07 - Bluetooth dan Lora

Bersama sama mempelajari fitur komunikasi bawaan ESP 32 bluetooth dan komunikasi umum IoT yaitu LORA

- Komunikasi bluetooth dan Lora

Komunikasi bluetooth dan Lora

Bluetooth Communication

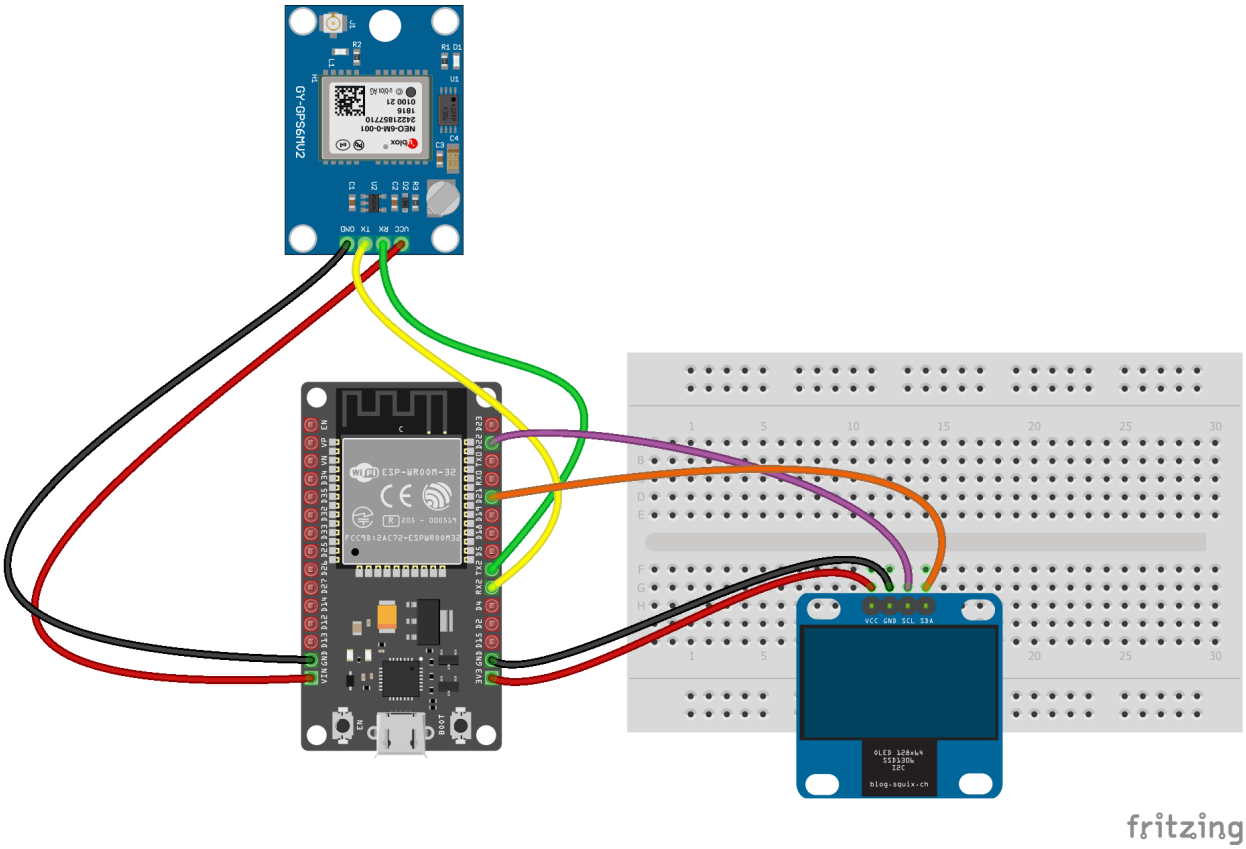
Pada percobaan kali ini kita akan menggunakan bluetooth untuk berkomunikasi antara gawai yang kita miliki dengan ESP 32 yang akan kita gunakan. Berikut adalah penjelasan fitur Bluetooth yang ada didalam ESP 32 (sumber: https://randomnerdtutorials.com).

A. Percobaan Bluetooth Classic

Peralatan yang kita butuhkan untuk percobaan kali ini

No.	Komponen
1	ESP 32 Dev-Board
2	Modul GPS Ublox Neo 6M
3	Kabel Micro USB
4	Breadboard
5	Jumper Wires

Skema rangkaian



Koneksi pin ESP-32 dengan modul GPS.

ESP - 32	Modul GPS Ublox Neo 6M
TX2 / GPIO 17	RX
RX2 / GPIO 16	TX
Vin (5V)	Vcc
GND	GND

Koneksi pin ESP32 dengan OLED 128 x 64 (SSD1306).

ESP - 32	OLED
GPIO 21	SDA
GPIO 22	SCL
3V3	Vcc
GND	GND

Sketch programnya adalah sebagai berikut:

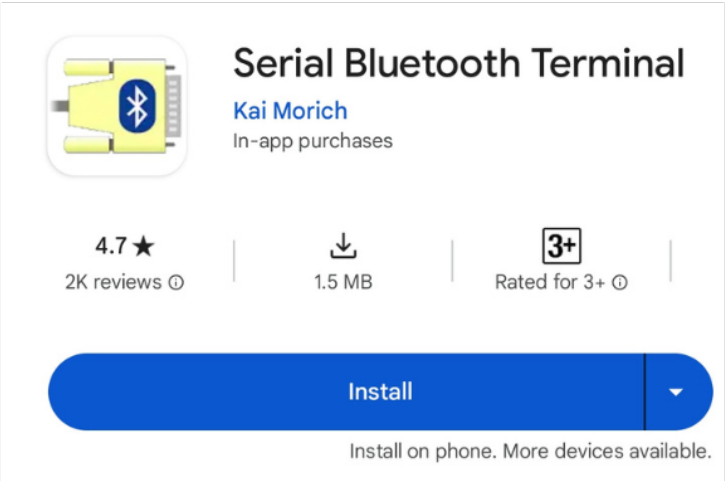
```
/* Sketch Kode untuk WEEKLY PROJECT 7 */
/* by Sandi Wibowo*/

#include <TinyGPSPlus.h>
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include "BluetoothSerial.h"

// Check if Bluetooth configs are enabled
#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
#endif

TinyGPSPlus gps;
BluetoothSerial SerialBT;

#define TX_PIN 17
#define RX_PIN 16
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1
#define SCREEN_ADDRESS 0x3C
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

String message = "";
char incomingChar;
String gpsString = "";



unsigned long previousMillis = 0;
const long interval = 10000;

void setup() {

  Serial.begin(115200);
  delay(1000);
  Serial2.begin(9600, SERIAL_8N1, RX_PIN, TX_PIN);
  if(!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
```

```
    Serial.println(F("SSD1306 allocation failed"));
    for(;;);
  }

  pinMode(LED_BUILTIN, OUTPUT);
  // Menginisiasi Serial Bluetooth
  SerialBT.begin("SMART-X");
  Serial.println("Bluetooth siap pairing");

  display.display();
  delay(5000);
  delay(2000);
  display.clearDisplay();
  display.setTextColor(WHITE);
  display.setTextSize(1);
  display.setCursor(25, 31);
  display.print("SMART-X ITB");
  display.display();
  delay(5000);
  display.clearDisplay();
  delay(500);

}

void tampilkanOled() {
  display.clearDisplay();
  display.setCursor(0, 0);
  display.print("SMART-X | GPS Rec");
  if (gps.location.isValid()){
    display.setCursor(0,11);
    display.print("Lat : ");
    display.print(gps.location.lat(),7);
    display.setCursor(0,21);
    display.print("Lon : ");
    display.print(gps.location.lng(),6);
    display.setCursor(0,31);
    display.print("Sat : ");
    display.print(gps.satellites.value());
    display.setCursor(55,31);
    display.print("Alt : ");
```

```
    display.print(gps.altitude.meters());
    display.setCursor(0,41);
    display.print("HDOP : ");
    display.print(gps.hdop.hdop());
  } else {
    display.setCursor(0,11);
    display.print("Invalid");
  }
  display.display();
}

void pesanBluetooth(){
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;
    gpsString = "Posisi : " + String(gps.location.lat(),6) + " " + String(gps.location.lng(),6);
    SerialBT.println(gpsString);
  }
}

void bluetoothReceive() {
  if (SerialBT.available()) {
    char incomingChar = SerialBT.read();
    if (incomingChar != '\n') {
      message += String(incomingChar);
    } else {
      message = "";
    }
  }
  display.setCursor(0, 51);
  display.print(message);
  display.display();
  if (message == "hdop") {
     SerialBT.println(gps.hdop.hdop());
  } else if (message == "sat") {
     SerialBT.println(gps.satellites.value());
  } else if (message == "alti") {
     SerialBT.println(gps.altitude.meters());
  } else if (message == "speed") {
     SerialBT.println(gps.speed.kmph());
```

```
  } else if (message == "dir") {
    SerialBT.println(gps.course.deg());
  }

}

void loop() {
  while (Serial2.available() > 0)
    if (gps.encode(Serial2.read()))
      bluetoothReceive();
      tampilkanOled();
      pesanBluetooth();
  if (millis() > 5000 && gps.charsProcessed() < 10)
  {
    Serial.println(F("ESP 32, Tidak terhubung Sensor"));
    while (true);
  }
}
```

Kita perlu menginstall aplikasi **Serial Bluetooth Terminal** di hanphone kita, aplikasi ini bisa kita unduh di Playstore.



Secara visual hasil dari percobaan kita, dapat dilihat di video berikut.

https://www.youtube.com/embed/UTjKrxQwxVQ

B. Percobaan *Bluetooth Low Energy* (BLE) - Server

ESP32 dilengkapi dengan Bluetooth Low Energy (BLE), sebuah teknologi nirkabel hemat daya yang memungkinkan komunikasi dengan perangkat lain. BLE memiliki beberapa keuntungan dibandingkan dengan Bluetooth klasik, termasuk:

- **Konsumsi daya yang lebih rendah:** BLE ideal untuk perangkat yang ditenagai baterai karena konsumsi dayanya jauh lebih rendah dibandingkan Bluetooth klasik.
- **Jangkauan yang lebih luas:** BLE dapat mencapai jangkauan hingga 100 meter, tergantung pada kondisi lingkungan.
- **Kecepatan data yang lebih tinggi:** BLE mampu mencapai kecepatan data hingga 2 Mbps, cukup untuk berbagai aplikasi.

ESP32 dapat digunakan sebagai perangkat BLE sentral atau periferal. Sebagai perangkat sentral, ESP32 dapat memindai dan terhubung ke perangkat BLE periferal lainnya. Sebagai perangkat periferal, ESP32 dapat terhubung ke perangkat BLE sentral, seperti smartphone atau tablet.

Beberapa contoh aplikasi BLE untuk ESP32:

- **Perangkat IoT:** ESP32 dapat digunakan untuk membangun perangkat IoT yang terhubung ke smartphone atau gateway BLE.
- **Beacon:** ESP32 dapat digunakan sebagai beacon untuk menyiarkan informasi ke perangkat BLE terdekat.
- **Pengendali jarak jauh:** ESP32 dapat digunakan sebagai pengendali jarak jauh untuk perangkat BLE lainnya.

ESP32 mendukung dua model komunikasi BLE:

1. Central:

- **Peran:** Perangkat ESP32 bertindak sebagai master dan dapat memindai, menemukan, dan terhubung ke perangkat BLE periferal.
- **Fungsi:** ESP32 dapat menginisiasi komunikasi dan mengontrol perangkat periferal.
- **Contoh:** ESP32 terhubung ke sensor BLE untuk menerima data.

2. Peripheral:

- **Peran:** Perangkat ESP32 bertindak sebagai sleve dan menunggu koneksi dari perangkat BLE sentral.
- **Fungsi:** ESP32 dapat menerima data dan perintah dari perangkat sentral.
- **Contoh:** ESP32 terhubung ke smartphone sebagai keyboard BLE.

Model komunikasi BLE:

- **GATT (Generic Attribute Profile):** Mendefinisikan struktur data dan layanan yang digunakan untuk pertukaran data.

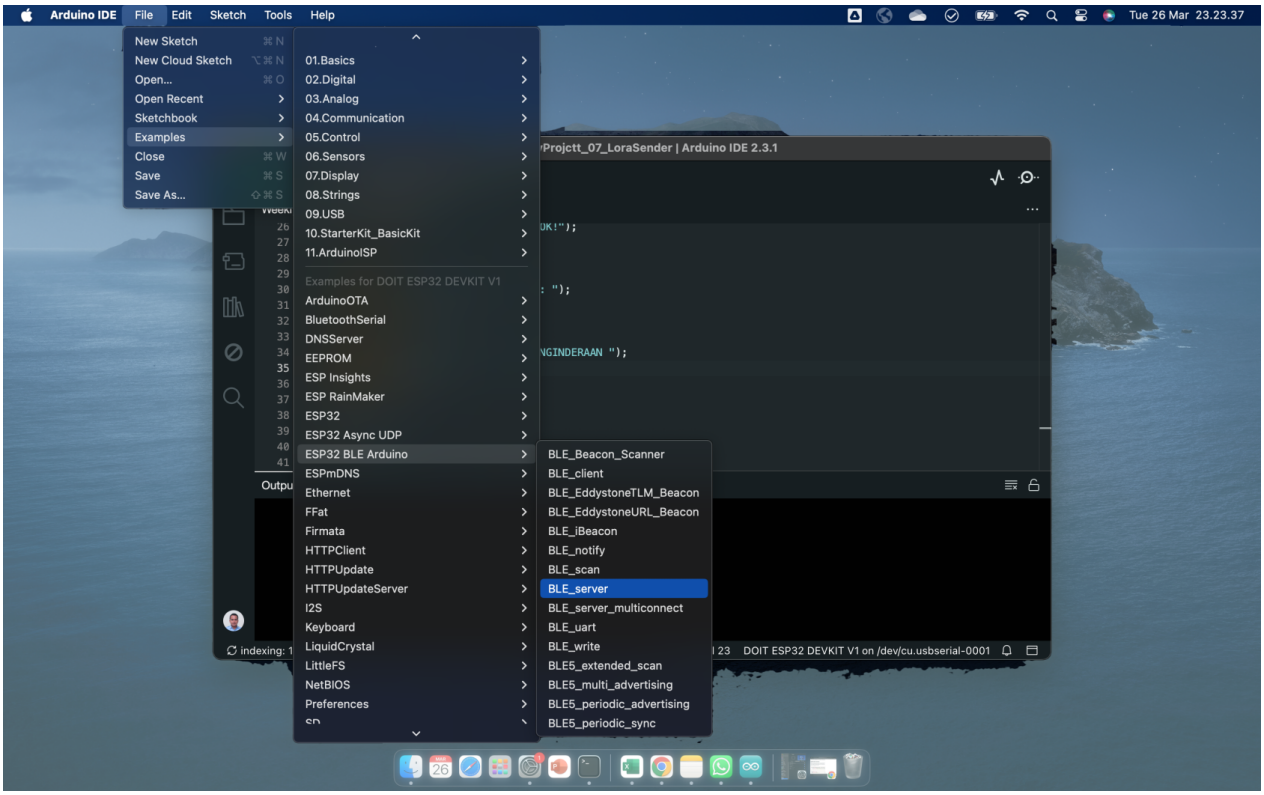
- **GAP (Generic Access Profile):** Menentukan prosedur untuk penemuan perangkat, koneksi, dan autentikasi.
- **ATT (Attribute Protocol):** Mendefinisikan operasi baca/tulis untuk mengakses data pada perangkat BLE.

Pilihan model komunikasi:

- **Central:** Cocok untuk perangkat yang perlu mengontrol perangkat BLE lain.
- **Peripheral:** Cocok untuk perangkat yang perlu menyediakan data ke perangkat BLE lain.

Alamat generator UUID	https://www.uuidgenerator.net/

Source code percobaan kali ini dapat kita ambil di menu **File > Examples > ESP 32 BLE Arduino > BLE_Server.**



Kita akan modifikasi source code nya menjadi seperti berikut:

```
#include <BLEDevice.h>
#include <BLEUtils.h>
#include <BLEServer.h>

// https://www.uuidgenerator.net/

#define SERVICE_UUID        "a4abe6c9-c250-425e-84f3-305f1010d0d3"
#define CHARACTERISTIC_UUID "9bfc173d-e259-4e1f-a8dd-ef0a71dead18"
```

```
void setup() {
  Serial.begin(115200);
  Serial.println("Starting BLE work!");

  BLEDevice::init("SMART-X BLE");
  BLEServer *pServer = BLEDevice::createServer();
  BLEService *pService = pServer->createService(SERVICE_UUID);
  BLECharacteristic *pCharacteristic = pService->createCharacteristic(
                                         CHARACTERISTIC_UUID,
                                         BLECharacteristic::PROPERTY_READ |
                                         BLECharacteristic::PROPERTY_WRITE
                                       );

  pCharacteristic->setValue("Hello World says Neil");
  pService->start();
  // BLEAdvertising *pAdvertising = pServer->getAdvertising();  // this still is working for backward compatibility
  BLEAdvertising *pAdvertising = BLEDevice::getAdvertising();
  pAdvertising->addServiceUUID(SERVICE_UUID);
  pAdvertising->setScanResponse(true);
  pAdvertising->setMinPreferred(0x06);  // functions that help with iPhone connections issue
  pAdvertising->setMinPreferred(0x12);
  BLEDevice::startAdvertising();
  Serial.println("Characteristic defined! Now you can read it in your phone!");
}

void loop() {
  // put your main code here, to run repeatedly:
  delay(2000);
}
```

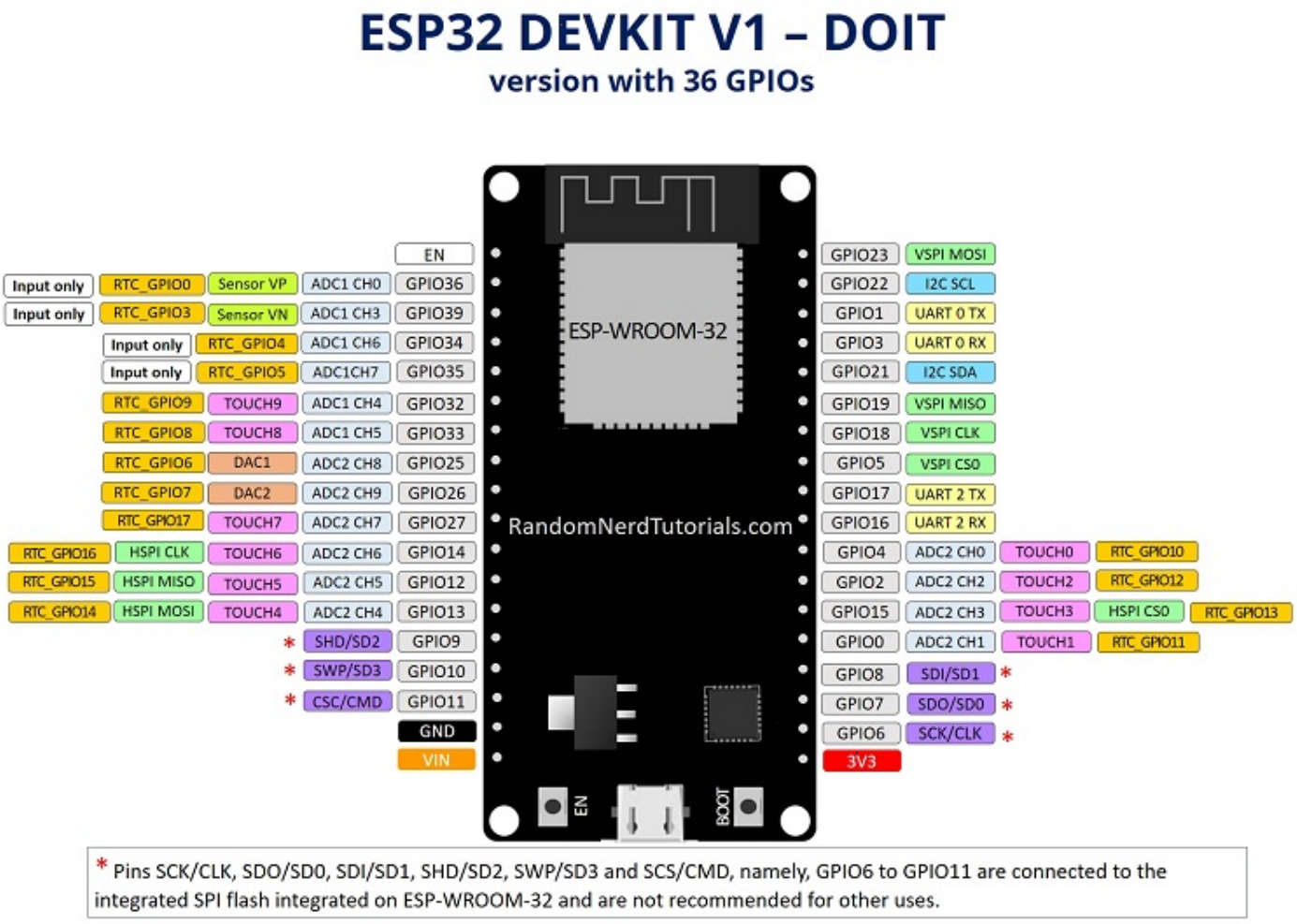
Video hasil percobaan sebagai berikut.

https://www.youtube.com/embed/yJ6qJPP4xJs?feature=youtu

> Referensi:
>
> 1. https://randomnerdtutorials.com/esp32-bluetooth-classic-arduino-ide/

LoRa Communication

Skema koneksi antara GPIO dan Pinout LoRa Ra-01 Ai Thinker (SX1278).

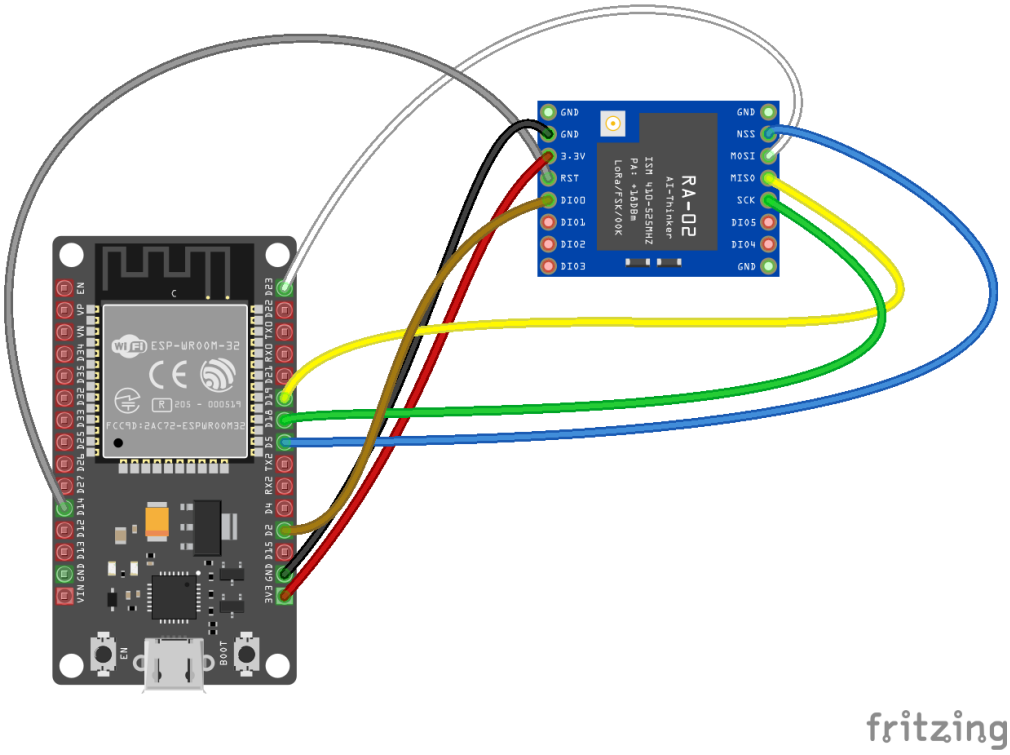


(source : randomnerdtutorials.com)

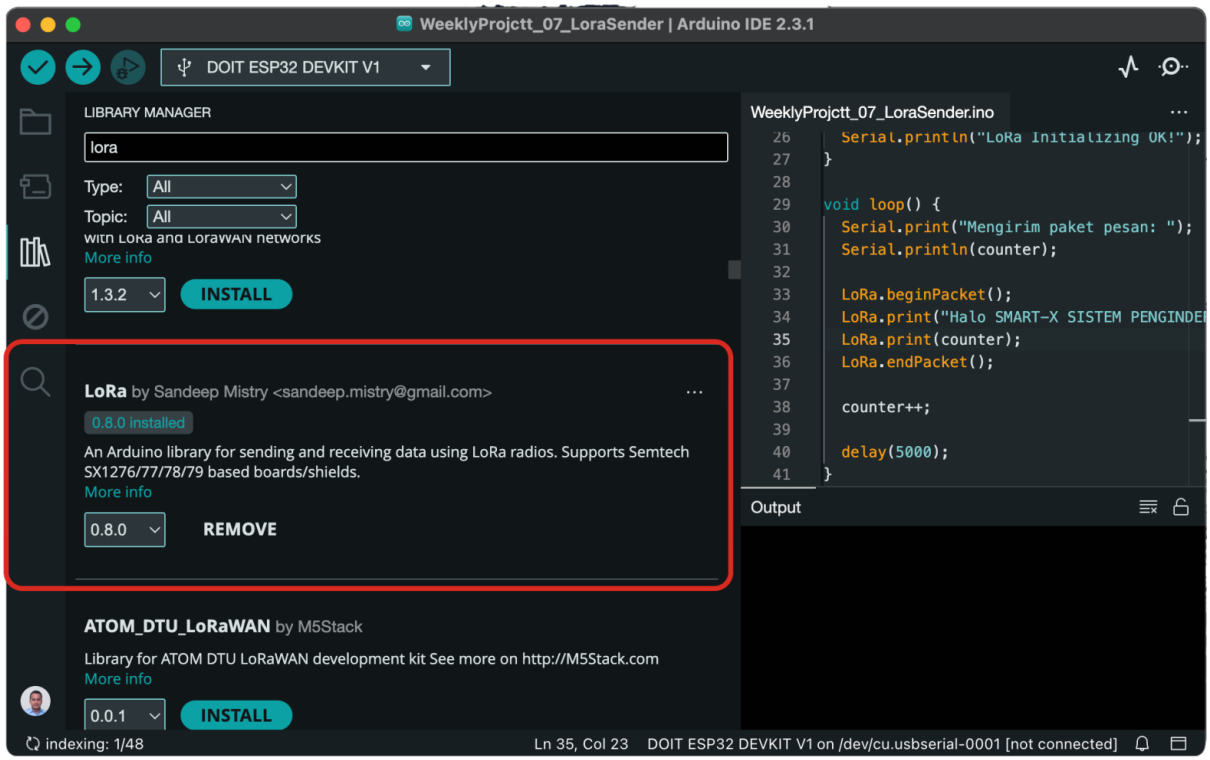
ESP 32	Ra 01 - Ai Thinker (SX 1278)
GPIO 23	MOSI
GPIO 2	DIO0
GPIO 19	MISO
GPIO 18	SCK

GPIO 14	RESET
GPIO 5	NSS
3V3	3V3
GND	GND

Skema rangkaian sebagaimana tabel diatas adalah seperti diagram berikut.



Penggunaan modul Long Range (LoRa) membutuhkan library yang harus kita download ke Arduino IDE yaitu Library LoRa.



Lora receiver penerima,

```
/* Sketch - ESP32 Lora Penerima Pesan */

#include <SPI.h>
#include <LoRa.h>

#define ss 5
#define rst 14
#define dio0 2

void setup() {
  Serial.begin(115200);
  while (!Serial);
  Serial.println("LoRa Penerima Pesan");

  LoRa.setPins(ss, rst, dio0);

  while (!LoRa.begin(433E6)) {
    Serial.println(".");
    delay(500);
  }
```

```
  LoRa.setSyncWord(0xF3);
  Serial.println("Inisialisasi Lora, BERHASIL!");
}

void loop() {
  int packetSize = LoRa.parsePacket();
  if (packetSize) {
    Serial.print("Menerima Pesan '");
    while (LoRa.available()) {
      String LoRaData = LoRa.readString();
      Serial.print(LoRaData);
    }
    Serial.print("' dengan Nilai RSSI : ");
    Serial.println(LoRa.packetRssi());
  }
}
```

Lora sender:

```
/* Sketch ESP-32 Pengirim Pesan */

#include <SPI.h>
#include <LoRa.h>

#define ss 5
#define rst 14
#define dio0 2

int counter = 0;

void setup() {
  //initialize Serial Monitor
  Serial.begin(115200);
  while (!Serial);

  Serial.println("Inisialisasi Lora ke Frekuensi 433Mhz");
  LoRa.setPins(ss, rst, dio0);

  while (!LoRa.begin(433E6)) {
    Serial.println(".");
```

```
    delay(500);
  }

  LoRa.setSyncWord(0xF3);
  Serial.println("LoRa Initializing OK!");
}

void loop() {
  Serial.print("Mengirim paket pesan: ");
  Serial.println(counter);

  LoRa.beginPacket();
  LoRa.print("Halo SMART-X SISTEM PENGINDERAAN ");
  LoRa.print(counter);
  LoRa.endPacket();

  counter++;

  delay(5000);
}
```

Berikut ini adalah video hasil percobaan pengiriman paket informasi dari ESP 32 pengirim (sender) ke ESP 32 penerima (receiver).

https://www.youtube.com/embed/jtTVLfxtBLo?feature=youtu

> Referensi:
>
> 1. https://randomnerdtutorials.com/esp32-lora-rfm95-transceiver-arduino-ide/