Beehive weight monitoring

3 year project

teacher:

Paulo Torres

Student:

Fábio Gonçalves

**Summary**

[1 Start steps on ESP8266 configuration 3](#_Toc29427635)

[1.1 Download and installation of the ARDUINO IDE 3](#_Toc29427636)

[Download Arduino IDE 3](#_Toc29427637)

[Installation of Arduino IDE 3](#_Toc29427638)

[1.2 Download and installation of the python 3 4](#_Toc29427639)

[Download Python 4](#_Toc29427640)

[Installation of Python 4](#_Toc29427641)

[1.3 Download and installation of the Drivers for ESP8266 5](#_Toc29427642)

[1.4 Configure IDE to allow programming Node MCU Board 7](#_Toc29427643)

[1.5 Install and configure Raspberry Pi as MQTT broker 10](#_Toc29427644)

[1.6 MQTT protocol 11](#_Toc29427645)

[1.7 DDNS 12](#_Toc29427646)

[1.8 Port forwarding or virtual server 14](#_Toc29427647)

[2 Weight thermal and humidity sensor system 15](#_Toc29427648)

[2.1 Used Materials 15](#_Toc29427649)

[2.2 Schematics 15](#_Toc29427650)

[1. Electrical schematic 15](#_Toc29427651)

[2. 50kg Load Cell 16](#_Toc29427652)

[1. Humidity and Temperature Sensor 17](#_Toc29427653)



# Start steps on ESP8266 configuration

To start programming the ESP8266, it’s essential to do these simple steps:

1. Download and installation of the ARDUINO IDE
2. Download and install python
3. Download and install drivers of ESP8266
4. Configure IDE to allow programming the Node MCU Board
5. Download and install the required Libraries of Node MCU Board
6. Programing Node MCU Board

## Download and installation of the ARDUINO IDE

To download and install the Arduino IDE it’s required access to internet or installation package of software.

Download Arduino IDE

Open web browser and go to this link: <https://www.arduino.cc/en/Main/Software> and download the version adapted to operational system (exist versions for Windows, Linux and Mac Os X).

Install the program using the package downloaded in Arduino.cc Web Site.

Installation of Arduino IDE

After the package download, start the installation, running the installation file on the system.

It will open a window with installation options, select all of them.

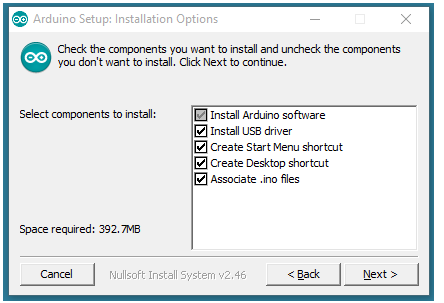


Fig. 1 - Installation Options

After this step, select the file location on destination folder by click on button “Browse …”, or click on install.

The Installation will proceed, and the installer will extract all necessary files to execute the “Arduino Software (IDE)”.

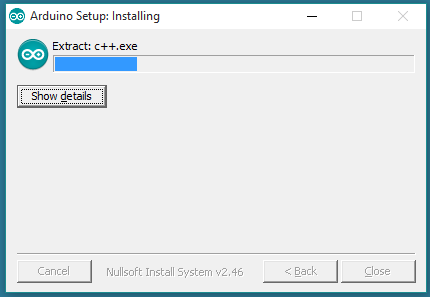


Fig. 2 - Arduino IDE installing

The installation will finish and it’s now possible to start Arduino IDE clicking on shortcut on Desktop or search in windows start menu by “Arduino”.

## Download and installation of the python 3

Download Python

In this part of the project, Python 3 allow to connect the libraries posteriorly downloaded in the next steps to Arduino IDE.

So to download the Python it’s essential to access <https://www.python.org/downloads/> and select the version. This version must be higher than version 3 of python and adapted to operational system.



Fig. 3 - Downloading steps of Python 3.7.1

After the file downloaded, run the executable file (only for Windows) and follow the next steps.

Installation of Python

The executable file will open a window like Fig.4 when it’s needed to select to “Add Python 3.7 to Path” and after that click on “Install Now”. Alternatively, for expert users the possibility to a run a “Customized installation” is allow.

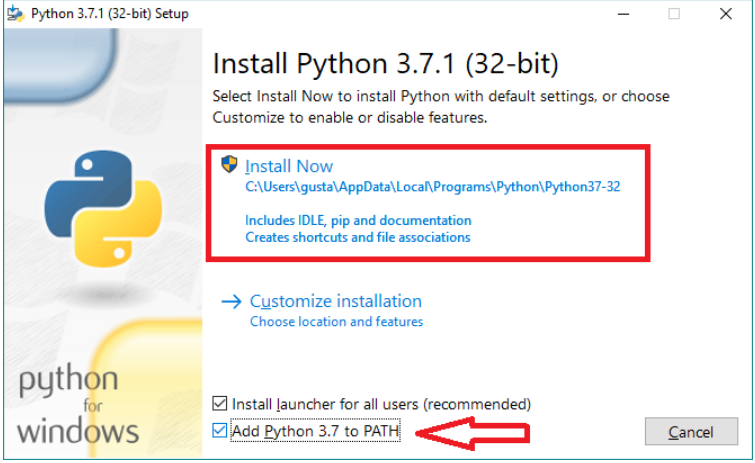


Fig. 4 - First steps to install Python

Wait a few moments, and when the installation was finished the python setup will show a message like that:

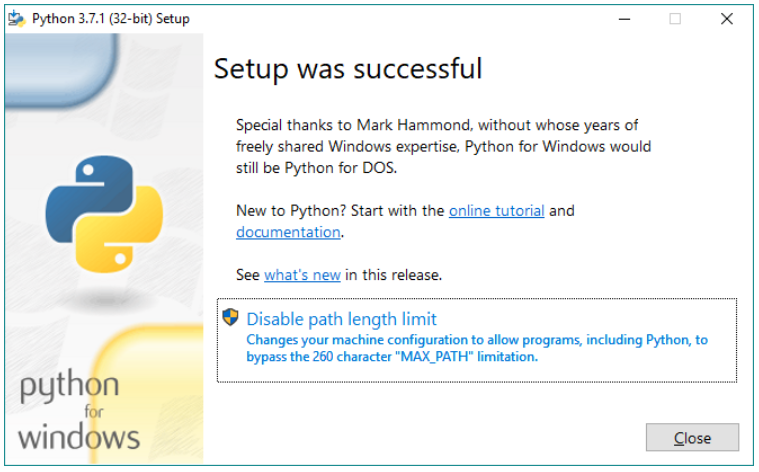


Fig. 5 - Finish message of Python installation

Now the python was successfully installed on machine, for check that, open command prompt by typing “cmd” on windows start menu and type again python. Command prompt will send a message and with the version installed and other specification of the python installation

## Download and installation of the Drivers for ESP8266

Probably in this step, Node MCU Board are not recognized for windows, so it’s necessary download the drivers using an executable file that could be downloaded here: <http://www.wch.cn/download/CH341SER_EXE.html> note that the site could be in a Chinese version.

For an “official” driver installation on Windows 10 it’s need to access the “Device manager” clicking with right button mouse on start menu and select “device manager”, an windows will be open and its possible to see a “trouble” in “other devices”, as you can see on Fig. 6.

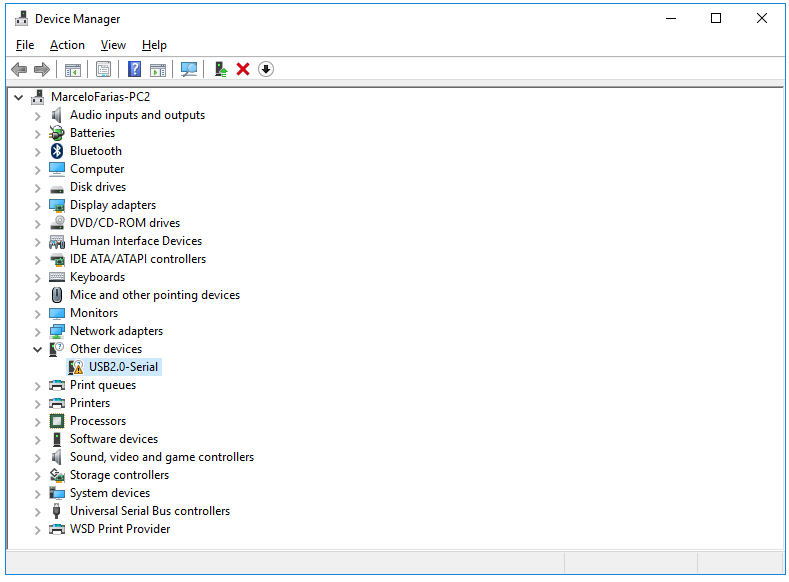


Fig. 6 - Windows Device Manager

By click with the right button on there, it’s possible to “install controllers” and windows will search a specific official driver. Alternatively, it’s possible to update controllers, and windows will search by the last software drivers.

After these steps, the “Device Manager” will update and now it’s possible to see an update windows that contains ports corresponding to Node MCU Board registered as COM & LPT ports, in this example the connected port its COM 21, note that COM Port number is automatically administered by the operation system.

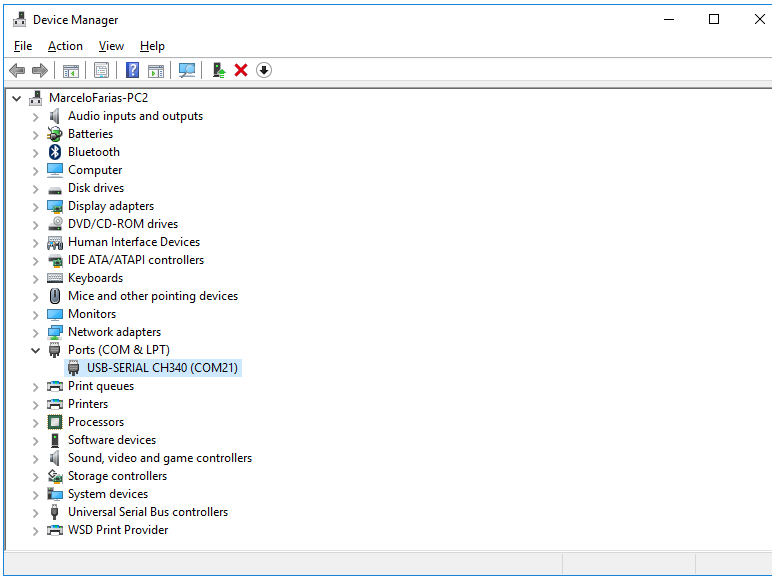


Fig. 7 - Device manager updated with the installed Controllers

## Configure IDE to allow programming Node MCU Board

In this phase is now possible to start the configuration of Arduino IDE to programming Node MCU Board. It means that the configuration of Arduino doesn’t support by default the board corresponding to ESP8266, so it has a lot of configurations to do essentials to programming and access the node MCU board. Note that this configurations steps allow the possibility to access some example files of codes that are not “accessible” for default on Arduino IDE.

In first step launch the Arduino IDE by clicking on IDE shortcut on desktop if it creates or launch by start menu on windows system by typing “Arduino”.

After running of Arduino IDE go to “File”🡪 “Preferences”. It will open the windows of Fig.8 and input this link: <http://arduino.esp8266.com/stable/package_esp8266com_index.json> on “Additional Board Manager URLs” as is showed on Fig.8.

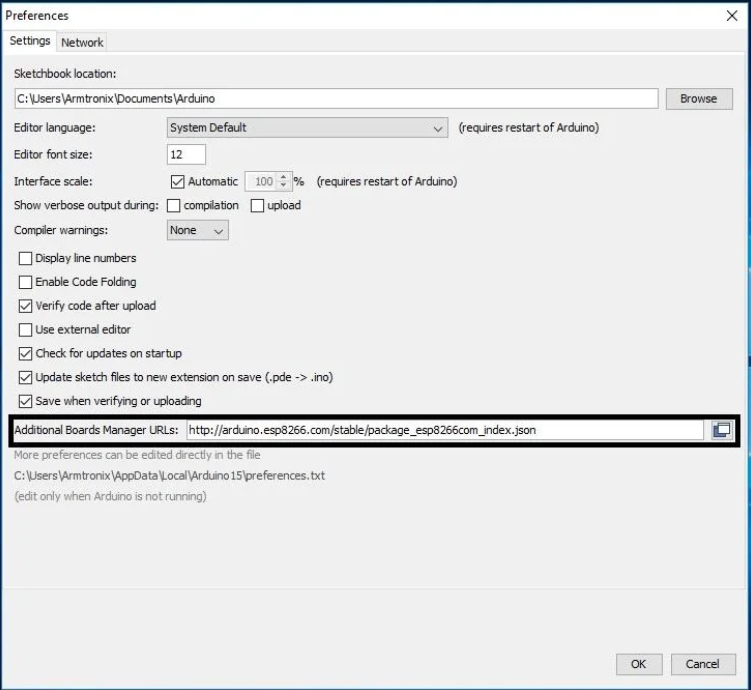


Fig. 8 - Arduino Preferences

The next step is installing the board on Arduino IDE, to do that, go to “Tools”🡪 “Board: …”🡪 “Board Manager”, as it’s possible to see on Fig.9.

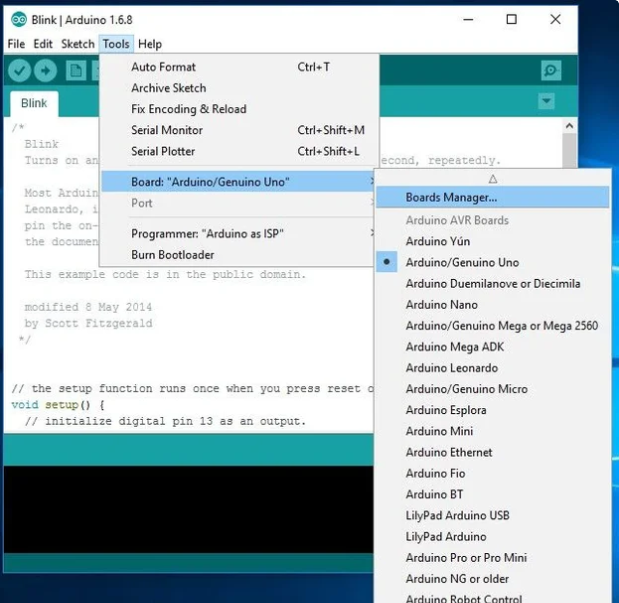


Fig. 9 - Arduino Tools🡪 Board Manager

Then a window called “Board Manager will open and it’s possible to search by the board, for that type “esp8266” on search box. When the search is complete, install the board called “esp8266 by esp8266 community” select the last version and click on install button. It is possible to see this step on Fig.10.

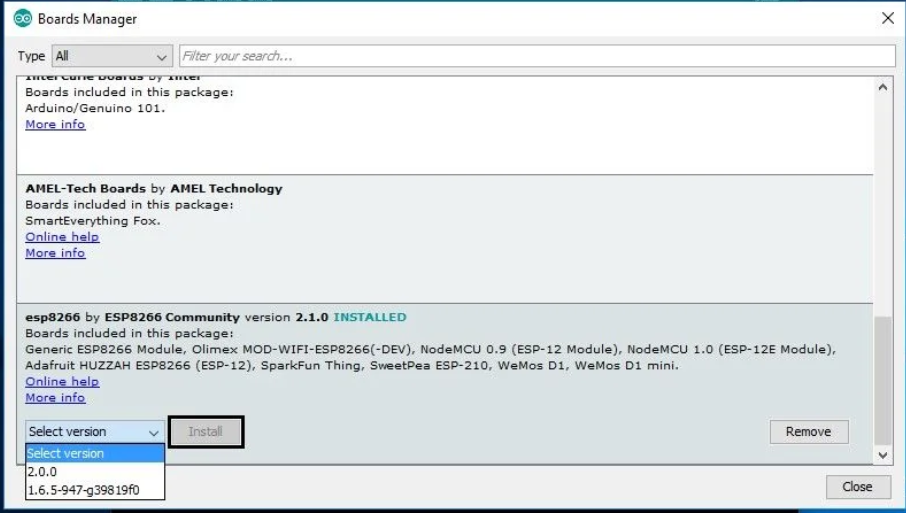


Fig. 10 - Install ESP8266

Wait for the program complete the installation. And after close “Board Manager” window and return to Arduino IDE interface. Go to “Tools”🡪 “Board…” and select the “NodeMCU 1.0 (ESP-12E Module)” as observed on Fig.11.

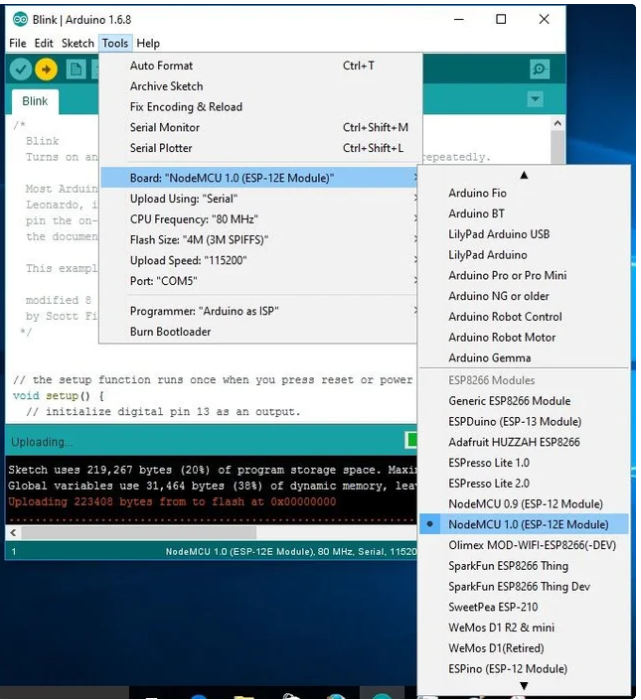


Fig. 11 - Board selection

Now the last step is select on Arduino IDE the correspondent COM for ESP8266 programming and see the Serial Monitor. For that go Back to Arduino Interface and select “Tools” 🡪 “Port…” and click on correspondent com of ESP8266 that is the same installed on Fig.7 of the last section of this document[[1]](#footnote-1), ass it’s possible to see on Fig.12. Exist the possibility of have different ports registered on Arduino IDE because of other services like Bluethooth.

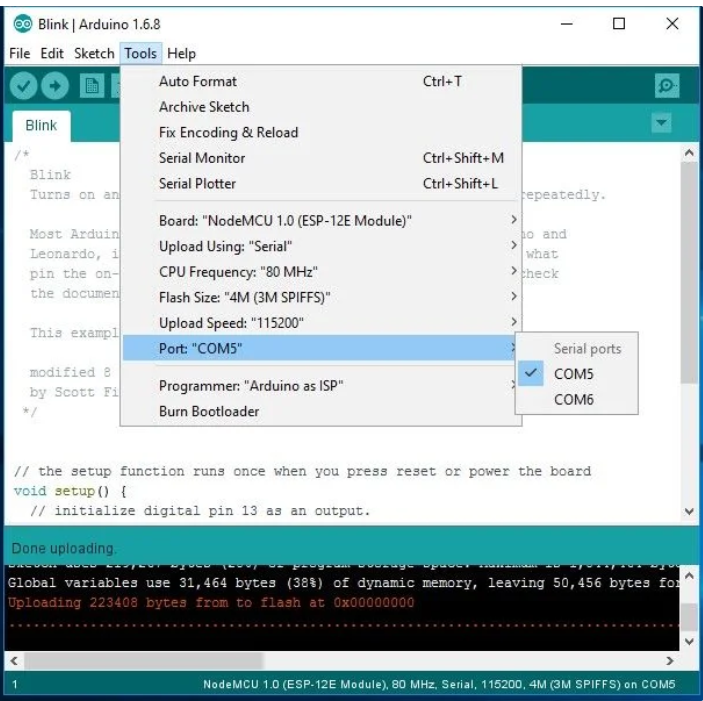


Fig. 12 - Select COM port of ESP8266

## Install and configure Raspberry Pi as MQTT broker

To install de “server” MQTT, it essential installing Mosquitto. It allows data transmission between Node MCU and Raspberry.

To do that, the first step is update and upgrade the Raspbian system. Using “Terminal” type:

* sudo apt-get update
* sudo apt-get upgrade

After this step Raspbian system is now updated, and proceed to the installation of Mosquitto and Mosquitto-clients running the next commands:

* sudo apt-get install -y mosquitto mosquitto-clients

And now active service running the follow command:

* sudo systemctl enable mosquitto.service.

It’s possible to see that with this command Raspbian activate the Mosquitto service and is now possible the non-secure communication. But a good practice is protecting the system against unauthorized access. Before any change stop the mosquitto service typing:

* sudo stop mosquitto

After creating a file that is the responsible to save the MQTT credentials like username and password. In future this password is encrypted.

* sudo mosquitto\_passwd -c /etc/mosquitto/passwd <username>

In this step the field <username> must be changed by the username of MQTT Broker and the the Raspbian will request the password twice for a correct verification insert the same password in two times.

The next step is telling the system where credentials are stored and block access to unauthenticated users, so run in terminal the next command:

* sudo nano /etc/mosquitto/mosquitto.conf

And a windows nano will open with the file “mosquitto.conf” that is the preferences file of Mosquitto. In that file insert the following lines

* password\_file /etc/mosquitto/passwd
* allow\_anonymous false

The field allow anonymous must be set to “false” to prevent anonymous access.

To end the installation and configuration of Mosquitto service start again the raspberry as broker:

* mosquitto -c /etc/mosquitto/mosquitto.conf

To test if the correct operation run the next command.

* mosquitto\_sub -h localhost -p 1883 -t my topic -u <username> -P <senha>

</senha></username>

## MQTT protocol

MQTT stands for MQ Telemetry Transport. It is a publish/subscribe, extremely simple and lightweight messaging protocol, designed for constrained devices and low-bandwidth, high-latency or unreliable networks. The design principles are to minimize network bandwidth and device resource requirements whilst also attempting to ensure reliability and some degree of assurance of delivery. These principles also turn out to make the protocol ideal of the emerging “machine-to-machine” (M2M) or “Internet of Things” world of connected devices, and for mobile applications where bandwidth and battery power are at a premium.

MQTT was invented by Dr Andy Stanford-Clark of IBM, and Arlen Nipper of Arcom (now Eurotech), in 1999.

You can pass a user name and password with an MQTT packet in V3.1 of the protocol. Encryption across the network can be handled with SSL, independently of the MQTT protocol itself (it is worth noting that SSL is not the lightest of protocols, and does add significant network overhead). Additional security can be added by an application encrypting data that it sends and receives, but this is not something built-in to the protocol, in order to keep it simple and lightweight.

Definition By: (mqtt.org, 2019)

It’s possible to use the follow most important methods in MQTT:

* Connect 🡪 Client requested a connection to a server
* Publish 🡪 Post a mensage
* Subscribe 🡪 Subscribe a specific topic
* Unsubscribe 🡪 Unsubscribe from the specific topic
* Disconnect 🡪Disconnect notification

The next scheme identifies the function of MQTT protocol:

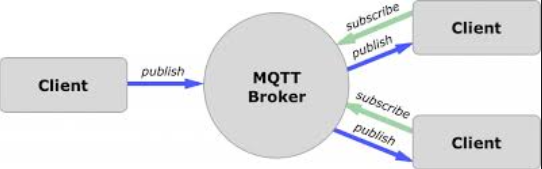


Fig. 13 – Architecture MQTT Protocol

## DDNS

The DDNS is a Dynamic Domain Name System protocols, so like a DNS protocol it allow to access to a server using a Domain Name, the DNS protocol is one of the most uses protocol in internet, because it make easy to people without know the ip address of a server access them using a specifics name domain, an example of this is when access of google search page using for example [www.google.com](http://www.google.com).

To allow this both of home routers have this installed protocol and are compatible with it.

For example, in 4G router when the port forwarding is correctly configure, other or the administrator can access to the PC, Raspberry, server or other end device by typing the correct IP of end device. So a possible configuration to do in router is:

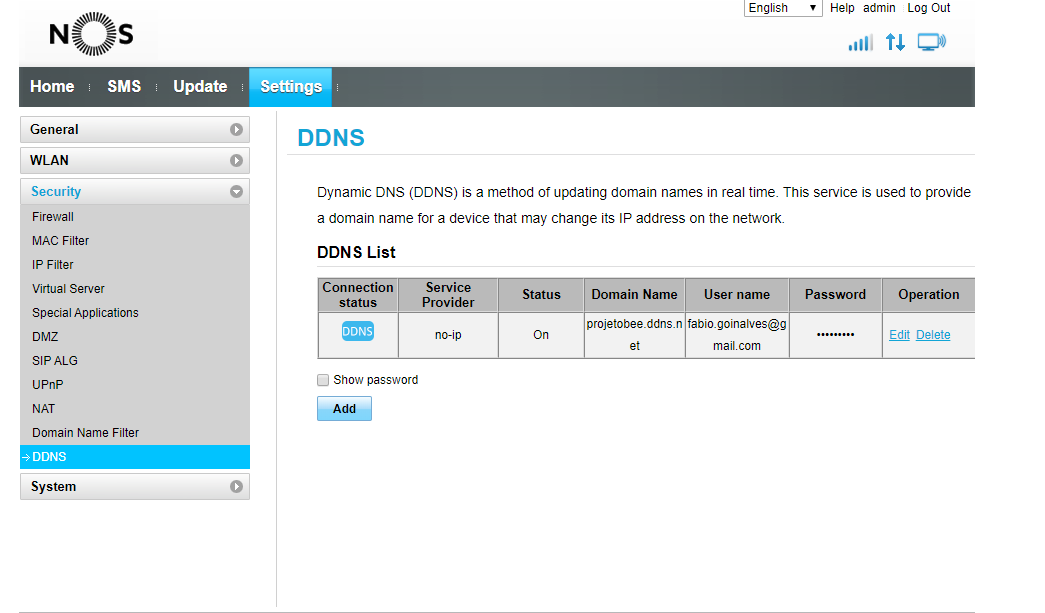


Fig. 14 – DDNS Router LTE NOS Configuration

But to allow that configuration is needed to configure a “service provider” in this example the router Huawey B310 allow to configure “no-ip” as a service provider. By access no ip web site on: <https://www.noip.com/> its possible to create a new account using e-mail address and password.

In no-ip web site it’s possible to configure a DDNS, using the tab “No-IP host names” and after click on “Create Hostname”. In this part, only insert, name of the host is pretended, for example the name of the project, in this case “projetobee”, select a possible domain, “ddns.net” for example, and after insert the public IP of the router, this public IP is the IP in this case: 83.132.64.193 address that router use to connect your home or company network to the internet, after this steps click on “Create Hostname” to confirm the creation. It possible to that the new IP DDNS creation is something like:

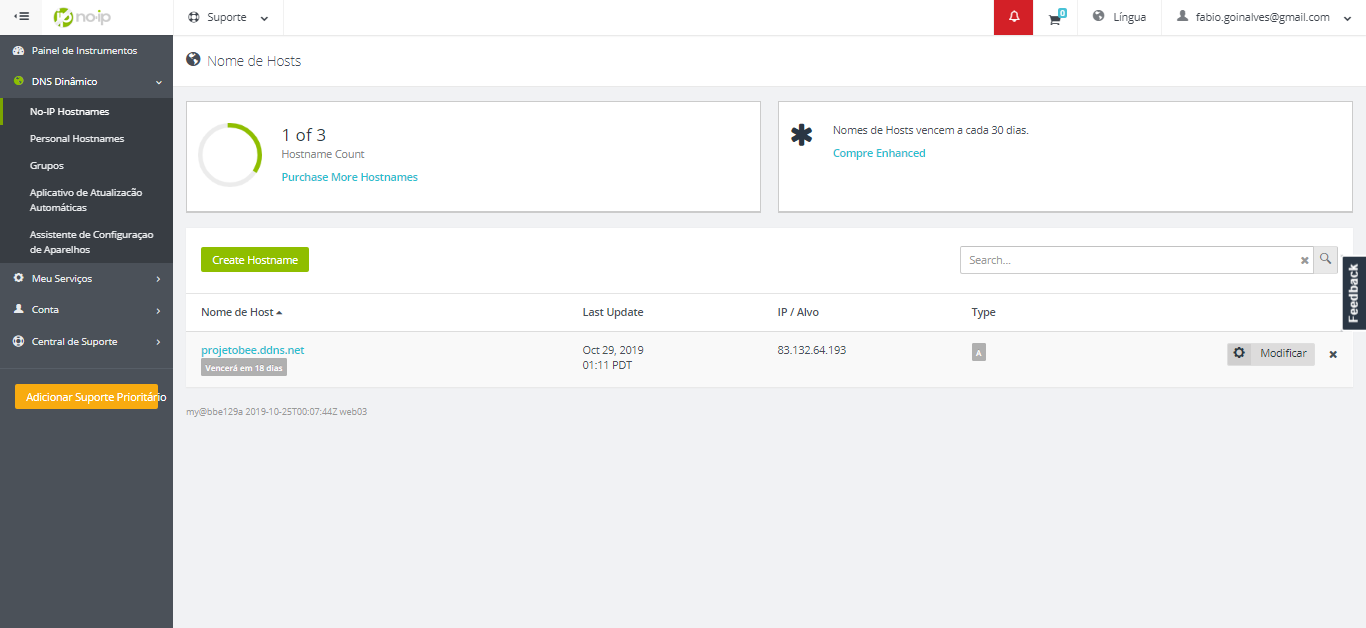
Now, on router DDNS configure all information, as are presented on this image. And now it’s possible to access the end device by outside of home or company network using a web browser. And the No-IP redirect the packages to the router when a search by “projetobee.ddns.net” is do it on other browser connected to the internet.

Fig. 15 – DDNS Configuration on noip Web Site

## Port forwarding or virtual server

The port forwarding allow to access an specific port by outside of the router. So the forward the packages that comes to them in one port to another port.

An example of that is when a router receive a HTTP package on wan port 80, it will forward all HTTP packages received for LAN port 80 to a specific address in case of Virtual Server. To understand that a possible implementation of this feature is showed on next figure:

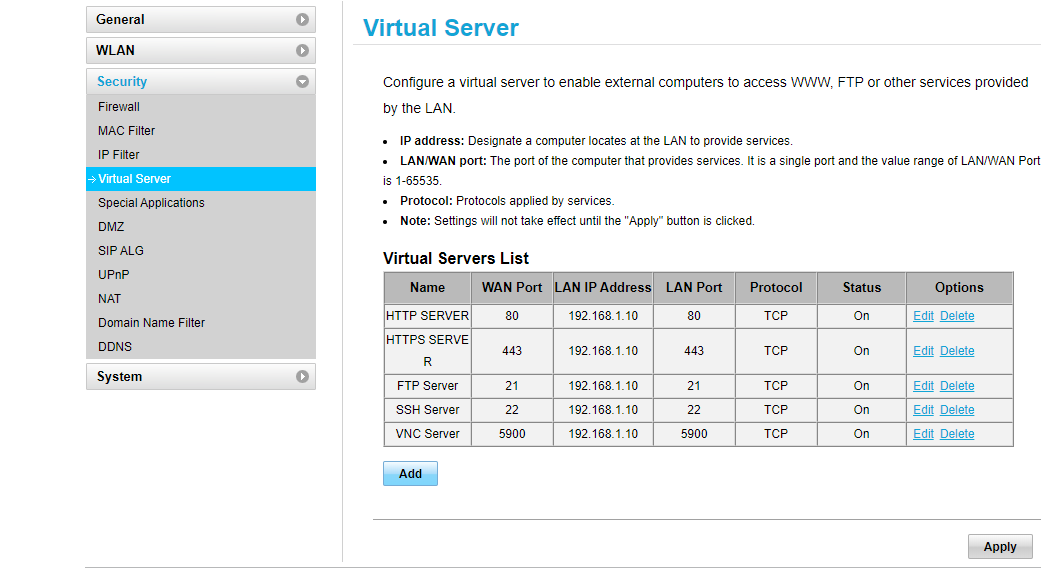


Fig. 16: Virtual Server Router NOS LTE configuration

Another example is the File Transfer Protocols (FTP) received on WAN Port 21 is forwarding to a specific IP in this case to 192.168.1.1.10 that is the FTP server in LAN port 21 of TCP protocol.

A big part of this TCP/UDP ports are listed on this url: <https://pt.wikipedia.org/wiki/Lista_de_portas_dos_protocolos_TCP_e_UDP>

# Weight thermal and humidity sensor system

## Used Materials

* Node MCU Board
* Protoboard or a previous designed PCB Board
* 50 Kg Load Cell
* HX711 (Analog to digital converter)
* Wood bars
* Wires
* Power supply for Node MCU Board
* DHT11

## Schematics

1. Electrical schematic

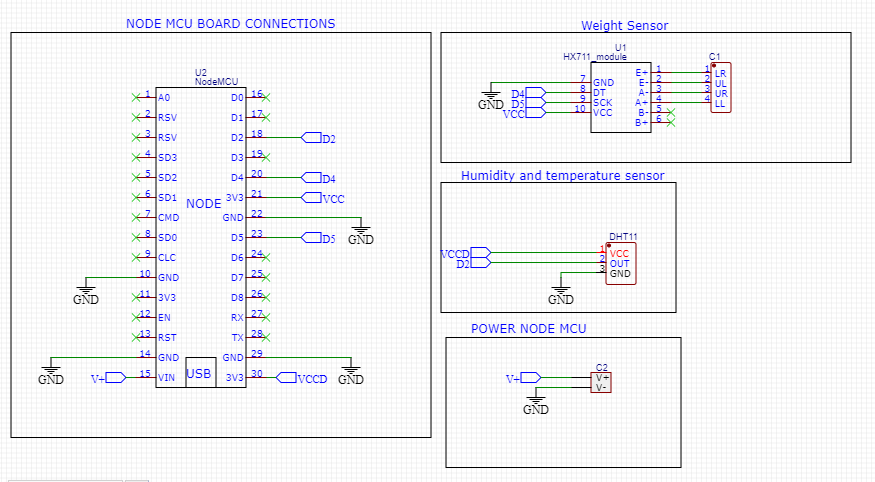


Fig. 17: Electrical Schematic

1. 50kg Load Cell

Mount the four load cell in the points of wood bars, with the rigid mount touching the wood and the load cell will be touched by the hive.



Fig. 18: Singular Load Cell

The location of load cells depends the type of hive that beekeeper have.

Load cell have 3 wires, each wire has a different color (red, black and white) it means that exist two variable resistors of 1kΩ.

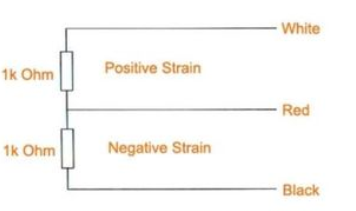


Fig. 19 Schematic Singular Load Cell

In first steps of implementation the wires will e connected in a specific logic; this logic allows to construct a Wheatstone bridge that will be implemented on HX711 (ADC).

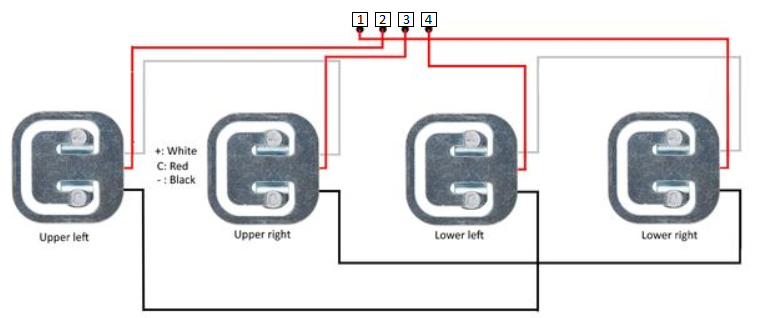


Fig. 20: Schematic Load Cells to connect HX711

In the previously figure, it’s possible to see the numeric identifications, this allows to identify the connection to HX711 as are descripted in the Next Table.

|  |  |  |
| --- | --- | --- |
| Numeric Identification | HX711 Pin | Human Identification |
| 1 | E+ | Lower Right (LR) |
| 2 | E- | Upper Left (UL) |
| 3 | A- | Upper Right (UR) |
| 4 | A+ | Lower Left (LL) |

1. Humidity and Temperature Sensor

“DHT11 Temperature & Humidity Sensor features a temperature & humidity sensor complex with a calibrated digital signal output. By using the exclusive digital signal acquisition technique and temperature & humidity sensing technology, it ensures high reliability and excellent long-term stability. This sensor includes a resistive-type humidity measurement component and an NTC temperature measurement component and connects to a high-performance 8-bit micro controller, offering excel lent quality, fast response, anti-interference ability and cost-effectiveness.

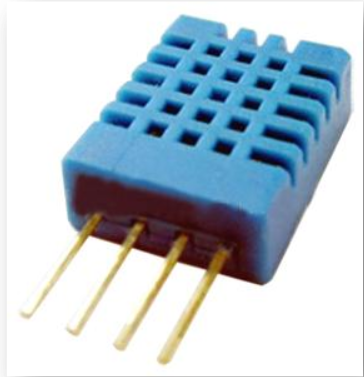


Fig. 21: Humidity and Temperature Sensor ( DHT11)

Each DHT11 elements strictly calibrated in the laboratory that is extremely accurate on humidity calibration. The calibration coefficients are stored as program in the OTP memory, which are used by the sensor’s internal signal detecting process.”

By: Mouser, DHT11 Technical Data Sheet Translated Version

For a question of simplicity, usability and cost, in this project are used a DHT11 module, that only have 3 pins (Power, Data and Ground) with a previously solder resistance as recommended by the seller.

When the connecting cable is shorter than 20 meters, a 5K pull-up resistor is recommended; when the connecting cable is longer than 20 meters, choose a appropriate pull-up resistor as needed.

By: Mouser, DHT11 Technical Data Sheet Translated Version

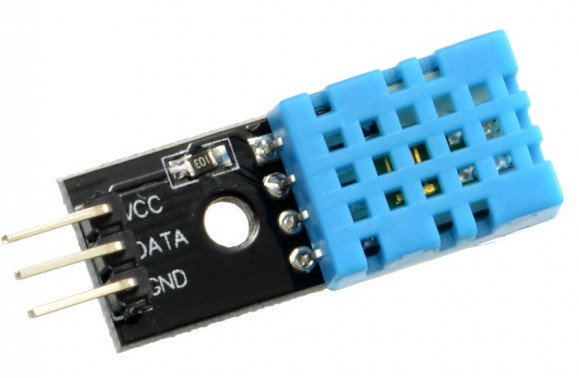
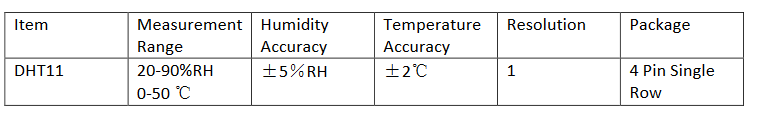
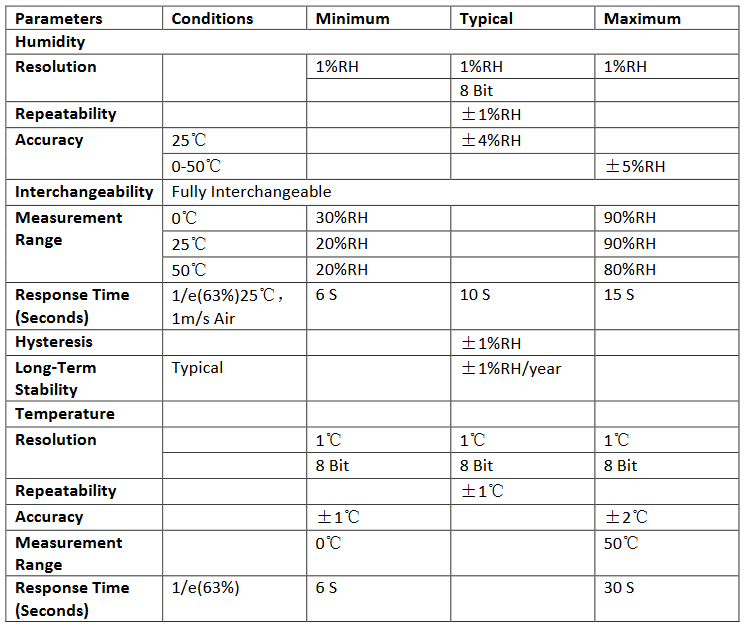


Fig. 22: Module DHT11

**Overview**



**Detailed Specifications**

****

As previously showed in this report the Data pin (OUT) of DHT11 is connected to the D2 pin of NODE MCU.

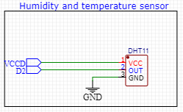


Fig. 23: Schematic for DHT11 and NodeMCU

In code a specific library was used on Node MCU to obtain the values of temperature and humidity, by using the next command lines:

* temperatura = dht.readTemperature();
* humidade = dht.readHumidity();

# LoRa Communications

## RFM95

With the objective of have long communications range, will be implemented LoRa technology. This technology will be implemented with RFM95.



Fig. 24:Real module RFM95W

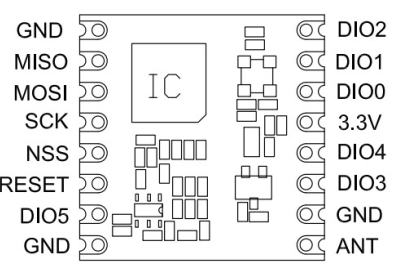


Fig. 25:Pinout Module RFM95

This RFM95 is a Low Power Long Range Transceiver Module, that could be used to multiple applications, as:

* Automated meter reading
* Home and building automation
* Wireless alarm and security systems
* Industrial monitoring and control
* Long range irrigation systems

## Electronic connections

The LoRa module are associated to a NODEMCU what means that it,s possible to use that as a gateway of different communications methods (LoRa and Wi-Fi). To do that it’s essential do a connection between both modules.

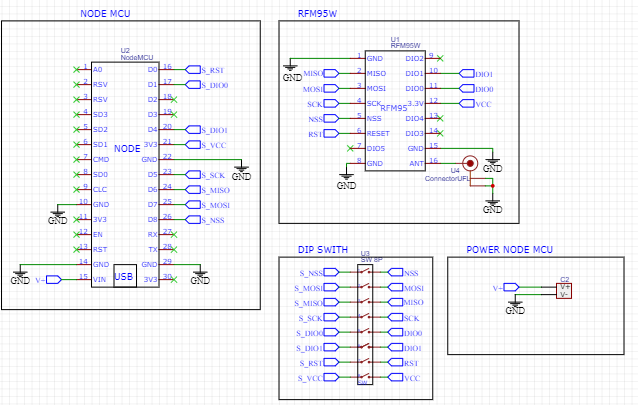


Fig. 26: Schematic NODEMCU and RFM95 connection

To make easy the preparation of the system, a table with the connections follow the previously electronic schema.

|  |  |
| --- | --- |
| **NODE MCU** | **RFM95w** |
| Gpio15/D8 | NSS |
| Gpio13/D7 | MOSI |
| Gpio12/D6 | MISO |
| Gpio14/D5 | SCK |
| Gpio05/D1 | DIO0 |
| Gpio02/D4 | DIO1 |
| Gpio16/D0 | RST |
| VCC | 3.3V |
| GND | GND |

## Breadboard implementation

In the first steps on the system preparations, it’s possible to see that the distance between pins it’s less them the hole pins distance of breadboard, typically on 2.54mm. Because of that and if the objective is the breadboard implementation, it’s essential a support to connect the RFM95 to Breadboard.



Fig. 27: Adapter SMD 16pin 2mm to 2.54mm

In the previously image it’s possible to see a support for ESP8266 to breadboard. This support will be implemented on RFM95 with two adaptations, one of them is remove all SMD resistors, and the other is do short circuit in R2 (center SMD resistor). After that only need to weld the RFM95 and the pins to the PCB support.

## Antenna Manufacturing

For this implementation that antenna used in first steps of prototype is manufactured with an UTP cable (Unshielded Twisted Pair), this antenna is a dipole of 8.6cm of ¼ λ, calculated by the follow equations:

The wavelength is 0.345 m, so the dimension of dipole is ¼ of wavelength, that means the antenna have a wire with 0.086m = 8.6 cm.

1. Note that the Port selected on Fig.7 and on Fig 12will be different **only** in this document. [↑](#footnote-ref-1)