Manas Pratim Biswas
Information Technology
Jadavpur University
(2020-2024)

This is a bonafide record of the research work that I have undergone under the mentorship of **Prof. Amlan Chakrabarti** and DST Women Scientist **Dola Gupta Bhattacharjee** in the two month period of June-July, 2022.

I hereby declare this documentation including all the code and attached images is a product of my work.

All the above codes can be found in this GitHub repository - [Link to all the codes]

Content

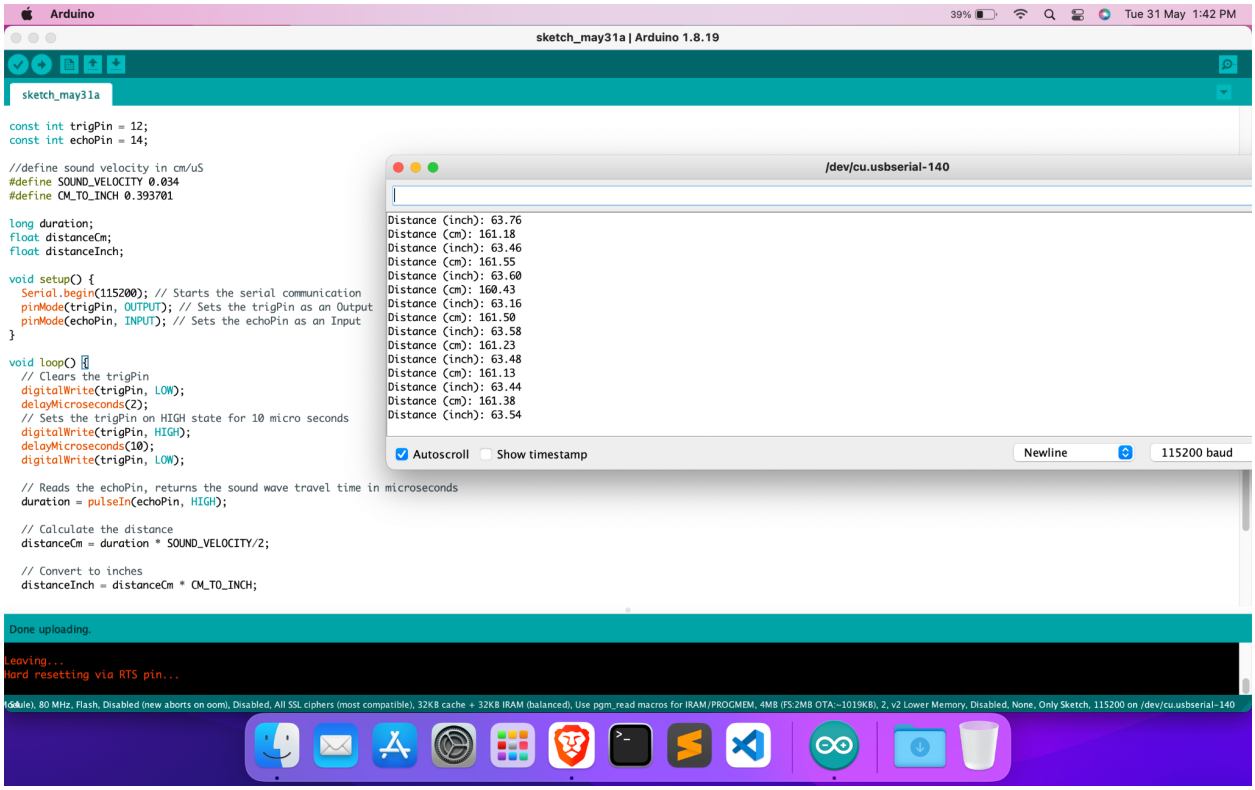
07/06/22	Multicasted multiple NodeMCUs to a single NodeMCU	14-20
10/06/22	Established communications between NodeMCUs with HC-SRO4 sensors	21-30
11/06/22	Connected NodeMCU with ThingSpeak platform	31-36
16/06/22	Started with Python Programming and NumPy	37
22/06/22	Learnt pandas, matplotlib and seaborn	37
24/06/22	Started with Machine Learning	37
26/06/22	Learnt Decision Trees and Random Forests	37
30/06/22	Learnt KNN and SVM	37
04/07/22	Made a project on Diabetes prediction system and deployed in on the web	37
15/07/22	Coded water leakage detection in a water tank	37-46
18/07/22	Synchronised and read data using water flow sensor and NodeMCU	47-49
29/07/22	Forecasting water usage	50-51

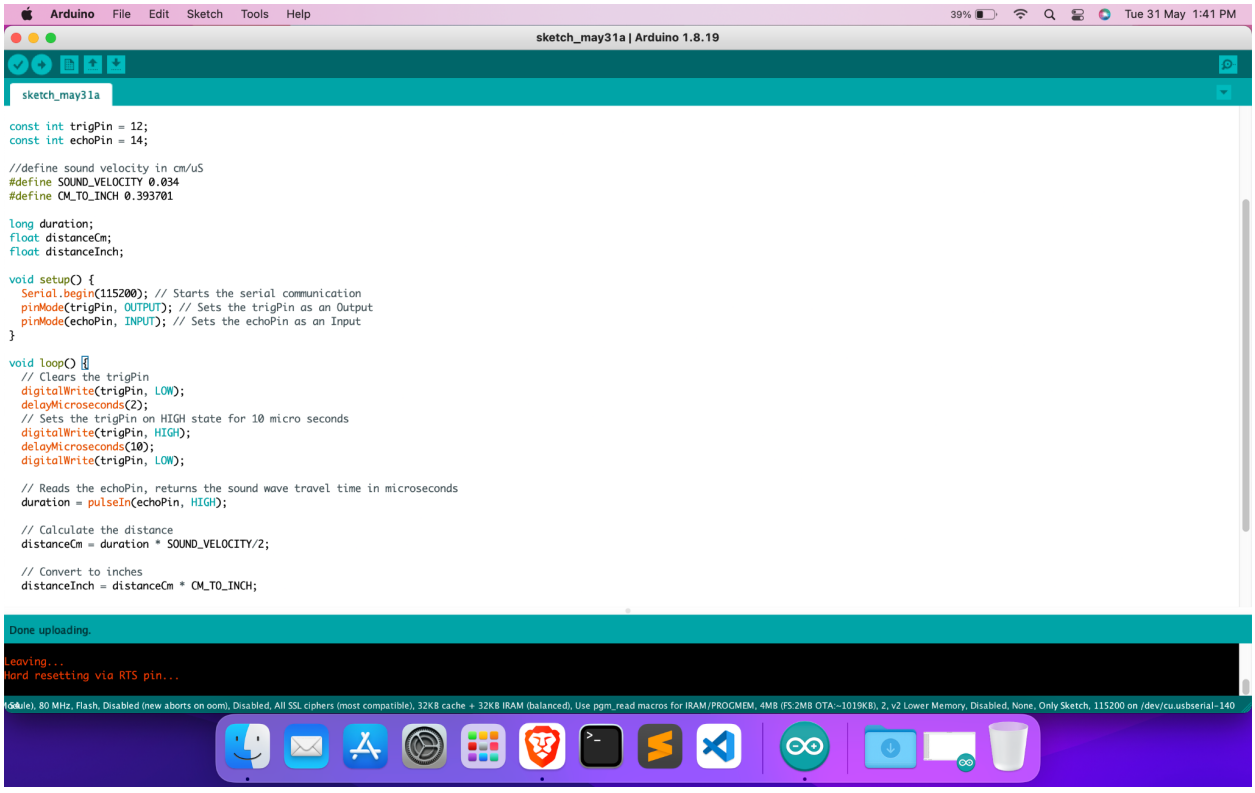
27th May

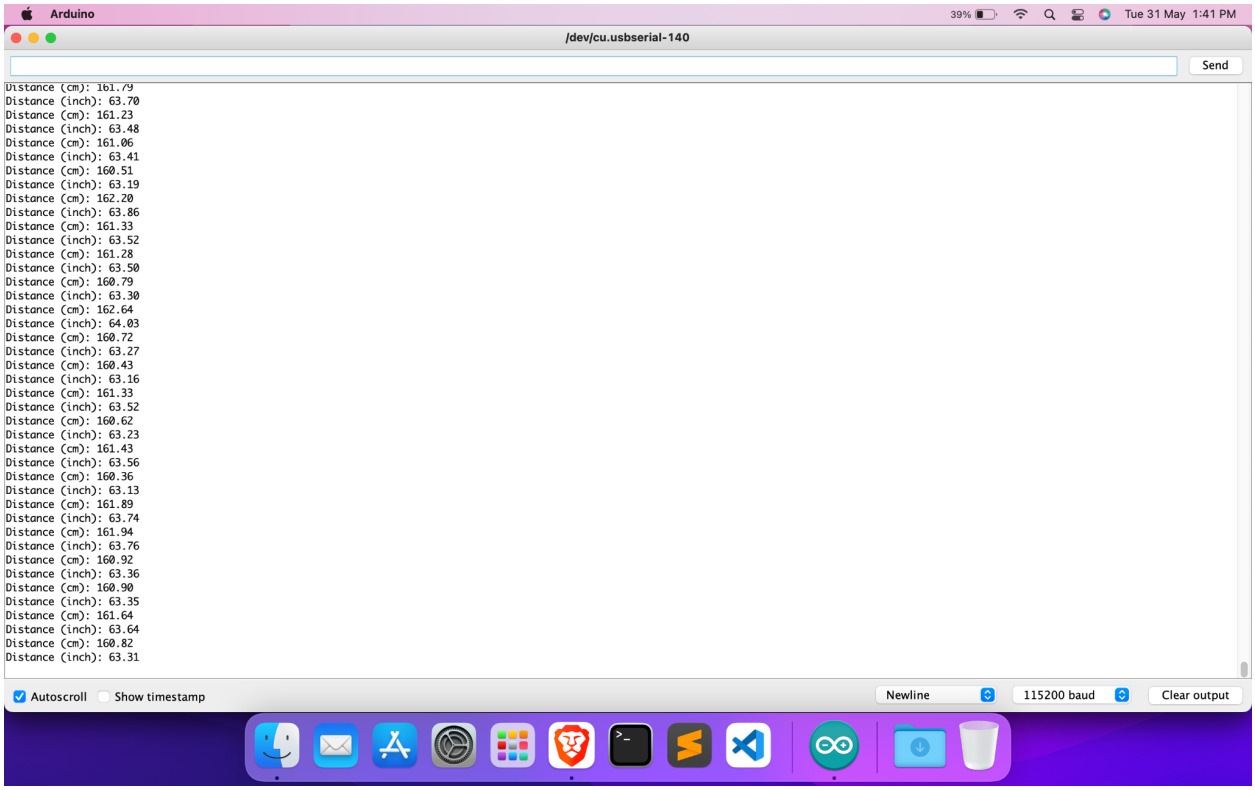
- Working of NodeMCU and the various pins present in NodeMCU
- Synchronization of NodeMCU with ArduinoIDE
- Synchronization of NodeMCU with Ultrasonic Depth Sensor

31st May

- Synchronized NodeMCU with ArduinoIDE
- Comparison between ESP-32 and NodeMCU
- Synchronization of NodeMCUs


Arduino
39%
Tue 31 May 1:42 PM
sketch_may31a | Arduino 1.8.19
sketch_may31a
const int trigPin = 12;
const int echoPin = 14;
//define sound velocity in cm/uS
#define SOUND_VELOCITY 0.034
#define CM_TO_INCH 0.393701
long duration;
float distanceCm;
float distanceInch;
void setup() {
Serial.begin(115200); // Starts the serial communication
pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
pinMode(echoPin, INPUT); // Sets the echoPin as an Input
}
void loop() {
// Clears the trigPin
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
// Sets the trigPin on HIGH state for 10 micro seconds
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
// Reads the echoPin, returns the sound wave travel time in microseconds
duration = pulseIn(echoPin, HIGH);
// Calculate the distance
distanceCm = duration * SOUND_VELOCITY/2;
// Convert to inches
distanceInch = distanceCm * CM_TO_INCH;
/dev/cu.usbserial-140
Distance (inch): 63.76
Distance (cm): 161.18
Distance (inch): 63.46
Distance (cm): 161.55
Distance (inch): 63.60
Distance (cm): 160.43
Distance (inch): 63.16
Distance (cm): 161.50
Distance (inch): 63.58
Distance (cm): 161.23
Distance (inch): 63.48
Distance (cm): 161.13
Distance (inch): 63.44
Distance (cm): 161.38
Distance (inch): 63.54
Autoscroll
Show timestamp
Newline
115200 baud
Done uploading.
Leaving...
Hard resetting via RTS pin...
Module), 80 MHz, Flash, Disabled (new aborts on oom), Disabled, All SSL ciphers (most compatible), 32KB cache + 32KB IRAM (balanced), Use pgm_read macros for IRAM/PROGMEM, 4MB (FS:2MB OTA:~1019KB), 2, v2 Lower Memory, Disabled, None, Only Sketch, 115200 on /dev/cu.usbserial-140


Arduino
File
Edit
Sketch
Tools
Help
39%
Tue 31 May 1:41 PM
sketch_may31a | Arduino 1.8.19
sketch_may31a
const int trigPin = 12;
const int echoPin = 14;
//define sound velocity in cm/uS
#define SOUND_VELOCITY 0.034
#define CM_TO_INCH 0.393701
long duration;
float distanceCm;
float distanceInch;
void setup() {
Serial.begin(115200); // Starts the serial communication
pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
pinMode(echoPin, INPUT); // Sets the echoPin as an Input
}
void loop() {
// Clears the trigPin
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
// Sets the trigPin on HIGH state for 10 micro seconds
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
// Reads the echoPin, returns the sound wave travel time in microseconds
duration = pulseIn(echoPin, HIGH);
// Calculate the distance
distanceCm = duration * SOUND_VELOCITY/2;
// Convert to inches
distanceInch = distanceCm * CM_TO_INCH;
Done uploading.
Leaving...
Hard resetting via RTS pin...
Module), 80 MHz, Flash, Disabled (new aborts on oom), Disabled, All SSL ciphers (most compatible), 32KB cache + 32KB IRAM (balanced), Use pgm_read macros for IRAM/PROGMEM, 4MB (FS:2MB OTA:~1019KB), 2, v2 Lower Memory, Disabled, None, Only Sketch, 115200 on /dev/cu.usbserial-140



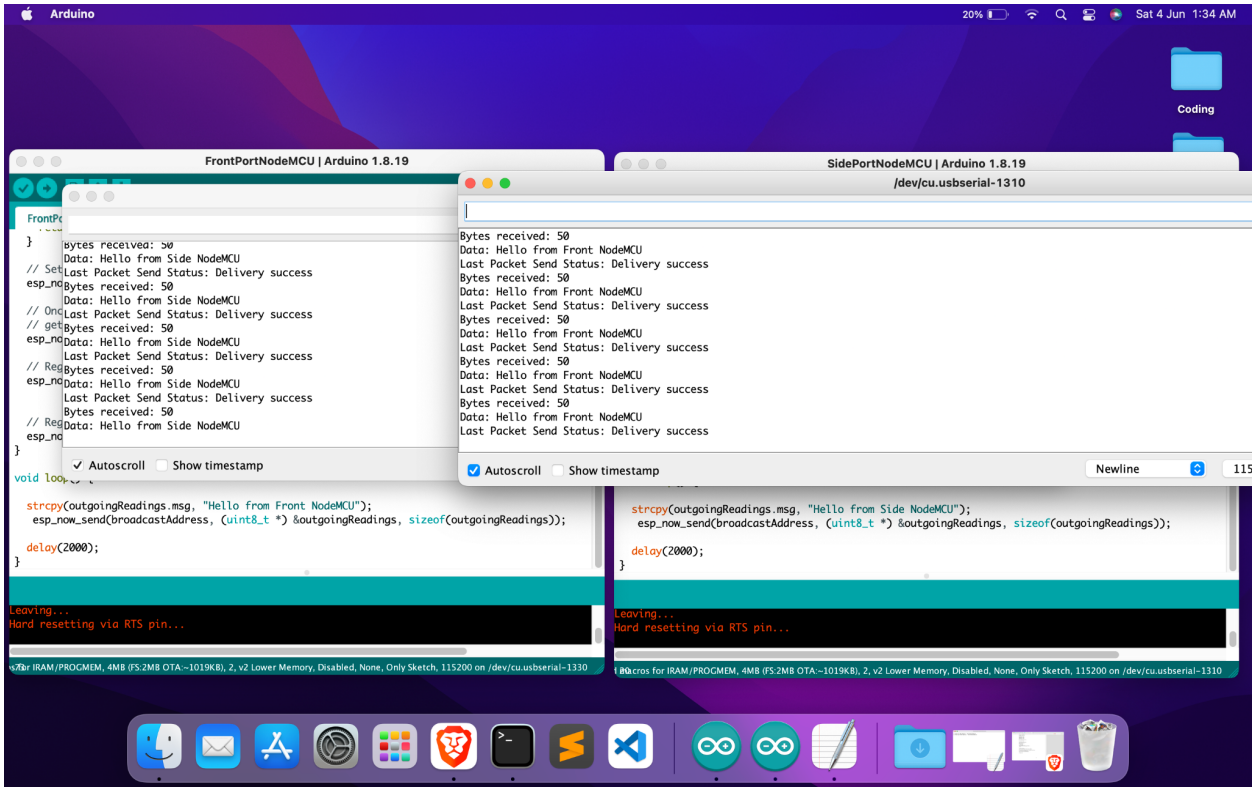
Code to integrate HC-SRO4 sensor with NodeMCU:

```
const int trigPin = 12;
const int echoPin = 14;

//define sound velocity in cm/uS
#define SOUND_VELOCITY 0.034
#define CM_TO_INCH 0.393701

long duration;
float distanceCm;
float distanceInch;

void setup() {
  Serial.begin(115200); // Starts the serial communication
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
}

void loop() {
```

```
  // Clears the trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);

  // Calculate the distance
  distanceCm = duration * SOUND_VELOCITY/2;

  // Convert to inches
  distanceInch = distanceCm * CM_TO_INCH;

  // Prints the distance on the Serial Monitor
  Serial.print("Distance (cm): ");
  Serial.println(distanceCm);
  Serial.print("Distance (inch): ");
  Serial.println(distanceInch);

  delay(1000);
}
```

3rd June

- Identifying the MAC address of NodeMCU
- Two-way communication between NodeMCU
- Many-to-one communication between NodeMCU
- [Reference Codes]



Code to get MAC address:

```
#ifdef ESP32
  #include <WiFi.h>
#else
  #include <ESP8266WiFi.h>
#endif
```

```
void setup(){
  Serial.begin(115200);
  Serial.println();
  delay(5000);
  Serial.print("ESP Board MAC Address:  ");
  Serial.println(WiFi.macAddress());
}

void loop(){

}
```

Code for two-way communication between NodeMCUs:

Sender code:

```
/*
  Flow of the code
  1 - Put WiFi in STA Mode
  2 - Initialise ESPNOW
  3 - Set Role to Combo
  4 - Add peer device
  5 - Define Send Callback Function
  6 - Define Receive Callback Function
*/

#include <Wire.h>
#ifdef ESP32
  #include <WiFi.h>
#else
  #include <ESP8266WiFi.h>
#endif
#include <espnow.h>

// REPLACE WITH THE MAC Address of your receiver
uint8_t broadcastAddress[] = {0xC8, 0xC9, 0xA3, 0x69, 0xDE, 0x1D};


typedef struct struct_message {
  char msg[50];
} struct_message;
```

```
// Create a struct_message called outgoingReadings to hold outgoing data
struct_message outgoingReadings;

// Create a struct_message called incomingReadings to hold incoming data
//struct_message incomingReadings;

// Variable to store if sending data was successful
String success;

// Callback when data is sent
void OnDataSent(uint8_t *mac_addr, uint8_t sendStatus) {
  Serial.print("Last Packet Send Status: ");
  if (sendStatus == 0) {
    Serial.println("Delivery success");
  }
  else {
    Serial.println("Delivery fail");
  }
}

// Callback when data is received
//void OnDataRecv(uint8_t * mac, uint8_t *incomingData, uint8_t len) {
//  memcpy(&incomingReadings, incomingData, sizeof(incomingReadings));
//  Serial.print("Bytes received: ");
//  Serial.println(len);
//  Serial.print("Data: "); Serial.println(incomingReadings.msg);
//}


void setup() {
  // Init Serial Monitor
  Serial.begin(115200);

  // Set device as a Wi-Fi Station
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();

  // Init ESP-NOW
  if (esp_now_init() != 0) {
    Serial.println("Error initialising ESP-NOW");
    return;
  }
```

```
  // Set ESP-NOW Role
  esp_now_set_self_role(ESP_NOW_ROLE_COMBO);

  // Once ESPNow is successfully Init, we will register for Send CB to
  // get the status of Transmitted packet
  esp_now_register_send_cb(OnDataSent);

  // Register peer
  esp_now_add_peer(broadcastAddress, ESP_NOW_ROLE_COMBO, 1, NULL, 0);



  // Register for a callback function that will be called when data is
received
  // esp_now_register_recv_cb(OnDataRecv);
}

void loop() {

  strcpy(outgoingReadings.msg, "Hello from Side NodeMCU");
   esp_now_send(broadcastAddress, (uint8_t *) &outgoingReadings,
sizeof(outgoingReadings));

  delay(2000);
}
```

Receiver Code:

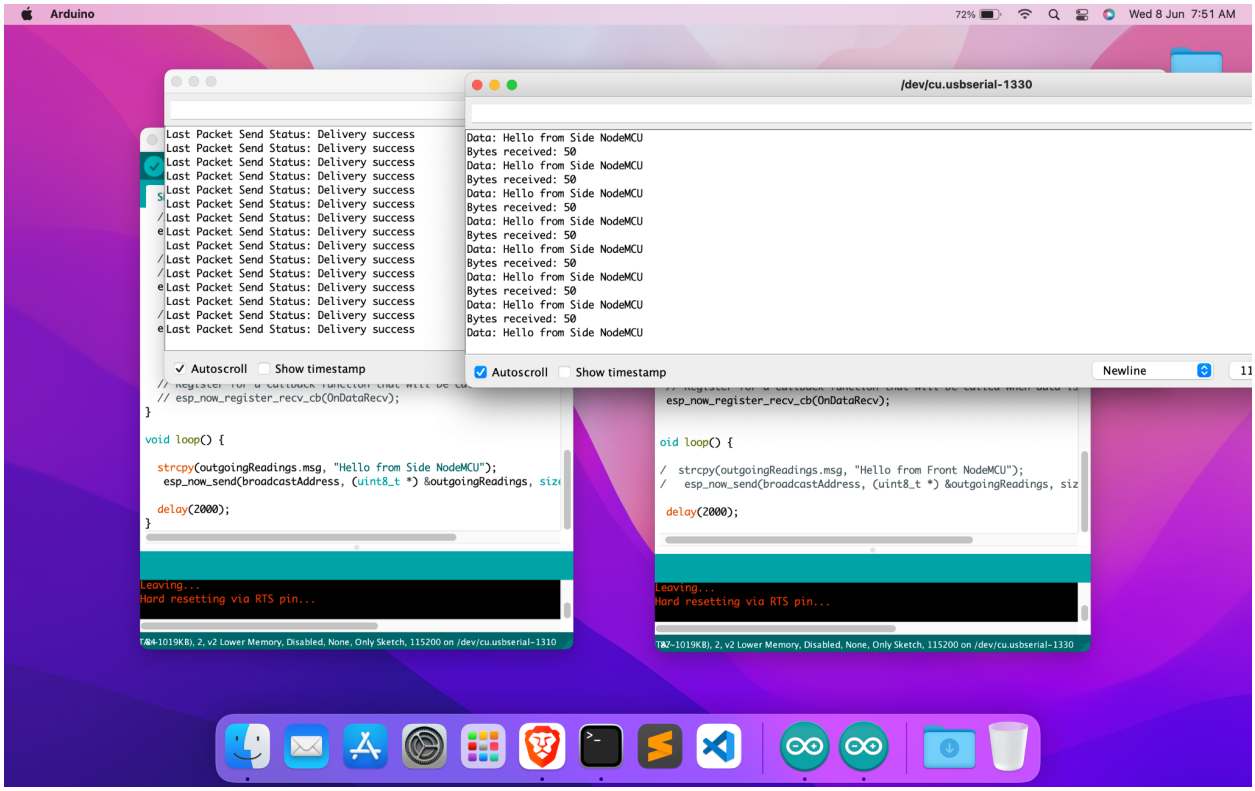
```
/*
  Flow of the code
  1 - Put WiFi in STA Mode
  2 - Initialise ESPNOW
  3 - Set Role to Combo
  4 - Add peer device
  5 - Define Send Callback Function
  6 - Define Receive Callback Function
*/

#include <Wire.h>
#ifdef ESP32
```

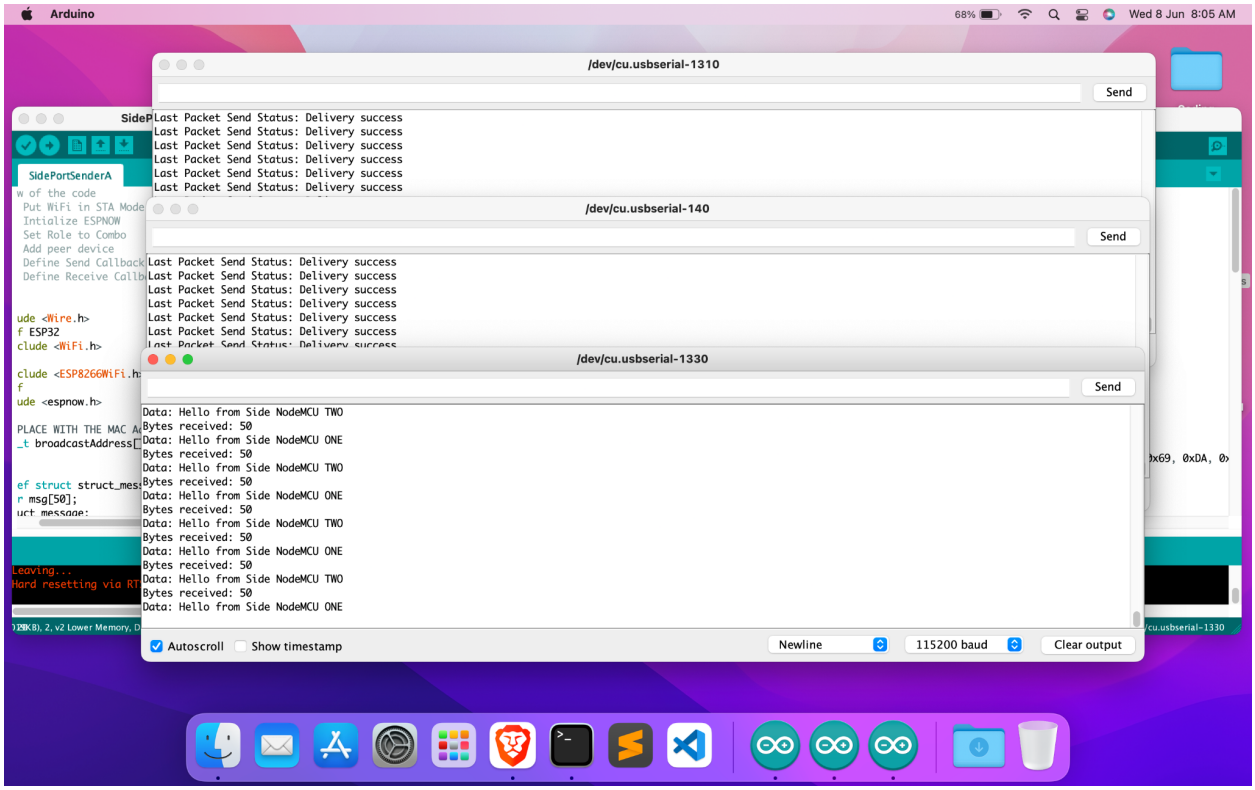
```
  #include <WiFi.h>
#else
  #include <ESP8266WiFi.h>
#endif
#include <espnow.h>

// REPLACE WITH THE MAC Address of your receiver
uint8_t broadcastAddress[] = {0xC8, 0xC9, 0xA3, 0x69, 0xDA, 0xCA};

typedef struct struct_message {
  char msg[50];
} struct_message;

// Create a struct_message called outgoingReadings to hold outgoing data
//struct_message outgoingReadings;

// Create a struct_message called incomingReadings to hold incoming data
struct_message incomingReadings;

// Variable to store if sending data was successful
String success;

// Callback when data is sent
//void OnDataSent(uint8_t *mac_addr, uint8_t sendStatus) {
//  Serial.print("Last Packet Send Status: ");
//  if (sendStatus == 0) {
//    Serial.println("Delivery success");
//  }
//  else {
//    Serial.println("Delivery fail");
//  }
//}

// Callback when data is received
void OnDataRecv(uint8_t * mac, uint8_t *incomingData, uint8_t len) {
  memcpy(&incomingReadings, incomingData, sizeof(incomingReadings));
  Serial.print("Bytes received: ");
  Serial.println(len);
  Serial.print("Data: "); Serial.println(incomingReadings.msg);
}


void setup() {
```

```
  // Init Serial Monitor
  Serial.begin(115200);

  // Set device as a Wi-Fi Station
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();

  // Init ESP-NOW
  if (esp_now_init() != 0) {
    Serial.println("Error initialising ESP-NOW");
    return;
  }

  // Set ESP-NOW Role
  esp_now_set_self_role(ESP_NOW_ROLE_COMBO);

  // Once ESPNow is successfully Init, we will register for Send CB to
  // get the status of Transmitted packet
  // esp_now_register_send_cb(OnDataSent);

  // Register peer
  esp_now_add_peer(broadcastAddress, ESP_NOW_ROLE_COMBO, 1, NULL, 0);


  // Register for a callback function that will be called when data is
received
  esp_now_register_recv_cb(OnDataRecv);
}

void loop() {

//  strcpy(outgoingReadings.msg, "Hello from Front NodeMCU");
//   esp_now_send(broadcastAddress, (uint8_t *) &outgoingReadings,
sizeof(outgoingReadings));

  delay(2000);
}
```

7th June

- One-to-One Communication of NodeMCUs
- Multicasting multiple NodeMCUs to a single NodeMCU





Code for One-Way receiver :

```
/*
  Flow of the code
  1 - Put WiFi in STA Mode
  2 - Initialize ESPNOW
  3 - Set Role to Combo
  4 - Add peer device
  5 - Define Send Callback Function
  6 - Define Receive Callback Function
*/

#include <Wire.h>
#ifdef ESP32
  #include <WiFi.h>
#else
  #include <ESP8266WiFi.h>
#endif
#include <espnow.h>
```

```
// REPLACE WITH THE MAC Address of your receiver
uint8_t broadcastAddress[] = {0xC8, 0xC9, 0xA3, 0x69, 0xDA, 0xCA};

typedef struct struct_message {
  char msg[50];
} struct_message;

// Create a struct_message called outgoingReadings to hold outgoing data
//struct_message outgoingReadings;

// Create a struct_message called incomingReadings to hold incoming data
struct_message incomingReadings;

// Variable to store if sending data was successful
String success;

// Callback when data is sent
//void OnDataSent(uint8_t *mac_addr, uint8_t sendStatus) {
//  Serial.print("Last Packet Send Status: ");
//  if (sendStatus == 0) {
//    Serial.println("Delivery success");
//  }
//  else {
//    Serial.println("Delivery fail");
//  }
//}

// Callback when data is received
void OnDataRecv(uint8_t * mac, uint8_t *incomingData, uint8_t len) {
  memcpy(&incomingReadings, incomingData, sizeof(incomingReadings));
  Serial.print("Bytes received: ");
  Serial.println(len);
  Serial.print("Data: "); Serial.println(incomingReadings.msg);
}


void setup() {
  // Init Serial Monitor
  Serial.begin(115200);

  // Set device as a Wi-Fi Station
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();
```

```
  // Init ESP-NOW
  if (esp_now_init() != 0) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }

  // Set ESP-NOW Role
  esp_now_set_self_role(ESP_NOW_ROLE_COMBO);

  // Once ESPNow is successfully Init, we will register for Send CB to
  // get the status of Transmitted packet
  // esp_now_register_send_cb(OnDataSent);

  // Register peer
  esp_now_add_peer(broadcastAddress, ESP_NOW_ROLE_COMBO, 1, NULL, 0);


  // Register for a callback function that will be called when data is
received
  esp_now_register_recv_cb(OnDataRecv);
}

void loop() {

//  strcpy(outgoingReadings.msg, "Hello from Front NodeMCU");
//   esp_now_send(broadcastAddress, (uint8_t *) &outgoingReadings,
sizeof(outgoingReadings));

  delay(2000);
}
```

Code for One-way sender :

```
/*
  Flow of the code
  1 - Put WiFi in STA Mode
  2 - Initialize ESPNOW
  3 - Set Role to Combo
  4 - Add peer device
  5 - Define Send Callback Function
```

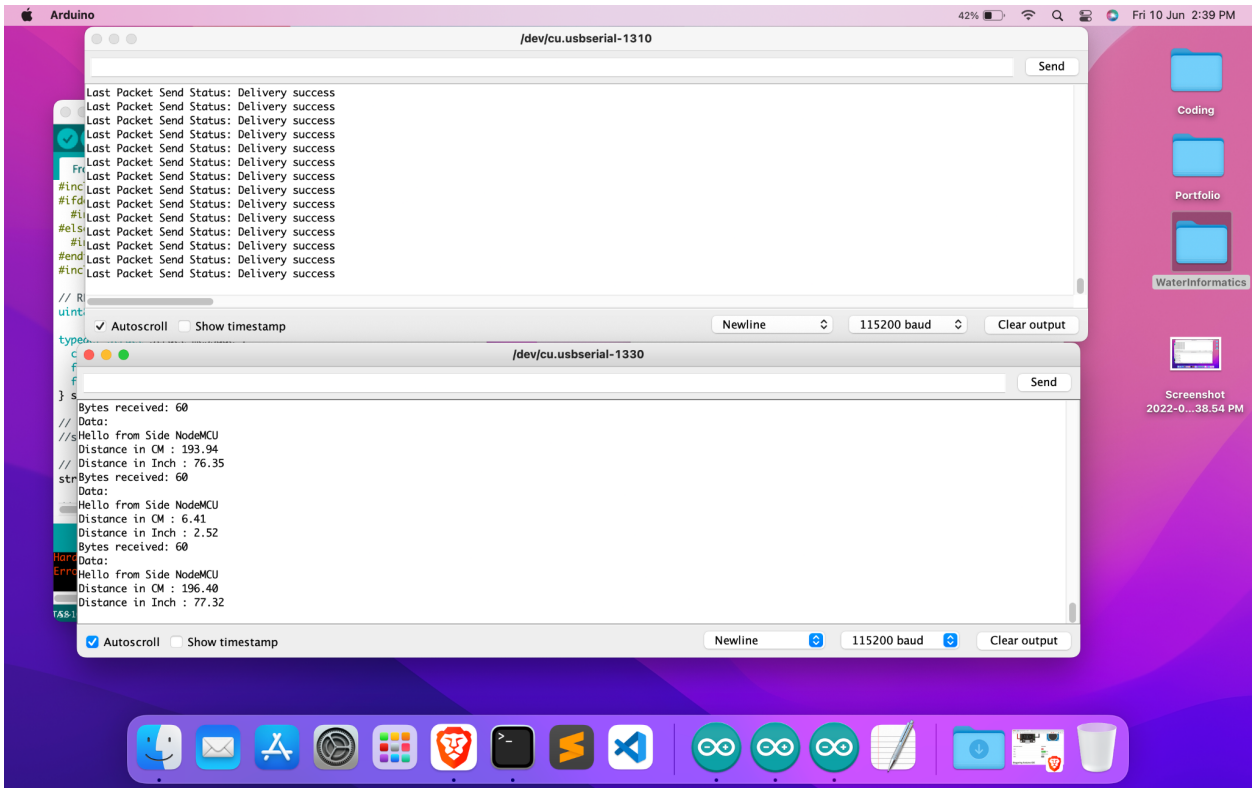
```
  6 - Define Receive Callback Function
*/

#include <Wire.h>
#ifdef ESP32
  #include <WiFi.h>
#else
  #include <ESP8266WiFi.h>
#endif
#include <espnow.h>

// REPLACE WITH THE MAC Address of your receiver
uint8_t broadcastAddress[] = {0xC8, 0xC9, 0xA3, 0x69, 0xDE, 0x1D};


typedef struct struct_message {
  char msg[50];
} struct_message;

// Create a struct_message called outgoingReadings to hold outgoing data
struct_message outgoingReadings;

// Create a struct_message called incomingReadings to hold incoming data
//struct_message incomingReadings;

// Variable to store if sending data was successful
String success;

// Callback when data is sent
void OnDataSent(uint8_t *mac_addr, uint8_t sendStatus) {
  Serial.print("Last Packet Send Status: ");
  if (sendStatus == 0) {
    Serial.println("Delivery success");
  }
  else {
    Serial.println("Delivery fail");
  }
}

// Callback when data is received
//void OnDataRecv(uint8_t * mac, uint8_t *incomingData, uint8_t len) {
//  memcpy(&incomingReadings, incomingData, sizeof(incomingReadings));
//  Serial.print("Bytes received: ");
```

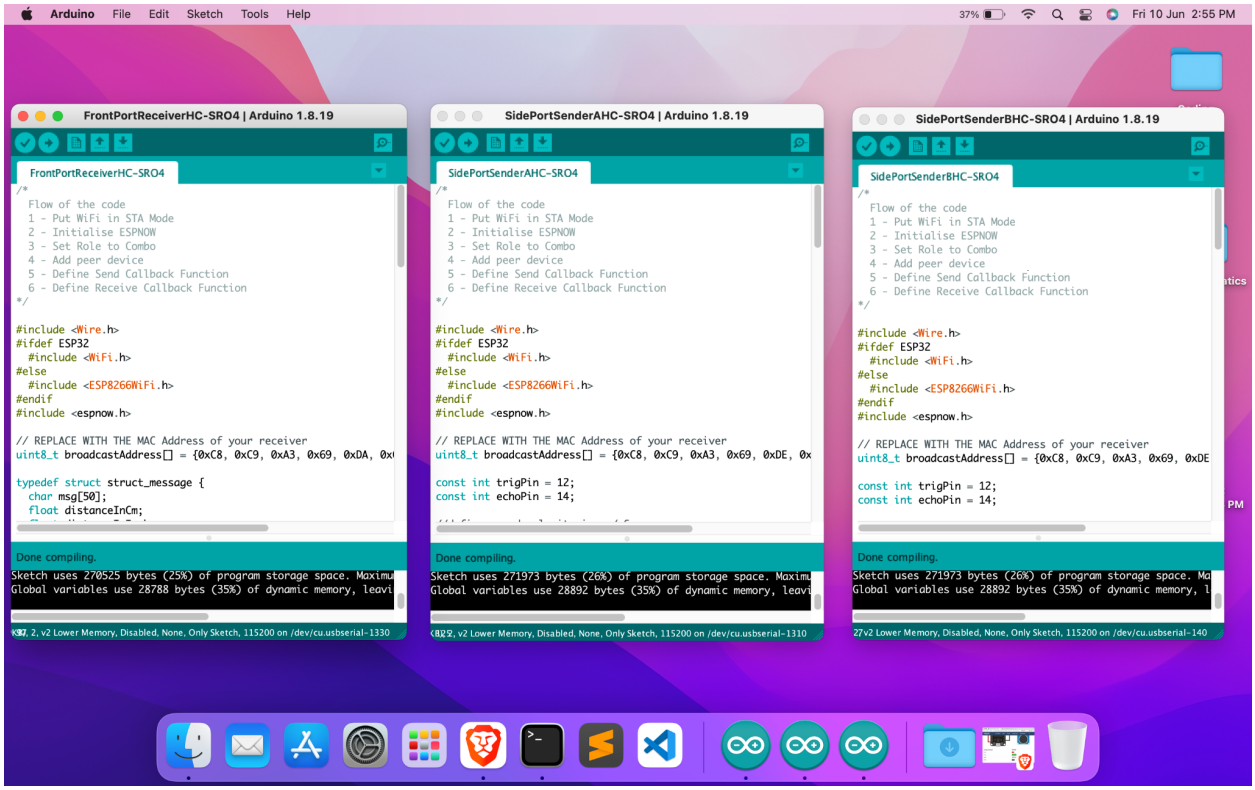
```
//  Serial.println(len);
//  Serial.print("Data: "); Serial.println(incomingReadings.msg);
//}


void setup() {
  // Init Serial Monitor
  Serial.begin(115200);

  // Set device as a Wi-Fi Station
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();

  // Init ESP-NOW
  if (esp_now_init() != 0) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }

  // Set ESP-NOW Role
  esp_now_set_self_role(ESP_NOW_ROLE_COMBO);

  // Once ESPNow is successfully Init, we will register for Send CB to
  // get the status of the Transmitted packet
  esp_now_register_send_cb(OnDataSent);

  // Register peer
  esp_now_add_peer(broadcastAddress, ESP_NOW_ROLE_COMBO, 1, NULL, 0);



  // Register for a callback function that will be called when data is
received
  // esp_now_register_recv_cb(OnDataRecv);
}

void loop() {

  strcpy(outgoingReadings.msg, "Hello from Side NodeMCU");
   esp_now_send(broadcastAddress, (uint8_t *) &outgoingReadings,
sizeof(outgoingReadings));

  delay(2000);
```

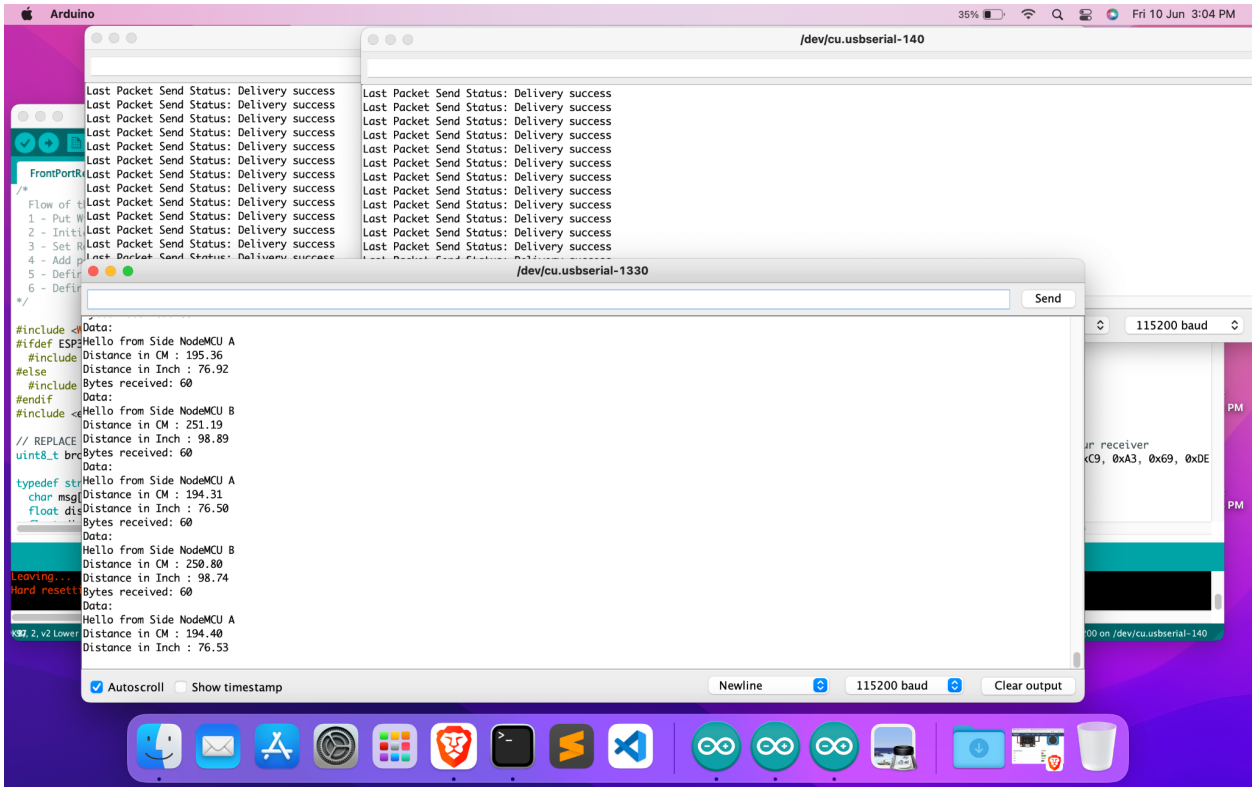
```
}
```

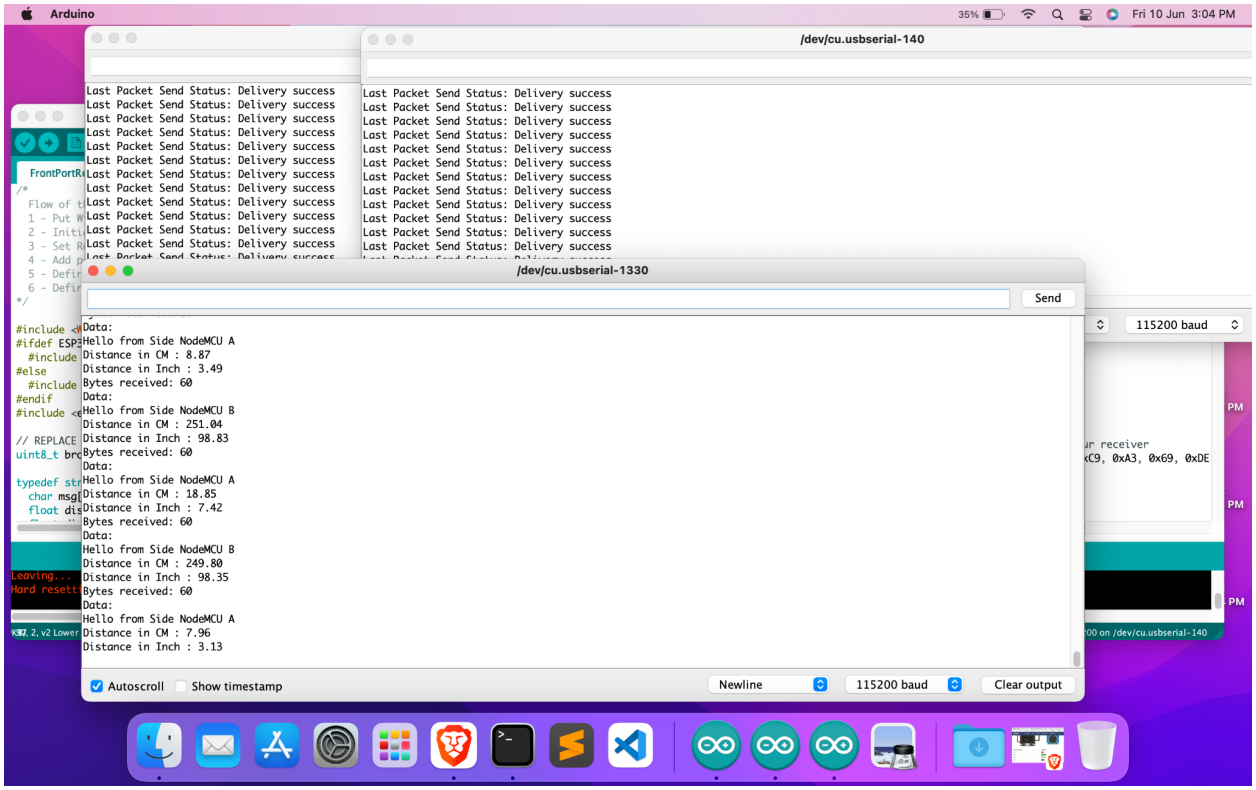
10 June:

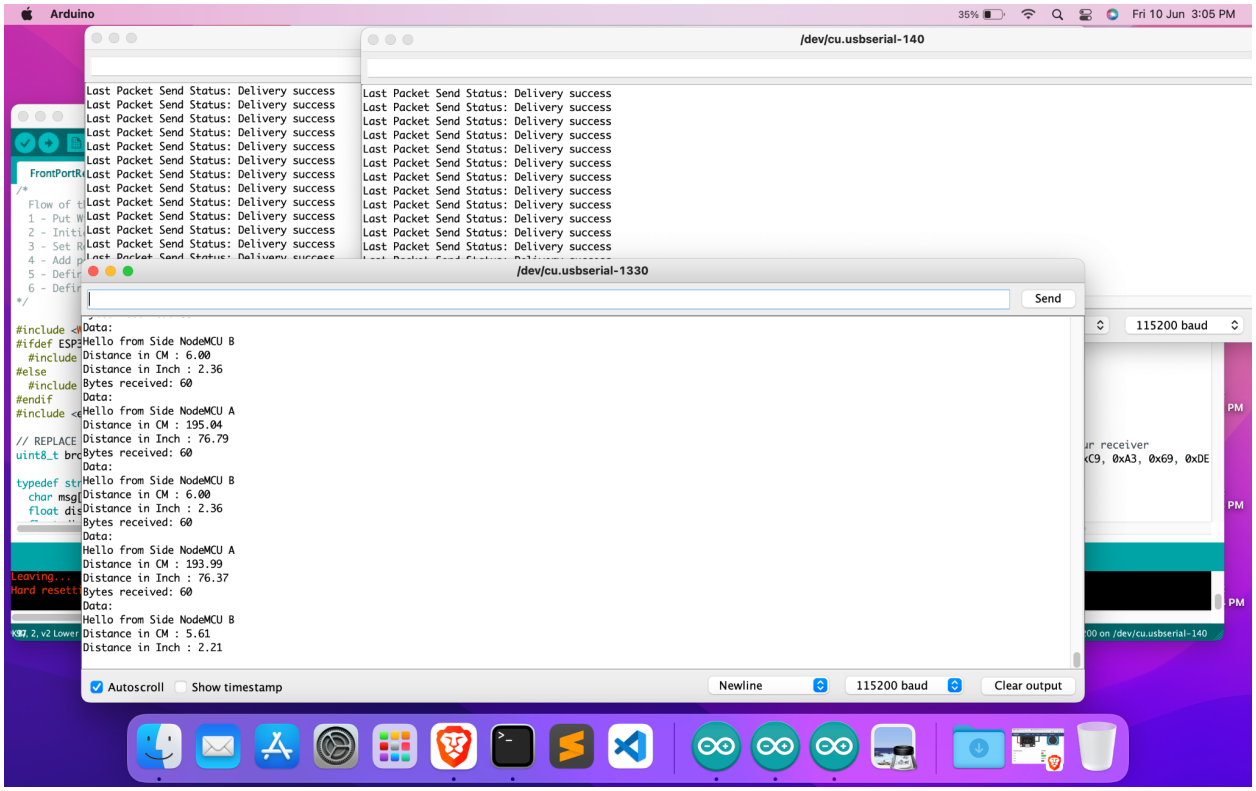
- Connected HC-SRO4 with NodeMCU and established one-to-one communication
- Connected HC-SRO4 with NodeMCU and established multicasting

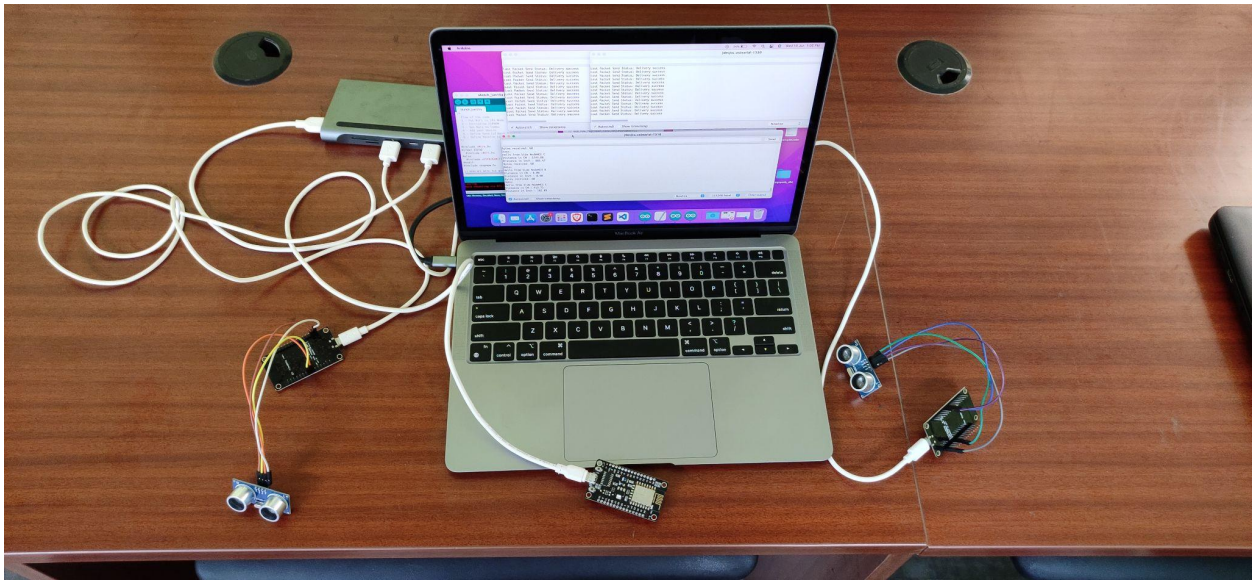



Arduino File Edit Sketch Tools Help
Fri 10 Jun 2:55 PM
FrontPortReceiverHC-SRO4 | Arduino 1.8.19
SidePortSenderAHC-SRO4 | Arduino 1.8.19
SidePortSenderBHC-SRO4 | Arduino 1.8.19
Done compiling.
Sketch uses 270525 bytes (25%) of program storage space.
Sketch uses 271973 bytes (26%) of program storage space.
Global variables use 28788 bytes (35%) of dynamic memory
Global variables use 28892 bytes (35%) of dynamic memory


Arduino
Fri 10 Jun 3:04 PM
/dev/cu.usbserial-140
/dev/cu.usbserial-1330
Send
Data:
Hello from Side NodeMCU A
Distance in CM : 195.36
Distance in Inch : 76.92
Bytes received: 60
Hello from Side NodeMCU B
Distance in CM : 251.19
Distance in Inch : 98.89
Hello from Side NodeMCU A
Distance in CM : 194.31
Distance in Inch : 76.50
Hello from Side NodeMCU B
Distance in CM : 250.80
Distance in Inch : 98.74
Hello from Side NodeMCU A
Distance in CM : 194.40
Distance in Inch : 76.53
Autoscroll
Show timestamp
Newline
115200 baud
Clear output


Arduino
Fri 10 Jun 3:04 PM
/dev/cu.usbserial-140
Last Packet Send Status: Delivery success
/dev/cu.usbserial-1330
Send
Data:
Hello from Side NodeMCU A
Distance in CM : 8.87
Distance in Inch : 3.49
Bytes received: 60
Data:
Hello from Side NodeMCU B
Distance in CM : 251.04
Distance in Inch : 98.83
Bytes received: 60
Data:
Hello from Side NodeMCU A
Distance in CM : 18.85
Distance in Inch : 7.42
Bytes received: 60
Data:
Hello from Side NodeMCU B
Distance in CM : 249.80
Distance in Inch : 98.35
Bytes received: 60
Data:
Hello from Side NodeMCU A
Distance in CM : 7.96
Distance in Inch : 3.13
Autoscroll
Show timestamp
Newline
115200 baud
Clear output


Arduino
Fri 10 Jun 3:05 PM
/dev/cu.usbserial-140
Last Packet Send Status: Delivery success
/dev/cu.usbserial-1330
Send
Data:
Hello from Side NodeMCU B
Distance in CM : 6.00
Distance in Inch : 2.36
Bytes received: 60
Data:
Hello from Side NodeMCU A
Distance in CM : 195.04
Distance in Inch : 76.79
Bytes received: 60
Data:
Hello from Side NodeMCU B
Distance in CM : 6.00
Distance in Inch : 2.36
Bytes received: 60
Data:
Hello from Side NodeMCU A
Distance in CM : 193.99
Distance in Inch : 76.37
Bytes received: 60
Data:
Hello from Side NodeMCU B
Distance in CM : 5.61
Distance in Inch : 2.21
Autoscroll
Show timestamp
Newline
115200 baud
Clear output



Code for Side-Port sender :

```
*
  Flow of the code
  1 - Put WiFi in STA Mode
  2 - Initialise ESPNOW
  3 - Set Role to Combo
  4 - Add peer device
  5 - Define Send Callback Function
  6 - Define Receive Callback Function
*/

#include <Wire.h>
#ifdef ESP32
  #include <WiFi.h>
#else
  #include <ESP8266WiFi.h>
#endif
#include <espnow.h>

// REPLACE WITH THE MAC Address of your receiver
uint8_t broadcastAddress[] = {0xC8, 0xC9, 0xA3, 0x69, 0xDE, 0x1D};
```

```
const int trigPin = 12;
const int echoPin = 14;

//define sound velocity in cm/uS
#define SOUND_VELOCITY 0.034
#define CM_TO_INCH 0.393701

long duration;
float distanceCm;
float distanceInch;

typedef struct struct_message {
  char msg[50];
  float distanceInCm;
  float distanceInInch;
} struct_message;

// Create a struct_message called outgoingReadings to hold outgoing data
struct_message outgoingReadings;

// Create a struct_message called incomingReadings to hold incoming data
//struct_message incomingReadings;

// Variable to store if sending data was successful
String success;

// Callback when data is sent
void OnDataSent(uint8_t *mac_addr, uint8_t sendStatus) {
  Serial.print("Last Packet Send Status: ");
  if (sendStatus == 0) {
    Serial.println("Delivery success");
  }
  else {
    Serial.println("Delivery fail");
  }
}

// Callback when data is received
//void OnDataRecv(uint8_t * mac, uint8_t *incomingData, uint8_t len) {
//  memcpy(&incomingReadings, incomingData, sizeof(incomingReadings));
//  Serial.print("Bytes received: ");
//  Serial.println(len);
```

```
//  Serial.print("Data: "); Serial.println(incomingReadings.msg);
//}


void setup() {
  // Init Serial Monitor
  Serial.begin(115200);

  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input

  // Set device as a Wi-Fi Station
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();

  // Init ESP-NOW
  if (esp_now_init() != 0) {
    Serial.println("Error initialising ESP-NOW");
    return;
  }

  // Set ESP-NOW Role
  esp_now_set_self_role(ESP_NOW_ROLE_COMBO);

  // Once ESPNow is successfully Init, we will register for Send CB to
  // get the status of Transmitted packet
  esp_now_register_send_cb(OnDataSent);

  // Register peer
  esp_now_add_peer(broadcastAddress, ESP_NOW_ROLE_COMBO, 1, NULL, 0);



  // Register for a callback function that will be called when data is
received
  // esp_now_register_recv_cb(OnDataRecv);
}

void loop() {

  // Clears the trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
```

```
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);

  // Calculate the distance
  distanceCm = duration * SOUND_VELOCITY/2;

  // Convert to inches
  distanceInch = distanceCm * CM_TO_INCH;

  strcpy(outgoingReadings.msg, "Hello from Side NodeMCU");
  outgoingReadings.distanceInCm = distanceCm;
  outgoingReadings.distanceInInch = distanceInch;

  esp_now_send(broadcastAddress, (uint8_t *) &outgoingReadings,
sizeof(outgoingReadings));

  delay(2000);
}
```

Code for Front-Port receiver :

```
*
  Flow of the code
  1 - Put WiFi in STA Mode
  2 - Initialise ESPNOW
  3 - Set Role to Combo
  4 - Add peer device
  5 - Define Send Callback Function
  6 - Define Receive Callback Function
*/

#include <Wire.h>
#ifdef ESP32
  #include <WiFi.h>
#else
  #include <ESP8266WiFi.h>
```

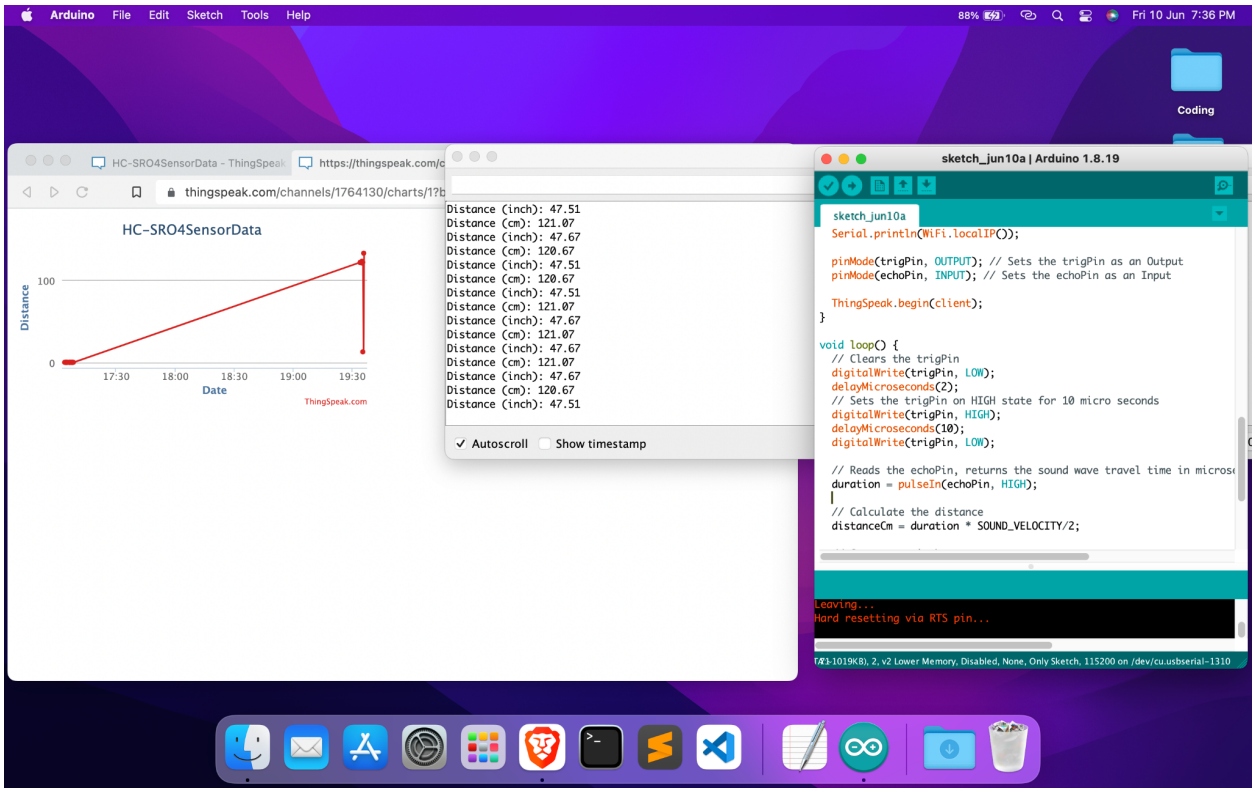
```
#endif
#include <espnow.h>

// REPLACE WITH THE MAC Address of your receiver
uint8_t broadcastAddress[] = {0xC8, 0xC9, 0xA3, 0x69, 0xDE, 0x1D};

const int trigPin = 12;
const int echoPin = 14;

//define sound velocity in cm/uS
#define SOUND_VELOCITY 0.034
#define CM_TO_INCH 0.393701

long duration;
float distanceCm;
float distanceInch;

typedef struct struct_message {
  char msg[50];
  float distanceInCm;
  float distanceInInch;
} struct_message;

// Create a struct_message called outgoingReadings to hold outgoing data
struct_message outgoingReadings;

// Create a struct_message called incomingReadings to hold incoming data
//struct_message incomingReadings;

// Variable to store if sending data was successful
String success;

// Callback when data is sent
void OnDataSent(uint8_t *mac_addr, uint8_t sendStatus) {
  Serial.print("Last Packet Send Status: ");
  if (sendStatus == 0) {
    Serial.println("Delivery success");
  }
  else {
    Serial.println("Delivery fail");
  }
}
```

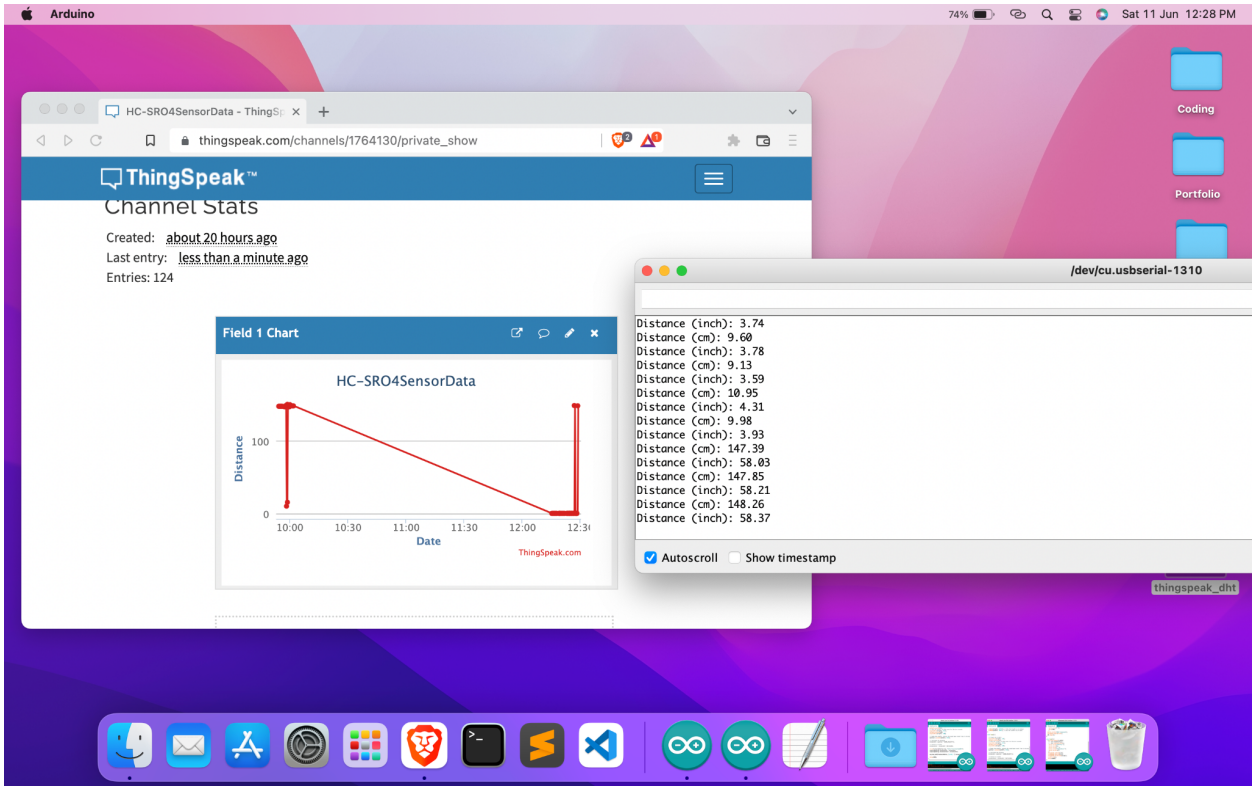
```
// Callback when data is received
//void OnDataRecv(uint8_t * mac, uint8_t *incomingData, uint8_t len) {
//  memcpy(&incomingReadings, incomingData, sizeof(incomingReadings));
//  Serial.print("Bytes received: ");
//  Serial.println(len);
//  Serial.print("Data: "); Serial.println(incomingReadings.msg);
//}


void setup() {
  // Init Serial Monitor
  Serial.begin(115200);

  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input

  // Set device as a Wi-Fi Station
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();

  // Init ESP-NOW
  if (esp_now_init() != 0) {
    Serial.println("Error initialising ESP-NOW");
    return;
  }

  // Set ESP-NOW Role
  esp_now_set_self_role(ESP_NOW_ROLE_COMBO);

  // Once ESPNow is successfully Init, we will register for Send CB to
  // get the status of Transmitted packet
  esp_now_register_send_cb(OnDataSent);

  // Register peer
  esp_now_add_peer(broadcastAddress, ESP_NOW_ROLE_COMBO, 1, NULL, 0);



  // Register for a callback function that will be called when data is
received
  // esp_now_register_recv_cb(OnDataRecv);
}
```

```
void loop() {

  // Clears the trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);

  // Calculate the distance
  distanceCm = duration * SOUND_VELOCITY/2;

  // Convert to inches
  distanceInch = distanceCm * CM_TO_INCH;

  strcpy(outgoingReadings.msg, "Hello from Side NodeMCU");
  outgoingReadings.distanceInCm = distanceCm;
  outgoingReadings.distanceInInch = distanceInch;

  esp_now_send(broadcastAddress, (uint8_t *) &outgoingReadings,
sizeof(outgoingReadings));

  delay(2000);
}
```

June 11 :

- Connected NodeMCU with HC-SRO4 sensor to send data to ThingSpeak Platform





Code :

Approach 1 :

```
#ifdef ESP32
  #include <WiFi.h>
#else
  #include <ESP8266WiFi.h>
#endif

#include <ThingSpeak.h>

WiFiClient client;

long myChannelNumber = 1764130;
const char myWriteAPIKey[] = "7F1U0PTLV5PNO5E2";

const char wifiName[] = "Manaspb";
const char password[] = "manas123";
```

```
const int trigPin = 12;
const int echoPin = 14;

//define sound velocity in cm/uS
#define SOUND_VELOCITY 0.034
#define CM_TO_INCH 0.393701

long duration;
float distanceCm;
float distanceInch;

void setup() {
  Serial.begin(115200); // Starts the serial communication

  WiFi.begin(wifiName,password);
  while(WiFi.status() != WL_CONNECTED)
  {
      delay(200);
      Serial.print("..");
  }

  Serial.println();
  Serial.println("NodeMCU is connected:");
  Serial.println(WiFi.localIP());

  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input

  ThingSpeak.begin(client);
}

void loop() {
  // Clears the trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Reads the echoPin, returns the sound wave travel time in microseconds
```

```
  duration = pulseIn(echoPin, HIGH);

  // Calculate the distance
  distanceCm = duration * SOUND_VELOCITY/2;

  // Convert to inches
  distanceInch = distanceCm * CM_TO_INCH;

  // Prints the distance on the Serial Monitor
  Serial.print("Distance (cm): ");
  Serial.println(distanceCm);
  Serial.print("Distance (inch): ");
  Serial.println(distanceInch);

  ThingSpeak.writeField(myChannelNumber,1,distanceCm,myWriteAPIKey);

  delay(1000);
}
```

Approach 2 :

```
#ifdef ESP32
  #include <WiFi.h>
#else
  #include <ESP8266WiFi.h>
#endif

#include <ThingSpeak.h>

WiFiClient client;

long myChannelNumber = 1764130;
const char myWriteAPIKey[] = "7F1U0PTLV5PNO5E2";

const char wifiName[] = "Manaspb";
const char password[] = "manas123";
const char ip[] = "184.106.153.149";

const int trigPin = 12;
const int echoPin = 14;
```

```
//define sound velocity in cm/uS
#define SOUND_VELOCITY 0.034
#define CM_TO_INCH 0.393701

long duration;
float distanceCm;
float distanceInch;

void setup() {
  Serial.begin(115200); // Starts the serial communication

  WiFi.begin(wifiName,password);
  while(WiFi.status() != WL_CONNECTED)
  {
      delay(200);
      Serial.print("..");
  }

  Serial.println();
  Serial.println("NodeMCU is connected:");
  Serial.println(WiFi.localIP());

  ThingSpeak.begin(client);

  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
}

void loop() {

  // Clears the trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);

  // Calculate the distance
```

```
  distanceCm = duration * SOUND_VELOCITY/2;

  // Convert to inches
  distanceInch = distanceCm * CM_TO_INCH;

  // Prints the distance on the Serial Monitor
  Serial.print("Distance (cm): ");
  Serial.println(distanceCm);
  Serial.print("Distance (inch): ");
  Serial.println(distanceInch);

  if(client.connect(ip,80)){
  ThingSpeak.setField(1,distanceCm);
  ThingSpeak.writeFields(myChannelNumber,myWriteAPIKey);
  client.stop();
  delay(3000);
  }
}
```

June 16:

- Learned basic syntax of python
- Learned about NumPy library

June 22:

- Learned about pandas, matplotlib, and seaborn libraries

June 24:

- Started with Machine Learning
- Learned Linear Regression

June 26:

- Learned Decision Trees and Random Forest

June 30:

- Learned KNN and SVM

July 4:

- Made a project on Diabetes Prediction System using Support Vector Machine algorithm
- Deployed the model on the web using Streamlit service
- Link to the code
- Link to the prediction website

July 15:

- Developed code for detecting leakage in a water tank using HC-SRO4 sensor and water flow sensor
- Link to the code

Code for Side-Port water flow sensor:

```
#include <Wire.h>
#ifdef ESP32
  #include <WiFi.h>
#else
  #include <ESP8266WiFi.h>
#endif
#include <espnow.h>

// REPLACE WITH THE MAC Address of your receiver
```

```
uint8_t broadcastAddress[] = {0xC8, 0xC9, 0xA3, 0x69, 0xDE, 0x1D};

#define SENSOR  D4
long currentMillis = 0;
long previousMillis = 0;
int interval = 1000;
float calibrationFactor = 4.5;
volatile byte pulseCount;
byte pulse1Sec = 0;
float flowRate;
unsigned int flowMilliLitres;
unsigned long totalMilliLitres;
void IRAM_ATTR pulseCounter()
{
  pulseCount++;
}

typedef struct struct_message {
  int id;
  char msg[50];
  unsigned long totalMillisFlown;
  float distanceInCm;
  float distanceInInch;
} struct_message;

// Create a struct_message called outgoingReadings to hold outgoing data
struct_message outgoingReadings;

// Create a struct_message called incomingReadings to hold incoming data
//struct_message incomingReadings;

// Variable to store if sending data was successful
String success;

// Callback when data is sent
void OnDataSent(uint8_t *mac_addr, uint8_t sendStatus) {
  Serial.print("Last Packet Send Status: ");
  if (sendStatus == 0) {
    Serial.println("Delivery success");
  }
  else {
    Serial.println("Delivery fail");
  }
```

```
}

// Callback when data is received
//void OnDataRecv(uint8_t * mac, uint8_t *incomingData, uint8_t len) {
//  memcpy(&incomingReadings, incomingData, sizeof(incomingReadings));
//  Serial.print("Bytes received: ");
//  Serial.println(len);
//  Serial.print("Data: "); Serial.println(incomingReadings.msg);
//}


void setup() {
  // Init Serial Monitor
  Serial.begin(115200);

  pinMode(SENSOR, INPUT_PULLUP);
  pulseCount = 0;
  flowRate = 0.0;
  flowMilliLitres = 0;
  totalMilliLitres = 0;
  previousMillis = 0;
  attachInterrupt(digitalPinToInterrupt(SENSOR), pulseCounter, FALLING);

  // Set device as a Wi-Fi Station
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();

  // Init ESP-NOW
  if (esp_now_init() != 0) {
    Serial.println("Error initialising ESP-NOW");
    return;
  }

  // Set ESP-NOW Role
  esp_now_set_self_role(ESP_NOW_ROLE_COMBO);

  // Once ESPNow is successfully Init, we will register for Send CB to
  // get the status of Transmitted packet
  esp_now_register_send_cb(OnDataSent);

  // Register peer
  esp_now_add_peer(broadcastAddress, ESP_NOW_ROLE_COMBO, 1, NULL, 0);
```

```
  // Register for a callback function that will be called when data is
received
  // esp_now_register_recv_cb(OnDataRecv);
}

void loop() {

  currentMillis = millis();
  if (currentMillis - previousMillis > interval) {
    pulse1Sec = pulseCount;
    pulseCount = 0;
    flowRate = ((1000.0 / (millis() - previousMillis)) * pulse1Sec) /
calibrationFactor;
    previousMillis = millis();
    flowMilliLitres = (flowRate / 60) * 1000;
    totalMilliLitres += flowMilliLitres;

  strcpy(outgoingReadings.msg, "Hello from Side NodeMCU wih WaterFlow
Sensor");
  outgoingReadings.totalMillisFlown = totalMilliLitres;
  outgoingReadings.id = 1;

  esp_now_send(broadcastAddress, (uint8_t *) &outgoingReadings,
sizeof(outgoingReadings));

  delay(2000);
  }
}
```

Code for Side-Port HC-SRO4 sensor:

```
#include <Wire.h>
#ifdef ESP32
  #include <WiFi.h>
#else
  #include <ESP8266WiFi.h>
#endif
#include <espnow.h>

// REPLACE WITH THE MAC Address of your receiver
```

```
uint8_t broadcastAddress[] = {0xC8, 0xC9, 0xA3, 0x69, 0xDE, 0x1D};

const int trigPin = 12;
const int echoPin = 14;

//define sound velocity in cm/uS
#define SOUND_VELOCITY 0.034
#define CM_TO_INCH 0.393701

long duration;
float distanceCm;
float distanceInch;

typedef struct struct_message {
  int id;
  char msg[50];
  unsigned long totalMillisFlown;
  float distanceInCm;
  float distanceInInch;
} struct_message;

// Create a struct_message called outgoingReadings to hold outgoing data
struct_message outgoingReadings;

// Create a struct_message called incomingReadings to hold incoming data
//struct_message incomingReadings;

// Variable to store if sending data was successful
String success;

// Callback when data is sent
void OnDataSent(uint8_t *mac_addr, uint8_t sendStatus) {
  Serial.print("Last Packet Send Status: ");
  if (sendStatus == 0) {
    Serial.println("Delivery success");
  }
  else {
    Serial.println("Delivery fail");
  }
}

// Callback when data is received
//void OnDataRecv(uint8_t * mac, uint8_t *incomingData, uint8_t len) {
```

```
//  memcpy(&incomingReadings, incomingData, sizeof(incomingReadings));
//  Serial.print("Bytes received: ");
//  Serial.println(len);
//  Serial.print("Data: "); Serial.println(incomingReadings.msg);
//}


void setup() {
  // Init Serial Monitor
  Serial.begin(115200);

  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input

  // Set device as a Wi-Fi Station
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();

  // Init ESP-NOW
  if (esp_now_init() != 0) {
    Serial.println("Error initialising ESP-NOW");
    return;
  }



// Set ESP-NOW Role
  esp_now_set_self_role(ESP_NOW_ROLE_COMBO);

  // Once ESPNow is successfully Init, we will register for Send CB to
  // get the status of Transmitted packet
  esp_now_register_send_cb(OnDataSent);

  // Register peer
  esp_now_add_peer(broadcastAddress, ESP_NOW_ROLE_COMBO, 1, NULL, 0);



  // Register for a callback function that will be called when data is
received
  // esp_now_register_recv_cb(OnDataRecv);
}
```

```
void loop() {

  // Clears the trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);

  // Calculate the distance
  distanceCm = duration * SOUND_VELOCITY/2;

  // Convert to inches
  distanceInch = distanceCm * CM_TO_INCH;

  strcpy(outgoingReadings.msg, "Hello from Side NodeMCU with HC-SRO4
sensor");
  outgoingReadings.distanceInCm = distanceCm;
  outgoingReadings.distanceInInch = distanceInch;
  outgoingReadings.id = 0;

  esp_now_send(broadcastAddress, (uint8_t *) &outgoingReadings,
sizeof(outgoingReadings));



delay(2000);
}
```

Code for front port Water flow and HC-SRO4 sensor:

```
#include <Wire.h>
#ifdef ESP32
  #include <WiFi.h>
#else
  #include <ESP8266WiFi.h>
#endif
#include <espnow.h>
```

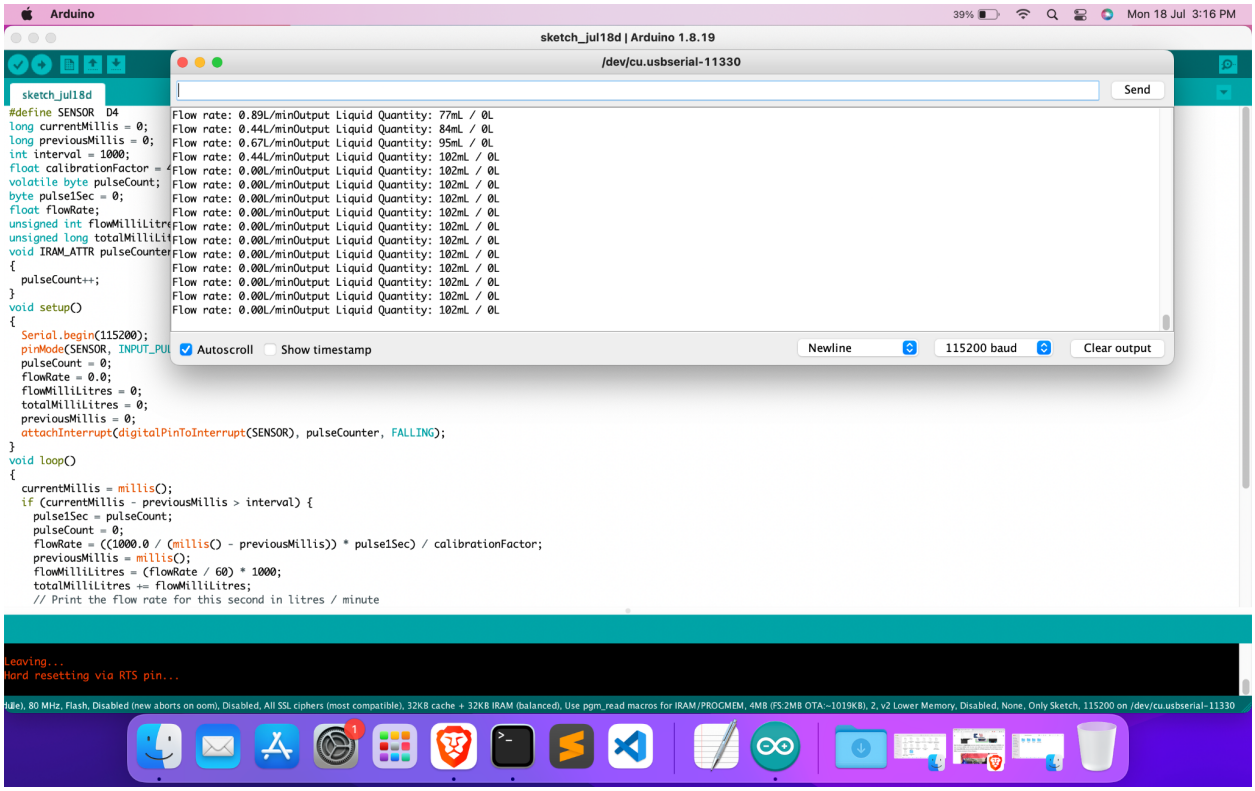
```
// REPLACE WITH THE MAC Address of your receiver
uint8_t broadcastAddress[] = {0xC8, 0xC9, 0xA3, 0x69, 0xDA, 0xCA};

#define PI 3.14
#define RADIUS 5
#define EPSILON 10

float volumeFromWaterSensor = 0;
float volumeFromDepthSensor = 0;
float previousHeight = 0.0;
float currentHeight = 0.0;
float heightDifference = 0.0;

typedef struct struct_message {
  int id;
  char msg[50];
  unsigned long totalMillisFlown;
  float distanceInCm;
  float distanceInInch;
} struct_message;

// Create a struct_message called outgoingReadings to hold outgoing data
//struct_message outgoingReadings;

// Create a struct_message called incomingReadings to hold incoming data
struct_message incomingReadings;

// Variable to store if sending data was successful
String success;

// Callback when data is sent
//void OnDataSent(uint8_t *mac_addr, uint8_t sendStatus) {
//  Serial.print("Last Packet Send Status: ");
//  if (sendStatus == 0) {
//    Serial.println("Delivery success");
//  }
//  else {
//    Serial.println("Delivery fail");
//  }
//}

// Callback when data is received
void OnDataRecv(uint8_t * mac, uint8_t *incomingData, uint8_t len) {
```

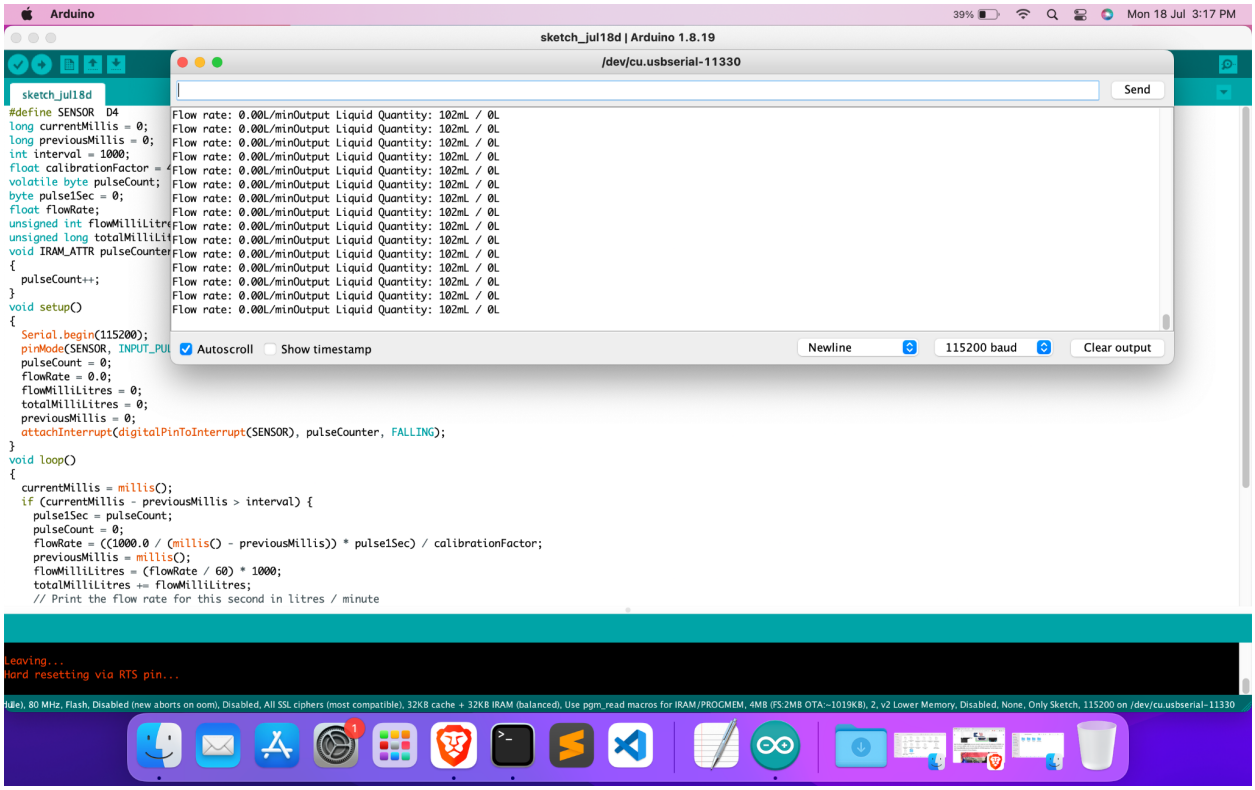
```
  memcpy(&incomingReadings, incomingData, sizeof(incomingReadings));
  Serial.print("Bytes received: ");
  Serial.println(len);

  if(millis()%10000==0)
  {
        if(abs(volumeFromDepthSensor-volumeFromWaterSensor) > EPSILON)
          Serial.print("There is a leakage");
        else
          Serial.print("There is no leakage");

        volumeFromDepthSensor = 0;
        volumeFromWaterSensor = 0;
  }
  if(incomingReadings.id==0)
  {
      currentHeight = incomingReadings.distanceInCm*10;
      heightDifference = currentHeight-previousHeight;
      previousHeight = currentHeight;

      volumeFromDepthSensor +=  PI*RADIUS*RADIUS*heightDifference;
  }

  else if(incomingReadings.id==1)
  {
      volumeFromWaterSensor = (float)
incomingReadings.totalMillisFlown*1000.0;
  }

  else
  {
      // Do nothing
  }
}


void setup() {
  // Init Serial Monitor
  Serial.begin(115200);

  // Set device as a Wi-Fi Station
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();
```

```
  // Init ESP-NOW
  if (esp_now_init() != 0) {
    Serial.println("Error initialising ESP-NOW");
    return;
  }

  // Set ESP-NOW Role
  esp_now_set_self_role(ESP_NOW_ROLE_COMBO);

  // Once ESPNow is successfully Init, we will register for Send CB to
  // get the status of Transmitted packet
  // esp_now_register_send_cb(OnDataSent);

  // Register peer
  esp_now_add_peer(broadcastAddress, ESP_NOW_ROLE_COMBO, 1, NULL, 0);


  // Register for a callback function that will be called when data is
received
  esp_now_register_recv_cb(OnDataRecv);
}

void loop() {

//  strcpy(outgoingReadings.msg, "Hello from Front NodeMCU");
//   esp_now_send(broadcastAddress, (uint8_t *) &outgoingReadings,
sizeof(outgoingReadings));

  delay(2000);
}
```

July 18:

- Read data using water flow sensor and NodeMCU





Code for water flow sensor and NodeMCU:

```
#define SENSOR  D4
long currentMillis = 0;
long previousMillis = 0;
int interval = 1000;
float calibrationFactor = 4.5;
volatile byte pulseCount;
byte pulse1Sec = 0;
float flowRate;
unsigned int flowMilliLitres;
unsigned long totalMilliLitres;
void IRAM_ATTR pulseCounter()
{
  pulseCount++;
}
void setup()
{
  Serial.begin(115200);
  pinMode(SENSOR, INPUT_PULLUP);
```

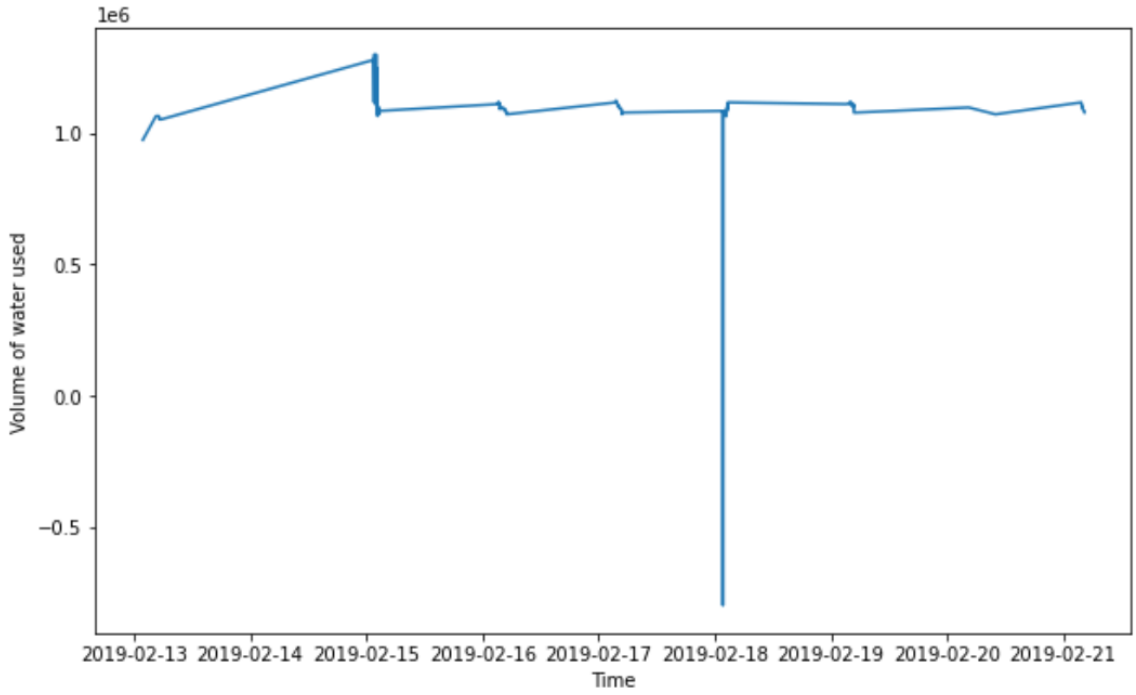
```
  pulseCount = 0;
  flowRate = 0.0;
  flowMilliLitres = 0;
  totalMilliLitres = 0;
  previousMillis = 0;
  attachInterrupt(digitalPinToInterrupt(SENSOR), pulseCounter, FALLING);
}
void loop()
{
  currentMillis = millis();
  if (currentMillis - previousMillis > interval) {
    pulse1Sec = pulseCount;
    pulseCount = 0;
    flowRate = ((1000.0 / (millis() - previousMillis)) * pulse1Sec) /
calibrationFactor;
    previousMillis = millis();
    flowMilliLitres = (flowRate / 60) * 1000;
    totalMilliLitres += flowMilliLitres;
    // Print the flow rate for this second in litres / minute
    Serial.print("Flow rate: ");
    Serial.print(flowRate);  // Print the integer part of the variable
    Serial.print("L/min");
    // Print the cumulative total of litres flowed since starting
    Serial.print("Output Liquid Quantity: ");
    Serial.print(totalMilliLitres);
    Serial.print("mL / ");
    Serial.print(totalMilliLitres / 1000);
    Serial.println("L");
  }
}
```

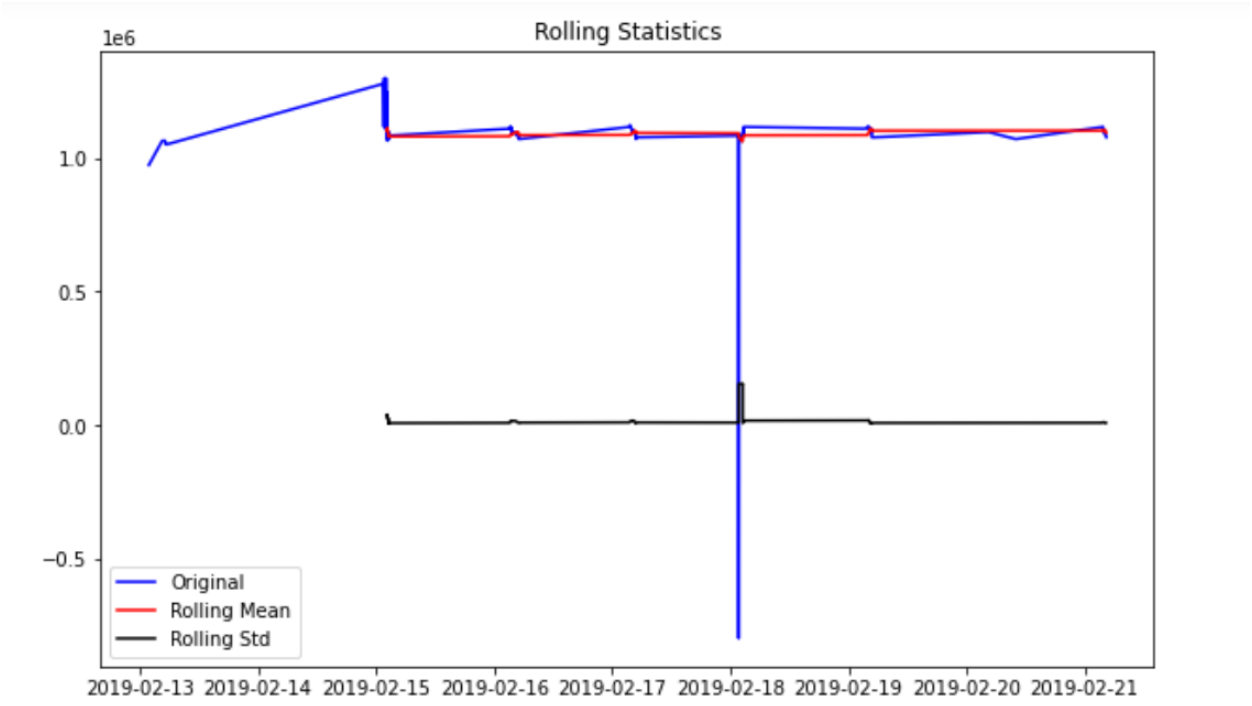
July 25:

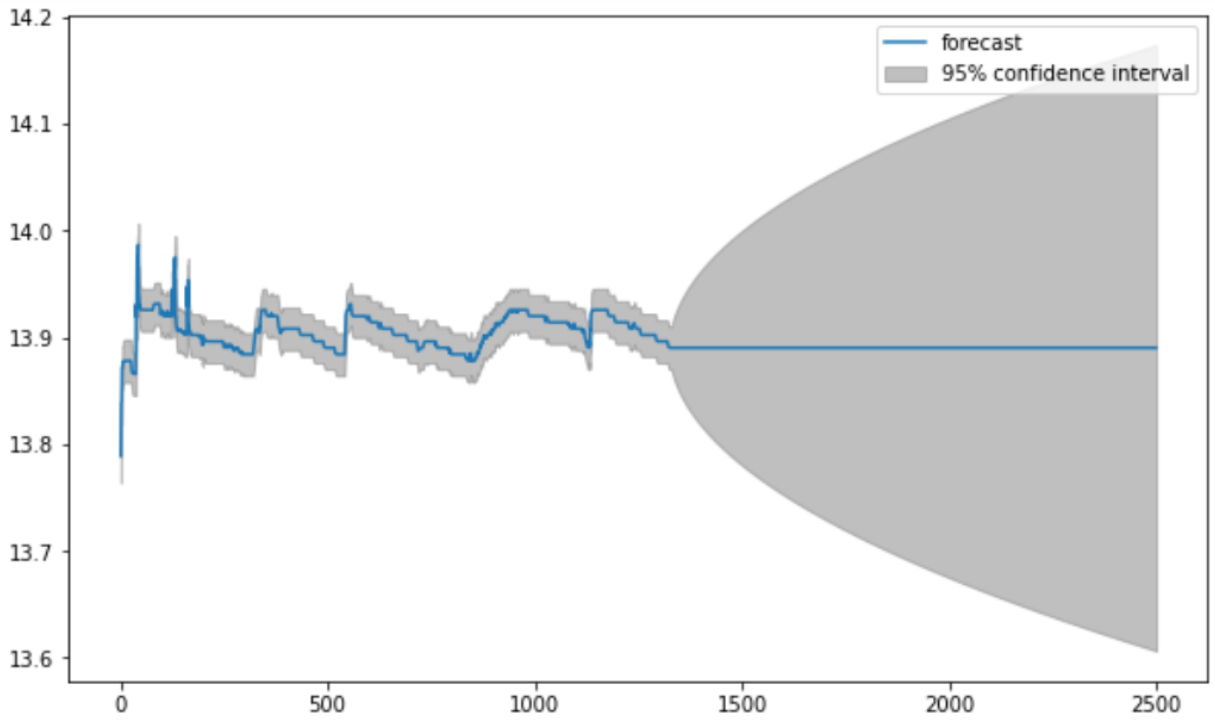
- Read and understood the the approach used in the research paper provided to me

July 29:

- Came up with a future forecasting model using ARIMA
- The detailed code and approach can be found in this Jupyter Notebook







Forecasting the Future Water usage