BAB 10
Komunikasi Serial

10.1 Tujuan Praktikum

1. Mahasiswa mampu mengenal komunikasi serial
2. Mahasiswa mampu memahami penggunan komunikasi serial
3. Mahaiswa mampu mengimplementasikan komunikasi serial

10.2 Kebutuhan Praktikum

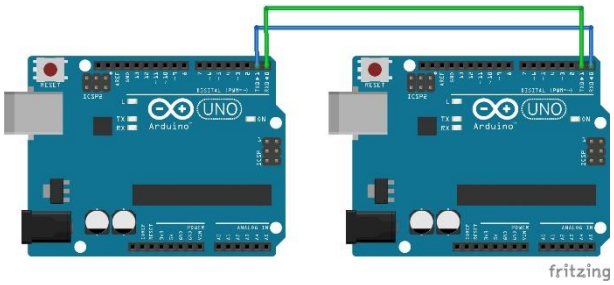
1. Komputer/Laptop
2. Arduino IDE
3. Trainer Mikrokontroler
4. Modul Sistem Embedded 2023

10.3 Landasan Teori

A. Komunikasi Serial

1. Pengertian

Komunikasi serial digunakan oleh arduino untuk dapat berkomunikasi dengan komputer seperti pada *com* monitor, itu adalah salah satu fungsi dari komunikasi serial antar arduino dengan komputer. Selain dapat berkomunikasi dengan komputer secara *default*, komunikasi serial juga digunakan untuk berkomunisai dengan perangkat lain seperti mikrokontroller lain, modul *bluetooth*, sensor berbasis serial, dll. Dengan adanya komunikasi serial ini maka arduino tak hanya bisa mengolah data dari pin input dan outputnya saja, tetapi juga bisa dikomunikasikan secara dua arah dengan perangkat komputer untuk menampilkan hasil pengolahan datanya.



Gambar 10.1 Komunikasi Serial Antar Arduino

Komunikasi serial Arduino memungkinkan kita dapat mengontrol arduino melalui komputer agar dapat memantau sesuatu yang terjadi padanya. Komunikasi yang terjadi secara serial hanya membutuhkan 2 pin saja yaitu RX

dan TX. RX biasa disebut sebagai *Receive* sedangkan TX disebut sebagai *Transmit*. Pin komunikasi serial Arduino terletak pada pin digital 0 (RX) dan 1 (TX), yang terhubung juga pada USB to Serial. Jika pin serial tidak digunakan maka pin tersebut bisa digunakan sebagai *input* / *output*.

Dalam sekali transmisi, komunikasi serial dapat mengirim beberapa bit data secara langsung. Dalam menggunakan komunikasi serial, kita juga harus menyamakan nilai *baudrate*. *Baudrate* merupakan istilah yang digunakan untuk kecepatan aliran data. Satuan baudrate adalah bps (*bit per second*). Untuk dapat berkomunikasi dengan komputer, terdapat beberapa pilihan *baudrate* pada arduino yaitu 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, atau 115200.

2. Jenis – Jenis Perintah Komunikasi Serial

   Saat kita menggunakan komunikasi serial ada beberapa jenis perintah yang bisa digunakan, yaitu :

   a. *If* (Serial)

      Perintah ini berfungsi untuk mengecek apakah port serial sudah siap atau belum.

   b. *Serial.begin*(nilai baudrate)

      Serial.begin digunakan untuk menentukan kecepatan dan penerimaan data melalui *port* serial. Kecepatan yang umumnya digunakan adalah 9600 *bit per second* (9600 bps).

   c. *Serial.end()*

      Serial.end digunakan untuk menghentikan program yang akan diperintah oleh komunikasi serial.

   d. *Serial.find()*

      Perintah ini digunakan untuk mencari *string* dalam *buffer* data.

   e. *Serial.findUntil()*

      Fungsi perintah ini adalah untuk mencari *buffer* data hingga data sesuai kriteria yang diberikan bisa ditemukan.

   f. *Serial.available()*

      Serial.available berguna untuk menghasilkan jumlah *byte* di *port* serial yang belum terbaca. Jika *port* serial datanya sedang kosong, fungsi ini bakal menghasilkan nol atau datanya tidak bisa terbaca lagi.

g. *Serial.read()*

Serial.read berguna untuk membaca satu *byte* data yang terdapat di *port* serial. Setelah pemanggilan *Serial.read()*, jumlah data di *port* serial berkurang satu.

h. *Serial.print*(data)

Serial.print digunakan untuk mengirimkan data ke *port* serial. Jika datanya kita masukkan ke *port* serial, maka yang dikirim akan menyesuaikan format tersebut. Dalam hal ini, format yang digunakan bisa berupa bilangan, *character*, *string*.

i. *Serial.println*(data)

Seria.println memiliki fungsi yang hampir sama dengan serial *print*. Bedanya, setelah data dicetak, selanjutnya data akan di *print* di garis baru dan letaknya dibawah data terakhir (seperti *newline*).

j. *Serial.readBytes()*

Perintah ini digunakan untuk membaca data *byte* yang diterima

k. *Serial.write()*

Fungsi perintah ini adalah untuk membaca data biner dari *port* serial. Biasanya data yang terkirim dalam bentuk *byte* atau deretan data *byte*.

l. *Serial.flush()*

Serial.flush digunakan untuk pengosongan data pembacaan pada *buffer*.

m. *Serial.parseFloat()*

Serial.parseFloat berfungsi untuk bilangan titik mengambang atau real.

n. *Serial.parseln()*

Serial.parseln digunakan untuk menghasilkan nilai bulat.

o. *Serial.setTimeout()*

Penggunaan perintah ini biasanya untuk mengatur batas maksimum waktu tunggu (*timeout*) proses transmisi data

p. *Serial.peek()*

Berfungsi untuk mengambil data berikutnya di *buffer* penerima

q. *Serial.serialEvent()*

Perintah ini berfungsi layaknya interupsi serial yang biasanya dipanggil jika data datang atau diterima

3. Alternatif Komunikasi Serial

Ada beberapa alternatif dalam menggunakan komunikasi serial diantaranya UART, I2C, dan SPI. Masing – masing alternatif memiliki cara penggunaan yang berbeda. Berikut penjelasan dari masing – masing alternatif :

a. UART(*Universal Asynchronous Receiver Transmitter*)

Pada komunkasi serial UART, port yang dipakai adalah port komunikasi serial TX dan RX, atau pin D0 dan D1 pada Arduino. Cara menghubungkannya adalah silang, yaitu TX dihubungkan ke RX , RX dihubungkan ke TX. Pengiriman data dilakukan dengan menggunakan *library* Serial pada Arduino. Pada komunikasi ini hanya dapat dihubungkan 2 buah *board* arduino.

Komunikasi UART pada Arduino dapat menggunakan *hardware* dan *software*. Secara *hardware*, komunikasi UART terhubung ke pin D0 dan D1. Jika perlu tambahan komunikasi serial, dapat menggunakan *library SoftwareSerial*. Jika menggunakan *library SoftwareSerial*, *port* yang digunakan bebas.

b. I2C (*Inter-Integrated Circuit*)

Inter-Integrated Circuit atau I2C merupakan sebuah protokol komunikasi yang digunakan untuk pertukaran data antar perangkat mikrokontroler dan sensor atau pun perangkat lainnya. I2C dirancang untuk menghubungkan berbagai perangkat dalam sebuah sistem menggunakan jalur komunikasi bersama. I2C banyak digunakan di beberapa perangkat, seperti sensor, EEPROM, RTC dan berbagai jenis *chip* lainnya yang mendukung komunikasi melalui I2C.

Komunikasi I2C pada Arduino biasanya dilakukan menggunakan dua pin, yaitu SDA (*Serial Data Line*) dan SCL (*Serial Clock Line*). SDA digunakan untuk mengirim data dan SCL digunakan untuk mengirimkan sinyal *clock* yang akan mengatur timing komunikasi antar perangkat yang terhubung. Berikut ini uraian singkat mengenai bagaimana komunikasi I2C digunakan antara dua perangkat pada arduino.

Pada komunikasi I2C, terdapat salah satu perangkat yang berperan sebagai "*Master*" dan perangkat lainnya berperan sebagai "*Slave*". *Master* yaitu perangkat yang menginisiasi komunikasi dan mengontrol alur

komunikasi, sedangkan *slave* yaitu perangkat yang menerima instruksi dari master.

c. SPI (*Serial Peripheral Interface*)

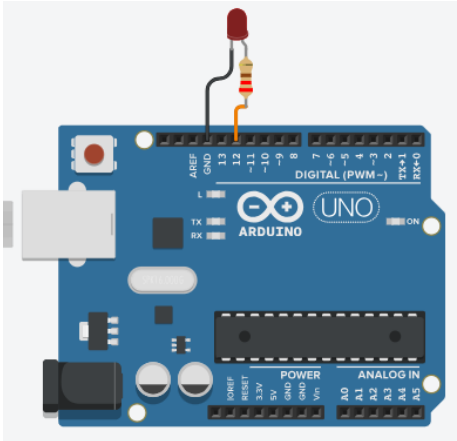
SPI (*Serial Peripheral Interface*) merupakan komunikasi seri *synchronous* yang berarti harus menggunakan clock yang sama untuk mengsinkronisasi deteksi *bit* pada *receiver*. Biasanya hanya digunakan untuk komunikasi jarak pendek dengan mikrokontroler lain yang terletak pada papan rangkaian yang sama. Bus SPI dikembangkan untuk menyediakan komunikasi dengan kecepatan tinggi dengan menggunakan pin mikrokontroler yang sedikit. Koneksi SPI yaitu device yang terhubung satu sama lain akan bersifat *Full Duplex* yang berarti ada *device* yang bertindak sebagai *Master* dan *Slave*.

Master device yaitu perangkat yang memulai sambungan dengan cara menginisialisasi SPI *address* dari *slave device*. Kemudian *master* dan *slave* bisa mengirim atau pun menerima data. Hal ini dikarenakan komunikasi *full duplex* yang artinya *master* dan slave bisa menerima atau pun mengirim data. *Slave device* bisa mengirim atau menerima data dalam waktu yang bersamaan.

10.4 Tugas Praktikum

A. Komunikasi Serial Dengan Komputer

1. Desain



Gambar 10.2 Desain Rangkaian

2. Rute Wiring

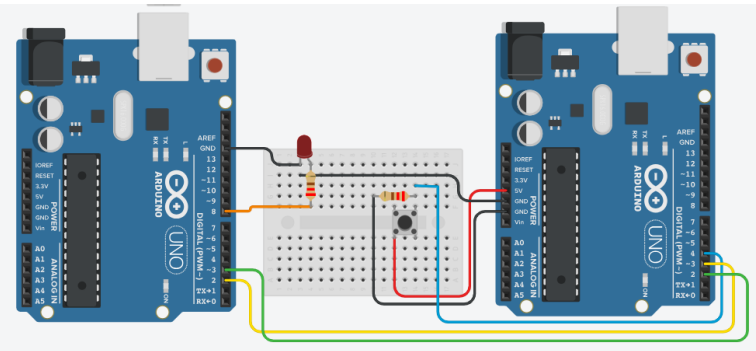
Komponen	Pin Konmponen	Pin Arduino
LED	Cathode (-)	GND
	Anode (+)	12

3. Source Code

```
int led = 12;
String data;
void setup()
{
  pinMode(led, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  while(Serial.available() > 0){
      delay(10);
    char input = Serial.read();
    data += input;
  }
  if(data.length() >0 ){
    Serial.println(data);
    if(data == "on"){
      digitalWrite(led, HIGH);
      }else if(data == "off"){
      digitalWrite(led, LOW);
      }
    data = "";
  }
  delay(1000);
}
```

B. Komunikasi Serial Dengan Dua Arduino (UART)

1. Desain



Gambar 10.3 Desain Rangkaian

2. Rute Wiring

a. Arduino 1

Komponen	Pin Konmponen	Pin Arduino
LED	Cathode (-)	GND
	Anode (+)	12
Arduino 2	GND	GND

b. Arduino 2

Komponen	Pin Konmponen	Pin Arduino
Button	Terminal 1a (+)	VCC (5V)
	Terminal 2b	GND
	Terminal 2b	4
Arduino 1	GND	GND

3. Source Code

a. Arduino 1

```
#include <SoftwareSerial.h>
SoftwareSerial newSerial(2,3);
int ledPin = 8;
int val;
String data;

bool status_led = false;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(2, INPUT);
  pinMode(3, OUTPUT);
  Serial.begin(9600);
  newSerial.begin(9600);
}
void loop() {
  while (newSerial.available() > 0) {
    char input = newSerial.read();
    data += input;
  }

  if(data.length() > 0){
    val = data.toInt();
    Serial.print("Data diterima: ");
    Serial.println(val);
    if (val == 0) {
      status_led = !status_led;
    }
        digitalWrite(ledPin, status_led);
    data ="";
  }
}
```

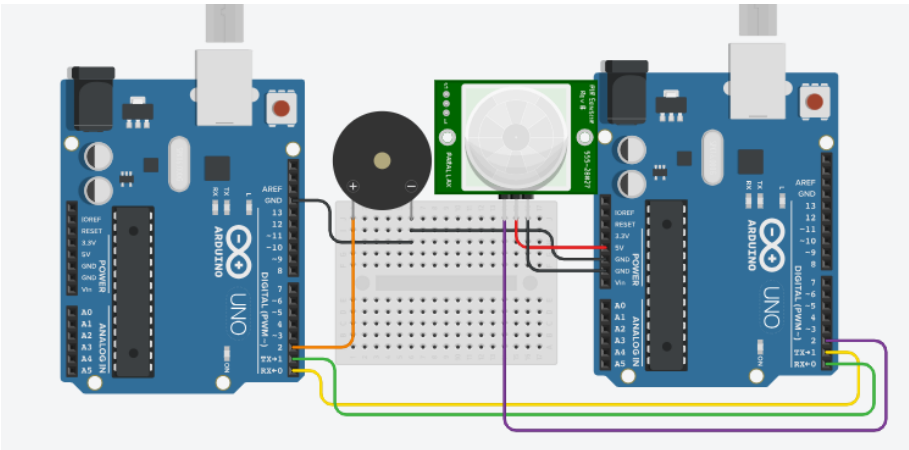
b. Arduino 2

```
#include <SoftwareSerial.h>
SoftwareSerial newSerial(2,3);
int button1 = 4;

void setup() {
  pinMode(button1, INPUT);
  pinMode(2, INPUT);
  pinMode(3, OUTPUT);
  Serial.begin(9600);
  newSerial.begin(9600);
```

```
}

void loop() {
  int valBtn = digitalRead(4);
  newSerial.print(valBtn);
  Serial.println(valBtn);
  delay(500);
}
```

10.5 Tugas Rumah

Buatlah sebuah program pada tinkercad dengan menerapkan komunikasi serial. Contoh desain rangkaian :



ketentuan :

1. Pada arduino pertama menggunakan 1 sensor.
2. Pada arduino kedua menggunakan 1 aktuator.
3. Tidak boleh sama seperti contoh desain rangkaian.