Slave Code:

```
/*********
  Rui Santos
  Complete project details at https://RandomNerdTutorials.com/esp-now-many-to-one-esp32/

  Permission is hereby granted, free of charge, to any person obtaining a copy
  of this software and associated documentation files.

  The above copyright notice and this permission notice shall be included in all
  copies or substantial portions of the Software.
*********/

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

// Structure example to receive data
// Must match the sender structure
typedef struct struct_message_1 {
  int id;
  double ESpO2;
  int beatAvg;
}struct_message_1;
typedef struct struct_message_2 {
  int id;
  float temperature;
  float humidity;
}struct_message_2;
typedef struct struct_message_3 {
  int id;
  float temperature;
}struct_message_3;

// Create a struct_message called myData
struct_message_1 myData1;
struct_message_2 myData2;
struct_message_3 myData3;

// Create a structure to hold the readings from each board
struct_message_1 board1;
struct_message_2 board2;
struct_message_3 board3;

// Create an array with all the structures
```

```
struct_message boardsStruct[3] = {board1, board2, board3};

// callback function that will be executed when data is received
void OnDataRecv(const uint8_t * mac_addr, const uint8_t *incomingData, int len) {
  char macStr[18];
  Serial.print("Packet received from: ");
  snprintf(macStr, sizeof(macStr), "%02x:%02x:%02x:%02x:%02x:%02x",
           mac_addr[0], mac_addr[1], mac_addr[2], mac_addr[3], mac_addr[4],
mac_addr[5]);
  Serial.println(macStr);
  memcpy(&myData, incomingData, sizeof(myData));
  Serial.printf("Board ID %u: %u bytes\n", myData.id, len);
  // Update the structures with the new incoming data
  if (mydata.id == 1){ // ESPO2 and BPM
    boardsStruct[myData.id-1].ESpO2 = myData.ESpO2;
    boardsStruct[myData.id-1].beatAvg = myData.beatAvg;
    Serial.printf("ESpO2 value: %.5f \n", boardsStruct[myData.id-1].ESpO2);
    Serial.printf("BPM value: %.5f \n", boardsStruct[myData.id-1].beatAvg);
    Serial.println();
  }
  else if (mydata.id == 2){ // DHT11 (Room Temperature and Humidity)
    boardsStruct[myData.id-1].temperature = myData.temperature;
    boardsStruct[myData.id-1].humidity = myData.humidity;
    Serial.printf("temperature value: %.5f \n", boardsStruct[myData.id-
1].temperature);
    Serial.printf("humidity value: %.5f \n", boardsStruct[myData.id-1].humidity);
    Serial.println();
  }
  else if (mydata.id == 3){ // Body Temperature
    boardsStruct[myData.id-1].temperature = myData.temperature;
    Serial.printf("temperature value: %.5f \n", boardsStruct[myData.id-
1].temperature);
    Serial.println();
  }
}

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

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

  //Init ESP-NOW
  if (esp_now_init() != ESP_OK) {
```

```
    Serial.println("Error initializing ESP-NOW");
    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() {
  delay(10000);
}
```

Master Code:

1. **MAX**

```
#include <esp_now.h>
#include <WiFi.h>
#include "MAX30105.h"  //MAX3010x library
#include <Wire.h>
#include "heartRate.h"  //Heart rate calculating algorithm
#include "ESP32Servo.h"
const byte RATE_SIZE = 10;
MAX30105 particleSensor;
byte rates[RATE_SIZE];
byte rateSpot = 0;
long lastBeat = 0;  //Time at which the last beat occurred
float beatsPerMinute;
double avered = 0;
double aveir = 0;
double sumirrms = 0;
double sumredrms = 0;
double SpO2 = 0;
double ESpO2 = 90.0;
double FSpO2 = 0.7;  //filter factor for estimated SpO2
double frate = 0.95;  //low pass filter for IR/red LED value to eliminate AC
component
int beatAvg;
int i = 0;
int Num = 30;
#define FINGER_ON 7000
#define MINIMUM_SPO2 90.0
```

```
// REPLACE WITH THE RECEIVER'S MAC Address
uint8_t broadcastAddress[] = { 0x94, 0xB5, 0x55, 0x2C, 0xFF, 0xA8 };

// Structure example to send data
// Must match the receiver structure
typedef struct struct_message {
  int id;  // must be unique for each sender board
  double ESpO2;
  int beatAvg;
} struct_message;

// Create a struct_message called myData
struct_message myData;

// Create peer interface
esp_now_peer_info_t peerInfo;

// callback when data is sent
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");
}

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

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

  // Init ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    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(peerInfo.peer_addr, broadcastAddress, 6);
  peerInfo.channel = 0;
  peerInfo.encrypt = false;
```

```
  // Add peer
  if (esp_now_add_peer(&peerInfo) != ESP_OK) {
    Serial.println("Failed to add peer");
    return;
  }
  if (!particleSensor.begin(Wire, I2C_SPEED_FAST))  //Use default I2C port,
400kHz speed
  {
    Serial.println("MAX30102");
    while (1);
  }

  //Set up the wanted parameters
  byte ledBrightness = 0x7F;
  byte sampleAverage = 4;
  byte ledMode = 2;
  int sampleRate = 800;
  int pulseWidth = 215;
  int adcRange = 16384;

  //Configure sensor with these settings
  particleSensor.setup(ledBrightness, sampleAverage, ledMode, sampleRate,
pulseWidth, adcRange);
  particleSensor.enableDIETEMPRDY();

  //Turn Red LED to low to indicate sensor is running
  particleSensor.setPulseAmplitudeRed(0x0A);

  //Turn off Green LED
  particleSensor.setPulseAmplitudeGreen(0);
}

void loop() {
  //Reading the IR value
  //(it will permit us to know if there's a finger on the sensor or not)
  long irValue = particleSensor.getIR();
  if (irValue > FINGER_ON)
  {
    myData.beatAvg = beatAvg;
    Serial.print(beatAvg);
    Serial.println(" BPM");
    if (beatAvg > 30)
    {
      myData.ESpO2 = ESpO2;
```

```
    Serial.print(String(ESpO2));
    Serial.println(" %");
  }
  else Serial.println("---- %");
  if (checkForBeat(irValue) == true)
  {
    myData.beatAvg = beatAvg;
    Serial.print(beatAvg);
    Serial.println(" BPM");
    if (beatAvg > 30)
    {
      myData.ESpO2 = ESpO2;
      Serial.print(String(ESpO2));
      Serial.println(" %");
    }
    else Serial.println("---- %");
    myData.beatAvg = beatAvg;
    Serial.print("beatAvg=");
    Serial.println(beatAvg);
    long delta = millis() - lastBeat;
    lastBeat = millis();
    beatsPerMinute = 60 / (delta / 1000.0);
    if (beatsPerMinute < 255 && beatsPerMinute > 20)
    {
      rates[rateSpot++] = (byte)beatsPerMinute;
      rateSpot %= RATE_SIZE;
      beatAvg = 0;
      for (byte x = 0; x < RATE_SIZE; x++) beatAvg += rates[x];
      beatAvg /= RATE_SIZE;
    }
  }
  uint32_t ir, red;
  double fred, fir;
  //Check the sensor, read up to 3 samples
  particleSensor.check();
  if (particleSensor.available())
  {
    i++;
    red = particleSensor.getFIFOIR();
    ir = particleSensor.getFIFORed();
    fred = (double)red;  //double
    fir = (double)ir;  //double
    //average red level by low pass filter
    avered = avered * frate + (double)red * (1.0 - frate);
    //average IR level by low pass filter
```

```
      aveir = aveir * frate + (double)ir * (1.0 - frate);
      //square sum of alternate component of red level
      sumredrms += (fred - avered) * (fred - avered);
      //square sum of alternate component of IR level
      sumirrms += (fir - aveir) * (fir - aveir);
      if ((i % Num) == 0)
      {
        double R = (sqrt(sumredrms) / avered) / (sqrt(sumirrms) / aveir);
        SpO2 = -23.3 * (R - 0.4) + 100;
        ESpO2 = FSpO2 * ESpO2 + (1.0 - FSpO2) * SpO2;  //low pass filter
        if (ESpO2 <= MINIMUM_SPO2) ESpO2 = MINIMUM_SPO2;  //indicator for finger
detached
        if (ESpO2 > 100) ESpO2 = 99.9;
        Serial.print("Oxygen % = ");
        Serial.println(ESpO2);
        sumredrms = 0.0;
        sumirrms = 0.0;
        SpO2 = 0;
        i = 0;
      }
      particleSensor.nextSample();
    }
  }
  else
  {
    for (byte rx = 0; rx < RATE_SIZE; rx++) rates[rx] = 0;
    beatAvg = 0;
    rateSpot = 0;
    lastBeat = 0;
    avered = 0;
    aveir = 0;
    sumirrms = 0;
    sumredrms = 0;
    SpO2 = 0;
    ESpO2 = 90.0;
    Serial.println("No FInger!");  //Finger Please
  }

  // Set values to send
  myData.id = 1;

  // Send message via ESP-NOW
  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");
  }
}
```

2. DHT11

```
#include <esp_now.h>
#include <WiFi.h>
#include "DHT.h"
#define DHTPIN 18
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

// REPLACE WITH THE RECEIVER'S MAC Address
uint8_t broadcastAddress[] = {0x94, 0xB5, 0x55, 0x2C, 0xFF, 0xA8};

// Structure example to send data
// Must match the receiver structure
typedef struct struct_message {
    int id; // must be unique for each sender board
    float humidity = dht.readHumidity();
    float temperature = dht.readTemperature();
} struct_message;

// Create a struct_message called myData
struct_message myData;

// Create peer interface
esp_now_peer_info_t peerInfo;

// callback when data is sent
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");
}

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

```
  // Set device as a Wi-Fi Station
  WiFi.mode(WIFI_STA);
  // Init ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    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(peerInfo.peer_addr, broadcastAddress, 6);
  peerInfo.channel = 0;
  peerInfo.encrypt = false;
  // Add peer
  if (esp_now_add_peer(&peerInfo) != ESP_OK){
    Serial.println("Failed to add peer");
    return;
  }
   Serial.println(F("DHTxx test!"));
dht.begin();
}

void loop() {
float h = dht.readHumidity();
float t = dht.readTemperature();
if (isnan(h) || isnan(t))
{
Serial.println(F("Failed to read from DHT sensor!"));
return;
}
Serial.print(F("Room Humidity: "));
Serial.print(h);
Serial.print(F("% Room Temperature: "));
Serial.print(t);
Serial.println(F("°C "));
  // Set values to send
  myData.id = 1;
  myData.humidity = h;
  myData.temperature =t;
  // Send message via ESP-NOW
  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");
  }
  delay(10000);
}
```

3. DS18B20:

```
/*********
  Rui Santos
  Complete project details at https://RandomNerdTutorials.com/esp-now-many-to-one-esp32/

  Permission is hereby granted, free of charge, to any person obtaining a copy
  of this software and associated documentation files.

  The above copyright notice and this permission notice shall be included in all
  copies or substantial portions of the Software.
*********/

#include <esp_now.h>
#include <WiFi.h>
#include <OneWire.h>
#include <DallasTemperature.h>
const int oneWireBus = 5; //GPIO where the DS18B20 is connected to
OneWire oneWire(oneWireBus); //Setup a oneWire instance to communicate with any OneWire devices
DallasTemperature sensors(&oneWire); //Pass our oneWire reference to Dallas Temperature sensor
float temperatureC;

// REPLACE WITH THE RECEIVER'S MAC Address
uint8_t broadcastAddress[] = {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};

// Structure example to send data
// Must match the receiver structure
typedef struct struct_message {
    int id; // must be unique for each sender board
    float temperature;
} struct_message;

// Create a struct_message called myData
struct_message myData;
```

```
// Create peer interface
esp_now_peer_info_t peerInfo;

// callback when data is sent
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");
}

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

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

  // Init ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    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(peerInfo.peer_addr, broadcastAddress, 6);
  peerInfo.channel = 0;
  peerInfo.encrypt = false;

  // Add peer
  if (esp_now_add_peer(&peerInfo) != ESP_OK){
    Serial.println("Failed to add peer");
    return;
  }
  sensors.begin(); //Start the DS18B20 sensor
}

void loop() {
  sensors.requestTemperatures();
  temperatureC = sensors.getTempCByIndex(0);
  Serial.print(F("Body Temperature: "));
```

```
  Serial.print(temperatureC);
  Serial.println(F("°C "));
  // Set values to send
  myData.id = 1;
  myData.temperature = temperatureC;

  // Send message via ESP-NOW
  esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &myData,
sizeof(myData));

  if (result == ESP_OK) {
    Serial.print("Sent with success");
  }
  else {
    Serial.print("Error sending the data");
  }
  delay(5000);
}
```

Mac Adress Code:

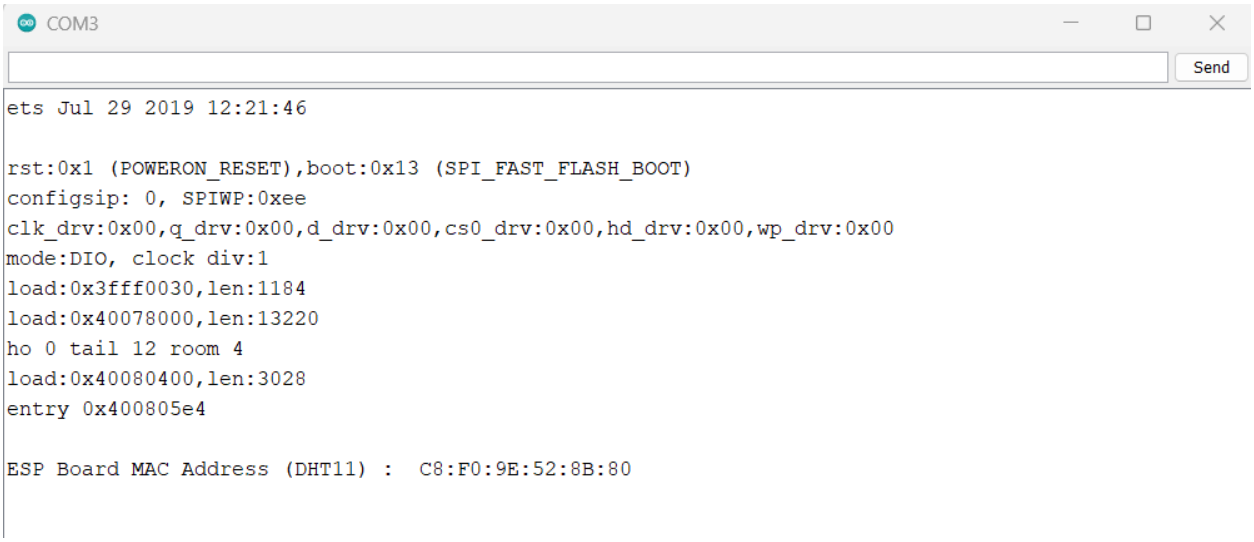
```
// Complete Instructions to Get and Change ESP MAC Address:
https://RandomNerdTutorials.com/get-change-esp32-esp8266-mac-
address-arduino/

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

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

void loop(){

}
```

Mac Address of Each Microcontroller:

```
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0030,len:1184
load:0x40078000,len:13220
ho 0 tail 12 room 4
load:0x40080400,len:3028
entry 0x400805e4

ESP Board MAC Address (MAX30102):  C8:F0:9E:52:60:10
```

```
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0030,len:1184
load:0x40078000,len:13220
ho 0 tail 12 room 4
load:0x40080400,len:3028
entry 0x400805e4

ESP Board MAC Address (DS18B20):  C8:F0:9E:52:62:A8
```

```
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0030,len:1184
load:0x40078000,len:13220
ho 0 tail 12 room 4
load:0x40080400,len:3028
entry 0x400805e4

ESP Board MAC Address (Master):  94:B5:55:2C:FF:A8
```


COM3
Send
ets Jul 29 2019 12:21:46
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0030,len:1184
load:0x40078000,len:13220
ho 0 tail 12 room 4
load:0x40080400,len:3028
entry 0x400805e4
ESP Board MAC Address (DHT11) : C8:F0:9E:52:8B:80