R/OSE ELECTRONICS AND MICROCONTROLLER MANUAL

An In-Progress Document

There are two major ‘classes’ of microcontroller in this project. The basic responsibility of each ‘unit’ microcontroller is to collect bio data from the plants and send it. The ‘hub’ microcontroller receives data from all the ‘units’ and tells the robot what to do. Like so:

Apart from collecting and transferring data these microcontrollers will be simultaneously operating several other operations.

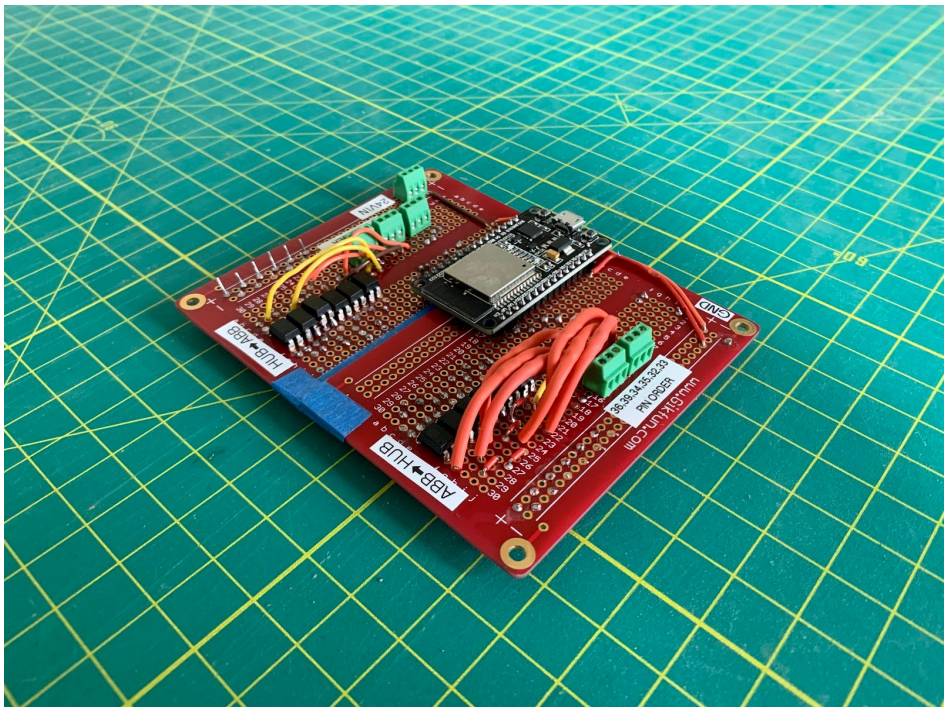
The ‘hub’ microcontroller needs to have the following functionality:

- Collect, store, organize moisture data
- Tell robot which units need to be watered
- *ideally* post data online

The microcontroller on each ‘unit’ needs to have the following functionality:

- Knows time of day
- Turns grow lights on and off
- Collect moisture data
- Send moisture data with ESP-NOW

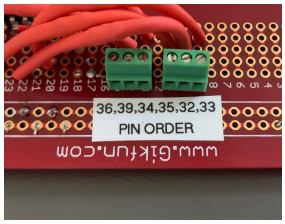
The Hub Setup and Wiring Guide

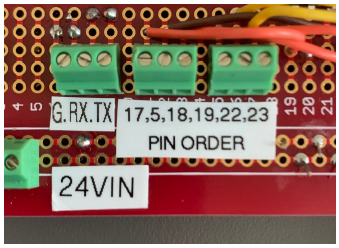


Hub 1.0 - AKA ‘redboi’ MAC address - 8C:AA:B5:85:65:24

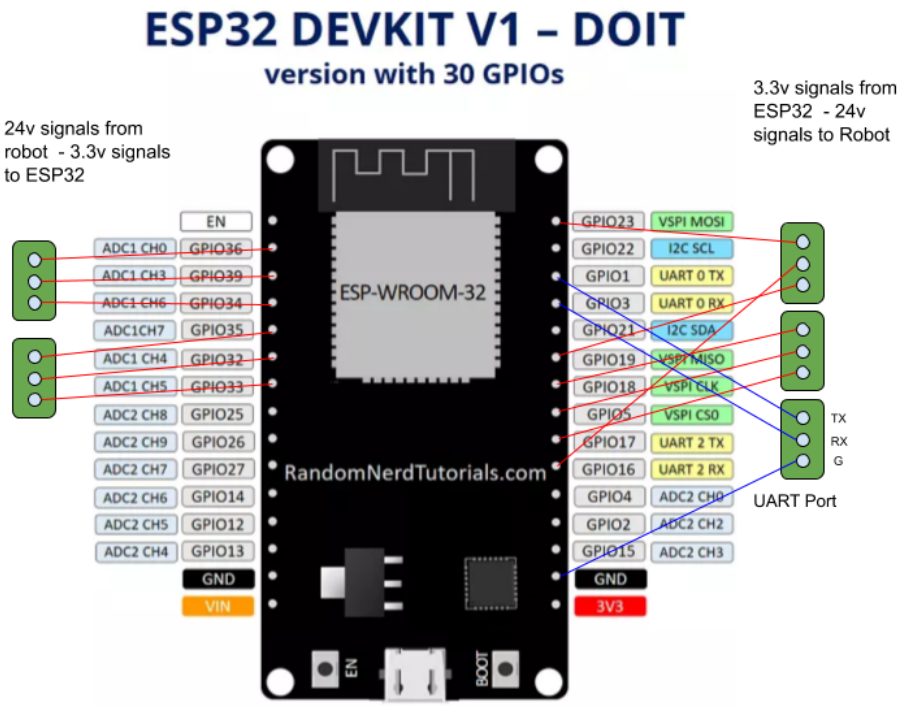
specs :

- ESP32 Devkit Doit V1
- Six 24v -3.3v optocoupler modified inputs terminals
- Six 3.3v -24v optocoupler modified output terminals
- One 3.3 v line level UART 2 way port.





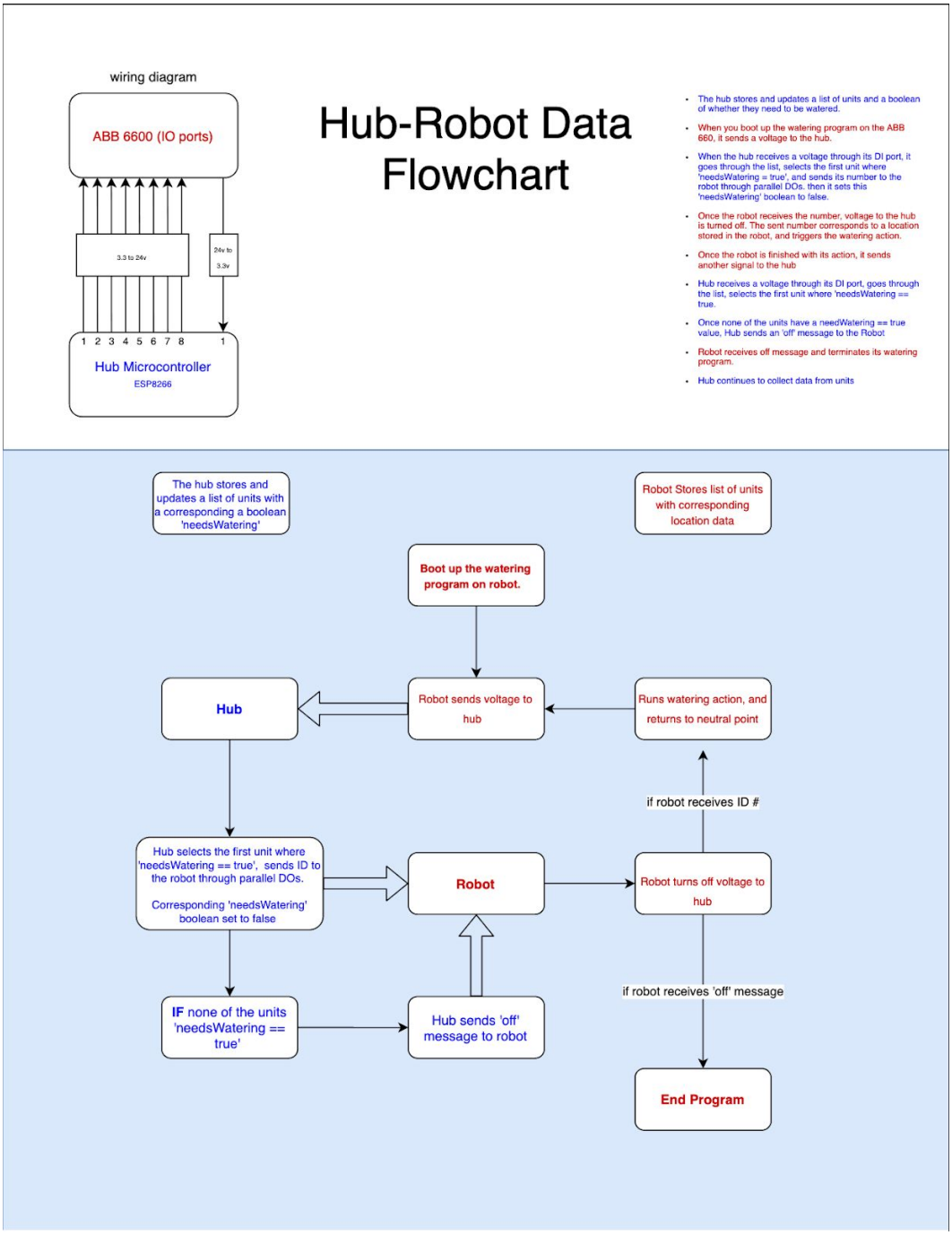
Pinouts/wiring guide -



Available ports:

Basic Setup

Hub - Robot Data Flowchart



Notes on Moisture Threshold Adjustments

- Given that serial communication between microcontroller and robot faces many timing issues.
- We've thus made the decision to send data from microcontroller to robot using parallel communication through the io ports to send 8 bit messages when triggered.

- The bot sends a ‘ready’ signal to the hub, and the hub tells it which units need watering. But, communicating anything other than an identification tag to the robot from the hub becomes a difficult task.
- However, each type of plant theoretically has different watering requirements. One unit might need to be watered more often than another, and we can take care of this issue by changing the threshold under which the robot waters the plant.
- So a decision needed to be made regarding where the threshold is defined: in each unit, or in the hub?
- There are pros and cons to both decisions, but ultimately, setting that threshold in the hub means that many thresholds can be set at once, rather than reprogramming units individually.
- Doing it the other way around could improve ease of use if we also designed a user interface for setting moisture thresholds. This might be a good idea in the future, but for now we can just maintain a table in the hub.

Table of Mac Addresses

Unit Number	MAC Address	Moisture Threshold
ESP32 Hub	**8C:AA:B5:85:65:24**	NA
1	48:3F:DA:0C:BF:97	550
2	48:3F:DA:0C:84:76	550
3	48:3f:da:7e:43:df	

Upload Code:

- The code for the 'hub' is based on the 'receiver' sketch described in the following tutorial: https://randomnerdtutorials.com/esp-now-many-to-one-esp8266-nodemcu/. We also need the 'hub' to format the data it receives and pass it on to the robot.

  Insert coding documentation here

- Open the following file with the Arduino IDE: (post link when ready)

- Make sure you've installed all necessary libraries. Some will be installed when you install add the board to the IDE, others may need to be downloaded separately. If you try to upload a sketch and the IDE indicates that it cannot find a library, download it. You can do this by going to **Tools > Manage Libraries**, and search for the library in question. Alternatively, you can download a zip file, go to **Sketch > Include library > Add .zip Library** and drag the file in.

- The 'Hub' is a receiver of information. Because of this you will need to make sure that it can identify the MAC address of each unit. So whenever you add a unit, you will need to update the hub with this new info.

  Insert instructions here

- Once you have assigned the proper MAC addresses from your units, upload the code, plug in a 'unit' and make sure the connection is working by checking your serial monitor.

Setting up the ‘Unit’ Microcontrollers

Basic Setup

- Find out the MAC address of the microcontroller. Follow the same process that was previously described for the ‘hub’.

- We’ll need to use this MAC address in the ‘hub’ code in order to identify which unit is doing the sending. There will be multiple units so we have to carefully keep track of these addresses. Maintaining a table which lists each unit number with its corresponding MAC address will save a lot of confusion.

- Note: As far as I know, 20 is the limit for the number of units that can communicate simultaneously with a hub, but we can find ways around this in the future.

- Before you start building the unit out, make sure your board is communicating with the hub. Do this by uploading the following code and checking your serial monitor.

  Insert code here

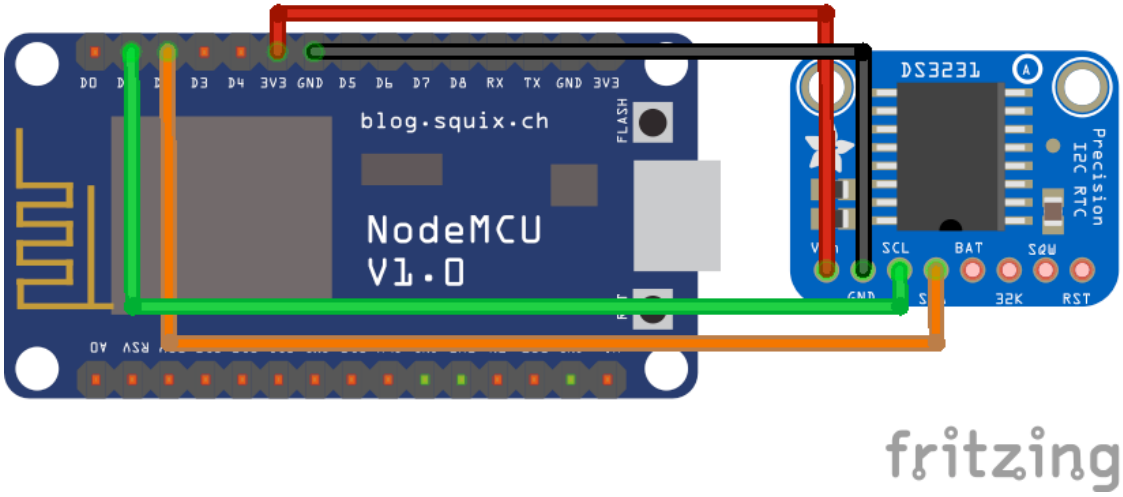
- Once you do so you can continue building your unit, which is a pretty involved process.

Getting your board to know the time

This will be done with an external RTC clock module, which requires some set up. Here’s a tutorial:

https://iot-guider.com/esp8266-nodemcu/learn-interfacing-ds3231-rtc-module-with-nodemcu/

Here’s the hookup diagram, the DS3231 module we will use may look slightly different, but the concept is the same.



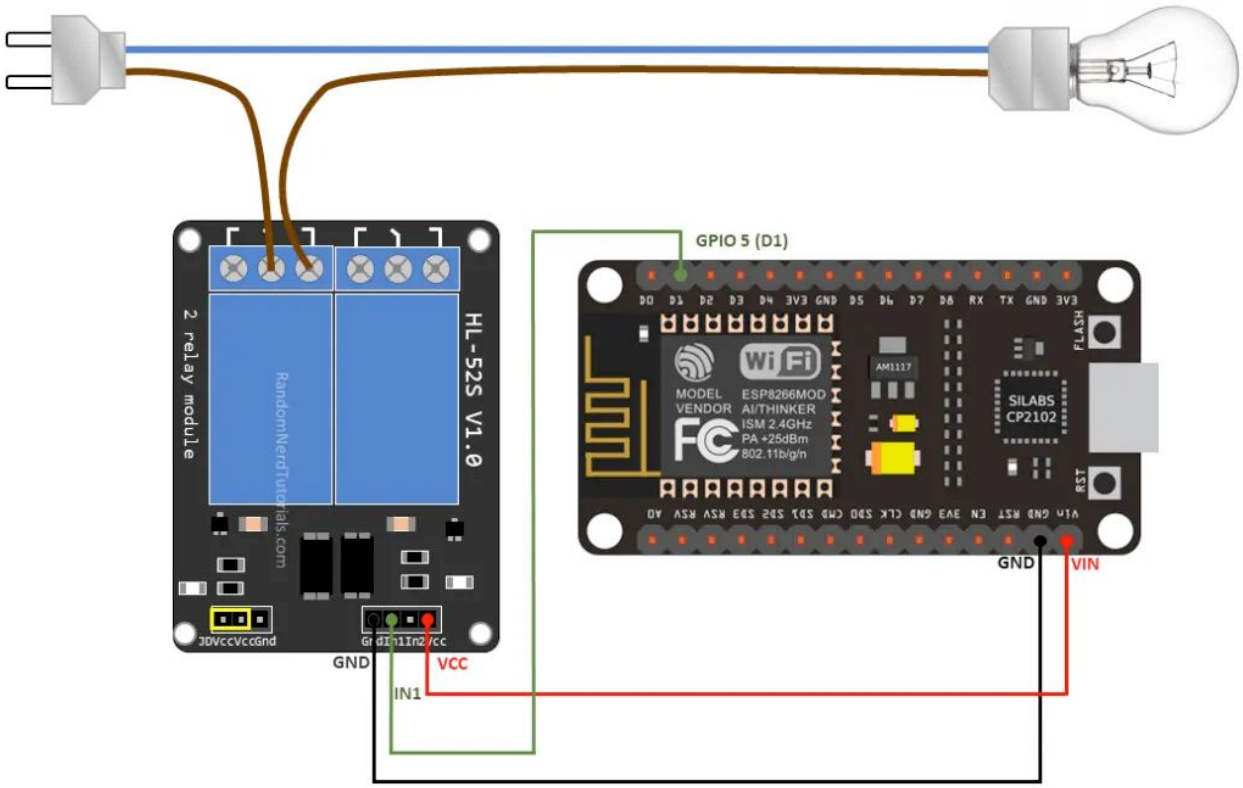
You will also need to make sure you have the following library downloaded:

1. Download Adafruit RTC Library Zip file from below given link
    - https://github.com/Makuna/Rtc/archive/master.zip
2. Go to **Sketch > Include library > Add .zip Library**
3. Select the downloaded zip file
4. If the Library is successfully added, Arduino IDE will show “Library added to your libraries. Check Include Library menu” on the status bar.

Learn more information about the RTC Library.
Other than that everything regarding the clock will be included in the program.

Turning grow lights on and off

Our grow lights will be hooked up as such, except that there is a 24V DC converter between outlet and the light.



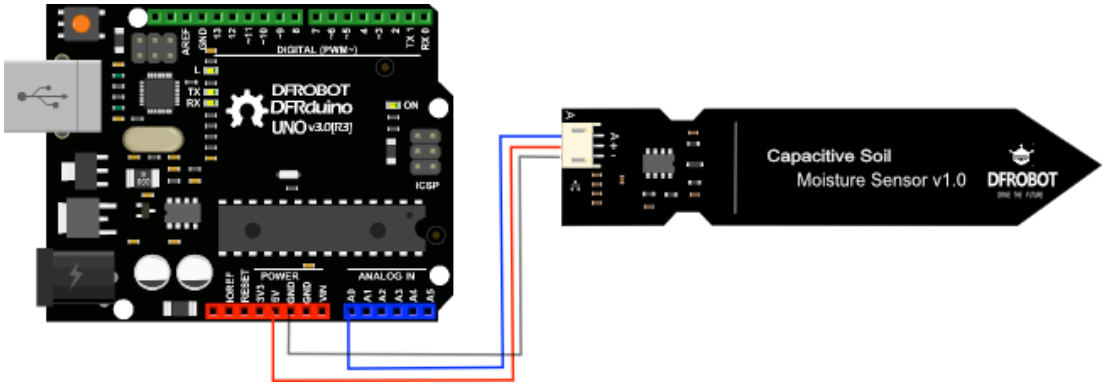
To do this, we will want to strip the + prong of the AC power cable, and hook it into the relay as demonstrated. Any of the available GPIO ports on the microcontroller should work.

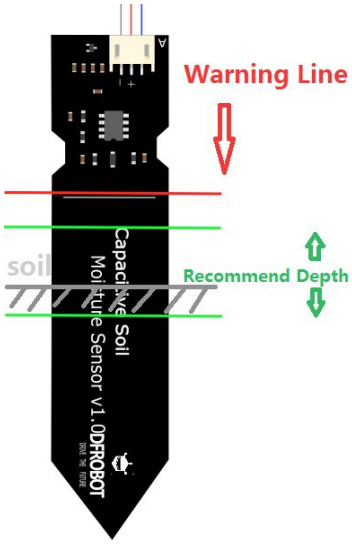
Side note: When we build the superstructure, we will need to buy a bigger power supply, something like this will be able to safely power 32 individual grow light strips, and strip the DC portion, but that is unnecessary for experimentation.

Collect Moisture Data

https://wiki.dfrobot.com/Capacitive_Soil_Moisture_Sensor_SKU_SEN0193

Using the A0 Pin on the Esp8266, we will hook up the capacitive soil sensor as such:





Then we have to calibrate it using a simple sketch -

```
void setup()
{
 Serial.begin(115200); // open serial port, set the baud rate as 115200
}
void loop()
{
 int val;
 val = analogRead(A0); //connect sensor to Analog 0
 Serial.println(val); //print the value to serial port
 delay(100);
}
```

To Calibrate:

1. Record the sensor value when the probe is exposed to the air as "Value 1". This is the boundary value of dry soil “Humidity: 0%RH”
2. Take a cup of water and insert the probe into it no further than the red line in the diagram
3. Record the sensor value when the probe is exposed to the water as "Value 2". This is the boundary value of moist soil “Humidity: 100%RH”

DO NOT expose electrical components on top of the module to water. Heat shrink necessary to protect the moisture sensor on completed units

Sending Moisture Data -

This code will be based on the ‘sender’ sketch in the following tutorial.
https://randomnerdtutorials.com/esp-now-many-to-one-esp8266-nodemcu/

Insert documentation here

Make chart of how the robot records location data for each plant