1. **MEMPEROLEH MAC ADDRESS ESP32**

|  |
| --- |
| #include "WiFi.h"    void setup(){  Serial.begin(115200);  WiFi.mode(WIFI\_MODE\_STA);  Serial.println(WiFi.macAddress());  }    void loop(){  } |

1. **ESP-NOW One-Way Point-to-Point Communication**

ESP32 sender

|  |
| --- |
| #include <esp\_now.h>  #include <WiFi.h>  // REPLACE WITH YOUR RECEIVER 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 {  char a[32];  int b;  float c;  bool d;  } struct\_message;  // Create a struct\_message called myData  struct\_message myData;  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;  }  }    void loop() {  // Set values to send  strcpy(myData.a, "THIS IS A CHAR");  myData.b = random(1,20);  myData.c = 1.2;  myData.d = false;    // 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(2000);  } |

Analisa

*#include* <esp\_now.h>

*#include* <WiFi.h>

Memasukan library esp\_now.h dan wifi.h

uint8\_t broadcastAddress[] = {0x24, 0x6F, 0x28, 0x02, 0xC3, 0x1C};

MAC penerima

typedef struct struct\_message {

  char a[32];

  int b;

  float c;

  bool d;

} struct\_message;

Merupakan jenis data yang akan dikirimkan oleh ESP32 sender ke ESP32 penerima, dalam contoh ini tipe data yang dikirimkan ada ‘character, integer, float, bool’

struct\_message myData;

Data variabel baru dengan nama struct\_message yang disebut myData yang akan menyimpan nilai variable tadi.

esp\_now\_peer\_info\_t peerInfo;

Buat variabel bertipe esp\_now\_peer\_info\_t untuk menyimpan informasi tentang peer.

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");

}

Gunakan fungsi ‘OnDataSent’ untuk memberikan respon balik saat pesan dikirim. Fungsi ini memberitahu pesan terkirim atau tidak.

void setup() {

*// Init Serial Monitor*

  Serial.begin(115200);

Konfigurasi Serial Monitor untuk debug dengan data rate 115200

  WiFi.mode(WIFI\_STA);

Konfigurasi ESP sebagai station

*if* (esp\_now\_init() != ESP\_OK) {

    Serial.println("Error initializing ESP-NOW");

*return*;

  }

Mulai ESP-Now

  esp\_now\_register\_send\_cb(OnDataSent);

Setelah sukses memulai ESP-Now, panggil fungsi OnDataSet yang telah dibuat sebelumnya

*// 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*;

  }

Konfigurasi untuk menghubungkan perangkat ESP lainnya.

  strcpy(myData.a, "THIS IS A CHAR");

  myData.b = random(1,20);

  myData.c = 1.2;

  myData.d = false;

Konfigurasi variable yang telah dibuat sebelumnya.

  esp\_err\_t result = esp\_now\_send(broadcastAddress, (uint8\_t \*) &myData, sizeof(myData));

Kirim pesan dengan konfigurasi seperti diatas

*if* (result == ESP\_OK) {

    Serial.println("Sent with success");

  }

*else* {

    Serial.println("Error sending the data");

  }

  delay(2000);

Cek apakah pesan berhasil dikirim dan menjeda 2 detik untuk notifikasi

ESP32 receive

|  |
| --- |
| #include <esp\_now.h>  #include <WiFi.h>  // Structure example to receive data  // Must match the sender structure  typedef struct struct\_message {  char a[32];  int b;  float c;  bool d;  } struct\_message;  // Create a struct\_message called myData  struct\_message myData;  // callback function that will be executed when data is received  void OnDataRecv(const uint8\_t \* mac, const uint8\_t \*incomingData, int len) {  memcpy(&myData, incomingData, sizeof(myData));  Serial.print("Bytes received: ");  Serial.println(len);  Serial.print("Char: ");  Serial.println(myData.a);  Serial.print("Int: ");  Serial.println(myData.b);  Serial.print("Float: ");  Serial.println(myData.c);  Serial.print("Bool: ");  Serial.println(myData.d);  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() {  } |

Analisa

*#include* <esp\_now.h>

*#include* <WiFi.h>

Memasukan library esp\_now.h dan wifi.h

typedef struct struct\_message {

char a[32];

int b;

float c;

bool d;

} struct\_message;

Merupakan jenis data yang akan dikirimkan oleh ESP32 sender ke ESP32 penerima, dalam contoh ini tipe data yang dikirimkan ada ‘character, integer, float, bool’

struct\_message myData;

Data variabel baru dengan nama struct\_message yang disebut myData yang akan menyimpan nilai variable tadi.

void OnDataRecv(const uint8\_t \* mac, const uint8\_t \*incomingData, int len) {

Menyalin data incomingData ke variable myData

memcpy(&myData, incomingData, sizeof(myData));

Struktur myData berisi beberapa variable di dalamnya yang isinya dikirim oleh ESP32 sender. Untuk mengakses variabel a misalnya, kita hanya perlu memanggil myData.a.

Serial.print("Bytes received: ");

Serial.println(len);

Serial.print("Char: ");

Serial.println(myData.a);

Serial.print("Int: ");

Serial.println(myData.b);

Serial.print("Float: ");

Serial.println(myData.c);

Serial.print("Bool: ");

Serial.println(myData.d);

Serial.println();

}

Memanggil variable yang telah dikonfigurasi selanjutnya.

void setup() {

*// Initialize Serial Monitor*

Serial.begin(115200);

Menginisiasi serial dengan data rate 115200.

WiFi.mode(WIFI\_STA);

Membuat ESP32 sebagai WiFi Station.

*if* (esp\_now\_init() != ESP\_OK) {

Serial.println("Error initializing ESP-NOW");

*return*;

}

Menginisiasi fungsi ESP-Now.

esp\_now\_register\_recv\_cb(OnDataRecv);

}

Mengunggakan fungsi OnDataRecv yang telah dibuat untuk menjadi umpan balik dalam komunikasi.

1. **One-Way, One-to-Many Communication**
   1. **Mengirim pesan yang sama ke beberapa board ESP32**

ESP32 sender

|  |
| --- |
| *#include* <esp\_now.h>  *#include* <WiFi.h>  *// REPLACE WITH YOUR ESP RECEIVER'S MAC ADDRESS*  uint8\_t broadcastAddress1[] = {0x24, 0x6F, 0x28, 0x02, 0xC3, 0x1C}; *//24:6F:28:02:C3:1C*  uint8\_t broadcastAddress2[] = {0x24, 0x0A, 0xC4, 0XC6, 0x06, 0x54}; *//24:0A:C4:C6:06:54*  uint8\_t broadcastAddress3[] = {0x8C, 0xCE, 0x4E, 0xC8, 0x29, 0x1B}; *//8C:CE:4E:C8:29:1B*  typedef struct test\_struct {    int x;    int y;  } test\_struct;  test\_struct test;  esp\_now\_peer\_info\_t peerInfo;  *// callback when data is sent*  void OnDataSent(const uint8\_t \*mac\_addr, esp\_now\_send\_status\_t status) {    char macStr[18];    Serial.print("Packet to: ");  *// Copies the sender mac address to a string*    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.print(macStr);    Serial.print(" send status:\t");    Serial.println(status == ESP\_NOW\_SEND\_SUCCESS ? "Delivery Success" : "Delivery Fail");  }    void setup() {    Serial.begin(115200);      WiFi.mode(WIFI\_STA);    *if* (esp\_now\_init() != ESP\_OK) {      Serial.println("Error initializing ESP-NOW");  *return*;    }      esp\_now\_register\_send\_cb(OnDataSent);    *// register peer*    peerInfo.channel = 0;    peerInfo.encrypt = false;  *// register first peer*    memcpy(peerInfo.peer\_addr, broadcastAddress1, 6);  *if* (esp\_now\_add\_peer(&peerInfo) != ESP\_OK){      Serial.println("Failed to add peer");  *return*;    }  *// register second peer*    memcpy(peerInfo.peer\_addr, broadcastAddress2, 6);  *if* (esp\_now\_add\_peer(&peerInfo) != ESP\_OK){      Serial.println("Failed to add peer");  *return*;    }  */// register third peer*    memcpy(peerInfo.peer\_addr, broadcastAddress3, 6);  *if* (esp\_now\_add\_peer(&peerInfo) != ESP\_OK){      Serial.println("Failed to add peer");  *return*;    }  }    void loop() {    test.x = random(0,20);    test.y = random(0,20);      esp\_err\_t result = esp\_now\_send(0, (uint8\_t \*) &test, sizeof(test\_struct));    *if* (result == ESP\_OK) {      Serial.println("Sent with success");    }  *else* {      Serial.println("Error sending the data");    }    delay(2000);  } |

Analisa

*#include* <esp\_now.h>

*#include* <WiFi.h>

Masukan library esp\_now.h & wifi.h

uint8\_t broadcastAddress1[] = {0x24, 0x6F, 0x28, 0x02, 0xC3, 0x1C};

uint8\_t broadcastAddress2[] = {0x24, 0x0A, 0xC4, 0XC6, 0x06, 0x54};

uint8\_t broadcastAddress3[] = {0x8C, 0xCE, 0x4E, 0xC8, 0x29, 0x1B};

Memasukan Mac Address ESP32 tujuan.

typedef struct test\_struct {

  int x;

  int y;

} test\_struct;

Membuat struktur data yang akan dikirim.

test\_struct test;

Membuat variable test\_struct menjadi variable baru dengan nama test.

esp\_now\_peer\_info\_t peerInfo;

Membuat variable esp\_now\_peer\_info\_t untuk memberikan informasi tentang koneksi.

void OnDataSent(const uint8\_t \*mac\_addr, esp\_now\_send\_status\_t status) {

  char macStr[18];

  Serial.print("Packet to: ");

*// Copies the sender mac address to a string*

  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.print(macStr);

  Serial.print(" send status:\t");

  Serial.println(status == ESP\_NOW\_SEND\_SUCCESS ? "Delivery Success" : "Delivery Fail");

}

Defisinikan variable OnDataSent yang berfungsi untuk mengeksekusi pesan yang dikirim dan memberikan umpan balik apabila pesan berhasil terkirim atau gagal terkirim.

ESP32 receive

|  |
| --- |
| *#include* <esp\_now.h>  *#include* <WiFi.h>  *//Structure example to receive data*  *//Must match the sender structure*  typedef struct test\_struct {  int x;  int y;  } test\_struct;  *//Create a struct\_message called myData*  test\_struct myData;  *//callback function that will be executed when data is received*  void OnDataRecv(const uint8\_t \* mac, const uint8\_t \*incomingData, int len) {  memcpy(&myData, incomingData, sizeof(myData));  Serial.print("Bytes received: ");  Serial.println(len);  Serial.print("x: ");  Serial.println(myData.x);  Serial.print("y: ");  Serial.println(myData.y);  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() {  } |

* 1. **Mengirim pesan berbeda ke beberapa board ESP32**