EG114728 HD in AI and Robotics

MBS4544 Robot Sensing and Vision

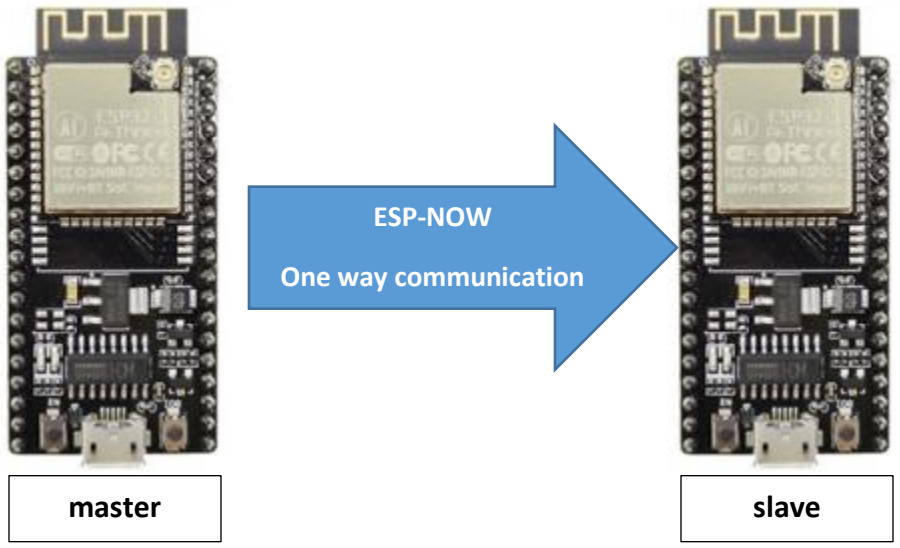
Objectives and learning outcomes

In the end of this section, you will be able to:

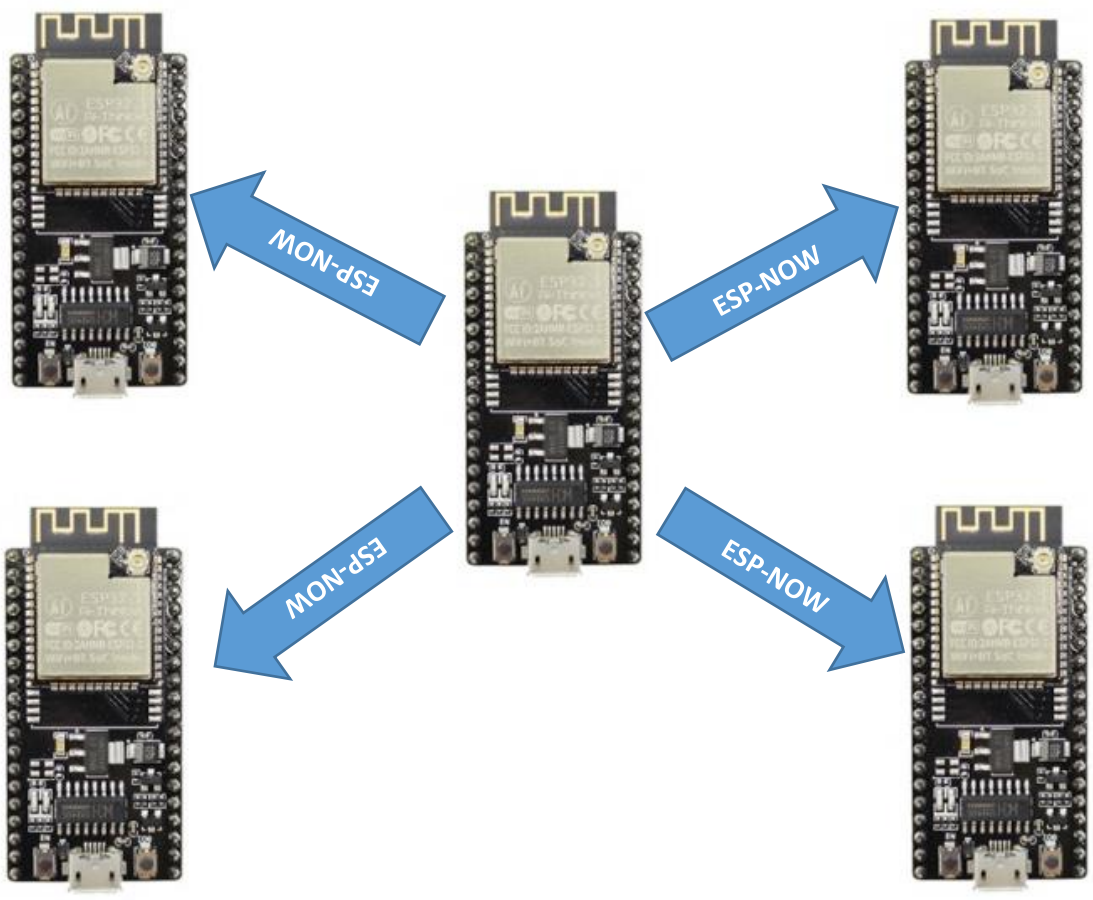
1. **Communicate wirelessly between two or more microcontrollers (ESP32) using a communication protocol called ESP-NOW**
2. **Develop test codes on two ESP32 to communicate with each other wirelessly via ESP-NOW protocol**
3. **Understand how to get measurement from the ToF sensor (VL53L0X)**
4. **Develop ESP32 codes to incorporate ToF sensor into ESP32-NOW master sketch and send data to another ESP32 (slave) using ESP-NOW protocol**
5. **Develop ESP32 codes to get data from IMU sensor (MPU9250 or MPU6050) and drive the robot car that is connected and controlled by another ESP32 using ESP-NOW protocol**

ESP-NOW One-Way Communication

This configuration is easy to implement and it is great to send data from one board to the other like sensor readings or ON and OFF commands to control GPIOs.



One ESP32 board sending the same or different commands to different ESP32 boards. This configuration is ideal to build something like a remote control. You can have several ESP32 boards around the house that are controlled by one main ESP32 board.



The middle ESP32 can be acted as slave and receiving data from several ESP32 boards. This configuration allows you to collect data from several sensor nodes into a single ESP32 board.

ESP-NOW One-Way Communication

With ESP-NOW, each board can be a sender and a receiver at the same time. So, you can establish two-way communication between boards.

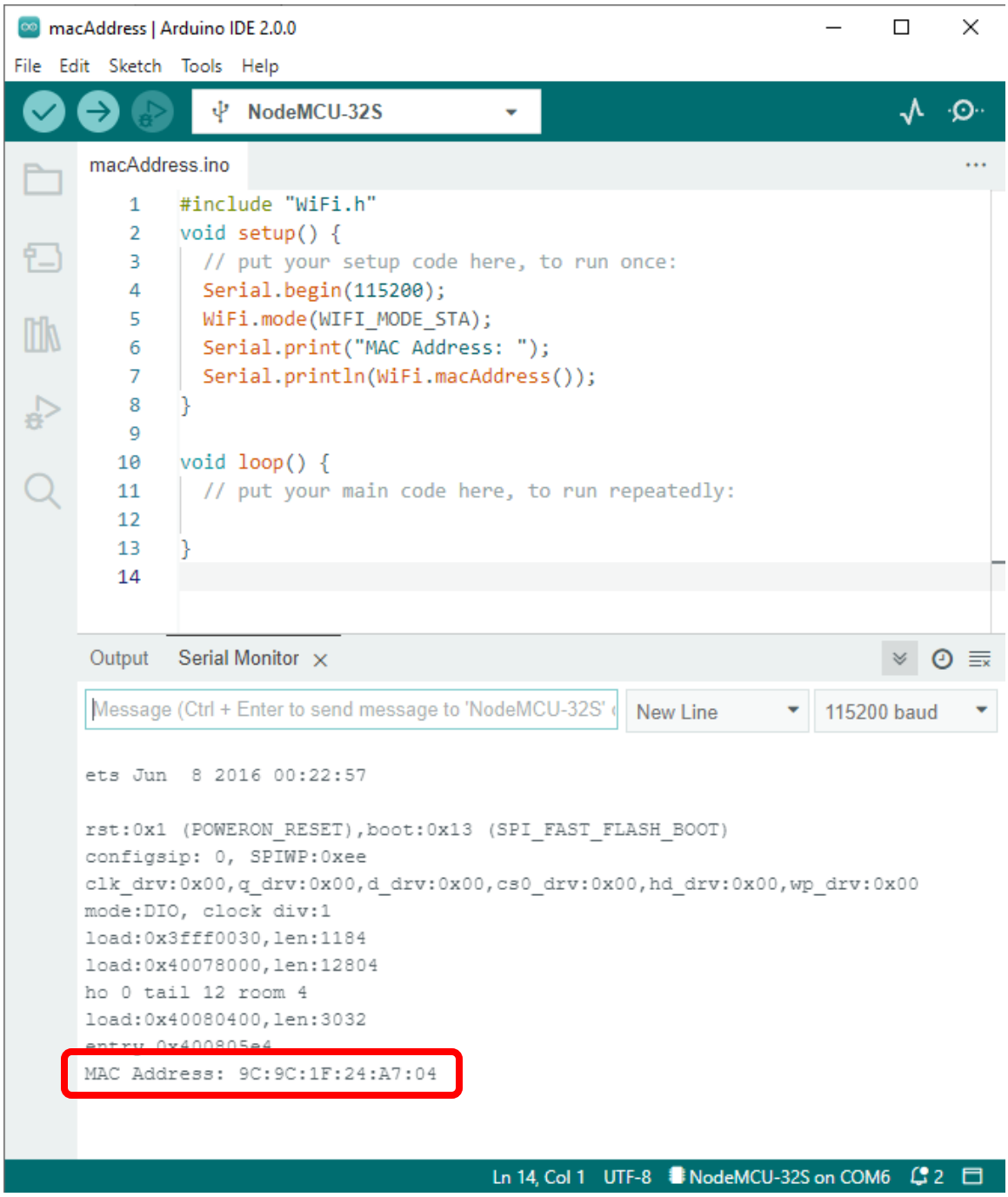
ESP32: Getting Board MAC Address

To communicate via ESP-NOW, you need to know the MAC (**Media Access Control**) Address of the ESP32 receiver. **Each ESP32 has a unique MAC Address** and that's how we identify each board to send data to it using ESP-NOW.

The following code will get your ESP32 MAC address:

```
#include "WiFi.h"
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  WiFi.mode(WIFI_MODE_STA);
  Serial.print("MAC Address: ");
  Serial.println(WiFi.macAddress());
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

After uploading the code, open the Serial Monitor at a baud rate of 115200 and press the ESP32 RST/EN button. The MAC address should be printed as follows:



ESP-NOW Example 1: Simple one-way point to point communication

Let's use a simple project that demonstrate how to send a message from one ESP32 to another. One ESP32 will be the "sender/master" and the other ESP32 will be the "receiver/slave".

In the sender/master sketch, we should include:

1. Initialize ESP-NOW;
2. Register a callback function upon sending data – the OnDataSent function will be executed when a message is sent. This can tell us if the message was successfully delivered or not;
3. Add a peer device (the receiver). For this, you need to know the receiver MAC address;
4. Send a message to the peer device.

On the receiver/slave side, the sketch should include:

1. Initialize ESP-NOW;
2. Register for a receive callback function (OnDataRecv). This is a function that will be executed when a message is received.
3. Inside that callback function, save the message into a variable to execute any task with that information.

ESP-NOW works with callback functions that are called when a device receives a message or when a message is sent.

A summary of the most essential ESP-NOW functions:

Function Name and Description
esp_now_init() - Initializes ESP-NOW. You must initialize Wi-Fi before initializing ESP-NOW.
esp_now_add_peer() - Call this function to pair a device and pass as an argument the peer MAC address.
esp_now_send() - Send data with ESP-NOW.
esp_now_register_send_cb() - Register a callback function that is triggered upon sending data. When a message is sent, a function is called – this function returns whether the delivery was successful or not.
esp_now_register_rcv_cb() - Register a callback function that is triggered upon receiving data. When data is received via ESP-NOW, a function is called.

ESP32 Sender/Master Sketch

```
#include <esp_now.h>
#include <WiFi.h>

#define CHANNEL 1
esp_now_peer_info_t slave;
```

```
uint8_t broadcastAddress[] = {0x8C,0xCE,0x4E,0xA6,0x41,0xC4}; // hard
code broadcast address for receiver

uint8_t data = 0;

// Structure example to send data
// Must match the receiver structure
typedef struct struct_message {
  uint8_t data;
} struct_message;

// Create a struct_message called myData
struct_message myData;

// callback when data is sent from Master to Slave
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  Serial.print("\r\nLast Packet Send Status:\t");
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Delivery Success" :
"Delivery Fail");
  Serial.print("Data Sent: ");
  Serial.println(data);
}

void setup() {
  Serial.begin(115200); // Init Serial Monitor
  WiFi.mode(WIFI_STA); // Set device as a Wi-Fi Station
  // Init ESP-NOW
  if (esp_now_init() != ESP_OK){
    Serial.println("Error init");
    return;
  }
  // Once ESPNow is successfully Init, we will register for Send CB to
  // get the status of Trasnmitted packet
  esp_now_register_send_cb(OnDataSent);
// Register peer
  memcpy(slave.peer_addr, broadcastAddress, 6);
  slave.channel = CHANNEL; // pick a channel
  slave.encrypt = false; // no encryption
  // Add peer
  if (esp_now_add_peer(&slave) !=ESP_OK){
    Serial.println("Failed to add peer");
    return;
  }
}
```

```
void loop() {
  myData.data = data;
  esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &myData,
sizeof(myData));
  if (result == ESP_OK) {
    Serial.println("Sent with success");
  }
  else {
    Serial.println("Error sending the data");
  }
  data++;
  delay(1000);
}
```

ESP32 Receiver/Slave Sketch

```
#include <esp_now.h>
#include <WiFi.h>

#define CHANNEL 1

uint8_t data;
uint16_t newData;

typedef struct struct_message {
  uint8_t data;
} struct_message;

// Create a struct_message called myData
struct_message myData;

// callback when data is recv from Master
void OnDataRecv(const uint8_t *mac_addr, const uint8_t *incomingData,
int data_len) {
  memcpy(&myData, incomingData, sizeof(myData));
  Serial.print("Received  :");
  Serial.println(myData.data);
  newData = myData.data * 10;
```

```
}

void setup() {
  Serial.begin(115200);// Initialize Serial Monitor
  WiFi.mode(WIFI_STA); // Set device as a Wi-Fi Station
  if (esp_now_init() != ESP_OK){
    Serial.println("err init....");
    return;
  }
  // Once ESPNow is successfully Init, we will register for recv CB to
  // get recv packer info
  esp_now_register_recv_cb(OnDataRecv);
}

void loop() {
 Serial.print("Data manipulated: ");
 Serial.println(newData);
 delay(200);
}
```

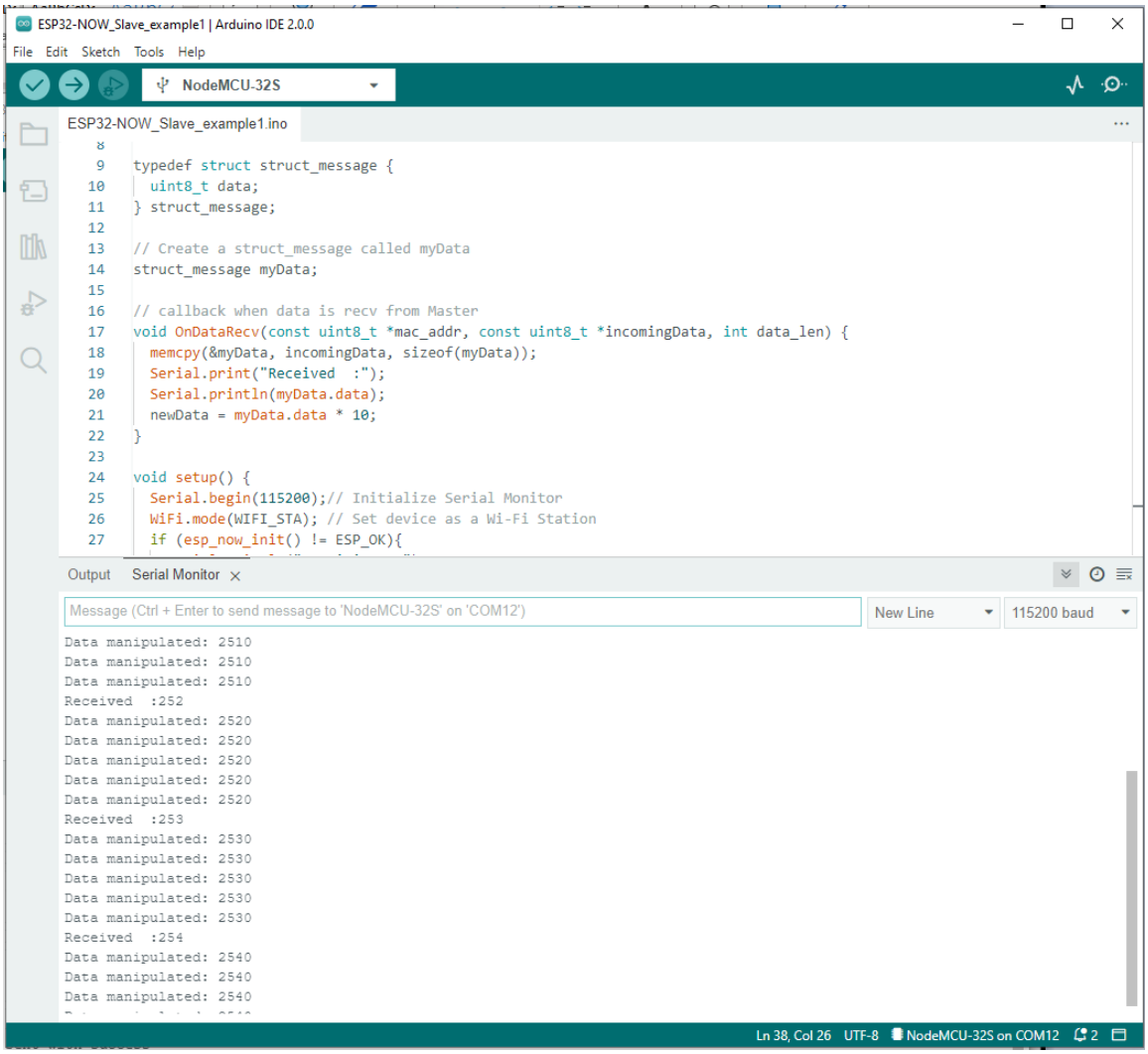
[NOTE: You may find the above sketches in my Github.]

Testing ESP-NOW Communication

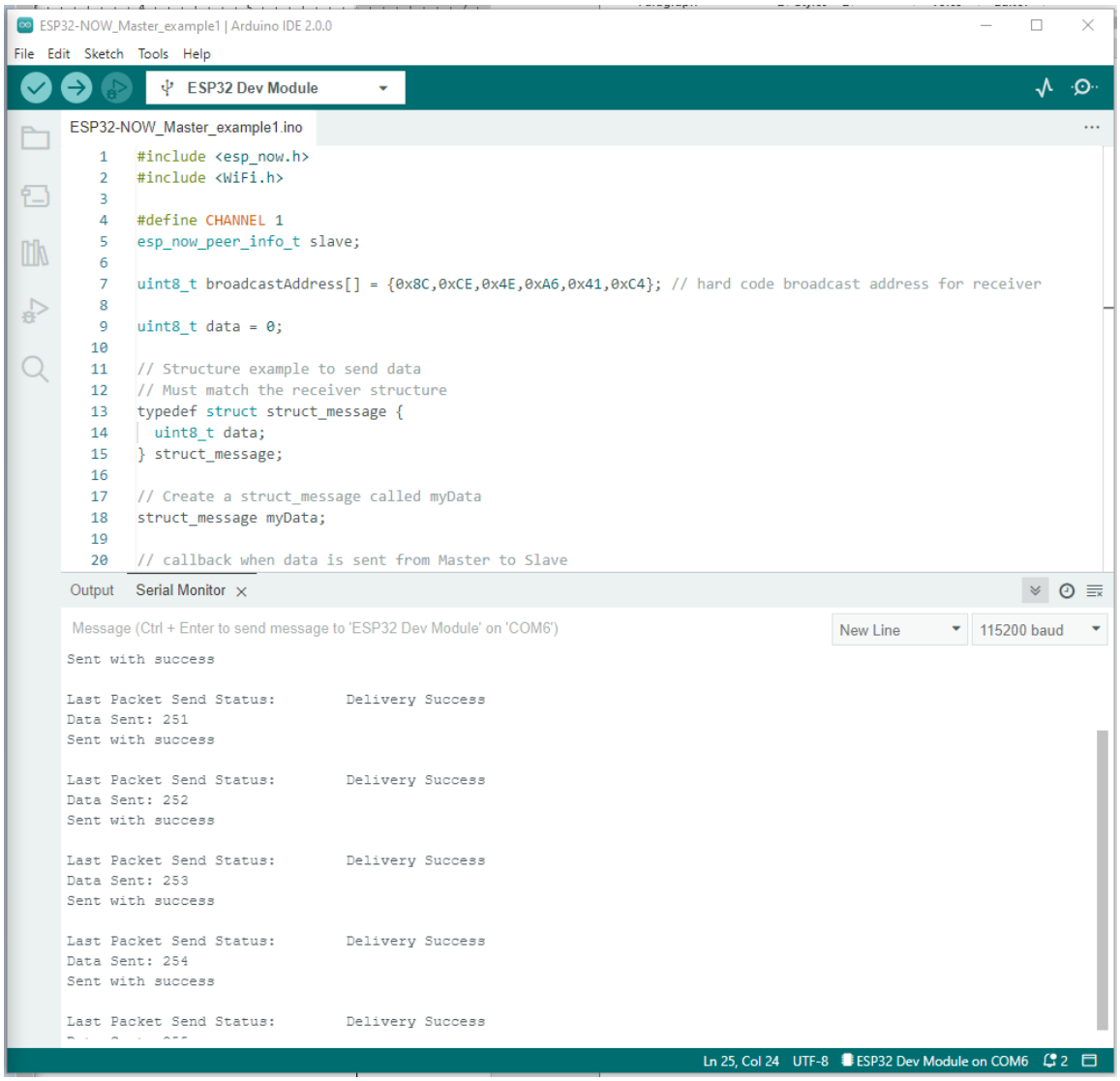
Upload the sender sketch to the sender ESP32 board and the receiver sketch to the receiver ESP32 board.

Now, open two Arduino IDE windows. One for the receiver, and another for the sender. Open the Serial Monitor for each board. Notice that the COM port for each board is different.

Testing the results on Sender/Master side:

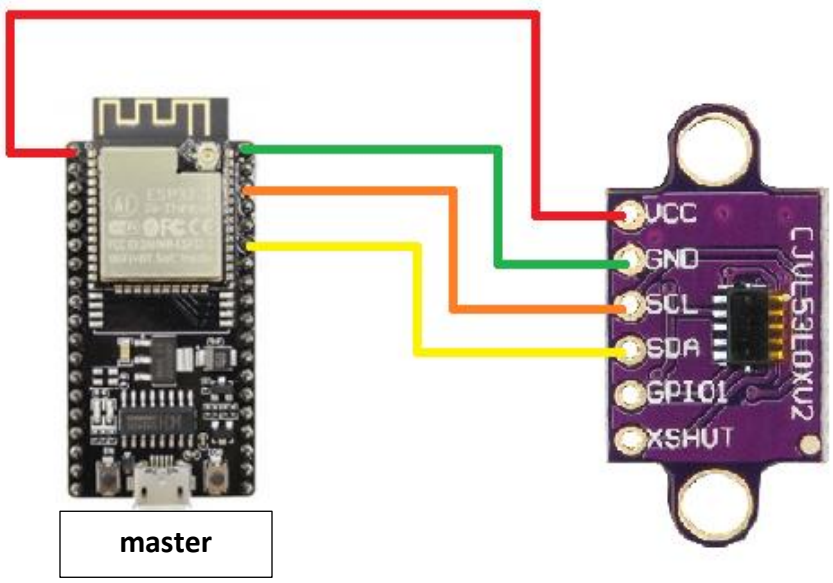


Testing the results on Receiver/Slave side:

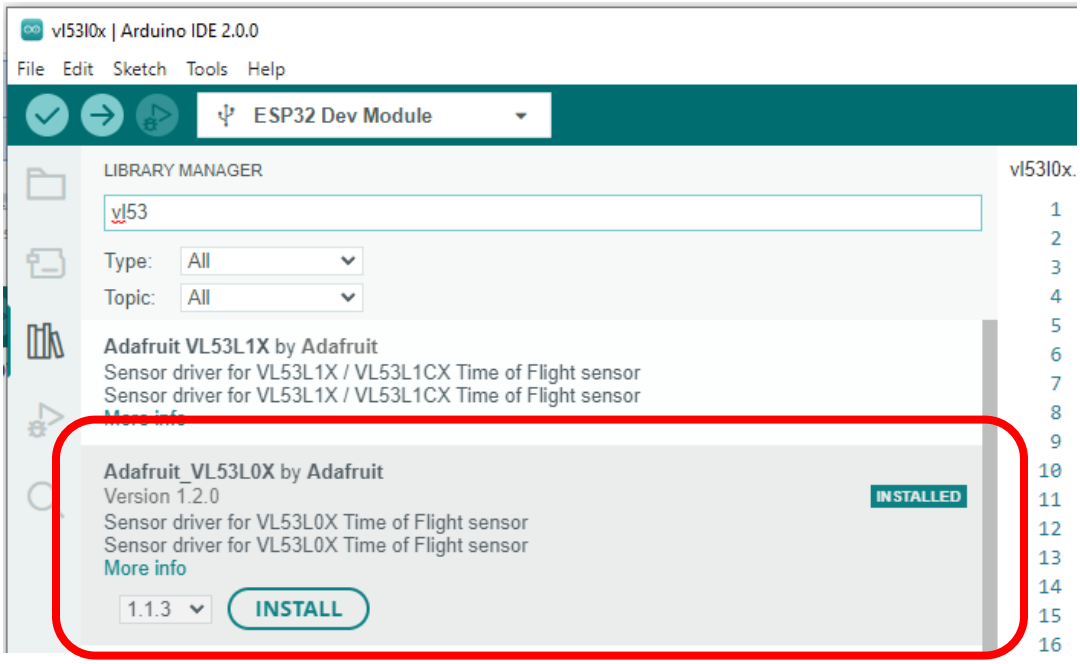

ESP32-NOW_Master_example1 | Arduino IDE 2.0.0
File Edit Sketch Tools Help
ESP32 Dev Module
ESP32-NOW_Master_example1.ino
#include <esp_now.h>
#include <WiFi.h>
#define CHANNEL 1
esp_now_peer_info_t slave;
uint8_t broadcastAddress[] = {0x8C,0xCE,0x4E,0xA6,0x41,0xC4}; // hard code broadcast address for receiver
uint8_t data = 0;
// Structure example to send data
// Must match the receiver structure
typedef struct struct_message {
uint8_t data;
} struct_message;
// Create a struct_message called myData
struct_message myData;
// callback when data is sent from Master to Slave
Output Serial Monitor
Message (Ctrl + Enter to send message to 'ESP32 Dev Module' on 'COM6')
New Line
115200 baud
Sent with success
Last Packet Send Status: Delivery Success
Data Sent: 251
Sent with success
Last Packet Send Status: Delivery Success
Data Sent: 252
Sent with success
Last Packet Send Status: Delivery Success
Data Sent: 253
Sent with success
Last Packet Send Status: Delivery Success
Data Sent: 254
Sent with success
Last Packet Send Status: Delivery Success
Ln 25, Col 24 UTF-8 ESP32 Dev Module on COM6 2

ESP-NOW Example - Sending ToF (Time of Flight) sensor data

Make the following connections.



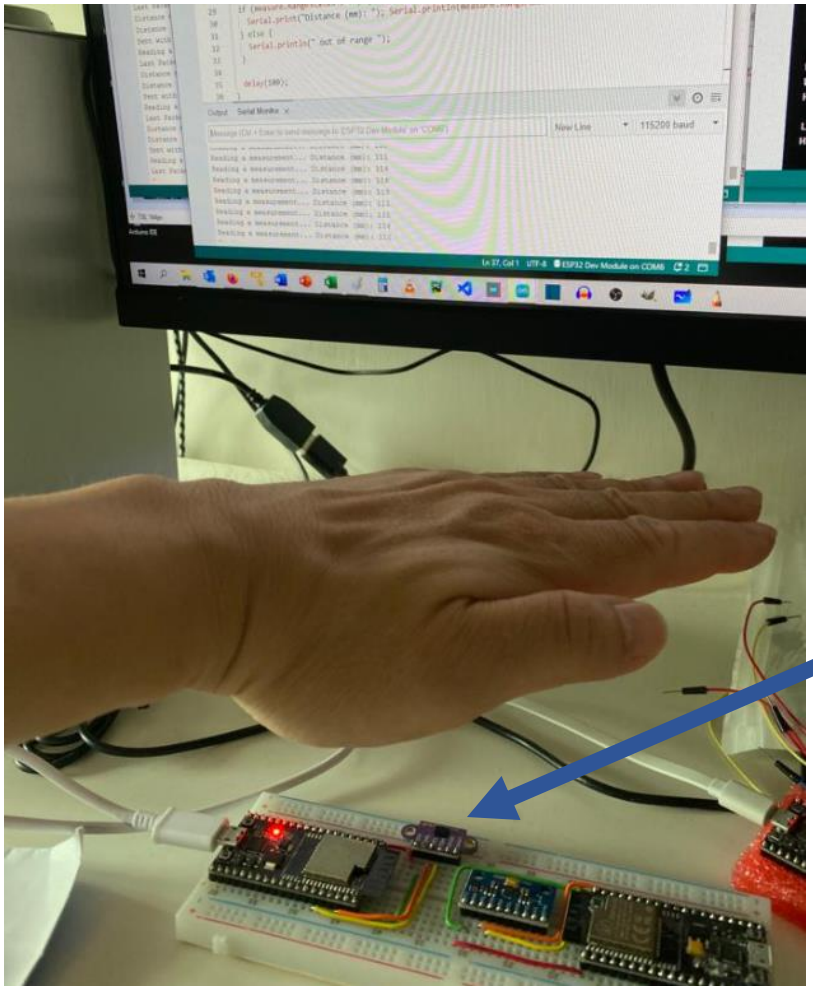
Add the VL53L0x library:



First of all, test the functionality of your circuit using the Adafruit example, File → Examples → ADAFRUIT_VL53L0X → vl53l0x.ino

(Note: the <Adafruit_VL53L0X.h> library includes the i2c platform, so you don't nee to include the <Wire.h> library)

Download the sketch to your ESP32 (use the "Master" one) and open the Serial Monitor. Test and see if you can get the distance measurement.



Check the sensor measurement. You should be able to see something similar to below screen:

```
35    delay(100);
36  }
```

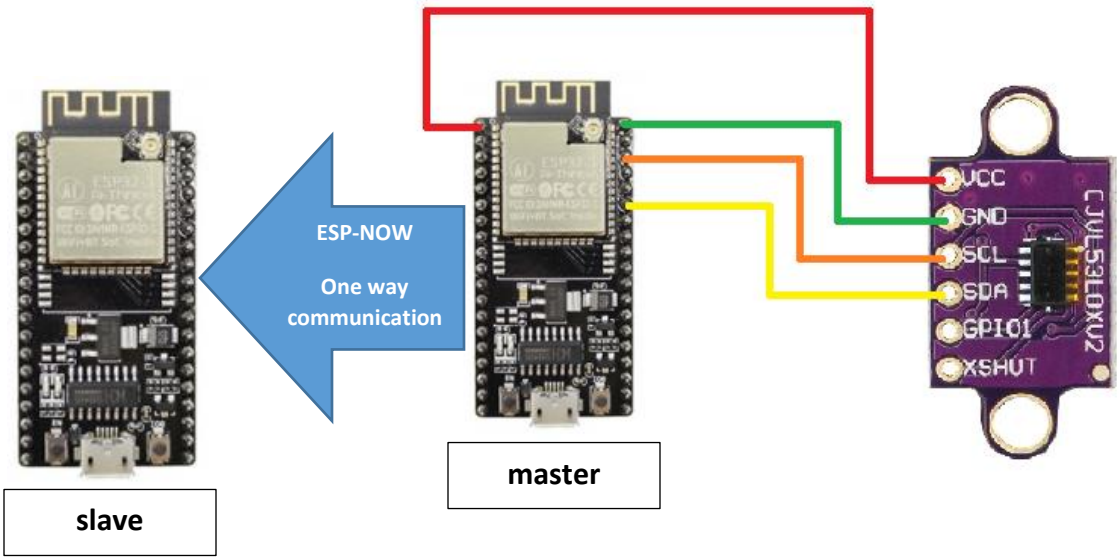
Output Serial Monitor ×

Message (Ctrl + Enter to send message to 'ESP32 Dev Module' on 'COM6')

```
Reading a measurement... Distance (mm): 144
Reading a measurement... Distance (mm): 132
Reading a measurement... Distance (mm): 132
Reading a measurement... Distance (mm): 136
Reading a measurement... Distance (mm): 138
Reading a measurement... Distance (mm): 138
Reading a measurement... Distance (mm): 137
Reading a measurement... Distance (mm): 138
```

Ln 37, C

Next, we will try to send the ToF measurement data via ESP-NOW protocol.



Let's incorporate the ToF sketch into the ESP32-NOW Master sketch.

```
#include <Adafruit_VL53L0X.h>
#include <esp_now.h>
#include <WiFi.h>
```

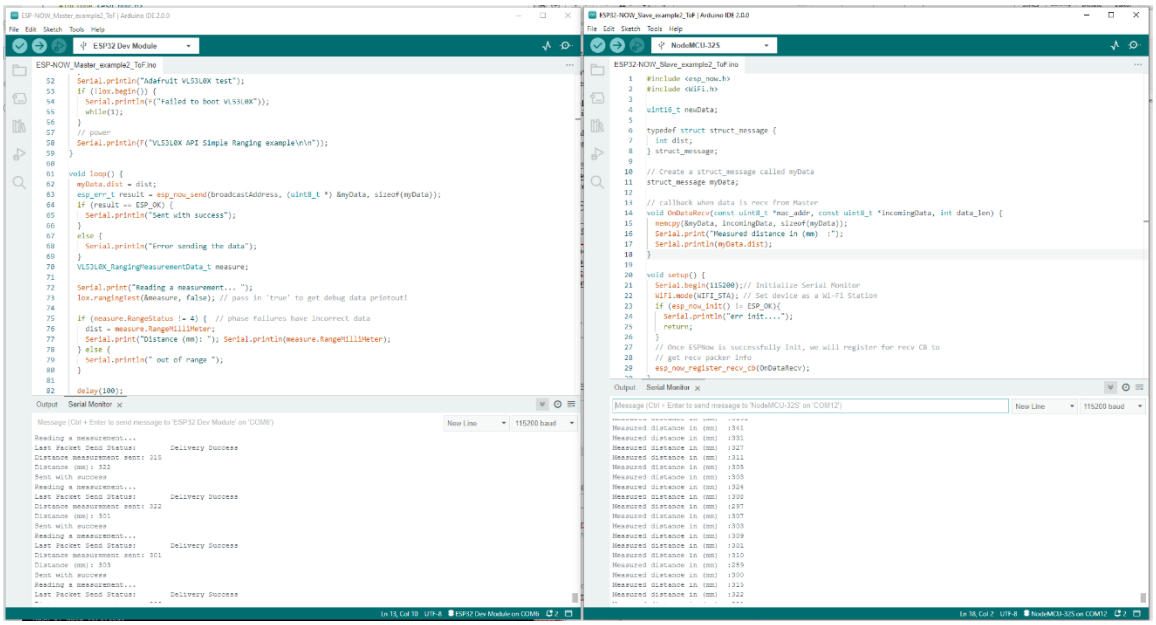
```
#define CHANNEL 1
esp_now_peer_info_t slave;

uint8_t broadcastAddress[] = {0x8C,0xCE,0x4E,0xA6,0x41,0xC4}; // hard
code broadcast address for receiver

Adafruit_VL53L0X lox = Adafruit_VL53L0X();

int dist;

// Structure example to send data
// Must match the receiver structure
typedef struct struct_message {
  int dist;
} struct_message;

// Create a struct_message called myData
struct_message myData;

// callback when data is sent from Master to Slave
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  Serial.print("\r\nLast Packet Send Status:\t");
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Delivery Success" :
"Delivery Fail");
  Serial.print("Distance measurement sent: ");
  Serial.println(dist);
}

void setup() {
  Serial.begin(115200); // Init Serial Monitor
  WiFi.mode(WIFI_STA); // Set device as a Wi-Fi Station
  // Init ESP-NOW
  if (esp_now_init() != ESP_OK){
    Serial.println("Error init");
    return;
  }
  // Once ESPNow is successfully Init, we will register for Send CB to
  // get the status of Trasnmitted packet
  esp_now_register_send_cb(OnDataSent);
// Register peer
  memcpy(slave.peer_addr, broadcastAddress, 6);
  slave.channel = CHANNEL; // pick a channel
  slave.encrypt = false; // no encryption
```

```
  // Add peer
  if (esp_now_add_peer(&slave) !=ESP_OK){
    Serial.println("Failed to add peer");
    return;
  }
  Serial.println("Adafruit VL53L0X test");
  if (!lox.begin()) {
    Serial.println(F("Failed to boot VL53L0X"));
    while(1);
  }
  // power
  Serial.println(F("VL53L0X API Simple Ranging example\n\n"));
}

void loop() {
  myData.dist = dist;
  esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &myData,
sizeof(myData));
  if (result == ESP_OK) {
    Serial.println("Sent with success");
  }
  else {
    Serial.println("Error sending the data");
  }
  VL53L0X_RangingMeasurementData_t measure;

  Serial.print("Reading a measurement... ");
  lox.rangingTest(&measure, false); // pass in 'true' to get debug data
printout!

  if (measure.RangeStatus != 4) {  // phase failures have incorrect data
    dist = measure.RangeMilliMeter;
    Serial.print("Distance (mm): ");
Serial.println(measure.RangeMilliMeter);
  } else {
    Serial.println(" out of range ");
  }

  delay(100);
}
```

Also, modify the ESP32-NOW Slave sketch.

```
#include <esp_now.h>
#include <WiFi.h>

uint16_t newData;

typedef struct struct_message {
  int dist;
} struct_message;

// Create a struct_message called myData
struct_message myData;

// callback when data is recv from Master
void OnDataRecv(const uint8_t *mac_addr, const uint8_t *incomingData,
int data_len) {
  memcpy(&myData, incomingData, sizeof(myData));
  Serial.print("Measured distance in (mm)  :");
  Serial.println(myData.dist);
}

void setup() {
  Serial.begin(115200);// Initialize Serial Monitor
  WiFi.mode(WIFI_STA); // Set device as a Wi-Fi Station
  if (esp_now_init() != ESP_OK){
    Serial.println("err init....");
    return;
  }
  // Once ESPNow is successfully Init, we will register for recv CB to
  // get recv packer info
  esp_now_register_recv_cb(OnDataRecv);
}

void loop() {
}
```

Test and see if everything work as expected. Below is my test screen for your reference.



Now, it is your turn to incorporate IMU data into ESP32-NOW communication between two ESP32 boards and the Slave ESP32 board should already been successfully connected and tested with the mecanum wheel car before.