

RIVER WATER POLLUTION MONITORING

Using ESP8266 and ESPnow Protocol

(Group-6)

Raghav Luthra (2019B4A8PS0639P)

Archit Bhatnagar(2019A7PS0133P)

Daksh Jitarwal (2019A3PS0276P)

BUILD FLOW STEPS :-

1. Interfacing sensors – DHT 11 sensor for temperature & humidity sensing.
The DHT11 is a basic, ultra-low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air and gives out a digital signal on the data pin (no analog input pins needed). We also use an Ultrasonic sensor to take distance values in place of a pH sensor. This is done at both master and slave nodes.
2. Connecting Devices - 2 ESP8266 modules communicate with each other using ESPnow MAC protocol without any external support, the master node sends data to the slave which in turn communicates both its data and the slave's data to the firebase using a Wifi hotspot. Both the inter-node communication and firebase communication happen simultaneously.
3. Website: This is a static website hosted on a local development server that displays the change in the data updated in real-time.
 1. Create a virtual environment using: `python -m venv venv`
 2. Access the virtual environment: `venv\Scripts\activate`

3. Install the dependencies: `pip install -r requirements.txt`
3. Run the flask server: `python server.py`
4. Open port 5000 to view the website.

All the sensors we use are default so we straightaway jump to explain ESP-NOW.

ESP-NOW–

ESP-NOW is a protocol developed by Espressif, which enables multiple devices to communicate with one another without using Wi-Fi. The protocol is similar to the low-power 2.4GHz wireless connectivity. The pairing between devices is needed prior to their communication. After the pairing is done, the connection is safe and peer-to-peer, with no handshake required. We are using it to communicate from the Master to the Slave.

The two NodeMCUs communicate with each other using esp_now Protocol

MAC address of the receiver is fed to the transmitter.

It then broadcasts the data to the receiver using the MAC address

`uint8_t broadcastAddress[] = { 0x5C, 0xCF, 0x7F, 0x3D, 0x03, 0xD2 };`

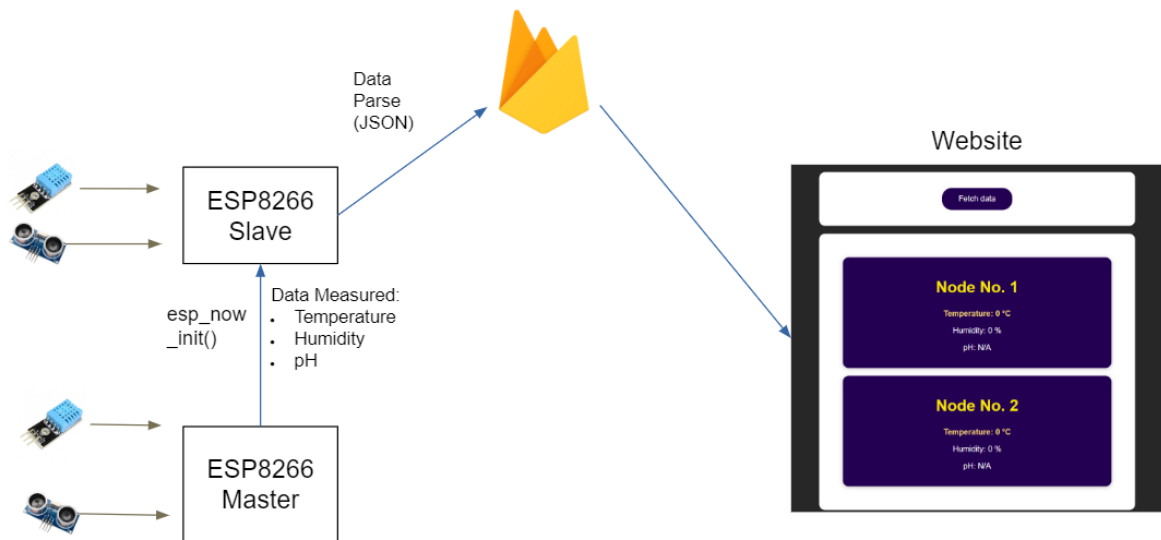
Code for obtaining MAC address is:-

```
#include "WiFi.h"
```

```
void setup() {  
    Serial.begin(115200);  
    WiFi.mode(WIFI_MODE_STA);  
    Serial.println(WiFi.macAddress());  
}
```

```
void loop() {  
}
```

SYSTEM ARCHITECTURE



The Master ESP8266 measures data using sensors and then sends them to the Slave ESP8266 (shown in Fig. 1).--

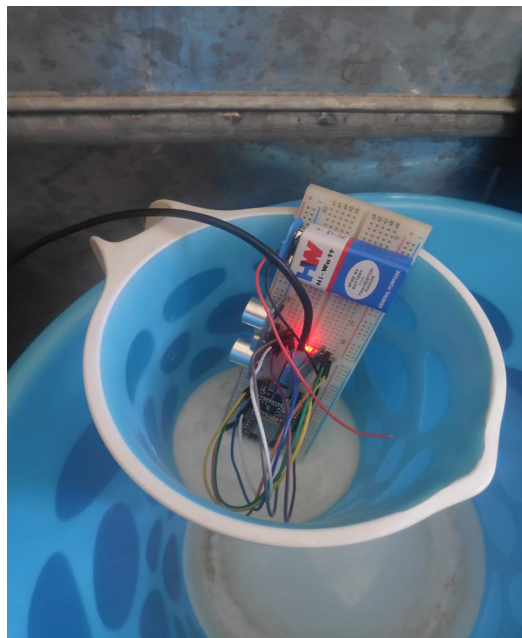


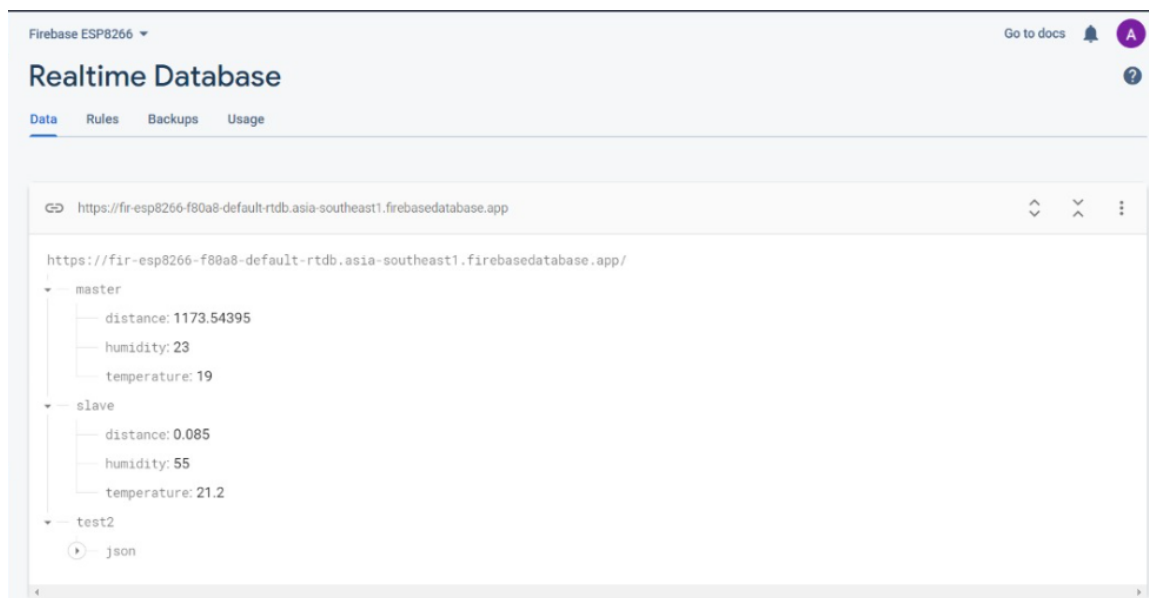
Fig. -1

The ESP8266 Slave measures its own properties and also receives the data from Master then updates both the data on firebase) (shown in Fig. 2)

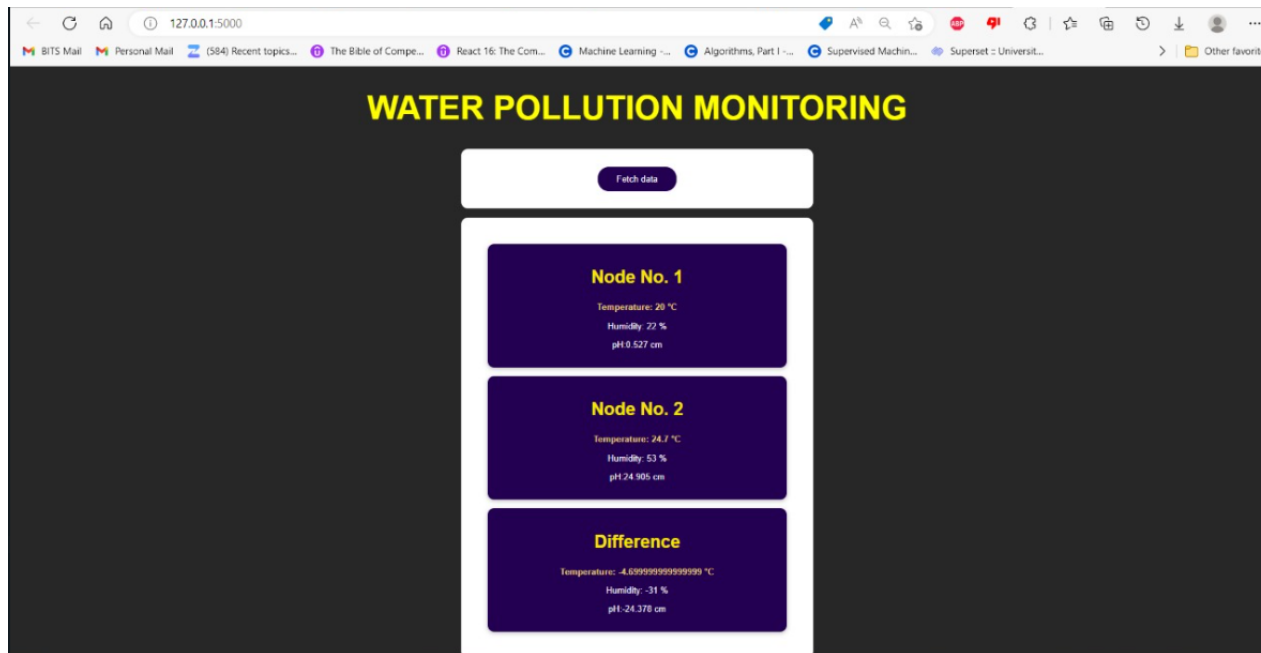


Fig. 2

Firestore real-time database stores and updates data values.



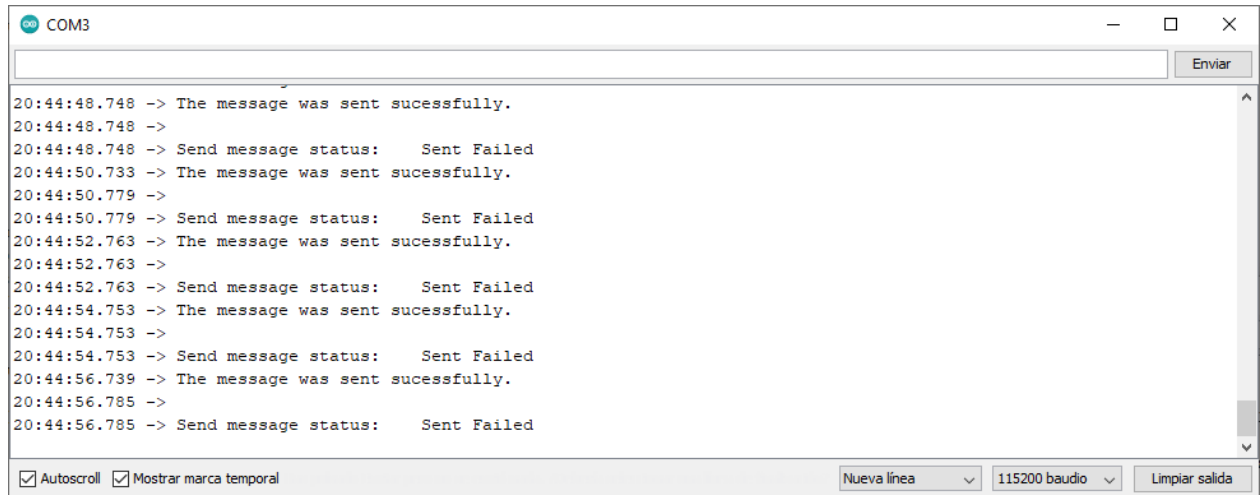
This data then gets displayed on the locally hosted Flask-based website(updates every 10 secs.)



Using ESP-Now and WiFi Simultaneously

When connecting the WiFi simultaneously, most of the ESP-Now packets are lost or do not arrive at all. This seems to be related to the way the WiFi works.

The Master Module is not connected to the wifi, and only sends packets to the slave over ESP-Now. It is observed that the master does not send the packets at all once the slave is connected to the wifi, and the following output is observed on the serial monitor of the master.



```
20:44:48.748 -> The message was sent sucessfully.
20:44:48.748 ->
20:44:48.748 -> Send message status:    Sent Failed
20:44:50.733 -> The message was sent sucessfully.
20:44:50.779 ->
20:44:50.779 -> Send message status:    Sent Failed
20:44:52.763 -> The message was sent sucessfully.
20:44:52.763 ->
20:44:52.763 -> Send message status:    Sent Failed
20:44:54.753 -> The message was sent sucessfully.
20:44:54.753 ->
20:44:54.753 -> Send message status:    Sent Failed
20:44:56.739 -> The message was sent sucessfully.
20:44:56.785 ->
20:44:56.785 -> Send message status:    Sent Failed
```

This error disappears as the `wifi.begin()` is commented out in the slave.

The slave does receive some packets after a lot of failed attempts from the master, but it seems to make a lot of noise and a little nuts.

The main cause of the problem was that the station mode WiFi goes to sleep mode while there is no work. Therefore, it stops listening to the ESP-Now packets and they are lost. This is solved by turning it into AP (Access point) mode.

```
WiFi.mode(WIFI_AP_STA);
```

Also, comment out the `Wifi.disconnect()` statement if it is present anywhere, as it puts the wifi to sleep mode when it becomes idle for even a little time.