MODUL 6
KOMUNIKASI SERIAL, SPI, DAN I2C

6.1 Tujuan Praktikum

Setelah mengikuti praktikum Modul 6, praktikan diharapkan dapat:

1. Mengetahui dan memahami pengertian Komunikasi Serial, I2C, dan SPI
2. Mengetahui dan memahami penggunaan Komunikasi Serial dan ESP32
3. Mengetahui dan memahami penggunaan Pin I2C ESP32

6.2 Alat dan Bahan

Alat dan bahan yang digunakan pada praktikum Modul 6 adalah sebagai berikut:

1. ESP32 S3
2. Kabel Jumper
3. OLED 128x64
4. Kabel USB to Type C

6.3 Dasar Teori

6.3.1 Komunikasi Serial

Komunikasi serial adalah metode transmisi data yang mengirimkan informasi satu per satu dari perangkat ke perangkat. Banyak standar serial yang berbeda telah dikembangkan selama bertahun-tahun untuk bandwidth perangkat berkecepatan rendah dan berkecepatan tinggi. Data biasanya dapat dipertukarkan dengan jarak yang jauh lebih jauh menggunakan komunikasi serial daripada komunikasi paralel . Komunikasi serial biasanya digunakan untuk menghubungkan printer, terminal, dan kamera ke komputer. Ini juga digunakan untuk antarmuka ke hard drive eksternal, drive disk video digital (DVD) dan perangkat memori flash.

6.3.1.1 Komunikasi Serial Menggukanan ESP32

Komunikasi serial pada ESP32 adalah komunikasi antara ESP32 dan komputer yang dapat dilakukan melalui port USB. Pada Arduino IDE, terdapat fasilitas Serial Monitor yang memungkinkan komunikasi dua arah untuk mengirim dan menerima data antara ESP32 dan komputer. Fasilitas ini bisa dimanfaatkan untuk berbagai keperluan, seperti mengontrol ESP32 atau memantau data dari perangkat yang terhubung.

Dengan menggunakan fasilitas ini, data dapat dikirim ke ESP32, dan sebaliknya ESP32 dapat mengirim data kembali ke komputer. Komunikasi serial yang terjadi secara serial hanya membutuhkan 2 wire, yaitu RX (Receive) dan TX (Transmit). Pada ESP32, pin komunikasi serial standar UART bisa digunakan pada pin GPIO yang diprogram sebagai RX dan TX

sesuai kebutuhan, karena ESP32 mendukung hingga 3 port UART yang bisa dikonfigurasi.

Pin RX (Receive) berfungsi untuk menerima data, dan pin TX (Transmit) berfungsi untuk mengirim data. Selain digunakan untuk komunikasi serial, pin-pin ini juga dapat digunakan sebagai I/O digital (Input/Output) jika komunikasi serial tidak digunakan.

Dalam pengiriman data secara digital terdapat dua buah ukuran yang penting untuk diketahui, yaitu Bit Rate dan Baud Rate

- **Bit Rate**: Jumlah bit yang dikirim atau diterima per satuan waktu (detik).
- **Baud Rate**: Banyaknya perubahan data (sinyal) yang terjadi per satuan waktu.

6.3.2 I2C

Komunikasi I2C merupakan singkatan dari *inter integrated circuit* merupakan komunikasi serial antara mikrokontroler yang memungkinkan kita untuk menghubungkan sejumlah *device (slave)* dengan *device utama (master)* dalam satu mode komunikasi.

Slave master berfungsi untuk menerima data serta mengirim data yang akan dibaca dan digunakan oleh master device untuk diproses. Slave Sender device berfungsi untuk membangkitkan dan mengirim clock pulse serta menentukan kapan dan dengan slave device mana berkomunikasi.

I2C merupakan protokol komunikasi serial standar resmi yang hanya membutuhkan dua jalur sinyal yang dirancang untuk komunikasi antara chip pada PCB. I2C pada awalnya dirancang untuk komunikasi 100kbps tetapi mode transmisi data yang lebih cepat telah dikembangkan selama bertahun-tahun untuk mencapai kecepatan hingga 3.4Mbps. Protokol I2C telah ditetapkan sebagai standar resmi, yang menyediakan kompatibilitas yang baik antara implementasi I2C dan kompatibilitas mundur yang baik.

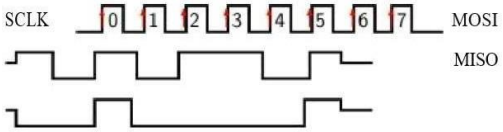
Perangkat utama dalam protokol komunikasi I2C ini ialah pulse serial data line (SDA) dan serial clock line (SCL).SaIah satu keunggulan dari komunikasi I2c ini, controller master dapat berkomunikasi dengan berbagai device dengan menggunakan dua kabel, kabel serial data (SDA) dan kabel serial clock (SCL).

6.3.3 SPI

Serial Peripheral Interface (SPI) adalah protokol synchronous serial data yang digunakan oleh mikrokontroler untuk berkomunikasi dengan satu atau lebih perangkat periferal dengan cepat pada jarak pendek. mi juga dapat digunakan untuk komunikasi antara dua mikrokontroler. Dengan koneksi SPI selalu ada satu Master Divice (biasanya mikrokontroler) yang mengontrol perangkat periferal. Biasanya ada tiga baris yang sanna untuk semua perangkat:

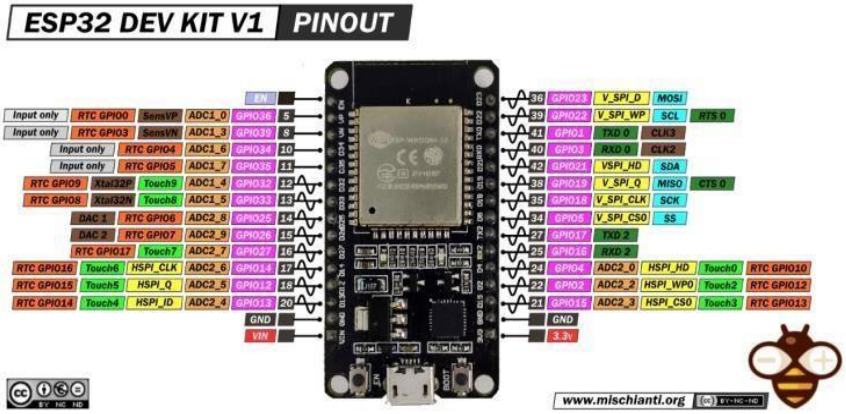
- MISO (Master In Slave Out) - untuk mengirim data ke master,
- MOSI (Master Out SIave In) - untuk mengirim data ke slave,
- SCK (Serial Clock) - Pulsa clock yang menyinkronkan transmisi data yang dihasilkan oleh master.
- SS (Slave Select)

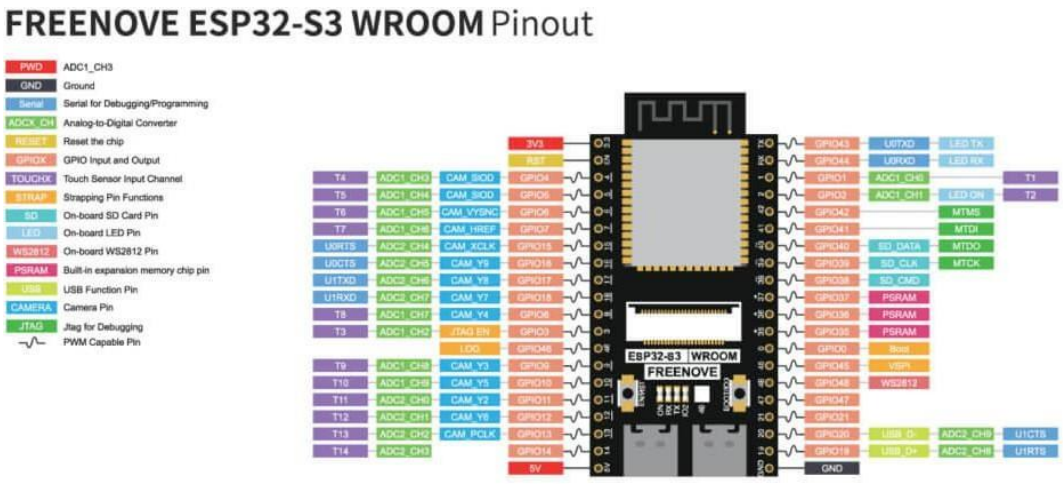
  SS merupakan pin pada setiap perangkat yang dapat digunakan oleh master untuk mengaktifkan dan menonaktifkan perangkat tertentu. Ketika pin Slave Select Low,maka akan berkomunikasi dengan master dan ketika Slave select High, maka akan mengabaikan master.



6.3.4 Pin I2C dan SPI

- ESP32





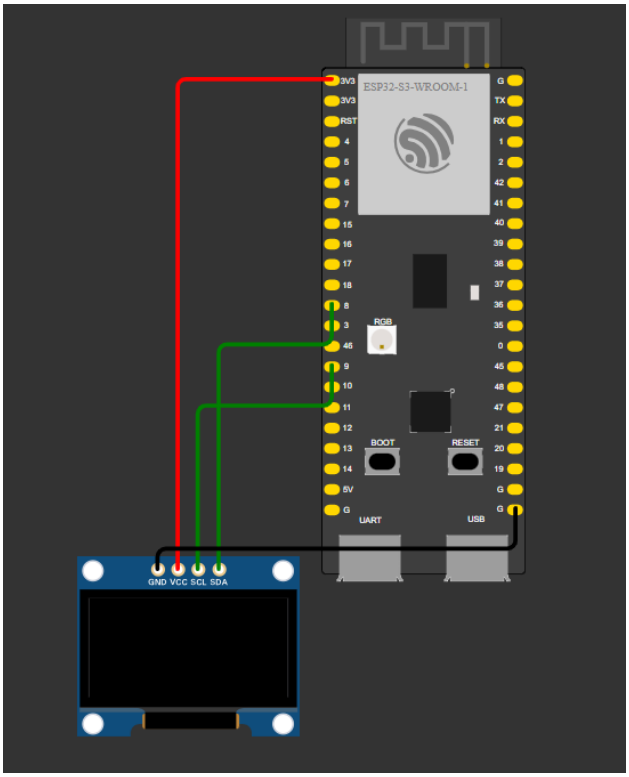
6.3.5 Perbedaan I2C dan SPI

Perbedaan Umum I2C dan SPI adalah pada I2C memungkinkan beberapa master dan slave, sedangkan pada SPI hanya dapat bekerja dengan satu master device yang mengendalikan banyak slave. Baik I2C maupun SPI, master device mengontrol clock untuk semua slave, tetapi perangkat slave I2C dapat memodifikasi clock bus utama.

6.4 Praktikum

A. "*Hello World"* di ESP32 S3 menggunakan OLED dan protocol I2C

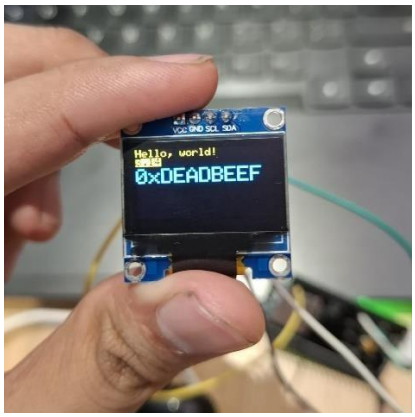
1. Buka Arduino IDE
2. Ikutilah Skematik Berikut:



3. Ketikkan kode berikut:

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels
#define OLED_RESET     -1
#define SCREEN_ADDRESS 0x3C
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

void setup() {
  Serial.begin(115200);
  Serial.println("Hello, ESP32-S3!");
  if(!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
    Serial.println(F("SSD1306 allocation failed"));
    for(;;); // Don't proceed, loop forever
  }

  display.clearDisplay();

  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0,0);
  display.println(F("Hello, world!"));

  display.setTextColor(SSD1306_BLACK, SSD1306_WHITE);
  display.println(3.141592);

  display.setTextSize(2);
  display.setTextColor(SSD1306_WHITE);
  display.print(F("0x")); display.println(0xDEADBEEF, HEX);

  display.display();
  delay(2000);
}

void loop() {

}
```

4. Save File, Lalu sambungkan ESP32 S3 ke Laptop.
5. Pilih Board : “ESP32S3 Dev Module” dan Port masukkan port yang digunakan.
6. Verify dan Upload code yang telah di ketik ke ESP32 S3 yang telah di rangkai.
7. Hasilnya akan jadi seperti ini :



B. Penggunaan I2C untuk Komunikasi dengan Tampilan OLED

1. Buka Arduino IDE.
2. Untuk Skematik tidak ada yang perlu diubah.
3. Ketikan kode berikut :

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels
#define OLED_RESET     -1
#define SCREEN_ADDRESS 0x3C
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

#define NUMFLAKES     10

#define LOGO_HEIGHT   16
#define LOGO_WIDTH    16
static const unsigned char PROGMEM logo_bmp[] =
{ 0b00000000, 0b11000000,
  0b00000001, 0b11000000,
  0b00000001, 0b11000000,
  0b00000011, 0b11100000,
  0b11110011, 0b11100000,
  0b11111110, 0b11111000,
  0b01111110, 0b11111111,
  0b00110011, 0b10011111,
  0b00011111, 0b11111100,
  0b00001101, 0b01110000,
  0b00011011, 0b10100000,
  0b00111111, 0b11100000,
  0b00111111, 0b11110000,
  0b01111100, 0b11110000,
  0b01110000, 0b01110000,
  0b00000000, 0b00110000 };

void setup() {
  Serial.begin(9600);

  if(!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
    Serial.println(F("SSD1306 allocation failed"));
    for(;;); // Don't proceed, loop forever
  }

  display.display();
  delay(2000);

  display.clearDisplay();
```

```
  display.drawPixel(10, 10, SSD1306_WHITE);

  display.display();
  delay(2000);

  testdrawline();      // Draw many lines
  testdrawrect();      // Draw rectangles (outlines)
  testfillrect();      // Draw rectangles (filled)
  testdrawcircle();    // Draw circles (outlines)
  testfillcircle();    // Draw circles (filled)
  testdrawroundrect(); // Draw rounded rectangles (outlines)
  testfillroundrect(); // Draw rounded rectangles (filled)
  testdrawtriangle();  // Draw triangles (outlines)
  testfilltriangle();  // Draw triangles (filled)
  testdrawchar();      // Draw characters of the default font
  testdrawstyles();    // Draw 'stylized' characters
  testscrolltext();    // Draw scrolling text
  testdrawbitmap();    // Draw a small bitmap image

  display.invertDisplay(true);
  delay(1000);
  display.invertDisplay(false);
  delay(1000);

  testanimate(logo_bmp, LOGO_WIDTH, LOGO_HEIGHT);
}

void loop() {
}

void testdrawline() {
  int16_t i;

  display.clearDisplay();

  for(i=0; i<display.width(); i+=4) {
    display.drawLine(0, 0, i, display.height()-1, SSD1306_WHITE);
    display.display();
    delay(1);
  }
  for(i=0; i<display.height(); i+=4) {
    display.drawLine(0, 0, display.width()-1, i, SSD1306_WHITE);
    display.display();
    delay(1);
  }
    delay(250);

    display.clearDisplay();

    for(i=0; i<display.width(); i+=4) {
      display.drawLine(0, display.height()-1, i, 0, SSD1306_WHITE);
      display.display();
      delay(1);
    }
    for(i=display.height()-1; i>=0; i-=4) {
      display.drawLine(0, display.height()-1, display.width()-1, i, SSD1306_WHITE);
      display.display();
      delay(1);
    }
    delay(250);

    display.clearDisplay();

    for(i=display.width()-1; i>=0; i-=4) {
      display.drawLine(display.width()-1, display.height()-1, i, 0, SSD1306_WHITE);
      display.display();
      delay(1);
    }
    for(i=display.height()-1; i>=0; i-=4) {
      display.drawLine(display.width()-1, display.height()-1, 0, i, SSD1306_WHITE);
      display.display();
      delay(1);
    }
    delay(250);

    display.clearDisplay();

    for(i=0; i<display.height(); i+=4) {
      display.drawLine(display.width()-1, 0, 0, i, SSD1306_WHITE);
      display.display();
      delay(1);
    }
    for(i=0; i<display.width(); i+=4) {
      display.drawLine(display.width()-1, 0, i, display.height()-1, SSD1306_WHITE);
      display.display();
      delay(1);
    }

    delay(2000);
  }
```

```
void testdrawrect(void) {
  display.clearDisplay();

  for(int16_t i=0; i<display.height()/2; i+=2) {
    display.drawRect(i, i, display.width()-2*i, display.height()-2*i, SSD1306_WHITE);
    display.display();
    delay(1);
  }

  delay(2000);
}

void testfillrect(void) {
  display.clearDisplay();

  for(int16_t i=0; i<display.height()/2; i+=3) {
    display.fillRect(i, i, display.width()-i*2, display.height()-i*2, SSD1306_INVERSE);
    display.display();
    delay(1);
  }

  delay(2000);
}

void testdrawcircle(void) {
  display.clearDisplay();

  for(int16_t i=0; i<max(display.width(),display.height())/2; i+=2) {
    display.drawCircle(display.width()/2, display.height()/2, i, SSD1306_WHITE);
    display.display();
    delay(1);
  }

  delay(2000);
}

void testfillcircle(void) {
  display.clearDisplay();

  for(int16_t i=max(display.width(),display.height())/2; i>0; i-=3) {
    display.fillCircle(display.width() / 2, display.height() / 2, i, SSD1306_INVERSE);
    display.display();
    delay(1);
  }

  delay(2000);
}

void testfillcircle(void) {
  display.clearDisplay();

  for(int16_t i=max(display.width(),display.height())/2; i>0; i-=3) {
    display.fillCircle(display.width() / 2, display.height() / 2, i, SSD1306_INVERSE);
    display.display();
    delay(1);
  }

  delay(2000);
}

void testdrawroundrect(void) {
  display.clearDisplay();

  for(int16_t i=0; i<display.height()/2-2; i+=2) {
    display.drawRoundRect(i, i, display.width()-2*i, display.height()-2*i,
      display.height()/4, SSD1306_WHITE);
    display.display();
    delay(1);
  }

  delay(2000);
}

void testfillroundrect(void) {
  display.clearDisplay();

  for(int16_t i=0; i<display.height()/2-2; i+=2) {
    // The INVERSE color is used so round-rects alternate white/black
    display.fillRoundRect(i, i, display.width()-2*i, display.height()-2*i,
      display.height()/4, SSD1306_INVERSE);
    display.display();
    delay(1);
  }

  delay(2000);
}

void testdrawtriangle(void) {
  display.clearDisplay();
```

```
  for(int16_t i=0; i<max(display.width(),display.height())/2; i+=5) {
    display.drawTriangle(
      display.width()/2  , display.height()/2-i,
      display.width()/2-i, display.height()/2+i,
      display.width()/2+i, display.height()/2+i, SSD1306_WHITE);
    display.display();
    delay(1);
  }

  delay(2000);
}

void testfilltriangle(void) {
  display.clearDisplay();

  for(int16_t i=max(display.width(),display.height())/2; i>0; i-=5) {
    // The INVERSE color is used so triangles alternate white/black
    display.fillTriangle(
      display.width()/2  , display.height()/2-i,
      display.width()/2-i, display.height()/2+i,
      display.width()/2+i, display.height()/2+i, SSD1306_INVERSE);
    display.display();
    delay(1);
  }

  delay(2000);
}

void testdrawchar(void) {
  display.clearDisplay();

  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 0);
  display.cp437(true);

  for(int16_t i=0; i<256; i++) {
    if(i == '\n') display.write(' ');
    else          display.write(i);
  }

  display.display();
  delay(2000);
}
void testdrawstyles(void) {
  display.clearDisplay();

  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0,0);
  display.println(F("Hello, world!"));

  display.setTextColor(SSD1306_BLACK, SSD1306_WHITE);
  display.println(3.141592);

  display.setTextSize(2);
  display.setTextColor(SSD1306_WHITE);
  display.print(F("0x")); display.println(0xDEADBEEF, HEX);

  display.display();
  delay(2000);
}

void testscrolltext(void) {
  display.clearDisplay();

  display.setTextSize(2);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(10, 0);
  display.println(F("scroll"));
  display.display();
  delay(100);

  display.startscrollright(0x00, 0x0F);
  delay(2000);
  display.stopscroll();
  delay(1000);
  display.startscrollleft(0x00, 0x0F);
  delay(2000);
  display.stopscroll();
  delay(1000);
  display.startscrolldiagright(0x00, 0x07);
  delay(2000);
  display.startscrolldiagleft(0x00, 0x07);
  delay(2000);
  display.stopscroll();
  delay(1000);
}
```

```
void testdrawbitmap(void) {
  display.clearDisplay();

  display.drawBitmap(
    (display.width()  - LOGO_WIDTH ) / 2,
    (display.height() - LOGO_HEIGHT) / 2,
    logo_bmp, LOGO_WIDTH, LOGO_HEIGHT, 1);
  display.display();
  delay(1000);
}

#define XPOS   0
#define YPOS   1
#define DELTAY 2

void testanimate(const uint8_t *bitmap, uint8_t w, uint8_t h) {
  int8_t f, icons[NUMFLAKES][3];

  for(f=0; f< NUMFLAKES; f++) {
    icons[f][XPOS]   = random(1 - LOGO_WIDTH, display.width());
    icons[f][YPOS]   = -LOGO_HEIGHT;
    icons[f][DELTAY] = random(1, 6);
    Serial.print(F("x: "));
    Serial.print(icons[f][XPOS], DEC);
    Serial.print(F(" y: "));
    Serial.print(icons[f][YPOS], DEC);
    Serial.print(F(" dy: "));
    Serial.println(icons[f][DELTAY], DEC);
  }

  for(;;) {
    display.clearDisplay();

    for(f=0; f< NUMFLAKES; f++) {
      display.drawBitmap(icons[f][XPOS], icons[f][YPOS], bitmap, w, h, SSD1306_WHITE);
    }

    display.display();
    delay(200);

    // Then update coordinates of each flake...
    for(f=0; f< NUMFLAKES; f++) {
      icons[f][YPOS] += icons[f][DELTAY];
      // If snowflake is off the bottom of the screen...
      if (icons[f][YPOS] >= display.height()) {
        // Reinitialize to a random position, just off the top
        icons[f][XPOS]   = random(1 - LOGO_WIDTH, display.width());
        icons[f][YPOS]   = -LOGO_HEIGHT;
        icons[f][DELTAY] = random(1, 6);
      }
    }
  }
}
```

4. Save file, kemudian lakukan verify & upload
5. Maka hasil nya akan jadi seperti ini :