrite(255)` requests a 100% duty cycle (always on), and `analogWrite(127)` is a 50% duty cycle (on half the time) for example.

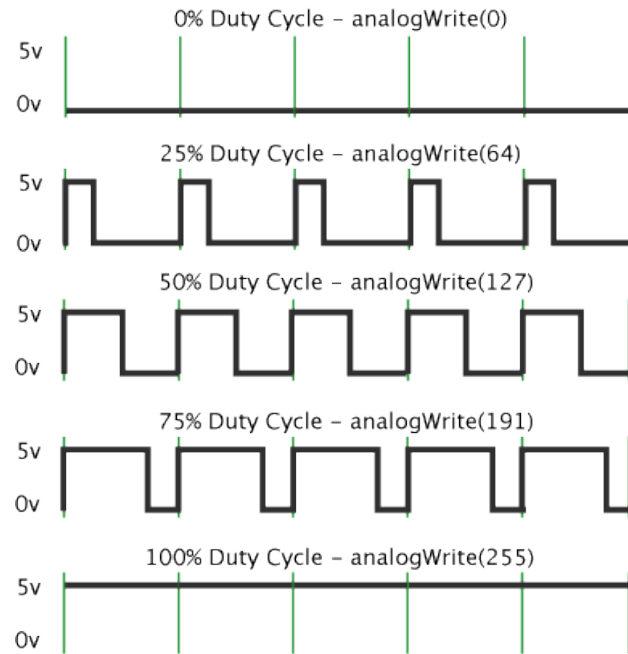


Figure 5 - PWM examples

2.4 The Transmitter Circuit

The IR transmitter circuits are going to be powered using a battery. Therefore, those circuit should consume as little power as possible while keeping the IR signals strong to achieve an acceptable control distance.

A simple transistor circuit can be used to drive the LED. A transistor with a suitable HFE and switching speed should be selected for this purpose.

The resistor values can simply be calculated using Ohm's law. We need to remember that the nominal voltage drops over an IR LED is approximately 1.1V.

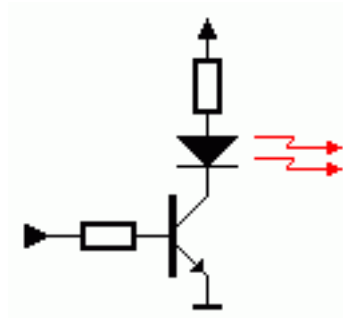


Figure 6 - Transmitter circuit

But, the circuit above has a great disadvantage – the current through the LED is decreasing as the battery voltage drops. This will result in a shorter control distance that can be covered. An emitter follower circuit, as shown below, can avoid that problem. The two diodes in series will limit the pulses on the base of the transistor to 1.2 Volts. The base-emitter voltage of the transistor subtracts 0.6 volts from that, resulting in a constant amplitude of 0.6 volts at the emitter when the IR transmitter is active. This constant amplitude across a constant resistor results in current pulses of a constant magnitude, regardless the battery voltage. Calculating the current through the LED is simply applying Ohm's law again.

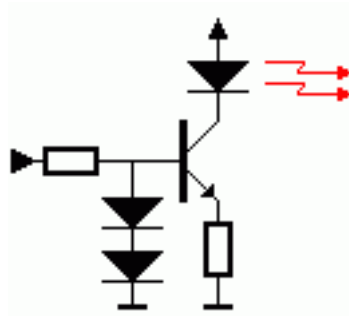


Figure 7 - emitter-follower circuit

2.5 The Receiver Circuit

The receiver part is divided into two parts. There's the AC receiver, which we are not going to touch because it's a built in by the AC designers, and there's the receiver we use to record the codes from the desired AC remote we want to control. The receiver circuit is much simpler than the transmitter because it only gets data and don't need to send anything. It is connected to the micro controller with wires so we don't need any kind of modulation. Although the receiver's modules are all packed in one chip called TSOP (see fig # below), it is important to understand how it works. To do so, let's have a look on the receiver's block diagram (fig #):

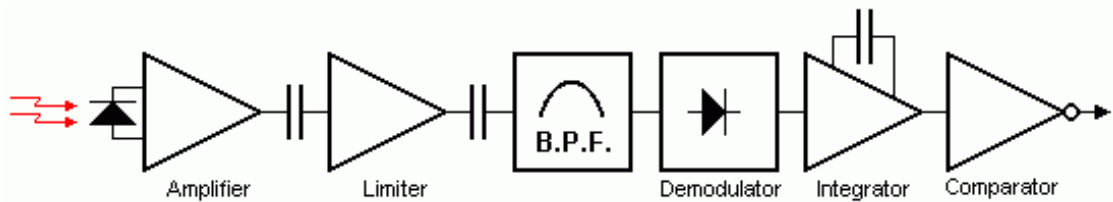


Figure 8 - IR Receiver Block Diagram

The received IR signal is picked up by the IR detection diode on the left side of the diagram. This signal is amplified and limited by the first 2 stages. The limiter acts as an AGC (Automatic Gain Control) circuit to get a constant pulse level, regardless of the distance to the handset.

As we can see, only the AC signal is sent to the Band Pass Filter. The Band Pass Filter is tuned to the carrier frequency of the handset unit. Common carrier frequencies range from 30kHz to 60kHz in consumer electronics, so it is important to pick the right one.

The next stages are a detector, integrator and comparator. The purpose of these three blocks is to detect the presence of the carrier frequency. If this carrier frequency is present the output of the comparator will be pulled low.

As said before, we are going to use a chip called TSOP that packs all those blocks together to a small sensor, shown in fig # below.



Figure 9 - TSOP 4838, an IR receiver

3 WiFi AC Project

3.1 Why we need it?

The main idea behind my project is to make your home smarter. That means that every element you have can be connected to an app and can be controlled from distance. I start with the AC, because it contains sending an IR codes and using database and much more complex than just turn switch on or off, but the idea is to continue and make all elements at home able to connect to the internet and therefore be controlled with a smartphone app.

3.2 Main Concept

The way I want to communicate with the AC is using IR codes. I'm going to use an IR LED and a special ability of the Arduino's I/O pins to draw current through the LED for a very specific segments of time. It's called PWM, which stands for Pulse Width Modulation. This code will be received in the AC receiver and command the AC to switch its mode.

The transmitter circuit will be operated by a micro controller called ESP that also able to connect to WiFi. The ESP will sample the Firebase database every 500 milliseconds to see if a new order from the user has received.

The user will operate the system from an app on a smartphone. Using the app, the user can create as much AC's profile as needed and record codes for them. When the user sends an order to the AC, the app connects to Firebase and change a set of values. Then, the micro-controller, at a sample time, will see the current order and operate the circuit to execute the user's command.

A block diagram of the system is shown below:

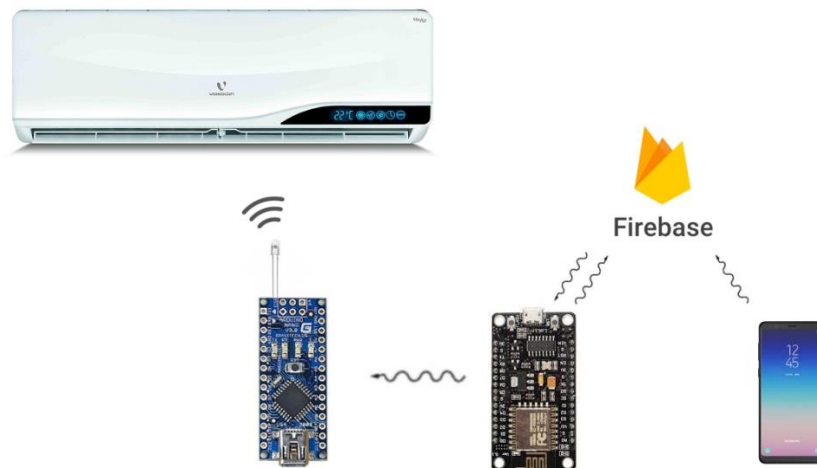


Figure 10 - WiFiAC Block Diagram

4. Firebase

4.1 What is Firebase?

Firebase is a backend-as-a-Service (BaaS) that started as a YC11 Startup and grew up into a next-generation app-development platform on Google Cloud Platform.

Firebase frees developers to focus crafting fantastic user experiences. Normally you need to manage servers and write APIs, but when it comes to Firebase – Firebase itself is your server, API and datastore, and all written so generically that you can modify it to suit most needs.

4.1.1 Firebase as a Realtime Database

Realtime data is the way of the future, nothing compares to it.

Most databases require you to make HTTP requests to get your data and give you that data only when you ask for it. Firebase sends new data as soon as it's updated. When a client saves a change to the data, all connected clients receive the updated data almost instantly. That can happen because Firebase doesn't connect to its client through HTTP, like most other databases, but through a WebSocket. WebSockets are much faster than HTTP and does not require to make individual calls. One WebSocket is enough for a lot of users, and all the data syncs through that single WebSocket as fast as the client's network can carry it.



Figure 11 - Firebase Logo

4.1.2 Firebase as a Files Storage

Firebase Storage provides a simple way to save binary files (most often images), but it could be anything else, and it's happening directly from the client.

Firebase Storage has its own system of security rules the GCloud bucket from unauthorized users, while granting detailed write privileges to authenticated clients.

4.1.3 Firebase Authentication

Firebase Auth has a built-in email-password authentication system. It supports OAuth2 for Google, Facebook, Twitter and GitHub. Firebase's OAuth2 system is well documented and mostly copy/paste. Firebase Auth integrates directly into Firebase Database, so it can control access to data.

4.1.4 Firebase Hosting

Firebase includes an easy-to-use hosting service for all static files. It serves them from a global CDN with HTTP/2.

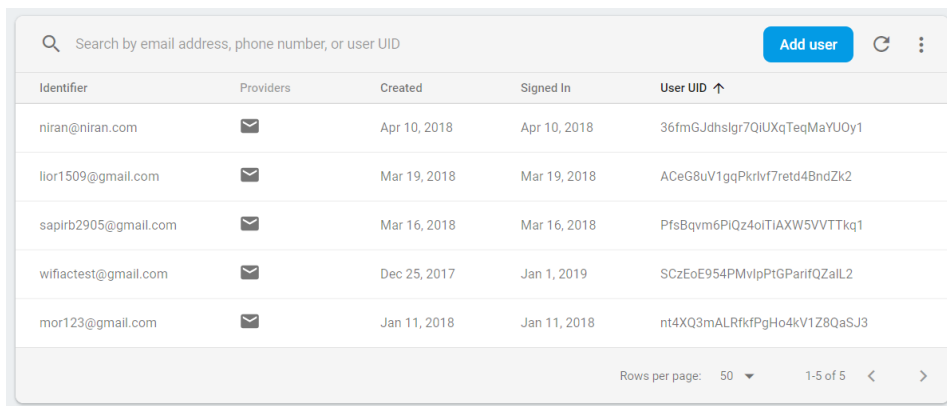
Firebase also hosting utilizes Superstatic, which can run locally for testing.

4.2 Firebase as WiFi-AC Database

Firebase is a crucial element of the project. Everything is working through Firebase and all the data is saved inside it. Login in and signing up also available thanks to this amazing tool. Therefore, we have to keep it organized and easy to read.

4.2.1 Firebase Authentication

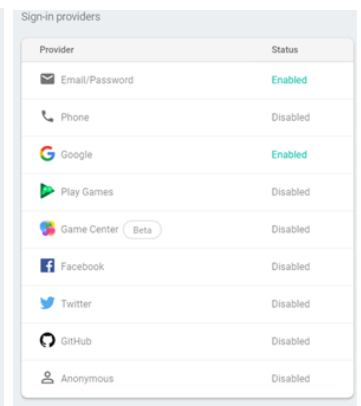
Firebase support a wide variety of authentication platforms such as Google, Facebook, Twitter, Email and more. In this project we currently using only Email and Google authentications. In figure # below we can see the authentication dashboard and in figure # we can see the users that already signed up.



The screenshot shows the 'Users' dashboard in the Firebase console. It features a search bar at the top with the text 'Search by email address, phone number, or user UID'. Below the search bar is a table with columns: Identifier, Providers, Created, Signed In, and User UID. The table lists five users with their respective email addresses and creation/sign-in dates. At the bottom right of the table, there are controls for 'Rows per page' (set to 50) and '1-5 of 5' rows.

Identifier	Providers	Created	Signed In	User UID
niran@niran.com	✉	Apr 10, 2018	Apr 10, 2018	36fmGJdhsigr7QiUXqTeqMaYUOy1
lior1509@gmail.com	✉	Mar 19, 2018	Mar 19, 2018	ACeG8uV1gqPkrIv7ret4BndZk2
sapirb2905@gmail.com	✉	Mar 16, 2018	Mar 16, 2018	PfsBqvm6PIQz4oiTIAxW5VVTtkq1
wifiactest@gmail.com	✉	Dec 25, 2017	Jan 1, 2019	SCzEoE954PMvlpPtGParifQZaIL2
mor123@gmail.com	✉	Jan 11, 2018	Jan 11, 2018	nt4XQ3mALRfkfPgHo4kV1Z8QaSJ3

Figure 12 - Users Dashboard

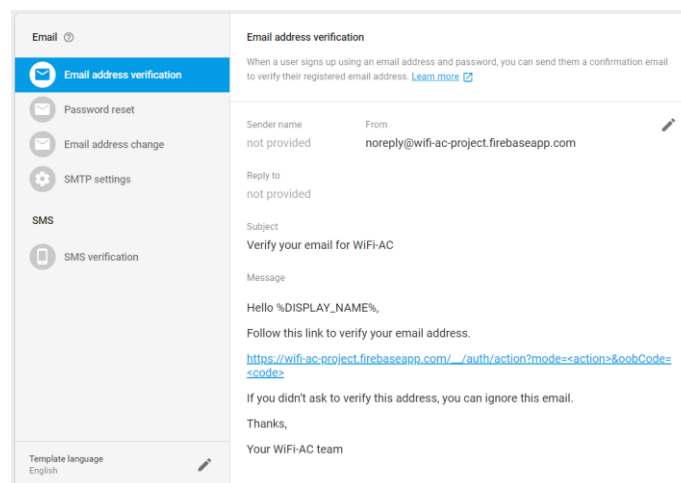


The screenshot shows the 'Sign-in providers' dashboard in the Firebase console. It is a table with two columns: 'Provider' and 'Status'. The providers listed are Email/Password (Enabled), Phone (Disabled), Google (Enabled), Play Games (Disabled), Game Center (Beta, Disabled), Facebook (Disabled), Twitter (Disabled), GitHub (Disabled), and Anonymous (Disabled).

Provider	Status
✉ Email/Password	Enabled
☎ Phone	Disabled
🌐 Google	Enabled
🎮 Play Games	Disabled
🎮 Game Center (Beta)	Disabled
📘 Facebook	Disabled
🐦 Twitter	Disabled
🔗 GitHub	Disabled
👤 Anonymous	Disabled

Figure 13 - Authentication Options

Firebase Authentication system also provides the database admin to send mails of verification and reset to the user. The mail templates can be change to the admin's preference. If SMS authentication is enabled, the text message also can be set. The dashboard window is shown in figure #.



The screenshot shows the 'Communication Templates' dashboard in the Firebase console. On the left is a sidebar with navigation links: Email address verification (selected), Password reset, Email address change, SMTP settings, SMS, and SMS verification. The main area displays the 'Email address verification' template. It includes fields for 'Sender name' (not provided) and 'From' (noreply@wifi-ac-project.firebaseio.com). The 'Subject' is 'Verify your email for WiFi-AC'. The 'Message' body contains a personalized greeting, a link to verify the email address, and a note about ignoring the email if not requested. At the bottom, there is a 'Template language' dropdown set to 'English'.

Figure 14 - Communication Templates

Finally, usage graph can be plotted to let the admin monitor the number of members and their locations. An example is shown below in figure #.

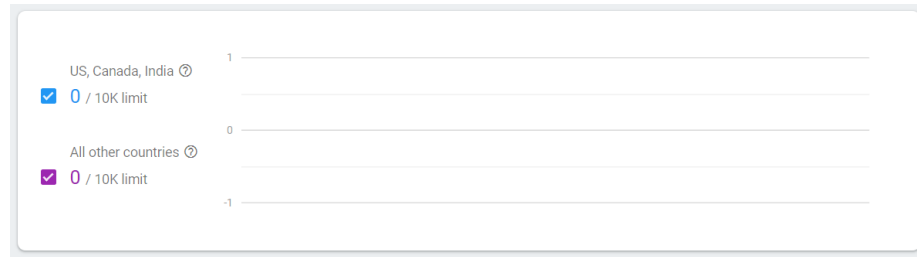


Figure 15 - Usage Graph

4.2.2 Firebase Database

As stated before, Firebase plays a critical role in this project. All user's data is saved inside it, except for the WiFi credentials to prevent data leaks. The database is organized as a tree with parent and child fields. Each user gets his own UID when signing up for the first time. This UID is used to identify the user and all his data saved under that UID tree. An example is shown below in figure #.

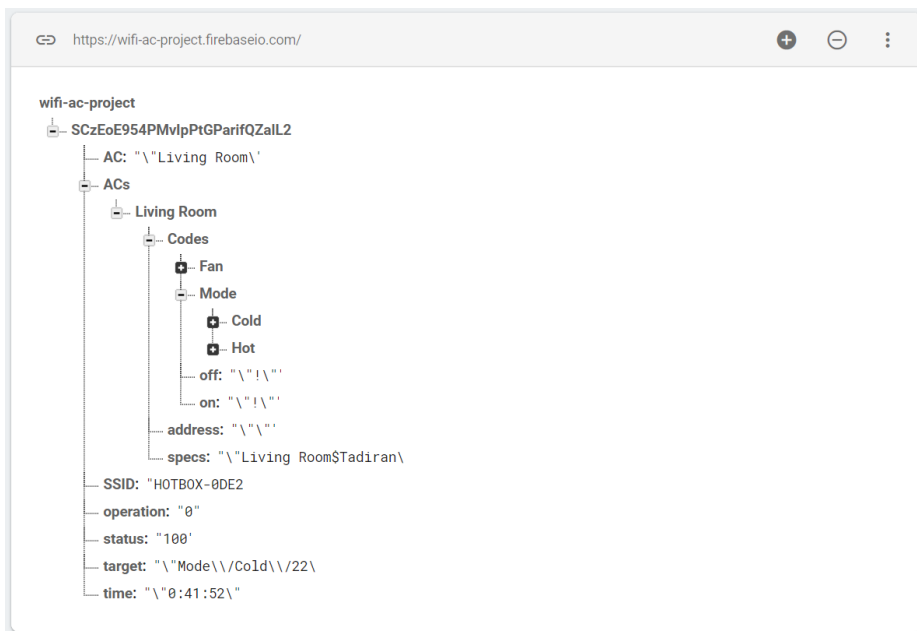


Figure 16 - WiFiAC Firebase Structure

In the example below, the UID is **SCzEoE954PMvlpPtGParifQZail2**, as can be seen at the top of the database tree. This way, the app can identify the user and he's ACs and able to send the right codes for each AC, per user.

The fields on the main tree of the DB are:

- **AC:** The current AC that is active in the app.
- **ACs:** All the ACs the user has entered in the app.
- **SSID:** The user's WiFi SSID.

- **operation:** Used to order the main circuit which execution to perform (read new code, send a code to AC)
- **status:** Used by the app and by the micro controller. Both are updating this value to tell each other what the current status is (200 = OK)
- **target:** Used by the main circuit to know which AC to send the code
- **time:** On later versions, this field will be used to set timers

inside the ACs branch we can see all the ACs that exists. If we expand one of the ACs, we can see the structure in which the codes are saved. As can be seen in fig #, inside each AC branch there is another branch called *Codes*. Inside that branch, there are another two branches (Fan and Mode) and two codes: on and off.

Inside the *Mode* branch we can see two more branches for Cold and Hot. On later versions there will be more branches for other functions of the AC.

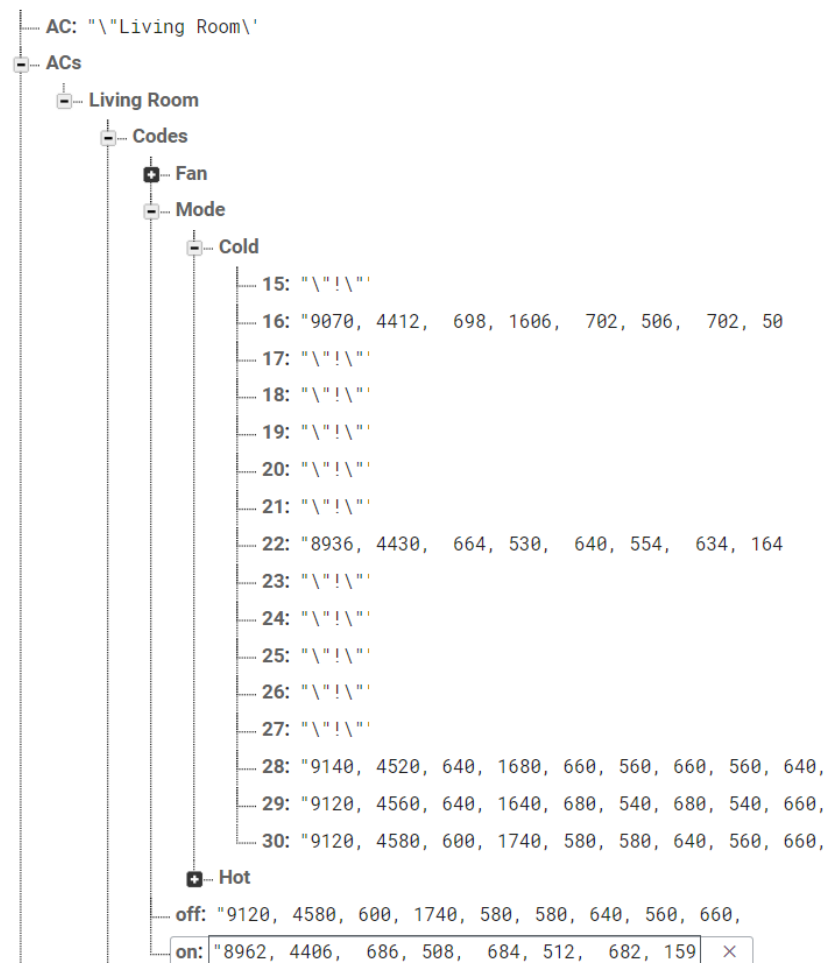


Figure 17 - ACs branches

5. MIT App Inventor

5.1 What is MIT App Inventor?

MIT App Inventor is an open-source web application originally provided by Google, and now maintained by the Massachusetts Institute of Technology (MIT).

MIT allows new programmer to create software application for the Android OS. It has a graphical interface which allows users to drag-drop blocks of codes to create a functional application that can run on an Android device.

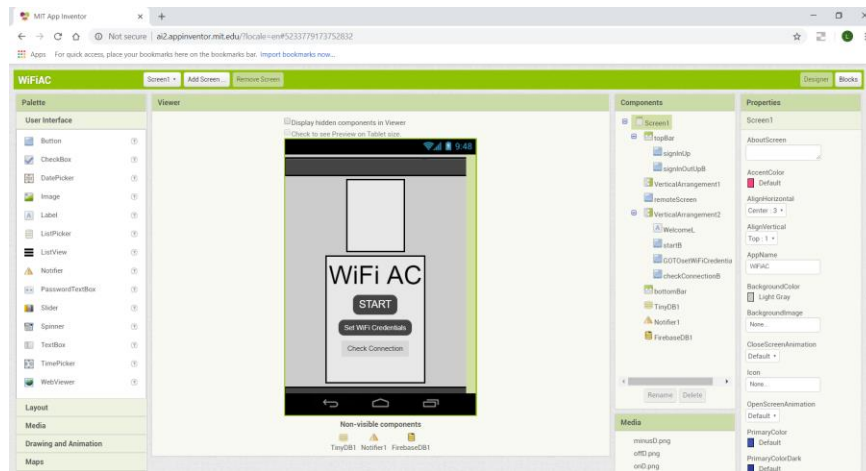


Figure 18 - MIT App Inventor UI Screen

MIT App Inventor provides two screens: Designer and Blocks.

Designer: In this screen we set the GUI (Graphical User Interface) of the app. We can place buttons, labels, images and other graphical elements. We can also place connectivity elements such as WiFi manager, Firebase and google sign in/up methods. To make the screen more organized, we can use layouts and tables. The last thing we add is a tinyDB element which allows us to save data between screens.

Blocks: In this screen we actually set all the functionality of the app. Each element we add in the designer screen gets its own blocks on the blocks screen. Using the blocks screen, we can define what will happen when a button is pressed, an instruction is arrived from firebase and so on.

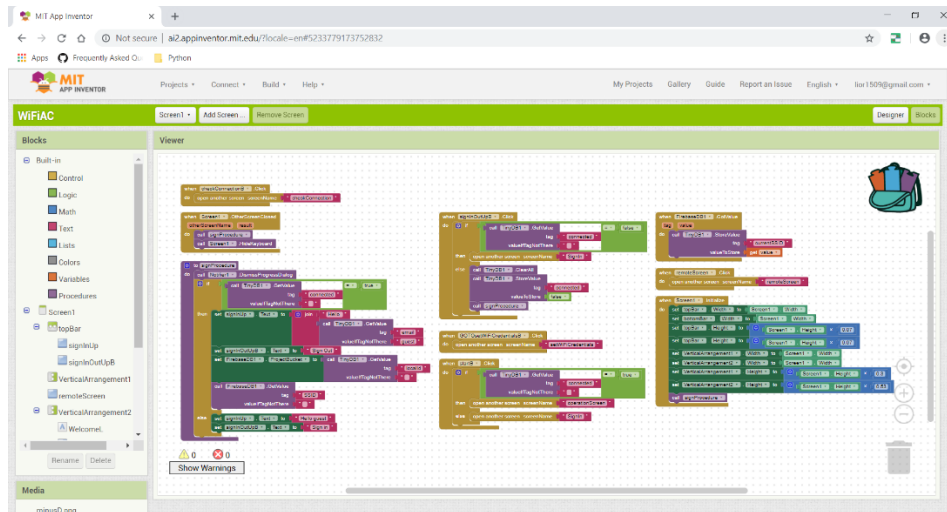


Figure 19 - MIT App Inventor Blocks Screen

5.2 Main Screen

The main screen welcomes the user to the app. It contains 3 buttons:

1. **Start:** start using the app.
2. **Set WiFi Credentials:** on first use or after reset, this button leads to a screen in which the user can set new WiFi credentials.
3. **Check Connection:** used to check if the ESP connected to the WiFi hotspot. This screen is for debugging purposes only.

Next Screen: Operation screen

Previous Screen: NA

5.2.1 Set WiFi Credentials Screen

As mentioned before, this screen is responsible for setting the WiFi SSID and password. In order to protect the user's credentials, the app will not save the data to Firebase which means every time the credentials are reset the user will need to enter them again.

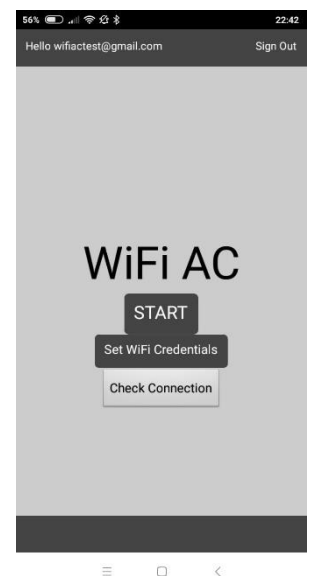


Figure 20 - Main Screen

5.3 Sign In/Up Screen

This screen is available only when the user didn't authenticate yet or never signed up. Signing in/up is available through Firebase authentication system. When a new user signing up, his mail and password are saved in the database storage. Then, when signing in, the mail and password provided by the user are checked in the Firebase db to see if the user exists. On later versions, I plan to add Facebook and Twitter signing options, also using Firebase.

Every user gets his own UID to help the Firebase authenticate him and send his own data while preventing data leaks.



Figure 21 - Sign in/up Screen

5.4 Operation Screen

After clicking on Start on main screen or after signing in/up, the app automatically navigates to Operation screen. In this screen, the user can add a new AC and record codes to it, clear all ACs exists on database, or choose an existing AC to control. The app automatically grabs all the user's ACs from Firebase database using the user UID.

5.4.1 Add New AC Screen

In this screen, as the name suggests, the user can add new AC code to the Firebase database and later record codes for it. As can be seen in figure #, the user need to enter the requested AC name and the AC's brand (this is optional). Each AC has its own WiFi-AC unit to operate it. The user also need to enter the unit address so that the main unit can communicate with it. In later version I plan to make this procedure by scanning a QR code.

If the user has selected an AC from the list, the app will open a new screen, called Remote Screen.

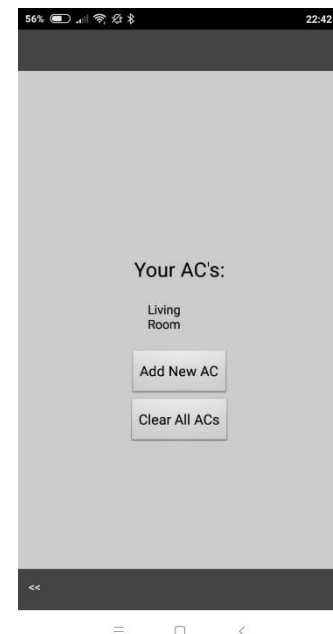


Figure 22 - Operation Screen

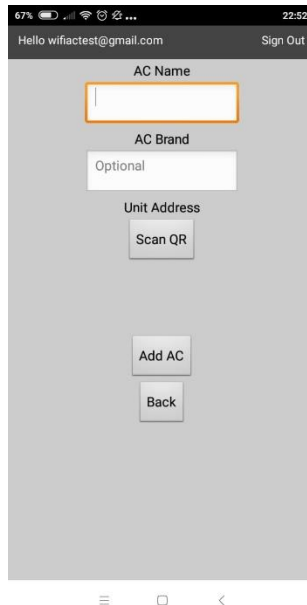


Figure 23 - Add New AC Screen

5.5 Remote Control Screen

The remote control screen is the core of the app. In this screen, the user actually sends orders to it's ACs. Also, in this screen the user record his AC's codes. The screen design looks like a remote control to make the operation easier. With Recording Mode on, every click automatically start a sequence of recording a new code to make it easier and faster to record new codes.

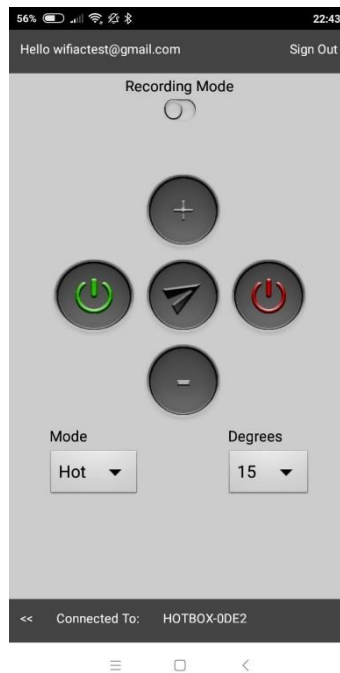


Figure 24 - Remote Control Screen

6. Arduino & ESP

6.1 What is Arduino?

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike.

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their needs. The software, too, is open-source, and it is growing through the contributions of users worldwide.



Figure 25 – Arduino Uno

Thanks to its simple and accessible user experience, Arduino has been used in thousands of different projects and applications. The Arduino software is easy-to-use for beginners, yet flexible enough for advanced users. It runs on Mac, Windows, and Linux. Teachers and students use it to build low cost scientific instruments, to prove chemistry and physics principles, or to get started with programming and robotics. Designers and architects build interactive prototypes, musicians and artists use it for installations and to experiment with new musical instruments. Makers, of course, use it to build many of the projects exhibited at the Maker Faire, for example. Arduino is a key tool to learn new things. Anyone - children, hobbyists, artists, programmers - can start tinkering just following the step by step instructions of a kit, or sharing ideas online with other members of the Arduino community.

6.2 Arduino Nano

The Arduino Nano is a small, complete and breadboard-friendly board based on the Atmega328. It is almost similar to the Arduino Uno, but lacks a DC power jack. The Arduino Nano is powered using a Mini-B USB cable.

The Arduino Nano is perfect for small projects and prototypes. Its tech specs are shown below.

Table 1 – Arduino Nano Summary

Microcontroller	ATmega328
Architecture	AVR
Operating Voltage	5 V
Flash Memory	32 KB of which 2 KB used by bootloader
SRAM	2 KB
Clock Speed	16 MHz
Analog IN Pins	8
EEPROM	1 KB
DC Current per I/O Pins	40 mA (I/O Pins)
Input Voltage	7-12 V
Digital I/O Pins	22 (6 of which are PWM)
PWM Output	6
Power Consumption	19 mA

The Arduino Nano's pinout diagram is also display below. As can be seen by the diagram, pin D3 is a PWM pin and can be used for generating the IR code by the IR LED.

The NRF24L01 communicates with the Nano's microcontroller using SPI. Therefore, the pins used for it are MOSI, MISO, SCK and SS (Slave Select). In addition, there is the CE (Chip Enable) pin to enable the module when the system wants to send a package. The pins used in this project are:

- CE – D9
- SS – D10
- MOSI – D11
- MISO – D12
- SCK – D13

In addition, like every other electrical element, the NRF24L01 needs power supply to work. by looking in the NRF24L01 datasheet, we can see the power supply range to make sure we don't burn the chip. The datasheet determines that the range is 1.9 to 3.6 Volts. Therefore, we connect it to the 3.3 Volts pin of the Arduino Nano.

For GND, we can select any GND pin of the Arduino Nano. It doesn't really matter because when designing the PCB board, we make a ground plane for all the circuit.

In chapter 9 below ([The Board](#)) one can see the complete board and all the wiring.

Single chip 2.4 GHz Transceiver **nRF24L01**

FEATURES

- True single chip GFSK transceiver
- Complete OSI Link Layer in hardware
- Enhanced ShockBurst™
- Auto ACK & retransmit
- Address and CRC computation
- On the air data rate 1 or 2Mbps
- Digital interface (SPI) speed 0-8 Mbps
- 125 RF channel operation
- Short switching time enable frequency hopping
- Fully RF compatible with nRF24XX
- 5V tolerant signal input pads
- 20-pin package (QFN20 4x4mm)
- Uses ultra low cost +/- 60 ppm crystal
- Uses low cost chip inductors and 2-layer PCB
- Power supply range: 1.9 to 3.6 V

APPLICATIONS

- Wireless mouse, keyboard, joystick
- Keyless entry
- Wireless data communication
- Alarm and security systems
- Home automation
- Surveillance
- Automotive
- Telemetry
- Intelligent sports equipment
- Industrial sensors
- Toys

Figure 26 - NRF24L01 Datasheet

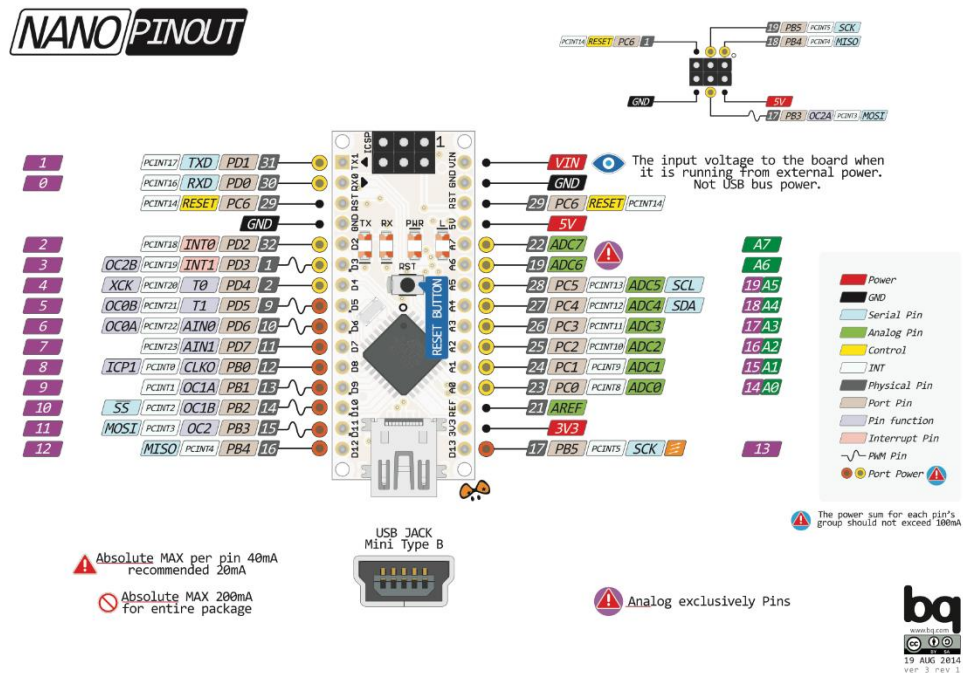




Figure 28 - NodeMCU Module

The NodeMCU v3 device features a 4 MB (32 Mb) flash memory organized in sectors of 4k each. The flash memory address starts at 0x40200000 and can be read and written from a Zerynth program using the internal flash module.

Table 2 - NodeMCU Summary

Microcontroller	Tensilica 32-bit RISC CPU Xtensa LX106
Operating Voltage	3.3 V
Input Voltage	7-12 V
Digital I/O Pins (DIO)	16
Analog Input Pins (ADC)	1
UART	1
SPI	1
I2C	1
Flash Memory	4 MB
SRAM	64 KB
Clock Speed	80 MHz
WiFi	IEEE 802.11 b/g/n: <ul style="list-style-type: none"> • Integrated TR switch, balun, LNA, power amplifier and matching network • WEP or WPA/WPA2 authentication, or open networks

The NodeMCU pinout diagram is display below. This circuit also contains the NRF24L01 and therefore using SPI pins. In addition, unlike the Nano's circuit, in this board there is also the TSOP module to record the codes. This module requires another DIO pin to receive the data.

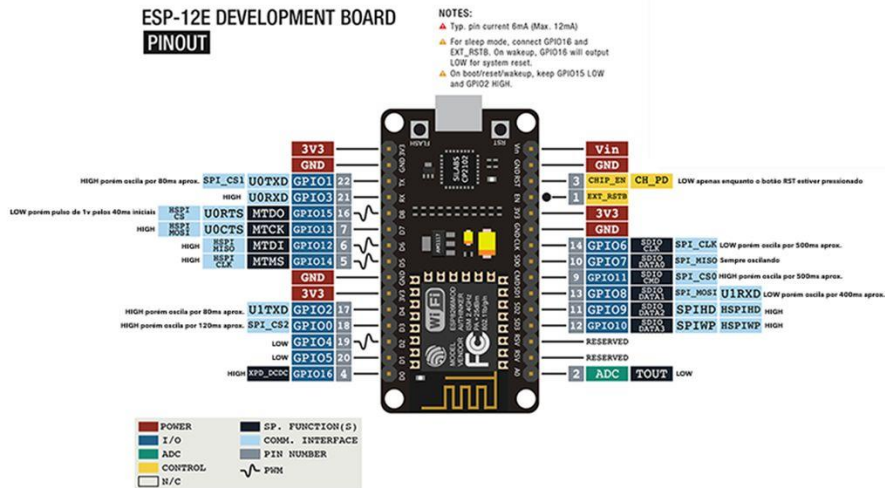


Figure 29 – NodeMCU Pinout

The SPI pins on the NodeMCU board are the following:

- SS – D3
- CE – D4
- SCK – D5
- MISO – D6
- MOSI – D7

The TSOP data pin on the NodeMCU is pin D2. Because the supply voltage range of the TSOP is 2.5 – 5.5 Volts, we can connect it directly to the DIO pin. The TSOP4838 datasheet is display in figure # below:

VISHAY www.vishay.com **TSOP22..., TSOP24..., TSOP48..., TSOP44...** Vishay Semiconductors

IR Receiver Modules for Remote Control Systems

DESIGN SUPPORT TOOLS [click logo to get started](#)

MECHANICAL DATA
Pinning for TSOP44..., TSOP48...:
1 = OUT, 2 = GND, 3 = V_S

FEATURES

- Improved immunity against HF and RF noise
- Low supply current
- Photo detector and preamplifier in one package
- Internal filter for PCM frequency
- Supply voltage: 2.5 V to 5.5 V
- Improved immunity against optical noise
- Insensitive to supply voltage ripple and noise
- Material categorization: for definitions of compliance please see www.vishay.com/doc?99912

DESCRIPTION

The TSOP22..., TSOP48..., TSOP24.. and TSOP44.. series are miniaturized IR receiver modules for infrared remote control systems. A PIN diode and a preamplifier are assembled on lead frame, the epoxy package contains an IR filter.

The demodulated output signal can be directly connected to

RoHS COMPLIANT
HALOGEN FREE
GREEN
(Pb-free)

Figure 30 - TSOP 4838 Datasheet

To make things easier to understand, I also add here the pinout of the NRF24L01 module. The datasheet can be seen in figure # above. Then again, all the connection can be seen in Chapter 9 below.

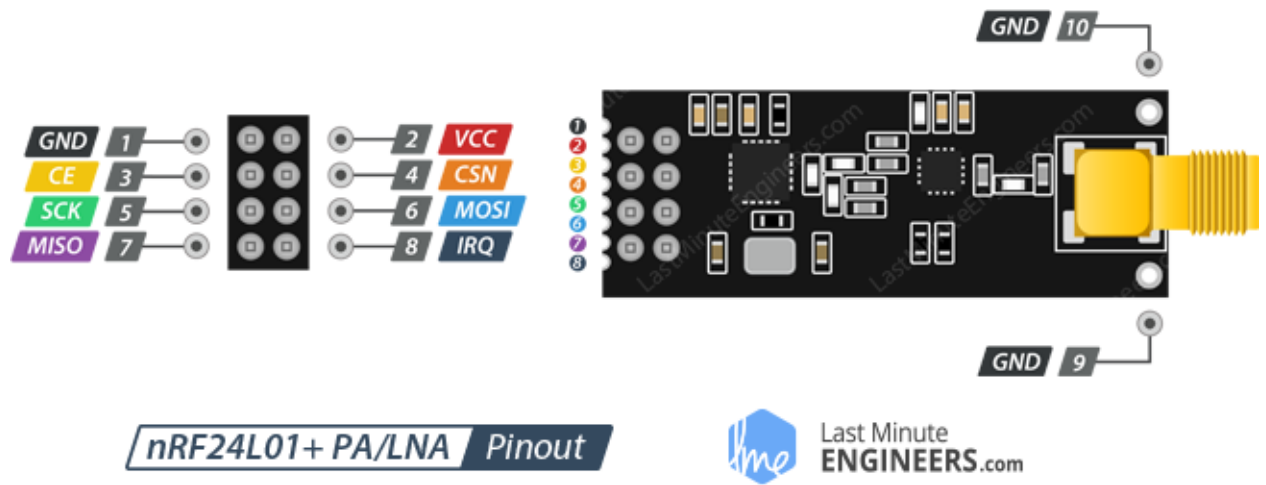


Figure 31 - NRF24L01 Pinout

7 3D Printing

7.1 What is 3D Printing?

3D printing or additive manufacturing is a process of making three dimensional solid objects from a digital file.

The creation of a 3D printed object is achieved using additive processes. In an additive process an object is created by laying down successive layers of material until the object is created. Each of these layers can be seen as a thinly sliced horizontal cross-section of the eventual object.

3D printing is the opposite of subtractive manufacturing which is cutting out / hollowing out a piece of metal or plastic with for instance a milling machine.

3D printing enables you to produce complex (functional) shapes using less material than traditional manufacturing methods.

7.2 Anet A8

The printer used in this project is Anet A8. This is a low-cost 3d printer with a wide community and support. This printer was bought especially for this project because of its easy-to-use interface and its high quality printing.

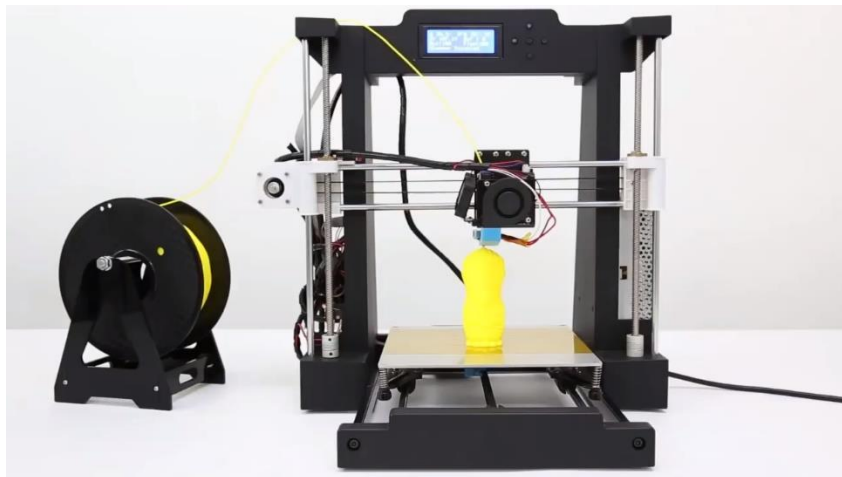


Figure 32 - The Anet A8 Printer

7.3 3D Printing in WiFi-AC Project

In my project the envelope of the system is crucial. No one wants an ugly “thing” on he’s wall. It is almost impossible to find the right envelope that fits to the dimensions of each units of the system and also offer a wide variety of colors and shapes. The best way to get exactly what I want to my project is to build it myself and then print in using a 3d printer. That way, I can be much more flexible with the envelope dimensions and shapes and can use whatever micro controller I want without worrying about its size. To build an envelope I used SolidWorks, while printing done using the Anet A8 3d printer.

7.3.1 Main Board Package

The main usually be at the living room so its package must be designed carefully. At the future, I plan to have a variety of designs for the package, but for now I designed one to test the system.

The software is used is SolidWorks and the package were printed using the Anet A8 3d printer.

7.3.2 The Transmitter Package

The transmitter package is a little bit different from the main board package because of the IR LED. We have to make sure the LED is visible so that the AC's receiver can get the code and operate the AC.

8 The Code

8.1 Motivation

The code that would “tell” the Arduino processor what to do and should cover all possible options. Therefore, the code must be long and complex. The motivation is to make it as simple as possible and easy to read and understand. In addition, the code should be flexible to changes and updates.

The code is written in C++ language and uses its functions. In this project I used advanced methods of C++ such as classes, objects, libraries, global variables, arrays and more.

Each unit has its own code. In order to be able to communicate between the various units, I build

8.2 Libraries

The Arduino libraries contain function that help control the different modules. Using of the open-source libraries, a lot of time can be saved instead of writing each module’s code. There are libraries for almost any module that exists, and all of them are open-coded.

The libraries used in this project are:

- ESP8266 libraries – ESP8266WiFi, ESP8266WebServer
- EEPROM
- FirebaseArduino
- SPI
- RF24
- IR libraries – IRsend, IRrecieve

In addition to these libraries, I also created a library for this project to be able to make the main code shorter and easier to understand. This library name is WiFiAC and it includes all the functions I’ve built and all the parameters I want to initialize when the module is turned on.

```
#include <ESP8266WiFi.h>           // For the access point
#include <ESP8266WebServer.h>       // For the web server
#include <WiFiClient.h>
#include <EEPROM.h>
#include <WiFiAC.h>
#include <FirebaseArduino.h>
#include <IRsend.h>
#include <RF24Network.h>
#include <RF24.h>
#include <SPI.h>
```

8.3 Objects

The main code is using objects to apply the functions of the libraries for the ESP Web Server, the IR send and receive functions, the WiFiAC library that was built for this project and the RF communication with the other modules.

```
// Objects
ESP8266WebServer server(80);
IRsend irsend(15);
WiFiAC;
RF24 myRadio(2,0); // NodeMCU 20.3.2018
RF24Network network(myRadio);
```

8.4 Main Code Functions

8.5 Setup() function

The Arduino's setup function is a built-in function that is running only once when the Arduino powers up. This function usually contains other functions that need to run when in the module startup such as connecting to WiFi, connecting to Firebase and so on.

First, we want to set the Arduino's baud rate to get and send information to the modules connected to it. To do that, we use the following function:

```
Serial.begin(115200);
```

As can be seen above, the baud rate we're using is 115200 in bits per second.

Next, we initialize the IR object. The way to do this is to use the begin() function of the IR library. Also, we want to set the LED pin to OUTPUT so we can set the LED on and off. We use an Arduino built-in function called pinMode().

```
Irsend.begin();
pinMode(ledPin, OUTPUT);
```

The next thing we want to do is try to connect to WiFi. But there is a problem here: we don't want the user to enter his WiFi credentials every time the Arduino starts up. The solution is to save that data to the EEPROM. EEPROM is a memory that is not erasing when the power is down. So, the first thing to do is to check the EEPROM for credentials data and try to connect. We do that with a function of the WiFiAC library that was written especially for that.

```
wifiac.writeEEPROM(0,32,ssid);
wifiac.writeEEPROM(32,32,pass);
wifiac.writeEEPROM(64,32,uid);
```

If the credentials are true, the next function will connect us to user's WiFi. If not, the user will set his credentials using the smartphone app. The ESP will switch to Access Point mode to allow the user connect to it using the smartphone and set the new data. Once the user enters the new data, the ESP will change its mode back to client mode and try to connect to the user's WiFi again.

```
while(!wifiac.begin(ssid, pass)){ // begin() - try to connect for 15 seconds
  Serial.println("Configuring Access Point...");
  Serial.println("setWiFi START");
```

```

WiFi.softAP("WiFi-AC", "12345678");
IPAddress apIP = WiFi.softAPIP();           // Get access point IP
Serial.print("Access Point IP is: "); Serial.println(apIP); // and print it to screen
setWiFiCredentials();
}

```

Next, we initialize the Firebase, RF and SPI objects to enable communication between modules and to database. We will get a notification to know if the NRF module has connected successfully so it can communicate with the other NRF around.

```

Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
// NRF Configurations
if(myRadio.begin())
  Serial.println("NRF ON");
else
  Serial.println("OFF");
SPI.begin();
network.begin(/*channel*/ 90, /*node address*/ this_node);

```

8.6 Loop() Function

The loop function, as it is called, run in loops as long as the ESP is on. Therefore, in this function we command the micro controller to sample the Firebase DB to see if new orders arrived. This function also contain the core code of the system. Its structure is a switch loop that tells the program what to do according to the operation number received from the Firebase DB. The options are: Record new code (Operation = 1), Get code from Firebase and send it to AC (Operation = 2) and reset WiFi credentials (Operation = 3). Most of the code's functions are from the WiFiAC library.

8.6.1 Case 1: Get a code from Firebase and send it to AC

This case is the main operation of the system. We need to get the code from Firebase DB, make some manipulation on it and send it to AC using the IR LED.

The DEBUG comments shows the lines of code that are only for the developer to see the operation step by step.

The flow is:

1. Removing the old code from memory
2. Get the path to go inside the Firebase DB
3. Get the length of the code using the countElements() function from the WiFiAC library
4. Converting the code's string to uint8 array to be able to send it to the other module using the NRF24L01 module. This module is limited in the quantity of bits it can send in each package, so we have to divide the code to segments
5. Setting the NRF24L01 module to send mode and then, using a while loop and a counter, sending the code segment by segment to the other module. Each package should wait to get acknowledge from the other module before the next package is sent.

6. After the delivery ends, reset the path to prepare for another delivery.

The code is listed below:

case 1:

```
Serial.println("Get code from Firebase and send it to AC");
```

```
code.remove(0);
```

```
path = uid;
```

```
// DEBUG
```

```
Serial.print("code: "); Serial.println(code);
```

```
path.concat("/AC");
```

```
AC = Firebase.getString(path);
```

```
// DEBUG
```

```
Serial.print("AC: "); Serial.println(AC);
```

```
AC = AC.substring(1, AC.length()-1);
```

```
// DEBUG
```

```
Serial.print("AC: "); Serial.println(AC);
```

```
path = uid;
```

```
path.concat("/target");
```

```
target = Firebase.getString(path);
```

```
// DEBUG
```

```
Serial.print("target1: "); Serial.println(target);
```

```
target = target.substring(1, target.length()-1);
```

```
target.replace('\\', '/');
```

```
// DEBUG
```

```
Serial.print("target2: "); Serial.println(target);
```

```
path = uid;
```

```
path.concat("/ACs/" + AC + "/Codes/" + target);
```

```
// DEBUG
```

```
AC = Firebase.getString(path);
```

```
// DEBUG
```

```
Serial.print("path: "); Serial.println(path);
```

```
code = Firebase.getString(path);
```

```
// DEBUG
```

```
Serial.print("code: "); Serial.println(code);
```

```
len = wifiac.countElements(code);
```

```
wifiac.stringToIntArray(code, buff);
```

```
wifiac.uint16To8(buff, buff8, len);
```

```
// NRF
```

```
radio.stopListening(); // First, stop listening so we can talk.
```

```
counter = (len/16)+1;
```

```
Serial.print("counter: "); Serial.println(counter);
```

```
while (j <= len){
```

```
  Serial.print("j: "); Serial.println(j);
```

```
  if(flag){
```

```
    sendBuff[0] = counter;
```

```
    flag = false;
```

```
  }
```

```
  else{
```

```
    for(int x=0; x<16; x++,j++){
```

```
      sendBuff[x] = buff8[j];
```

```
    }
```

```
    packSize = 16;
```

```
  }
```

```
  if ( radio.write(&sendBuff, packSize) ) { // Send the counter variable to the  
other radio
```

```
    for(int z=0; z<16; z++){
```

```
      Serial.print(sendBuff[z]); Serial.print(" ");
```

```
    }
```

```
    if (!radio.available()) { // If nothing in the buffer, we got an ack but it is  
blank
```

```
      Serial.print(F("Got blank response. round-trip delay: "));
```

```
    } else {
```

```
      while (radio.available() ) { // If an ack with payload was received
```

```
        radio.read( &gotByte, 1 ); // Read it, and display the response time
```

```
        unsigned long timer = micros();
```

```
        Serial.print(F("Got response "));
```

```
        Serial.print(gotByte);
```

```
        Serial.println(F(" microseconds"));
```

```
        counter++; // Increment the counter variable
```

```
      }
```

```
    }
```

```
  } else {
```

```
    Serial.println(F("Sending failed.)); // If no ack response, sending failed
```

```
    j-=16;
```

```

    }

    delay(100);
}

resetPath();
break;

```

8.6.2 Case 2: Record New Code

In this case, we are recording a new code and send it to Firebase DB.

As before, the DEBUG comments are used only for debugging by the developer.

The flow is:

1. Removing the old code from memory
2. Get the path to go inside the Firebase DB
3. Using the receiveCode() function from WiFiAC library, we set the micro-controller to read mode by using the IR receiver
4. If the code is not blank, we upload it to Firebase DB using the setString() function from Firebase library
5. After the upload ends, reset the path to prepare for another actions.

The code is listed below:

case 2:

```

Serial.println("Record code and send it to Firebase");
code.remove(0);
path = uid;

path.concat("/AC");
AC = Firebase.getString(path);

// DEBUG
Serial.print("AC: "); Serial.println(AC);

AC = AC.substring(1, AC.length()-1);

path = uid;
path.concat("/target");
target = Firebase.getString(path);
// DEBUG
//Serial.print("target1: "); Serial.println(target);

target = target.substring(1, target.length()-1);    // Remove quotation marks
target.replace("\\',/");

```

```

// DEBUG
//Serial.print("target2: "); Serial.println(target);

path = uid;
path.concat("/ACs/" + AC + "/Codes/" + target);

// DEBUG
//Serial.print("path: "); Serial.println(path);

code = wifiac.receiveCode();           // The problem is here!!!
if(code != ""){
  Firebase.setString(path, code);
}
delay(50);

resetPath();
break;

```

8.6.3 Case 3: Reset WiFi Credentials

The user's WiFi credentials are saved in EEPROM in order to be able to retrieve after the micro-controller lose power. In some cases, such as changing the system location to other house or after the user's WiFi setting are changed, the user will want to reset the WiFi credentials and set new ones. In order to do so, we need to delete the old credentials and let the user add new ones.

The flow is:

1. Clear EEPROM using clearEEPROM() function from WiFiAC library
2. Set the path to /SSID on Firebase DB to delete the credentials from the db as well
3. Reset path to prepare for new actions

Note: after resetting the credentials, the system should be rebooted so that the user can enter new settings.

The code is listed below:

case 3:

```

Serial.println("Reseting WiFi Credentials");
wifiac.clearEEPROM();
delay(50);
path.remove(0);
path.concat(uid);
path.concat("/SSID");
Firebase.setString(path, "");
delay(50);
resetPath();
delay(50);

```

break;

8.7 The Arduino IDE

The environment in which we use to compile and upload the code to the microcontroller called Arduino IDE. This freeware is incredibly minimalistic and yet provides almost everything for an Arduino-based project. The IDE provides a text editor to write the code. In addition, there is an output window to plot the status of the compilation, how much memory has been used, errors that were found in the program and much more useful data. The IDE also provides a serial monitor screen to communicate with the microcontroller and a serial plotter screen to give the user visualizations of variables in real-time.

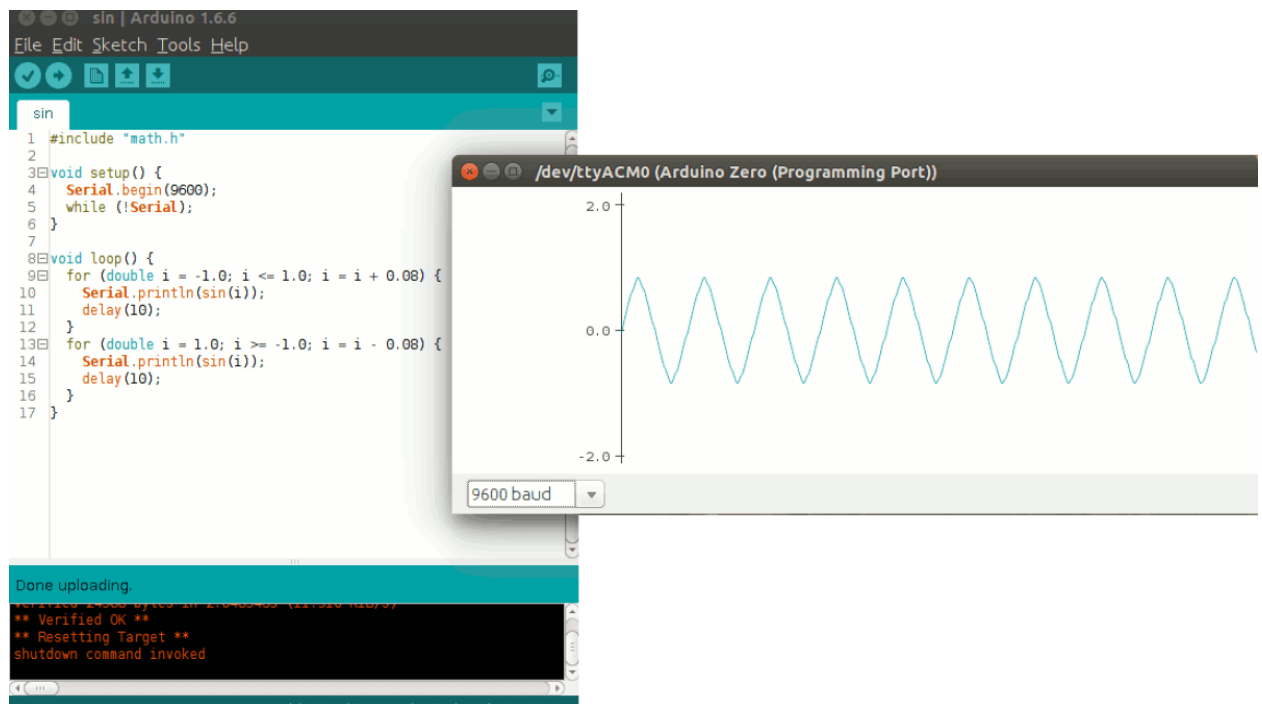


Figure 33 - Arduino IDE

Projects made using the Arduino are called sketches, and such sketches are usually written in a cut-down version of C++ (a number of C++ features are not included). Because programming a microcontroller is somewhat different from programming a computer, there are a number of device-specific libraries (e.g., changing pin modes, output data on pins, reading analog values, and timers). This sometimes confuses users who think Arduino is programmed in an “Arduino language.” However, the Arduino is, in fact, programmed in C++. It just uses unique libraries for the device.

While more advanced projects will take advantage of the built-in tools in the IDE, most projects will rely on the six buttons found below the menu bar:

- **Verify:** used to verifying the code without uploading it to the microcontroller

- **Upload:** used to upload the code to the microcontroller. When pressing this button, the IDE also verify and compile the code to check for syntax errors and other language violations
- **Dotted Paper:** used to create a new file
- **Upward arrow:** used to open an existing Arduino project
- **Downward arrow:** used to save the current project
- **Magnification:** used to open the Serial Monitor to communicate with the microcontroller and debug

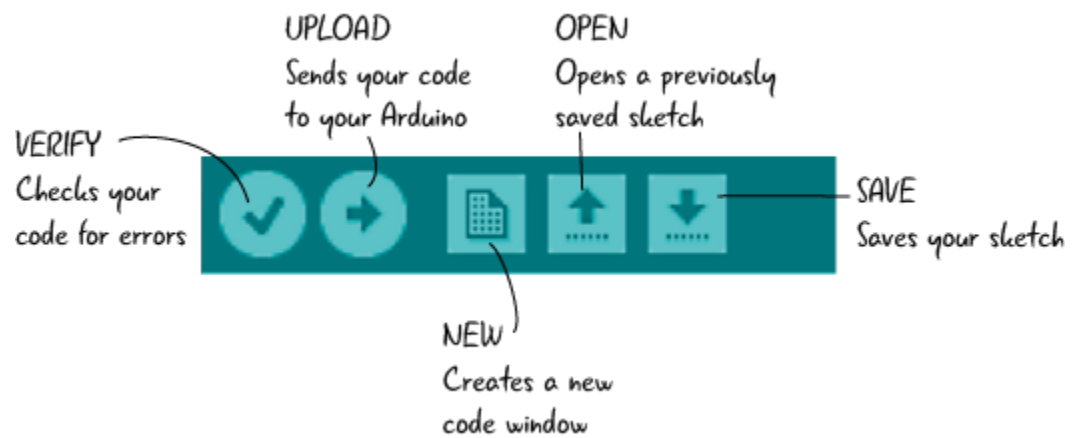


Figure 34 - Arduino's Buttons Panel

9 The Board

9.1 Motivation

PCB is an acronym for printed circuit board. It is a board that has lines and pads that connect various points together. In the picture below (Fig #), there are traces that electrically connect the various connectors and components to each other. A PCB allows signals and power to be routed between physical devices. Solder is the metal that makes the electrical connections between the surface of the PCB and the electronic components. Being metal, solder also serves as a strong mechanical adhesive.

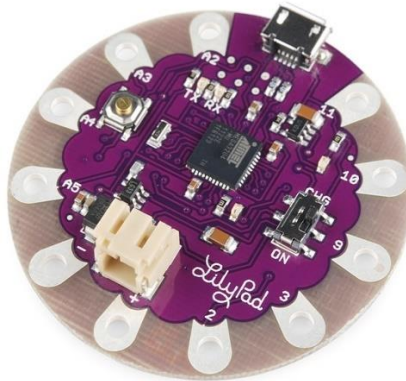


Figure 35 - LilyPad Arduino PCB

9.2 The Eagle

There are various ways to create a PCB, but all started at the design. Many PCB CAD softwares exists, but in this project, I chose to use Eagle. Eagle provides a professional tool for creating all kinds of PCB. In my project, I'm going to create a double-sided PCB with two layers. This way, I can bypass areas where two conductors meet by moving to the second layer. An example is shown below in fig #. The red trace is on the top layer and the blue is on the bottom. We can see that two conductors can go to the same direction without interfere with each other.

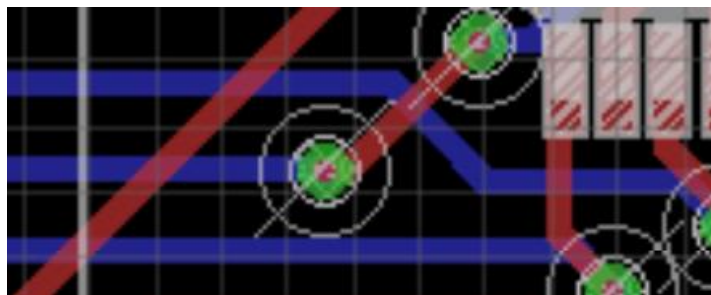


Figure 36 - Traces on both layers

9.3 Main Board

The main board contains the ESP module, NRF module, an LED and the IR Receiver. Like any other boards, it also contains resistor and capacitors. The board schema is show in figure # below.

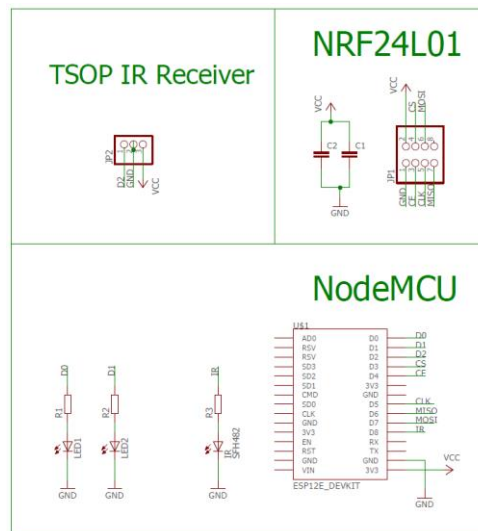


Figure 37 - Main Board Schema

After creating the schema of the board, the next step is to wire everything up, and that's the challenging part! We have to make sure we don't short any wire while keeping the board as simple as possible to solder and to understand. For that, it's almost impossible to user 1-layer pcb design, so we're going to use double-layer pcb design. The complete board after wiring everything is shown below in figure #.

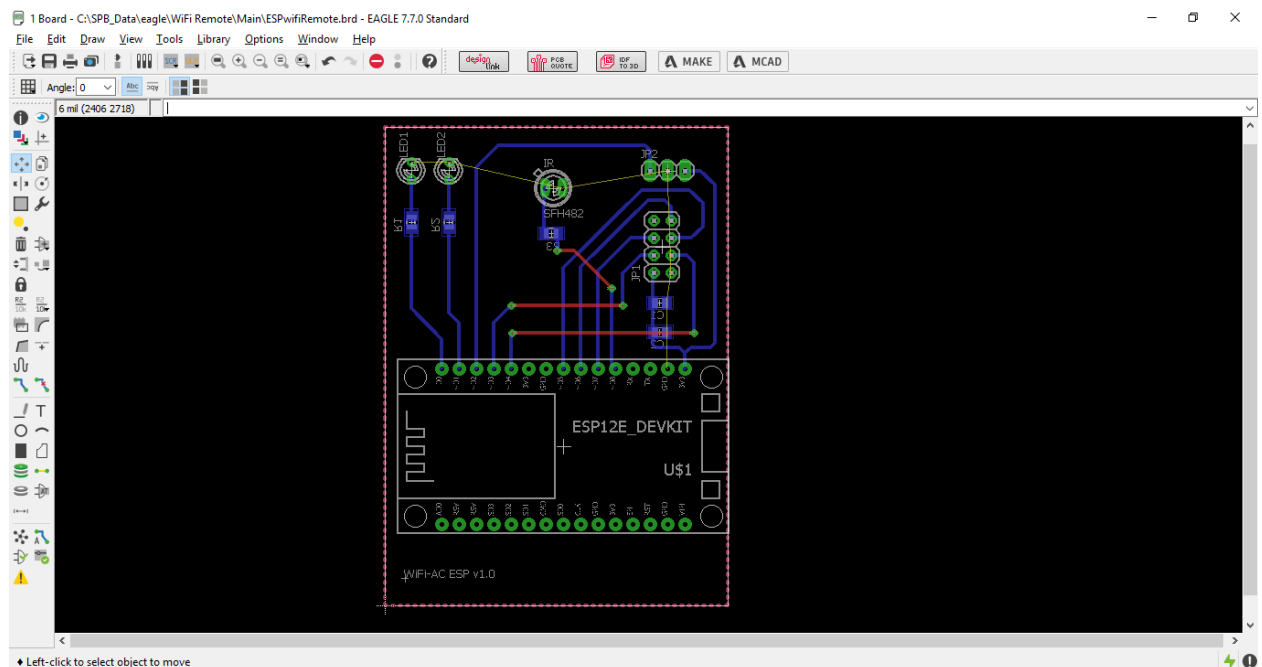


Figure 38 - Main Board PCB Layout

After the board is ready, there is still one thing to do – solder all the elements to it. In figures 35 and 36 below we can see the complete board – up and down.

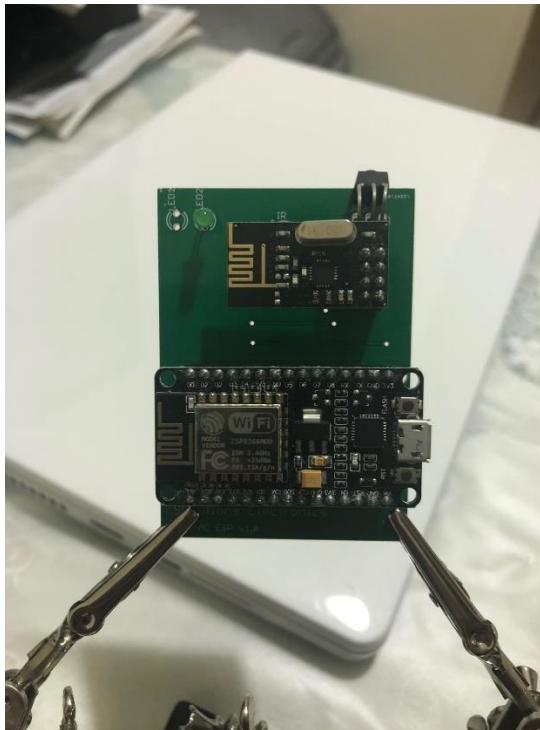


Figure 39 – Main Board Upper Layer

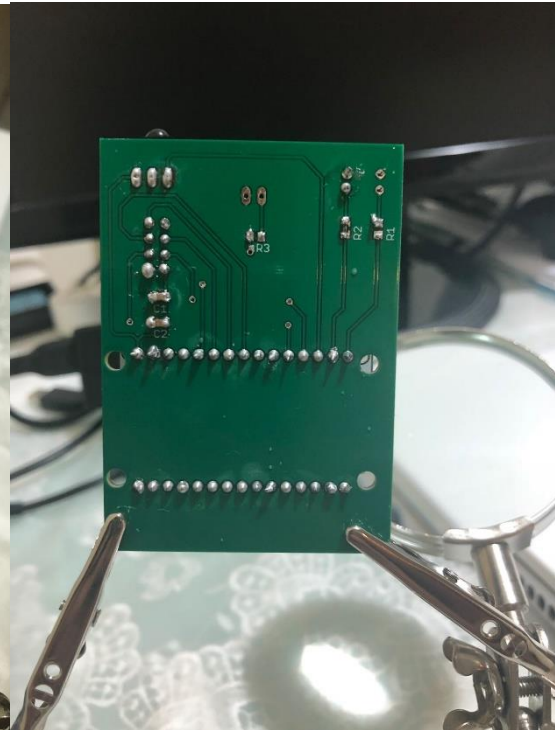


Figure 40 – Main Board Lower Layer

9.4 Transmitter Board

The transmitter board contains an IR LED, NRF module to communicate with the main board and a micro controller to operate the system. The microcontroller used is an Arduino Nano because of its small size and low power consumption. The board will be powered using a battery to not be depended on electrical sockets and allow flexibility when choosing a place to install the unit.

The schema of the transmitter board is show below in figure #:

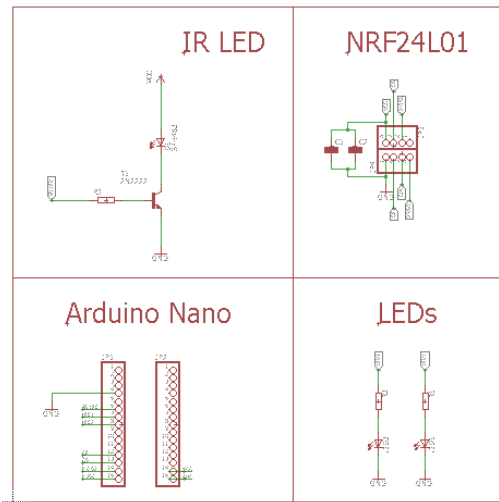


Figure 41 – Transmitter Board Schema

The complete board after wiring is shown below:

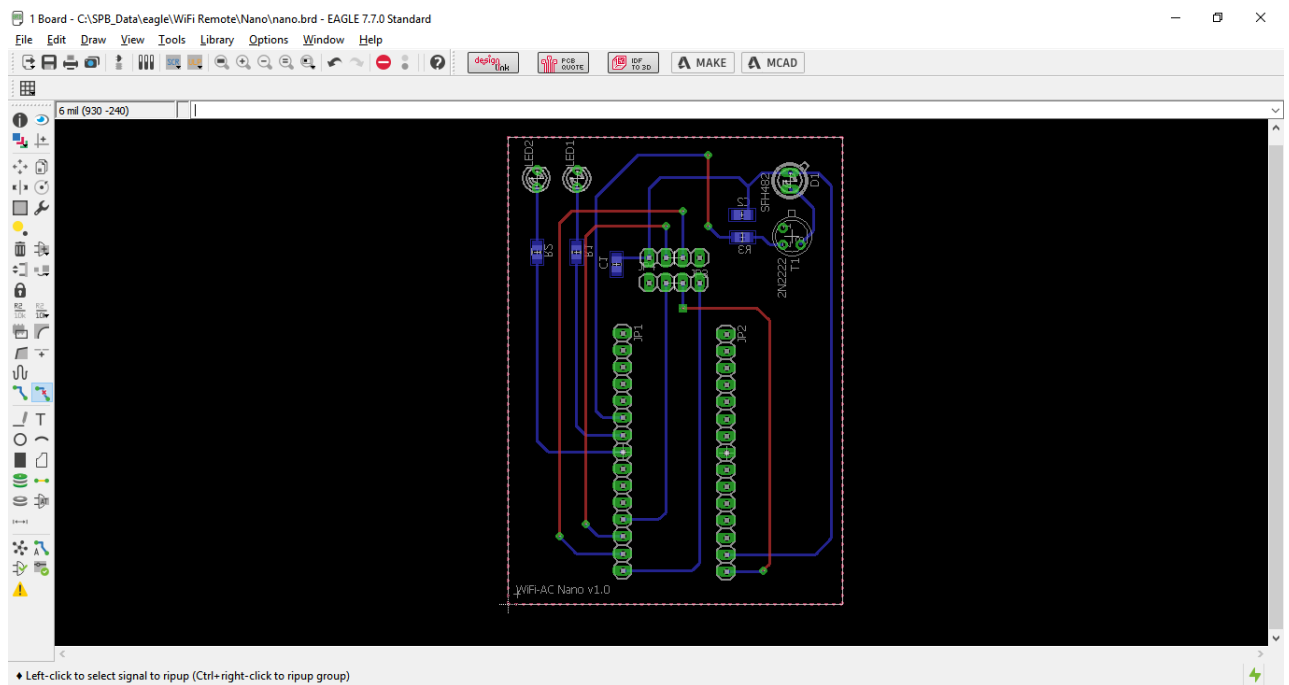


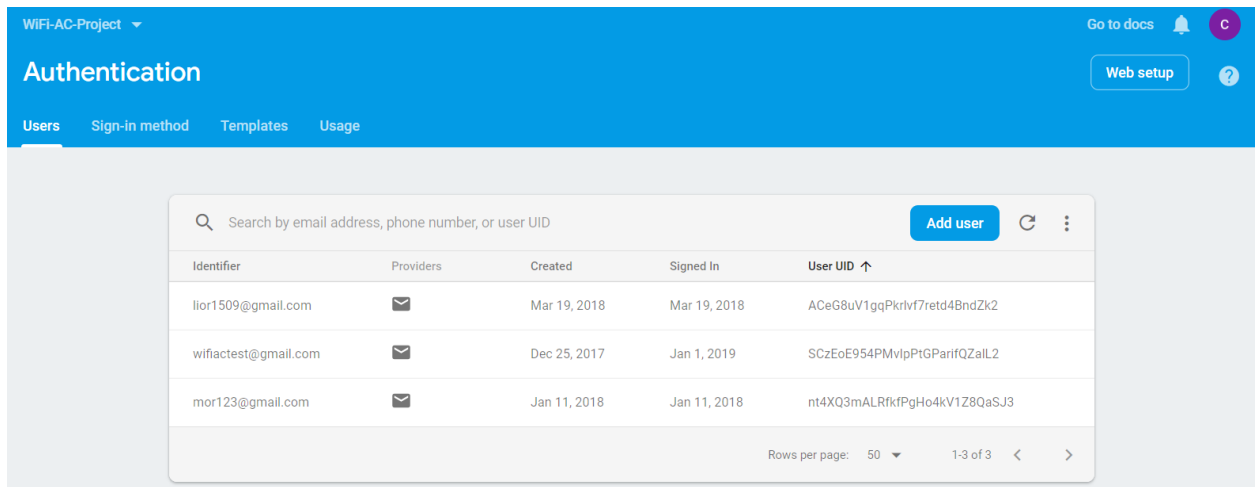
Figure 42 – Transmitter Board PCB

10 Tests

10.1 Testing the system

10.1.1 Creating an account

Using the smartphone app, the user can create his own account. Creating an account will generate a unique UID for identifying the user. The users list can be found and manage using the Firebase web interface.



Identifier	Providers	Created	Signed In	User UID ↑
llor1509@gmail.com	📧	Mar 19, 2018	Mar 19, 2018	ACeG8uV1gqPkriv7ret4BndZk2
wifactest@gmail.com	📧	Dec 25, 2017	Jan 1, 2019	SCzEoE954PMvlpPtGParifQZail2
mor123@gmail.com	📧	Jan 11, 2018	Jan 11, 2018	nt4XQ3mALRfkfPgHo4kV1Z8QaSJ3

Figure 43 - Firebase Users list

In the figure below we can see the users list. On the right side we can see the UID for each user. In addition, we can see when the user account was created and when the user last signed in.

10.1.2 Creating a new AC

After creating an account, the next step is to create a new AC. This is also done by the smartphone app, on the *Add New AC* screen. When a new AC is created, a new tree is also created on the Firebase database under the ACs tree. Under that tree all the AC codes are saved.

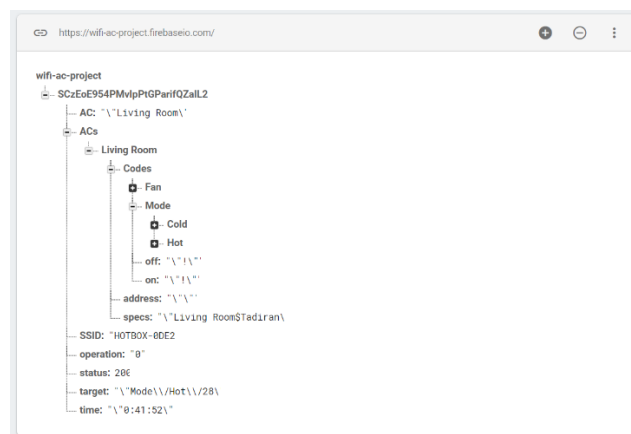


Figure 44 - Adding a new AC

10.1.3 Record new codes

Now it's time to record the AC's codes. To do so, the user enter the *Remote Control* screen. Each code is saved under the relevant settings. The *operation* field on the main tree will change to "2" to notify the microcontroller about the requested operation. In the figure below we can see the database fields changes. The orange fields are the ones which change or contain a field that is changing.

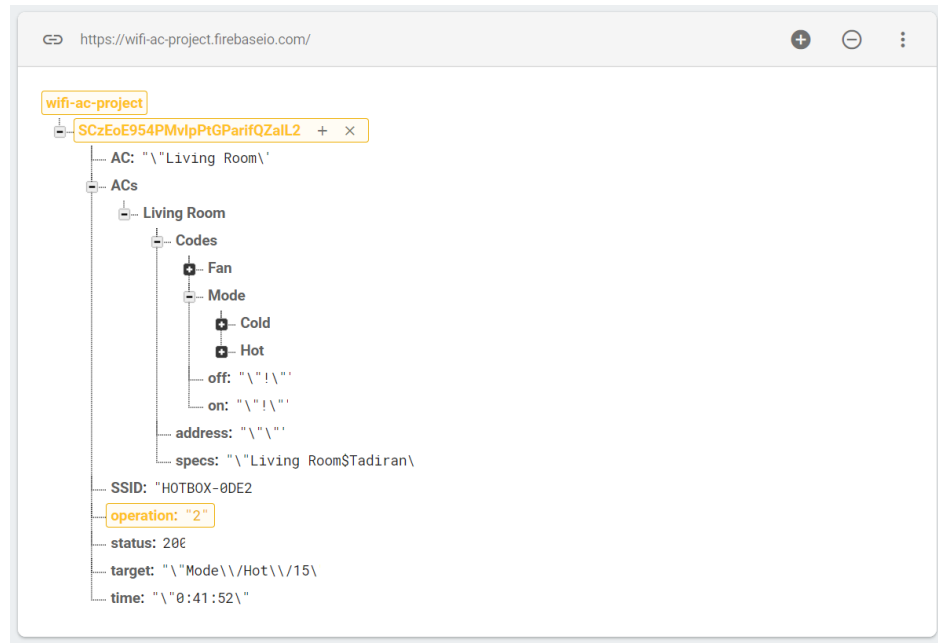


Figure 45 - Add a code to Firebase DB

To see the microcontroller working we need to connect it to the PC and open the Arduino IDE Serial Monitor (figure #).

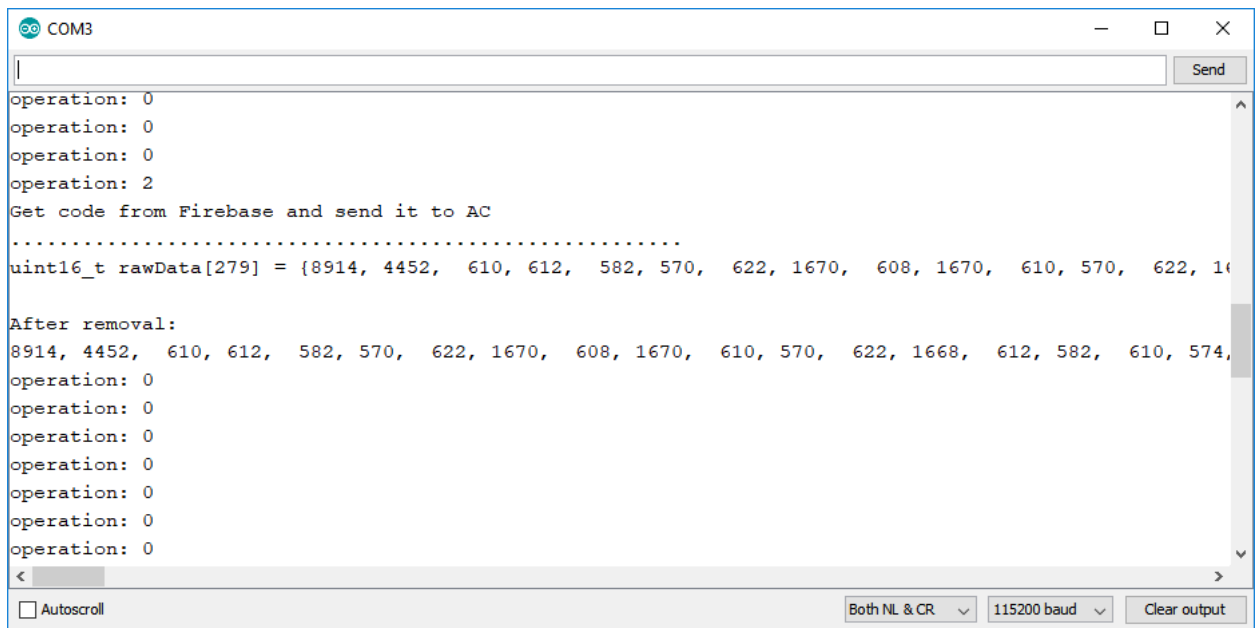


Figure 46 - Arduino IDE Serial Monitor

In the figure above we can see that the operation code changed to 2. After that, the microcontroller is waiting for a new code to arrive. When the code is recorded, the microcontroller will store it in the Firebase database at the relevant field.

10.1.4 Sending the code

Sending the code is divided into 2 parts: (1) send the code to the transmitter unit and (2) send the code to the AC

10.1.4.1 Send the code to the transmitter unit

To send the code from the WiFi unit to the transmitter unit we use the NRF24L01. This module is limited to 16 bits, so the code is divided into parts by the main unit microcontroller. After each segment of code is send, the microcontroller waits for an acknowledge from the transmitter unit to send the next one.

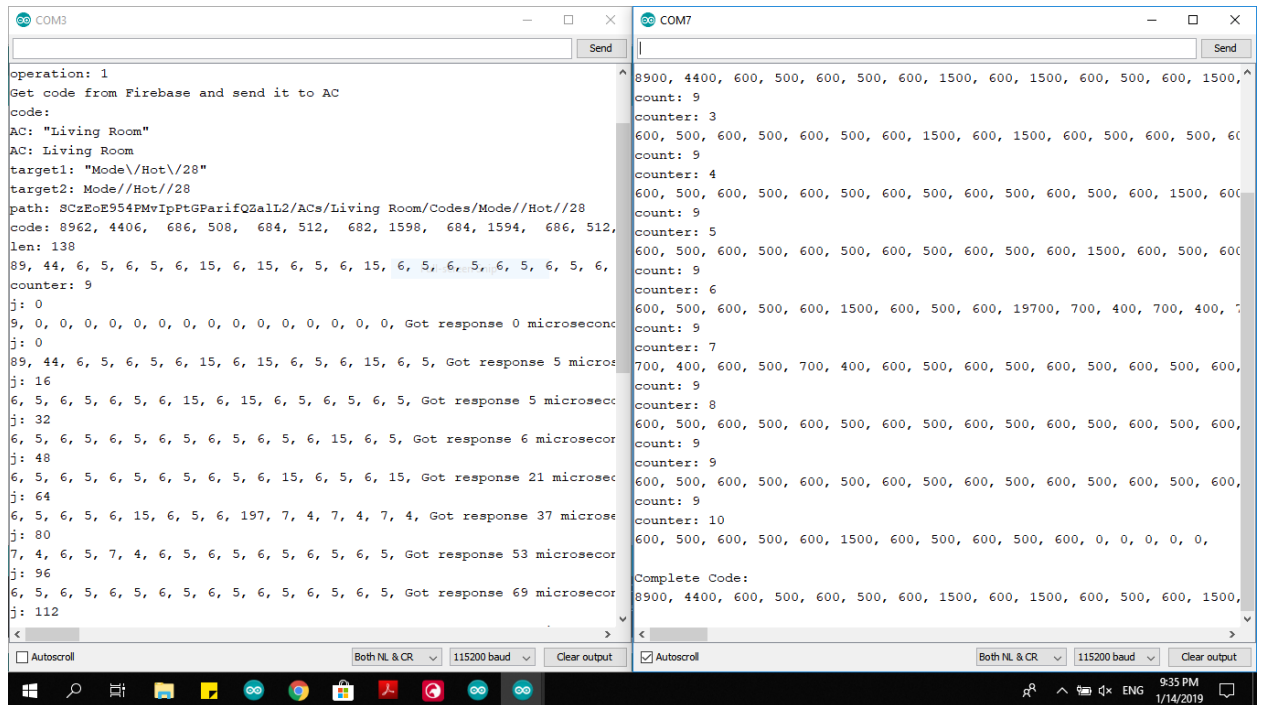


Figure 47 - Both units Serial Monitor

In the figure above, we can see the Serial Monitor of the two units. The left one is the WiFi unit, which sends the AC code to the transmitter unit on the right. As can be seen, the code is normalized and divided by 100 to make it shorter and therefore send it in lower amount of segments.

When the code reaches the transmitter side, it then sends to the AC using the IR LED and the PWM function mentioned before (

10.2 IR LED Range Test

In order to get the system working with an acceptable range, we must make sure there is enough current to drive the IR LED. In order to do that, we use an external power source and a 2n2222 PNP transistor to control it. The circuit is display below in figure #:

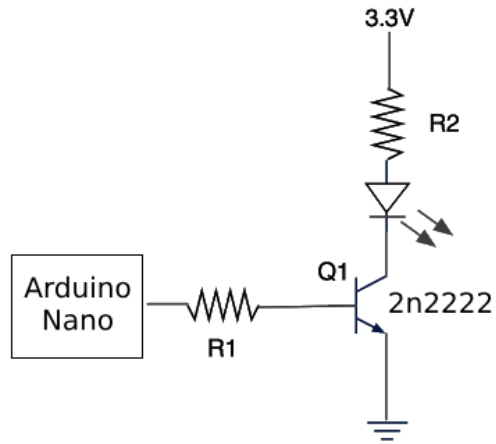


Figure 48 - IR Circuit

Using the transistor, we can control the current flow to the IR LED with the microcontroller I/O pin. This way we can drive much more current from the external source and not use the microcontroller I/O pins to drive the LED. This way we can achieve a larger range, as the microcontroller pins can't output the same amount of current as the external source can.

10.3 RF Range Test

The RF range must be long enough to enable communication between the units. The range is a function of the NRF24L01 parameters and can be changed for each chip. There isn't a defined way to find the best settings and the user should test it until finding the optimal settings.

11 Conclusions

11.1 IoT In Your Home

The WiFi AC project control the home AC by using IR codes. The reason I choose an AC is the complexity of the transmitted data. If we have a system that can handle so much data, turning on and off a switch is a piece of cake. Therefore, after building and operating such a complex system, every element at our home can become smart and connect to the internet. For example, to turn the lights on and off, all we have to do is just send 1 bit. We don't even need a database!

Another reason for choosing the home AC is that everything is wireless. The user does not need to touch the electrical net at his home. It makes the system be suitable for everyone.

11.2 Broker Unit

The main issue of this project is its range. To avoid connecting to the WiFi with too much modules, the code is sent from the main board to the transmitter boards around the building. In some cases, when the distance between the two boards is too big, or if there is a blocking element, such as a concrete wall, between them, the signal might not get to the other side. For that reason I plan to build another board to transfer the signal between them. This board will only contain a microcontroller and NRF24L01 module. When a packet is sent between the boards, there will be 1 bit that will be identify by the Broker unit. If that bit is 0, the Broker unit will not interrupt and nothing will happen. If the bit is set to 1, that means the sending board didn't get his acknowledge and need the Broker unit to help transfer the signal. The Broker unit will just send the package to its destination.

12 References

12.1 Figures List

Figure 1 - IoT Vision	5
Figure 2 - The Light's Spectrum	7
Figure 3 - IR code as a function of volts and time	8
Figure 4 - Modulation Circuit	8
Figure 5 - PWM examples	9
Figure 6 - Transmitter circuit	9
Figure 7 - emitter-follower circuit	10
Figure 8 - IR Receiver Block Diagram	10
Figure 9 - TSOP 4838, an IR receiver	11
Figure 10 - WiFiAC Block Diagram	12
Figure 11 - Firebase Logo	13
Figure 12 - Users Dashboard	14
Figure 13 - Authentication Options	14
Figure 14 - Communication Templates	14
Figure 15 - Usage Graph	15
Figure 16 - WiFiAC Firebase Structure	15
Figure 17 - ACs branches	16
Figure 18 - MIT App Inventor UI Screen	17
Figure 19 - MIT App Inventor Blocks Screen	18
Figure 20 - Main Screen	18
Figure 21 - Sign in/up Screen	19
Figure 22 - Operation Screen	19
Figure 23 - Add New AC Screen	20
Figure 24 - Remote Control Screen	20
Figure 25 - Arduino Uno	21
Figure 26 - NRF24L01 Datasheet	23
Figure 27 - Arduino Nano Pinout	23
Figure 28 - NodeMCU Module	24
Figure 29 - NodeMCU Pinout	25
Figure 30 - TSOP 4838 Datasheet	25
Figure 31 - NRF24L01 Pinout	26
Figure 32 - The Anet A8 Printer	27
Figure 33 - Arduino IDE	36
Figure 34 - Arduino's Buttons Panel	37
Figure 35 - Lilypad Arduino PCB	38
Figure 36 - Traces on both layers	38
Figure 37 - Main Board Schema	39
Figure 38 - Main Board PCB Layout	39
Figure 39 - Main Board Upper Layer Figure 40 - Main Board Lower Layer	40
Figure 41 - Transmitter Board Schema	41
Figure 42 - Transmitter Board PCB	41
Figure 43 - Firebase Users list	42

Figure 44 - Adding a new AC	42
Figure 45 - Add a code to Firebase DB.....	43
Figure 46 - Arduino IDE Serial Monitor.....	44
Figure 47 - Both units Serial Monitor	45
Figure 48 - IR Circuit	46

12.2 Tables List

Table 1 – Arduino Nano Summary	22
Table 2 - NodeMCU Summary.....	24

12.3 References

IoT - https://en.wikipedia.org/wiki/Internet_of_things

Arduino - <https://en.wikipedia.org/wiki/Arduino>

NodeMCU - http://www.nodemcu.com/index_en.html

ESP - <https://www.espressif.com/en/products/hardware/esp8266ex/overview>

IR Communication - <https://learn.sparkfun.com/tutorials/ir-communication/all>

MIT App Inventor - <http://appinventor.mit.edu/explore/about-us.html>

Firebase - <https://firebase.google.com/docs/>

3d Printing - <https://www.3dhubs.com/guides/3d-printing/>

PCB Design – <https://www.build-electronic-circuits.com/pcb-design/>

Eagle - <https://www.autodesk.com/products/eagle/overview>