KUMPULAN KODE KOMUNIKASI SERIAL I2C

Kode untuk ESP32/WEMOS atau ESP DevKitC lainnya

Master:

```
// ===== KODE UNTUK ESP32 MASTER (I2C) =====
#include <Wire.h>

#define SLAVE_ADDRESS 0x08
const int POT_PIN = 34; // Pin ADC1

void setup() {
  // Bisa tentukan pin custom: Wire.begin(SDA_PIN, SCL_PIN);
  Wire.begin(); // Gunakan default (SDA=21, SCL=22)
  Serial.begin(115200);
}

void loop() {
  int potVal = analogRead(POT_PIN); // 0-4095
  byte dataToSend = map(potVal, 0, 4095, 0, 255);

  Wire.beginTransmission(SLAVE_ADDRESS);
  Wire.write(dataToSend);
  Wire.endTransmission();

  Serial.print("Mengirim: ");
  Serial.print(dataToSend);

  Wire.requestFrom(SLAVE_ADDRESS, 1);
  if (Wire.available()) {
    byte reply = Wire.read();
    Serial.print(" | Balasan: ");
    Serial.println(reply);
  }
  delay(500);
}
```

Slave:

```
// ===== KODE UNTUK ESP32 SLAVE (I2C) =====
#include <Wire.h>

#define SLAVE_ADDRESS 0x08
const int LED_PIN = 23;

volatile byte receivedData = 0;
byte counter = 0;

void receiveEvent(int howMany) {
  if (Wire.available()) {
    receivedData = Wire.read();
  }
}

void requestEvent() {
  Wire.write(counter);
  counter++;
}

void setup() {
  Wire.begin(SLAVE_ADDRESS); // Join as Slave
  Wire.onReceive(receiveEvent);
  Wire.onRequest(requestEvent);

  pinMode(LED_PIN, OUTPUT);
  Serial.begin(115200);
  Serial.println("Slave I2C Siap!");
}

void loop() {
  // ESP32 analogWrite lebih canggih, perlu setup channel
  // Untuk simpel, kita gunakan digital ON/OFF saja
  // Jika nilai > 128, nyalakan LED
  if (receivedData > 128) {
    digitalWrite(LED_PIN, HIGH);
  } else {
    digitalWrite(LED_PIN, LOW);
  }
  delay(10); // Cek data
}
```

Kode untuk ARDUINO:

Master:

```
// ===== KODE UNTUK ARDUINO MASTER (I2C) =====
#include <Wire.h>

#define SLAVE_ADDRESS 0x08 // Alamat untuk Slave (bebas pilih, 0x08 - 0x77)
const int POT_PIN = A0;

void setup() {
  Wire.begin(); // Join I2C bus sebagai Master
  Serial.begin(9600);
}

void loop() {
  // 1. Baca pot (0-1023) dan map ke 1 byte (0-255)
  int potVal = analogRead(POT_PIN);
  byte dataToSend = map(potVal, 0, 1023, 0, 255);

  // 2. Kirim data ke Slave
  Wire.beginTransmission(SLAVE_ADDRESS);
  Wire.write(dataToSend);
  Wire.endTransmission();

  Serial.print("Mengirim: ");
  Serial.print(dataToSend);

  // 3. Minta 1 byte data balasan dari Slave
  Wire.requestFrom(SLAVE_ADDRESS, 1);
  if (Wire.available()) {
    byte reply = Wire.read();
    Serial.print(" | Balasan: ");
    Serial.println(reply);
  }

  delay(500);
}
```

Slave:

```
// ===== KODE UNTUK ARDUINO SLAVE (I2C) =====
#include <Wire.h>

#define SLAVE_ADDRESS 0x08 // Alamat HARUS SAMA dengan Master
const int LED_PIN = 9;

// 'volatile' karena diakses oleh interrupt
volatile byte receivedData = 0;
byte counter = 0; // Data balasan untuk Master

// Fungsi ini akan dipanggil OTOMATIS saat Master mengirim data
void receiveEvent(int howMany) {
  if (Wire.available()) {
    receivedData = Wire.read(); // Ambil data
  }
}

// Fungsi ini akan dipanggil OTOMATIS saat Master meminta data
void requestEvent() {
  Wire.write(counter); // Kirim nilai counter sebagai balasan
  counter++; // Increment counter
```

```
}

void setup() {
  Wire.begin(SLAVE_ADDRESS); // Join I2C bus sebagai Slave
  Wire.onReceive(receiveEvent); // Daftarkan fungsi 'receiveEvent'
  Wire.onRequest(requestEvent); // Daftarkan fungsi 'requestEvent'

  pinMode(LED_PIN, OUTPUT);
  Serial.begin(9600);
  Serial.println("Slave I2C Siap!");
}

void loop() {
  // Update LED di loop utama, bukan di dalam ISR
  analogWrite(LED_PIN, receivedData);

  // Aksi ini hanya untuk debugging di serial monitor
  static byte lastData = 0;
  if(receivedData != lastData) {
    Serial.print("Data diterima: ");
    Serial.println(receivedData);
    lastData = receivedData;
  }
}
```

Kode untuk Raspiberry Pi:

Master:

```
# ===== KODE UNTUK RASPBERRY PI MASTER (I2C) =====
import smbus
import time

# Alamat slave HARUS sama
SLAVE_ADDRESS = 0x08

# Buat objek bus I2C (1 untuk RasPi modern, 0 untuk model lama)
bus = smbus.SMBus(1)

print("Master Pi Siap...")

data_to_send = 0
while True:
    try:
        # Simulasikan nilai pot (0-255)
        data_to_send = (data_to_send + 5) % 256

        # 1. Kirim data ke Slave
        bus.write_byte(SLAVE_ADDRESS, data_to_send)
        print(f"Mengirim: {data_to_send}")

        # 2. Baca 1 byte balasan dari Slave
        reply = bus.read_byte(SLAVE_ADDRESS)
        print(f"Menerima Balasan: {reply}")

        time.sleep(0.5)

    except IOError:
        print("Error: Slave tidak ditemukan/merespon")
        time.sleep(1)
```

```
    except KeyboardInterrupt:
        print("\nProgram dihentikan")
        break
```

Slave:

```
# ===== KODE UNTUK RASPBERRY PI SEBAGAI SPI SLAVE =====

from rpi_spi_slave import SpiSlave
import time

# Inisialisasi SPI Slave
# Ini akan menggunakan pin SPI default:
# - MISO: GPIO 9
# - MOSI: GPIO 10
# - SCK:  GPIO 11
# - SS:   GPIO 8 (CE0)
try:
    slave = SpiSlave()
    print("Slave Pi Siap Menerima Data...")
    print("Menggunakan pin SPI default (GPIO 8, 9, 10, 11).")
    print("Tekan Ctrl+C untuk berhenti.")

    # Siapkan data balasan pertama (misal: 0)
    # Ini akan dikirim saat Master melakukan transfer PERTAMA
    slave.put([0x00])

    while True:
        # Menunggu data dari Master (ini adalah proses 'blocking')
        # .transfer() akan:
        # 1. Menerima data baru dari Master.
        # 2. Mengirim data yang ada di buffer (yang disiapkan di 'slave.put' sebelumnya).
        received_data = slave.transfer()

        if received_data:
            # Tampilkan data yang baru saja diterima
            print(f"Data Diterima: {received_data[0]}")

            # Siapkan data balasan untuk transaksi BERIKUTNYA
            # Contoh: kita kirim kembali (data yang diterima + 10)
            # Pastikan nilainya tetap 1 byte (0-255) menggunakan modulo % 256
            reply_data = [(received_data[0] + 10) % 256]

            # Masukkan data balasan ke buffer pengiriman
            slave.put(reply_data)

        # Beri jeda sedikit agar tidak membebani CPU
        time.sleep(0.001)

except KeyboardInterrupt:
    # Matikan slave dengan bersih saat Ctrl+C ditekan
    slave.close()
    print("\nProgram dihentikan. Slave SPI ditutup.")
```