



BUKU KERJA PRAKTIK MAHASISWA (BKPM)

WORKSHOP SISTEM TERTANAM

TKK30706

SEMESTER III

OLEH :

BENI WIDIAWAN

PROGRAM STUDI TEKNIK KOMPUTER

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI JEMBER

TAHUN 2022

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET DAN TEKNOLOGI
POLITEKNIK NEGERI JEMBER

LEMBAR PENGESAHAN

WORKSHOP SISTEM TERTANAM

Mengetahui,

Koord. Program Studi,

Koord./Tim Mata Kuliah,

Penulis,



Yogiswara, ST., MT.

NIP.197009292003121001



Beni Widiawan, S.ST., MT.

NIP.197808162005011002



Beni Widiawan, S.ST., MT.

NIP.197808162005011002

Menyetujui,

Ketua Jurusan Teknologi Informasi,

Hendra Yufit Riskiawan, S.Kom., M.Cs.

NIP. 198302032006041003

KATA PENGANTAR

Puji Syukur kehadiran Tuhan Yang Maha Esa karena atas limpahan rahmat-Nya sehingga kami dapat menyelesaikan Buku Kerja Praktikum Mahasiswa (BKPM) Workshop Sistem Tertanam. Materi Workshop terdiri dari bekerja dengan mikrokontroler arduino, ESP32, dan SBC Raspberry Pi. BKPM ini disusun berdasarkan metode Student Center Learning yaitu menempatkan mahasiswa sebagai pusat kegiatan belajar. BKPM ini terdiri dari Pokok Bahasan, Acara Praktikum, Tempat, Alokasi Waktu, Capaian Pembelajaran Mata Kuliah (CPMK), Indikator, Dasar Teori, Alat dan Bahan, Prosedur Kerja, Hasil dan Pembahasan, Kesimpulan, Rubrik Penilaian. Kami menyadari masih banyak kekurangan dalam penyusunan BKPM ini. Oleh karena itu, kami sangat mengharapkan kritik dan saran demi perbaikan dan kesempurnaan modul ini. Kami mengucapkan terima kasih kepada berbagai pihak yang telah membantu proses penyelesaian BKPM ini. Semoga BKPM ini dapat bermanfaat bagi kita semua.

DAFTAR ISI (menyusul)

Acara 1

Pokok Bahasan	: Workshop Mikrokomputer Arduino
Acara Praktikum/Praktik	: 1 dan 2
Tempat	: Laboratorium SKK
Alokasi Waktu	: 2 x (PB: 2x2x170; PT: 2x4x60; M: 1x2x60)

a. Capaian Pembelajaran Mata Kuliah (CPMK) :

1. Mampu mendesain dan membuat shield modul arduino hingga teknik masking dan pemasangan komponen sesuai skematik
2. Mampu membuat program pada mikrokomputer arduino dengan tingkat keberhasilan program 100%, sehingga memahami teoritis pemrograman mikrokomputer arduino

b. Indikator Penilaian :

1. Ketepatan dalam mengerjakan sesuai panduan
2. Keberhasilan percobaan sesuai panduan

c. Dasar Teori

Soil moisture sensor ini dapat mengetahui besarnya kelembaban didalam tanah. Sangat cocok digunakan untuk prototyping project monitoring kebun, pengontrolan pengairan/irigasi, ataupun project IoT (Internet of Things) agriculture.

Prinsip kerja sensor ini yaitu dengan mengalirkan arus pada dua probe maka resistansi yang terbaca berbanding lurus dengan jumlah kelembaban yang terdeteksi. Makin banyak cairan maka lebih mudah mengalirkan listrik dengan kata lain resistansinya kecil. Sebaliknya jika resistansinya besar maka listrik yang mengalir akan kecil yang kita asumsikan tanah tersebut makin kering.



DHT11 adalah sensor multifungsi, yaitu sensor yang memberikan informasi suhu dan kelembaban relatif disaat bersamaan. Penggunaan sensor DHT11 dapat digunakan untuk pengukuran keperluan secara umum. Sensor DHT11 memberikan informasi pembacaan yang cukup bisa diandalkan ketika berada pada lingkungan dengan kelembaban 20% RH ~ 90% RH, dan kondisi suhu lingkungan antara 0 °C ~ 50 °C. Penggunaan sangat cocok pada penggunaan keperluan setiap hari dengan kondisi yang tidak ekstrim.



Sensor Gas MQ-4 adalah komponen elektronika yang dapat mendeteksi kadar gas alam terkompresi / CNG (compressed natural gas) yang utamanya mengandung gas metana (methane, CH₄) yang merupakan bentuk zat paling sederhana dari hidrokarbon. Walaupun tidak bersifat racun, gas metana dapat berbahaya karena mudah terbakar (combustive / flammable gas). Gas ini tidak berbau dan tidak berwarna, menjadikannya sulit untuk dideteksi secara langsung oleh manusia.

Metana (CH₄) adalah gas tidak berbau yang menimbulkan efek rumah kaca. Koefisien daya tangkap panas metana lebih tinggi 25 kali dibanding karbon dioksida (CO₂).



ML8511 adalah modul sensor cahaya ultraviolet analog yang sangat mudah digunakan. Sensor ini menghasilkan tegangan analog (0 - 3.3V) sebanding dengan jumlah cahaya UV yang diterima. Ini berguna untuk device yang mengukur dan mengingatkan bahaya matahari (UV) atau mendeteksi indeks UV berkaitan dengan kondisi cuaca. Sensor ini mendeteksi cahaya 280-390nm. Rentang spektrum ini termasuk spektrum UVB (burning rays) dan sebagian UVA (tanning rays)



Modul RFID Card Reader/Writer ini sangatlah praktis untuk digunakan untuk membaca kartu dan keychain RFID karena ukurannya cukup kecil dan berdaya cukup rendah. Modul RFID Card Reader/Writer ini menggunakan teknologi MIFARE Type A 13.56 MHz (ISO/IEC 14443) A/MIFARE mode yang dirilis oleh NXP Semiconductor dengan sistem keamanan berbasis Crypto-1 (pada seri Classic) dan Triple-DES / AES (pada seri DESFire)

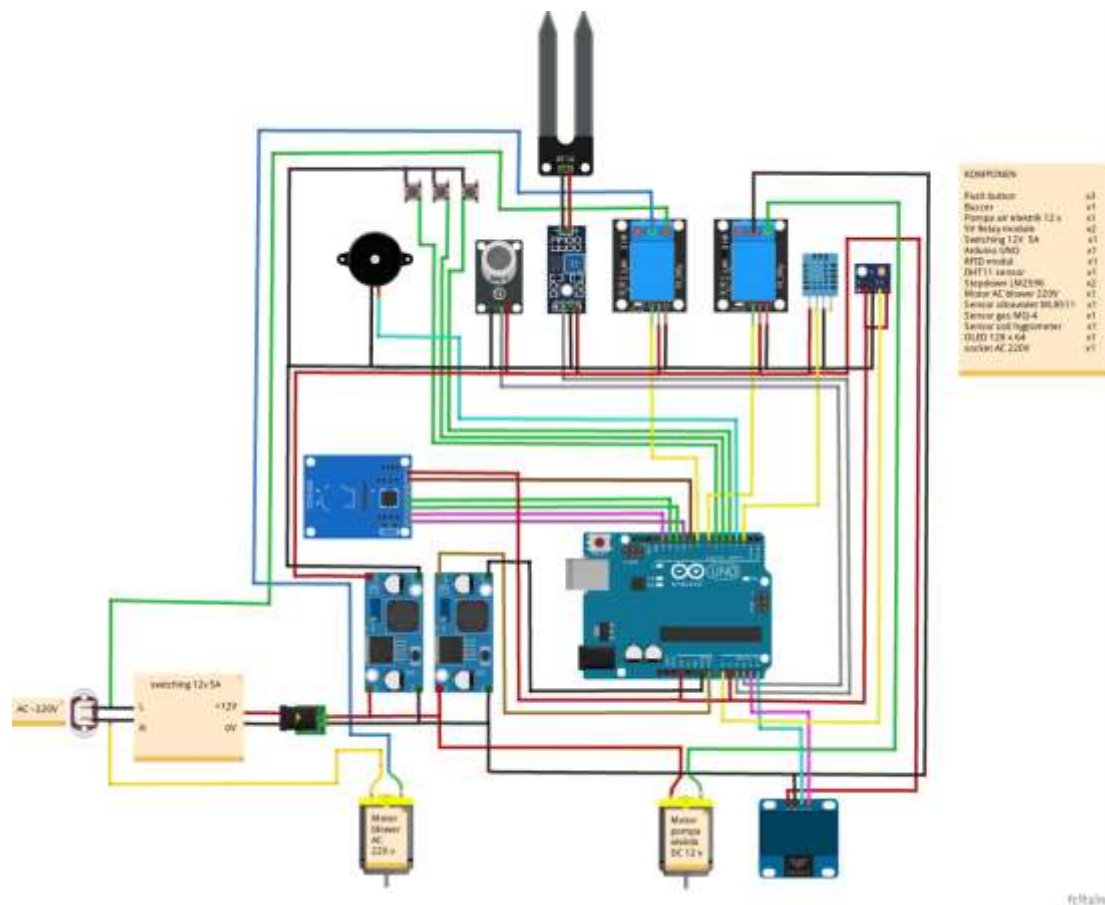
Aktuator adalah perangkat yang mengubah besaran listrik menjadi besaran fisis. Besaran fisis yang dimaksud adalah gerakan, cahaya, panas maupun tekanan ataupun magnetis. Aktuator menjadi bagian penting dari sistem kendali karena menentukan pergerakan dari sebuah proses. Aktuator digerakkan dengan kontrol dari [mikrokontroler](#), komputer ataupun PLC. Aktuator listrik yang mengubah besaran listrik menjadi gerakan mekanis diantaranya adalah Motor DC, Motor AC, Motor Stepper, Motor Servo, Selenoid Valve



d. Alat dan Bahan :

1. PCB fiber 10x20	1x
2. Arduino Uno	1x
3. Pushbutton	3x
4. Piezo Buzzer	1x
5. Pompa elektrik 12V	1x
6. Relay 5V 2Ch	1x
7. Switching 12V 5A	1x
8. RFID modul	1x
9. DHT11	1x
10. Stepdown LM2596	1x
11. Motor AC 220V	1x
12. UV sensor : ML8511	1x
13. Gas sensor : MQ-4	1x
14. Sensor Soil Hygrometer	1x
15. OLED 128x64	1x

e. Prosedur Kerja :



f. Hasil dan Pembahasan :

Praktikum 1 – Membaca RFID


```

#include <SPI.h>
#include <Wire.h>
#include <Adafruit_SSD1306.h>
#include <Adafruit_GFX.h>
#include <MFRC522.h>

boolean backlight = true;
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 32
#define OLED_RESET 4
#define NUMFLAKES 10
#define SS_PIN 10
#define RST_PIN 9
MFRC522 mfrc522(SS_PIN, RST_PIN);
#define ALARM 3
String content= "";

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
OLED_RESET);
void setup() {
  // put your setup code here, to run once:
  pinMode(ALARM,OUTPUT);
  SPI.begin();
  mfrc522.PCD_Init();
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.begin();
  display.clearDisplay();
  displayString(2,"POLIJE",35,10);
  delay(1000);
  Serial.begin(9600);
}

void loop() {
  readRfid();
  while(1){
    display.setTextSize(1);
    display.clearDisplay();
    display.setTextColor(WHITE,BLACK );
    display.setCursor(0, 0);
    display.println("Nama : ");
    display.setCursor(40, 0);
    display.println("USER");
    display.setCursor(0, 10);
    display.println("ID : ");
    display.setCursor(30, 10);
    display.println(content.substring(1));
    display.display();
  }
}

void displayString(int size,String kata, int x, int y)

```

```

{
    display.setTextColor( WHITE,BLACK);
    display.setTextSize(size);
    display.clearDisplay();
    display.setCursor(x, y);
    display.println(kata);
    display.setTextSize(size);
    display.display();
}
void readRfid() {
while(1){
// Look for new cards
    while( ! mfrc522.PICC_IsNewCardPresent())
    {
        displayString(1,"Tap your Card",30,10);
    }
// Select one of the cards
    while( ! mfrc522.PICC_ReadCardSerial())
    {
        displayString(1,"Tap your Card",30,10);
    }

//Show UID on serial monitor
    byte letter;
    for (byte i = 0; i < mfrc522.uid.size; i++)
    {
        content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));
        content.concat(String(mfrc522.uid.uidByte[i], HEX));
    }
    for(int i=0;i<2;i++){
        digitalWrite(ALARM,HIGH);
        delay(80);
        digitalWrite(ALARM,LOW);
        delay(80);
    }
    Serial.println(content.substring(1));
    content.toUpperCase();
    break;
}
}
void drawProgressbar(int x,int y, int width,int height, int progress)
{
    float bar = ((float)(width-1) / 100) * progress;
    display.drawRect(x, y, width, height, WHITE);
    display.fillRect(x+2, y+2, bar , height-4, WHITE);
}
void LOADING(){
    for(int i=0;i<100;i++){
        digitalWrite(ALARM,HIGH);
        delay(8);
    }
}

```

```

digitalWrite(ALARM,LOW);
delay(8);
display.clearDisplay();
drawProgressbar(0,10,120,10,i);
display.display();
delay(8);
}
for(int i=0;i<2;i++){
digitalWrite(ALARM,HIGH);
delay(80);
digitalWrite(ALARM,LOW);
delay(80);
}
}
void LOADING2(){
for(int i=0;i<100;i++){
digitalWrite(ALARM,HIGH);
delay(2);
digitalWrite(ALARM,LOW);
delay(2);
display.clearDisplay();
drawProgressbar(0,10,120,10,i);
display.display();
delay(2);
}
for(int i=0;i<2;i++){
digitalWrite(ALARM,HIGH);
delay(80);
digitalWrite(ALARM,LOW);
delay(80);
}
}
}

```

-
1. Jalankan modul SPD dan download program tersebut
 2. Tempelkan kartu (putih) pada RFID reader, dan apa yang muncul pada display

3. Tempelkan gantungan RFID (biru) pada RFID reader, dan tulis yang muncul pada display

Praktikum 2 – Sensor Soil Moisture

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

#define MOISTURE A2
boolean backlight = true;
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 32
#define OLED_RESET 4
#define NUMFLAKES 10
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
OLED_RESET);
int readSens;

void setup() {
  // put your setup code here, to run once:
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.begin();
  display.clearDisplay();
  displayString(2,"POLIJE",35,10);
  delay(1000);
  pinMode(MOISTURE, INPUT);
  Serial.begin(9600);
}

void loop() {
  readMoisture();
  displayIntMenuPage(readSens);
}

void readMoisture(){
  int readOne = analogRead(MOISTURE);
  readSens = map(readOne,0,1023,100,0);
  Serial.println(readSens);
  delay(500);
}
```

```

void displayIntMenuPage(float value)
{
    display.setTextSize(1);
    display.clearDisplay();
    display.setTextColor(WHITE,BLACK );
    display.setCursor(25, 0);
    display.println("SOIL SENSOR");
    display.drawFastHLine(0,10,127,WHITE);
    display.setCursor(5, 15);
    display.println("Value");
    display.setTextSize(2);
    display.setCursor(40, 14);
    display.println(value);
    display.setTextSize(2);
    display.setCursor(110, 14);
    display.println("%");
    display.setTextSize(2);
    display.display();
}

void displayString(int size,String kata, int x, int y)
{
    display.setTextColor( WHITE,BLACK);
    display.setTextSize(size);
    display.clearDisplay();
    display.setCursor(x, y);
    display.println(kata);
    display.setTextSize(size);
    display.display();
}

```

1. Jalankan modul SPD dan download program tersebut
2. Tuliskan apa yang muncul pada display

Praktikum 3 – Sensor Ultra Violet

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

boolean backlight = true;
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 32
#define OLED_RESET 4
#define NUMFLAKES 10
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
OLED_RESET);

int UVOUT = A0;
float uvIntensity;
void setup() {
    // put your setup code here, to run once:
    Serial.begin(9600);
    display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
    display.begin();
    display.clearDisplay();
    displayString(2,"POLIJE",35,10);
    delay(1000);
}

void loop() {
    // put your main code here, to run repeatedly:
    readUV_sensor();
    displayIntMenuPage(uvIntensity);
}
void displayIntMenuPage(float value)
{
    display.setTextSize(1);
    display.clearDisplay();
    display.setTextColor(WHITE,BLACK );
    display.setCursor(25, 0);
    display.println("UV SENSOR");
    display.drawFastHLine(0,10,127,WHITE);
    display.setCursor(5, 15);
    display.println("Value");
    display.setTextSize(2);
    display.setCursor(40, 14);
    display.println(value);
    display.setTextSize(2);
    display.setCursor(110, 14);
    display.println("mW/cm");
    display.setTextSize(2);
```

```

    display.display();
}
void displayString(int size,String kata, int x, int y)
{
    display.setTextColor( WHITE,BLACK);
    display.setTextSize(size);
    display.clearDisplay();
    display.setCursor(x, y);
    display.println(kata);
    display.setTextSize(size);
    display.display();
}
float readUV_sensor()
{
    int uvLevel = analogRead(UVOUT);
    int REF_3V3 = A1;
    int refLevel = averageAnalogRead(REF_3V3);

    float outputVoltage = 3.3 / refLevel * uvLevel;

    uvIntensity = mapfloat(outputVoltage, 0.99, 2.8, 0.0, 15.0);
    Serial.println(uvIntensity);
    delay(100);
}
int averageAnalogRead(int pinToRead)
{
    byte numberOfReadings = 8;
    unsigned int runningValue = 0;

    for(int x = 0 ; x < numberOfReadings ; x++)
        runningValue += analogRead(pinToRead);
    runningValue /= numberOfReadings;

    return(runningValue);
}
float mapfloat(float x, float in_min, float in_max, float out_min, float out_max)
{
    return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}

```

-
1. Jalankan modul SPD dan download program tersebut
 2. Tuliskan apa yang muncul pada display

Praktikum 4 – Sensor Gas MQ

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

#define MQ4 A3
boolean backlight = true;
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 32
#define OLED_RESET 4
#define NUMFLAKES 10
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
OLED_RESET);

int gas_sensor = A3;
float m = -0.318;
float b = 1.133;
float R0 = 4.0;
double percentage;
void setup() {

Serial.begin(9600);
display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
display.begin();
display.clearDisplay();
displayString(2,"POLIJE",35,10);
delay(1000);
}

void loop() {
// put your main code here, to run repeatedly:
readMQ();
displayIntMenuPage(percentage);

}
void readMQ() {
float sensor_volt;
float RS_gas;
float ratio;
float sensorValue = analogRead(MQ4);
sensor_volt = sensorValue*(5.0/1023.0);
RS_gas = ((5.0*10.0)/sensor_volt)-10.0;
ratio = RS_gas/R0;
double ppm_log = (log10(ratio)-b)/m;
double ppm = pow(10, ppm_log);
percentage = ppm/10000;
Serial.println(percentage);
```

```

    delay(100);
}
void displayIntMenuPage(float value)
{
    display.setTextSize(1);
    display.clearDisplay();
    display.setTextColor(WHITE,BLACK );
    display.setCursor(25, 0);
    display.println("GAS SENSOR");
    display.drawFastHLine(0,10,127,WHITE);
    display.setCursor(5, 15);
    display.println("Value");
    display.setTextSize(2);
    display.setCursor(40, 14);
    display.println(value);
    display.setTextSize(2);
    display.setCursor(110, 14);
    display.println("%");
    display.setTextSize(2);
    display.display();
}
void displayString(int size,String kata, int x, int y)
{
    display.setTextColor( WHITE,BLACK);
    display.setTextSize(size);
    display.clearDisplay();
    display.setCursor(x, y);
    display.println(kata);
    display.setTextSize(size);
    display.display();
}

```

1. Jalankan modul SPD dan download program tersebut
2. Tuliskan apa yang muncul pada display

Praktikum 5 – Sensor Suhu dan Kelembaban

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

#define DHTPIN 2
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
boolean backlight = true;
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 32
#define OLED_RESET 4
#define NUMFLAKES 10
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
OLED_RESET);

float suhu, kelembaban;

void setup() {
  // put your setup code here, to run once:
  dht.begin();
  Serial.begin(9600);
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.begin();
  display.clearDisplay();
  displayString(2,"POLIJE",35,10);
  delay(1000);
}

void loop() {
  // put your main code here, to run repeatedly:
  readDHT_sensor();
  displayIntMenuPage2(suhu,kelembaban);
}
void displayIntMenuPage2(float valueA, float valueB)
{
  display.setTextSize(1);
  display.clearDisplay();
  display.setTextColor(WHITE,BLACK );
  display.setCursor(25, 0);
  display.println("DHT SENSOR");
  display.drawFastHLine(0,10,127,WHITE);
  display.setCursor(15, 15);
```

```

display.println("SUHU = ");
display.setTextSize(1);
display.setCursor(55, 15);
display.println(valueA);
display.setTextSize(1);
display.setCursor(90, 15);
display.println("C");
display.setCursor(15, 25);
display.println("KELEMBABAN = ");
display.setTextSize(1);
display.setCursor(95, 25);
display.println(valueB);
display.setTextSize(1);
display.display();
}
void displayString(int size,String kata, int x, int y)
{
    display.setTextColor( WHITE,BLACK);
    display.setTextSize(size);
    display.clearDisplay();
    display.setCursor(x, y);
    display.println(kata);
    display.setTextSize(size);
    display.display();
}
void readDHT_sensor() {
    kelembaban = dht.readHumidity();
    suhu = dht.readTemperature();

    if (isnan(suhu) || isnan(kelembaban)) {
        return;
    }
    Serial.print(kelembaban);
    Serial.print(",");
    Serial.println(suhu);
    delay(500);
}

```

-
1. Jalankan modul SPD dan download program tersebut
 2. Tuliskan apa yang muncul pada display

Praktikum 6 – Menjalankan Pompa Air dan Kipas Pendingin

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_SSD1306.h>
#include <Adafruit_GFX.h>
int stateNowA = 0;
int stateBeforeA = 0;
int stateNowB = 0;
int stateBeforeB = 0;
int i = 0;
int j = 0;

String P = "ON";
String B = "ON";
int pompa = 8;
int blower = 7;
String button1State;
#define BTN_UP 4
#define BTN_DOWN 5
#define BTN_OK 6
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 32
#define OLED_RESET 4
#define NUMFLAKES 10
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
OLED_RESET);

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(BTN_UP,INPUT_PULLUP);
  pinMode(BTN_DOWN,INPUT_PULLUP);
  pinMode(BTN_OK,INPUT_PULLUP);
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  SPI.begin();
  Serial.begin(9600);
  pinMode(blower, OUTPUT);
  pinMode(pompa, OUTPUT);
  display.begin();
  display.clearDisplay();
  setContrast();
  delay(1000);
}
void loop(){
  serial();
  set();
  setRisetB();
  setRisetP();
}
```

```

void serial() {
  if (Serial.available()){
    button1State = Serial.readStringUntil('\n');
  }
  if (button1State == "Y"){
    B = "OFF";
    digitalWrite(blower,HIGH);
  }
  if (button1State == "N"){
    digitalWrite(blower,LOW);
    B = "ON";
  }
  if (button1State == "O"){
    digitalWrite(pompa,HIGH);
    P = "OFF";
  }
  if (button1State == "P"){
    digitalWrite(pompa,LOW);
    P = "ON";
  }
}

void set(){
  display.setTextColor( WHITE,BLACK);
  display.setTextSize(1);
  display.clearDisplay();
  display.setCursor(0, 0);
  display.println("BLOWER : ");
  display.setCursor(0, 15);
  display.println("POMPA : ");
  display.setCursor(64, 0);
  display.println(B);
  display.setCursor(64, 15);
  display.println(P);
  display.display();
}

void setContrast()
{
  display.display();
}

void setRisetB(){
  stateNowA = digitalRead(BTN_UP) ;

  if (stateNowA != stateBeforeA) {
    if (stateNowA==HIGH and i==0) {
      digitalWrite (7, HIGH);
      B = "OFF";
      j=1;
    }
    else if (stateNowA==LOW and j==1) {
      i=1;
    }
  }
}

```

```

    }
    else if (stateNowA==HIGH and i==1) {
        digitalWrite (7, LOW);
        B = "ON";
        j=0;
    }
    else if (stateNowA==LOW and j==0) {
        i=0;
    }
}
stateBeforeA=stateNowA;
}

void setRisetP(){
stateNowB = digitalRead(BTN_DOWN) ;
if (stateNowB != stateBeforeB) {
    if (stateNowB==HIGH and i==0) {
        digitalWrite (8, HIGH);
        P = "OFF";
        j=1;
    }
    else if (stateNowB==LOW and j==1) {
        i=1;
    }
    else if (stateNowB==HIGH and i==1) {
        digitalWrite (8, LOW);
        P = "ON";
        j=0;
    }
    else if (stateNowB==LOW and j==0) {
        i=0;
    }
}
stateBeforeB=stateNowB;
}

```

-
1. Jalankan modul SPD dan download program tersebut
 2. Tuliskan yang muncul pada display

Pompa air :

Kipas Pendingin :

g. Rubrik Penilaian :

Acara 2

Pokok Bahasan : Workshop Mikrokomputer SBC
Acara Praktikum/Praktik : 5, 6 dan 7
Tempat : Laboratorium SKK
Alokasi Waktu : 3 x (PB: 2x2x170; PT: 2x4x60; M: 1x2x60)

a. Capaian Pembelajaran Mata Kuliah (CPMK) :

1. Mampu melakukan instalasi dan memulai menggunakan SBC Raspberry Pi (instalasi, konfigurasi, pemrograman dasar dan pengenalan GPIO)
2. Mengerti dan mampu merakit dunia luar dengan SBC menggunakan GPIO, komunikasi serial dan USB. Serta mampu melakukan pemrograman sesuai panduan dengan tingkat keberhasilan 100%
3. Mengerti dan mampu menerapkan IoT pada SBC

b. Indikator Penilaian :

1. Ketepatan dalam mengerjakan sesuai panduan
2. Keberhasilan percobaan sesuai panduan

c. Dasar Teori :

Raspberry Pi, sering disingkat dengan nama Raspi, adalah komputer papan tunggal (single-board circuit; SBC) yang seukuran dengan kartu kredit yang dapat digunakan untuk menjalankan program perkantoran, permainan komputer, dan sebagai pemutar media hingga video beresousi tinggi. Raspberry Pi dikembangkan oleh yayasan nirlaba, Rasberry Pi Foundation, yang digawangi sejumlah pengembang dan ahli komputer dari Universitas Cambridge, Inggris.

Raspberry Pi memiliki website di www.raspberrypi.org yang memiliki cukup banyak informasi tentang produk, aksesoris sampai dengan contoh proyek pembelajaran yang dapat dikerjakan dengan Raspberry Pi. Berikut ini adalah tampilan website Raspberry Pi. Produk dan aksesoris Raspberry Pi dapat dilihat di website ini juga, ada banyak jenis Raspberry Pi yang dapat dipilih sesuai dengan kebutuhan. Aksesoris tambahan seperti Sense Hat, I/O Board Computer Module, Touch Display sampai dengan Case tersedia untuk melengkapi Raspberry Pi dalam memenuhi keinginan pengguna untuk merealisasikan ide kreatif dengan komputer mini ini.





GPIO atau general purpose input output adalah pin pada raspberry pi yang berfungsi sebagai input data maupun output data. Dari 40 pinout setelah digunakan untuk pin ground, 5V dan 3.3 V terdapat 28 pin yang tersisa. Dari 28 pin tersebut yang berfungsi sebagai GPIO sebanyak 17 pin. Pin lainnya merupakan pin alternatif seperti I2C bus, SPI Bus dan UART/Serial Bus. GPIO BCM0/pin 27 dan BCM1/pin 2 khusus digunakan untuk mengidentifikasi EEPROM pad modul HAT (Hardware on top).

GPIO memiliki output voltage sebesar 3.3 V dan tidak toleran terhadap over voltage input 5 V karena didalam board raspberry pi tidak terdapat over voltage protection. Jika ingin mengintegrasikan perangkat ini dengan mikrokontroler lain seperti arduino atau AVR yang bekerja di 5 V harus dibuatkan voltage shifter. Pemberian tegangan input pada GPIO yang melebihi kapasitas dapat merusak board raspberry pi. Setiap GPIO dapat dijadikan sebagai interrupt dengan trigger high/low/rise/fall/change.

Pada GPIO terdapat internal pull up sebesar 50 -65 kohm dan pull down sebesar 50-60 kohm yang dapat di enable atau di disable melalui software. Pin 3.3 v pada pinout raspberry pi dapat mengeluarkan arus maksimum 50mA sedangkan pada pin 5V dapat mengeluarkan arus hingga 300mA jika menggunakan supply 1A. beberapa pin GPIO juga bisa difungsikan sebagai output. PWM (Pulse Wide Modulation) dengan frekuensi output sampai orde Mhz. PWM ini dapat difungsikan untuk mengatur kecepatan motor DC dan sejenisnya.



Flask adalah salah satu framework micro website yang ditulis menggunakan bahasa pemrograman Python. Flask disebut dengan micro website karena tidak memerlukan beberapa tools/ library tambahan. Flask tidak memiliki lapisan abstraksi database, form validasi atau beberapa komponen pihak ketiga lainnya. Flask dibuat oleh Armin Ronacher yang berasal dari Poccoo sejak tahun 2010. Beberapa website seperti Pinterest dan

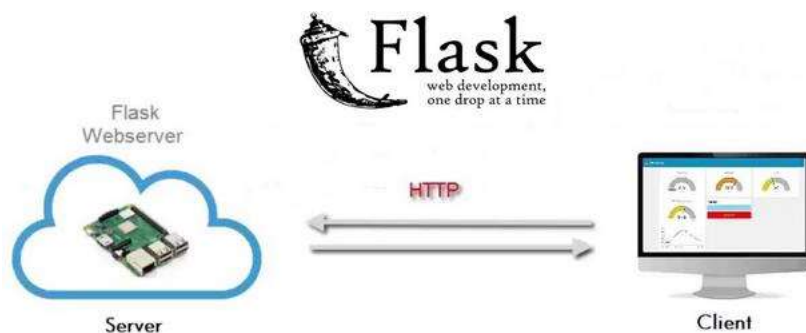
LinkedIn merupakan contoh website berbasisi Flask. Adapun beberapa keunggulan Flask yaitu,

- Lisensi BSD
- Memiliki server development dan debugger
- RESTful request dispatching
- Menggunakan template Jinja2
- Support secure cookies
- 100% WSGI 1.0 compliant
- Dokumentasi lengkap
- Kompatibel dengan Google App Engine

Flask adalah sebuah web framework yang ditulis dengan bahasa Python dan tergolong sebagai jenis microframework. Flask berfungsi sebagai kerangka kerja aplikasi dan tampilan dari suatu web. Dengan menggunakan Flask dan bahasa Python, pengembang dapat membuat sebuah web yang terstruktur dan dapat mengatur behaviour suatu web dengan lebih mudah.

Flask termasuk pada jenis microframework karena tidak memerlukan suatu alat atau pustaka tertentu dalam penggunaannya. Sebagian besar fungsi dan komponen umum seperti validasi form, database, dan sebagainya tidak terpasang secara default di Flask. Hal ini dikarenakan fungsi dan komponenkomponen tersebut sudah disediakan oleh pihak ketiga dan Flask dapat menggunakan ekstensi yang membuat fitur dan komponenkomponen tersebut seakan diimplementasikan oleh Flask sendiri.

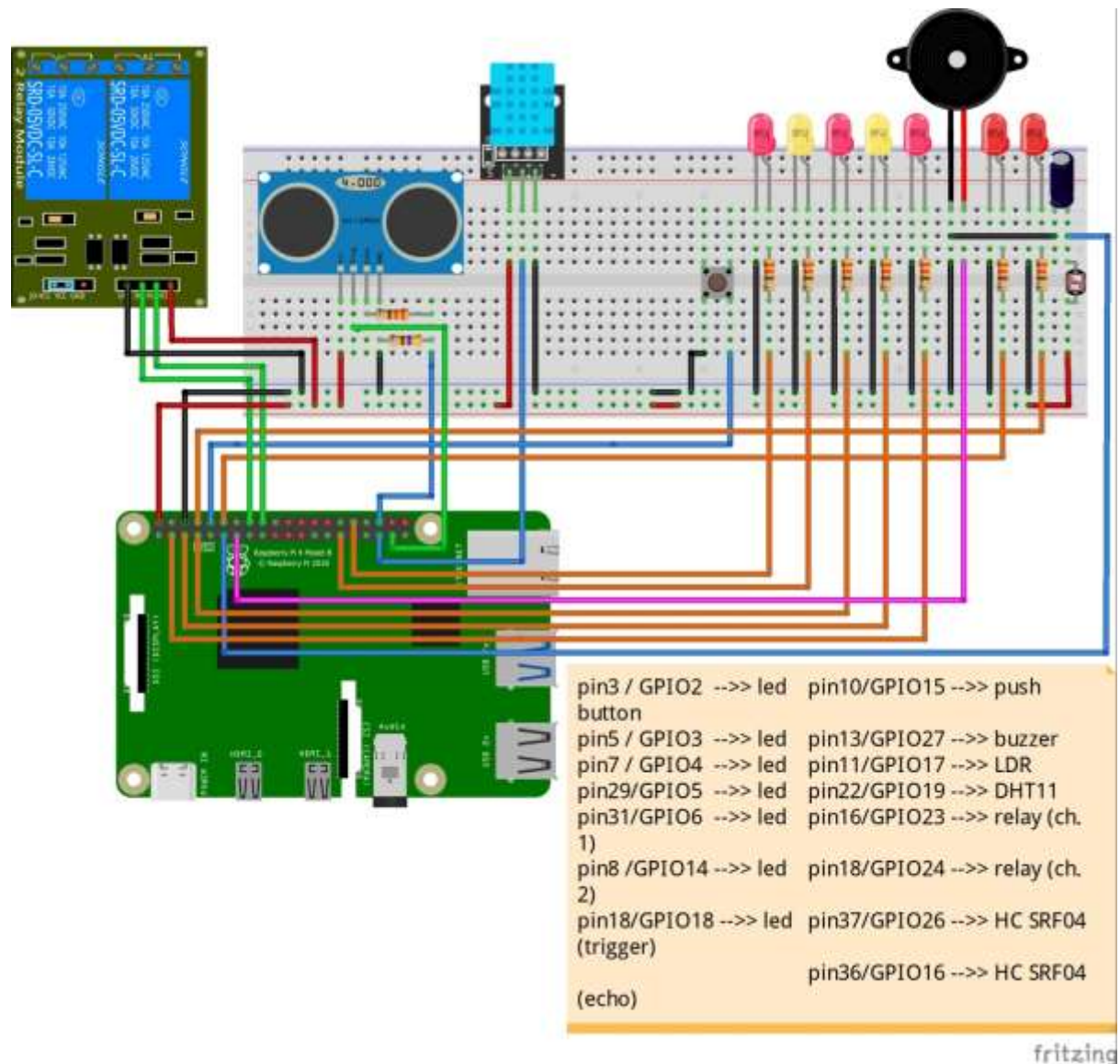
Selain itu, meskipun Flask disebut sebagai microframework, bukan berarti Flask mempunyai kekurangan dalam hal fungsionalitas. Microframework disini berarti bahwa Flask bermaksud untuk membuat core dari aplikasi ini sesederhana mungkin tapi tetap dapat dengan mudah ditambahkan. Dengan begitu, fleksibilitas serta skalabilitas dari Flask dapat dikatakan cukup tinggi dibandingkan dengan framework lainnya.



d. Alat dan Bahan :

Raspberry Pi 4	1x
Monitor (hdmi)	1x
Keyboard+mouse	1x
Power Supply 5V 3A type C	1x
Relay 5V 2Ch	1x
SRF-HC04	1x
DHT11	1x
LED 3mm	7x
Resistor 330Ohm	8x
Resistor 470Ohm	1x
Capacitor 10uF/16V	1x
Pushbutton	1x
LDR 3mm	1x

e. Prosedur Kerja :



f. Hasil dan Pembahasan :

A. Format Kartu micro-SD.

1. Kunjungi website SD Association dan download SD Formatter 4.0 (untuk Windows atau Mac) pada link berikut: <https://www.sdcard.org/downloads/>
2. Ikuti petunjuk untuk menginstal perangkat lunak.
3. Masukkan kartu micro-SD Anda ke komputer kemudian perhatikan huruf drive dimana kartu micro-SD dialokasikan untuk itu, misalnya F:/.
4. Pada SD Formatter, pilih huruf drive untuk kartu micro-SD dan lakukan format.

B. Instalasi Raspbian dengan NOOBS

1. NOOBS singkatan New Out of Box Software, dan jika Anda belum pernah bermainmain dengan GNU/Linux sebelumnya, maka disarankan Anda memulai dengan NOOBS.

2. Halaman Raspberry Pi Downloads. <https://www.raspberrypi.org/downloads/>
3. Klik pada NOOBS.
4. Klik pada tombol Download ZIP di bawah 'NOOBS (offline dan install via jaringan)', dan pilih folder untuk menyimpannya.
5. Ekstrak file dari ZIP.
6. Setelah kartu SD Anda telah diformat, copy semua file dalam folder NOOBS yang telah diekstrak ke drive kartu micro-SD.
7. File yang diperlukan kemudian akan ditransfer ke kartu SD Anda.
8. Ketika proses ini selesai, keluarkan kartu SD dengan aman (safely remove) dan masukkan ke Raspberry Pi.

C. Hubungkan Raspberry Pi 4 dengan Perangkat Penghubung

1. Hubungkan perangkat pendukung dengan Raspberry Pi 4

D. Pemrograman GPIO :

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setup(14, GPIO.OUT)
```

```
while (True):
    GPIO.output(14, True)
    time.sleep(1)
    GPIO.output(14, False)
    time.sleep(1)
```

```
import RPi.GPIO as GPIO
import time
pin = 14
GPIO.setmode(GPIO.BOARD)
GPIO.setup(pin, GPIO.OUT)
```

```
while(True):
    GPIO.output(pin, True)
    time.sleep(1)
    GPIO.output(pin, False)
    time.sleep(1)
```

```
import RPi.GPIO as GPIO
import time
GPIO.setwarnings(False)

input_button=15
GPIO.setmode(GPIO.BCM)
GPIO.setup(14, GPIO.OUT)
GPIO.setup(input_button, GPIO.IN, pull_up_down=GPIO.PUD_UP)

while True:
    GPIO.output(14, False)
    ab=GPIO.input(input_button)
    if ab==False:
        GPIO.output(14,True)
        print('pushbutton telah ditekan!!')
        time.sleep(0.2)
```

```
import RPi.GPIO as GPIO
import time
GPIO.setwarnings(False)

input_button=15
GPIO.setmode(GPIO.BCM)
GPIO.setup(14, GPIO.OUT)
GPIO.setup(input_button, GPIO.IN, pull_up_down=GPIO.PUD_UP)

status_button=False
input_lama=True

while True:
    status_baru=GPIO.input(input_button)
    if status_baru==False and input_lama==True:
        status_button=not status_button
        print('1')
        time.sleep(0.2)
    input_lama=status_baru
    GPIO.output(14, status_button)
```

```

import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)

ldr=17
tunda=2
def rc_time(ldr):
    count=0
    GPIO.setup(ldr, GPIO.OUT)
    GPIO.output(ldr, GPIO.LOW)
    time.sleep(tunda)
    GPIO.setup(ldr, GPIO.IN)
    while(GPIO.input(ldr)==GPIO.LOW):
        count+=1
    return count

try:
    while True:
        print(rc_time(ldr))
except KeyboardInterrupt:
    pass
finally:
    GPIO.cleanup()

```

```

import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)

ldr=17
led=14
tunda=2
nilai=0
GPIO.setup(led, GPIO.OUT)
GPIO.output(led, False)
def rc_time(ldr):
    count=0
    GPIO.setup(ldr, GPIO.OUT)
    GPIO.output(ldr, False)

```

```

time.sleep(tunda)
GPIO.setup(ldr, GPIO.IN)
while(GPIO.input(ldr)==GPIO.LOW):
    count+=1
return count

try:
    while True:
        print('nilai LDR : ')
        nilai=rc_time(ldr)
        print(nilai)
        if (nilai >= 2500):
            print('led menyala')
            GPIO.output(led, True)
        if (nilai <= 2500):
            print('led padam')
            GPIO.output(led, False)
except KeyboardInterrupt:
    pass
finally:
    GPIO.cleanup()

```

```

import RPi.GPIO as GPIO
import dht11

# initialize GPIO
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

instance =dht11.DHT11(pin=19)
result = instance.read()

if result.is_valid():
    print("Suhu: %-3.1f C" % result.temperature)
    print("Kelembaban: %-3.1f %" % result.humidity)
else:
    print("Error: %d" % result.error_code)
#GPIO.cleanup()

```

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

trigger=26
echo=16
GPIO.setup(trigger, GPIO.OUT)
GPIO.setup(echo, GPIO.IN)
GPIO.output(trigger, GPIO.LOW)

def get_range():
    GPIO.output(trigger, True)
    time.sleep(0.00001)
    GPIO.output(trigger, False)
    timeout_counter=int(time.time())
    start=time.time()

    while GPIO.input(echo) == 0 and (int(time.time())-timeout_counter) < 3:
        start=time.time()
    timeout_counter = int(time.time())
    stop=time.time()

    while GPIO.input(echo)== 1 and (int(time.time()) - timeout_counter) < 3:
        stop=time.time()

    elapsed = stop-start
    distance = elapsed * 34320
    distance = distance/2

    return distance

while True:
    jarak=get_range()
    print('jarak terukur : %.2f Cm' % jarak)
    time.sleep(1)
```

```

import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
button=15
ldr=17
led=18
led2=14
relay=23
relay2=24
tunda=2
nilai=0
GPIO.setup(led, GPIO.OUT)
GPIO.setup(relay, GPIO.OUT)
GPIO.output(led, False)
GPIO.output(relay, False)
GPIO.setup(led2,GPIO.OUT)
GPIO.setup(relay2, GPIO.OUT)
GPIO.output(led2, False)
GPIO.output(relay2, False)
GPIO.setup(button, GPIO.IN, pull_up_down=GPIO.PUD_UP)

status_button=False
input_lama=True

def rc_time(ldr):
    count=0
    GPIO.setup(ldr, GPIO.OUT)
    GPIO.output(ldr, False)
    time.sleep(tunda)
    GPIO.setup(ldr, GPIO.IN)
    while(GPIO.input(ldr)==GPIO.LOW):
        count+=1
    return count

try:
    while True:
        #LDR based

```

```

print('nilai LDR : ')
nilai=rc_time(ldr)
print(nilai)
if (nilai >= 2500):
    print('led dan saklar menyala')
    GPIO.output(led, True)
    GPIO.output(relay, False)
if (nilai <= 2500):
    print('led dan saklar padam')
    GPIO.output(led, False)
    GPIO.output(relay, True)
#Button based
status_baru=GPIO.input(button)
if status_baru==False and input_lama==True:
    status_button=not status_button
    print('1')
    time.sleep(0.2)
input_lama=status_baru
GPIO.output(led2, status_button)
status_button=not status_button
GPIO.output(relay2, status_button)
status_button=not status_button
except KeyboardInterrupt:
    pass
finally:
    GPIO.cleanup()

```

Instalasi Flask

Hal pertama yang harus dilakukan adalah menginstal Flask di Raspberry Pi 4, buka CLI terminal dan jalankan perintah:

```
sudo apt-get install python3-flask
```

Projek baru dimulai dengan membuat folder untuk menyimpan file, sebagai contoh:

```
mkdir rpiWebServer
```

Perintah di atas akan membuat folder bernama "rpiWebServer", dan tersimpan dalam sistem file python: /home/pi/rpiWebServer

Di folder ini, dibuat 2 sub-folder lain: **static** untuk CSS dan akhirnya file JavaScript dan **template** untuk File HTML. Buka folder baru rpiWebServer yang sudah dibuat:

```
cd rpiWebServer
```

Tambahkan buat 2 sub-folder baru:

```
mkdir static
```

dan,

```
mkdir templates
```

Struktur tree folder akan terlihat seperti ini:

```
/rpiWebServer
```

```
  /static
```

```
  /templates
```

WebServer python pertama kita dengan Flask

Buka IDE Python3 atau Thonny, Tuliskan kode "Hello Word" di bawah ini pada IDE Anda dan simpan sebagai contoh, sebagai helloWorld.py

```
from flask import Flask
```

```
app = Flask(__name__)
```

```
@app.route('/')
```

```
def index():
```

```
    return "Hello world <h1> ELJAVA MIFZAL HERBIYAN</h1>"
```

```
if __name__ == '__main__':
```

```
    app.run()
```

```
import RPi.GPIO as GPIO
```

```
from flask import Flask, render_template, request
```

```
app=Flask(__name__)
```

```
GPIO.setmode(GPIO.BCM)
```

```
GPIO.setwarnings(False)
```

```
led=14
```

```
buzzer=27
```

```
ldr=17
```

```
ledSts=0
```

```
buzzerSts=0
```

```
ldrSts=0
```

```
GPIO.setup(led, GPIO.OUT)
```

```

GPIO.setup(buzzer, GPIO.OUT)
#GPIO.setup(ldr, GPIO.IN)
GPIO.output(led, GPIO.LOW)
GPIO.output(buzzer, GPIO.LOW)
#GPIO.output(ldr, GPIO.LOW)

@app.route("/")
def rc_time(ldr):
    count=0
    GPIO.setup(ldr, GPIO.OUT)
    GPIO.output(ldr, False)
    time.sleep(tunda)
    GPIO.setup(ldr, GPIO.IN)
    while(GPIO.input(ldr)==GPIO.LOW):
        count+=1
    return count

@app.route("/")
def index():
    ledSts=GPIO.input(led)
    buzzerSts=GPIO.input(buzzer)

    templateData={'led': ledSts, 'buzzer': buzzerSts, 'ldr': ldrSts,}
    return render_template('index_1.html', **templateData)

@app.route("/<deviceName>/<action>")
def action(deviceName, action):
    if deviceName == 'led':
        actuator = led
    if deviceName == 'buzzer':
        actuator = buzzer
    if deviceName == 'ldr':
        actuator == ldr

    if action == "on":
        GPIO.output(actuator, GPIO.HIGH)
    if action == "off":
        GPIO.output(actuator, GPIO.LOW)

    ledSts=GPIO.input(led)
    buzzerSts=GPIO.input(buzzer)

```

```

ldrSts=GPIO.output(ldr)

templateData={'led': ledSts, 'buzzer': buzzerSts,'ldr': ldrSts,}
return render_template('index_1.html', **templateData)

if __name__=="__main__":
    app.run(debug=True, host='192.168.100.164')

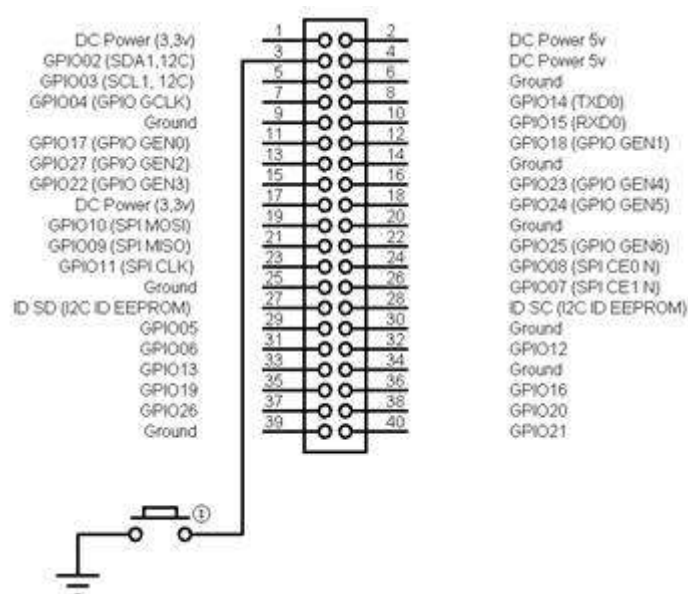
```

Antarmuka Sensor & Aktuator

Sensor dan aktuator merupakan dua perangkat yang tidak dapat dipisahkan dan banyak diterapkan pada berbagai proyek-proyek yang membutuhkan sistem otomatis atau automasi. Semua sensor dan aktuator memiliki fungsi yang sama yaitu mendeteksi atau menangkap sebuah sinyal dari alam untuk sensor dan merubah atau memindah sebuah objek untuk aktuator. Sehingga untuk memahami bagaimana cara sensor dan aktuator bekerja, kita tidak perlu membeli semua varian sensor dan aktuator yang tersedia di pasaran, namun cukup dengan satu sensor dan aktuator sederhana saja yang kemudian dapat dikembangkan sesuai dengan kebutuhan.

Antarmuka Push Button

Pada antarmuka *push button* ini akan membaca status dari sebuah push button yang terhubung dengan pin GPIO Raspberry dan menampilkan statusnya pada terminal. Push button dapat mewakili sebuah sensor digital karena sifatnya yang apabila ditekan *push button* akan mengeluarkan sinyal HIGH/LOW. Berikut gambar skematik dari antarmuka initialize



Gambar 1.2 Gambar Skematik Antarmuka Push Button

Dari skematik diatas, bahan yang diperlukan adalah:

- | | |
|--|----|
| 1. Raspberry Pi 2 single board circuit | 1x |
| 2. Kabel UTP (<i>Cross Connection</i>) | 1x |
| 3. PC Windows based | 1x |
| 4. Power Supply 2A | 1x |
| 5. Push Button | 1x |

Adapun program untuk membaaca status push button lalu menampilkannya pada terminal adalah sebagai berikut

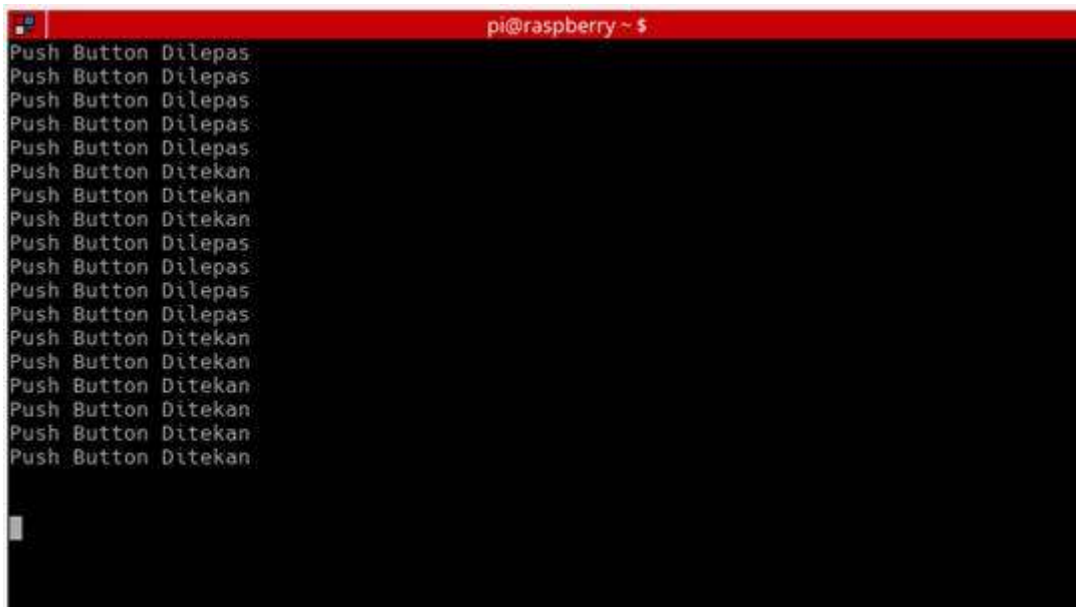
Program Antarmuka Push Button

```
# -----  
# Program Antarmuka GPIO - Push Button  
# File: gpio_btn.py  
# -----  
  
from RPi import GPIO  
from time import sleep  
GPIO.setmode(GPIO.BOARD)  
button=3  
GPIO.setup(button, GPIO.IN, pull_up_down=GPIO.PUD_UP)  
GPIO.setup(LED, GPIO.OUT)  
while(1):  
    if GPIO.input(button)==0:  
        print"Push Button Dilepas"  
    else GPIO.input(button)==1:  
        print"Push Button Ditekan"
```

Simpan program diatas dengan nama file *gpio_btn.py* lalu jalankan dengan mengetikan perintah berikut pada terminal

```
pi@raspberrypi ~ $ #Menjalankan program gpio_btn.py  
pi@raspberrypi ~ $ sudo python gpio_btn.py
```

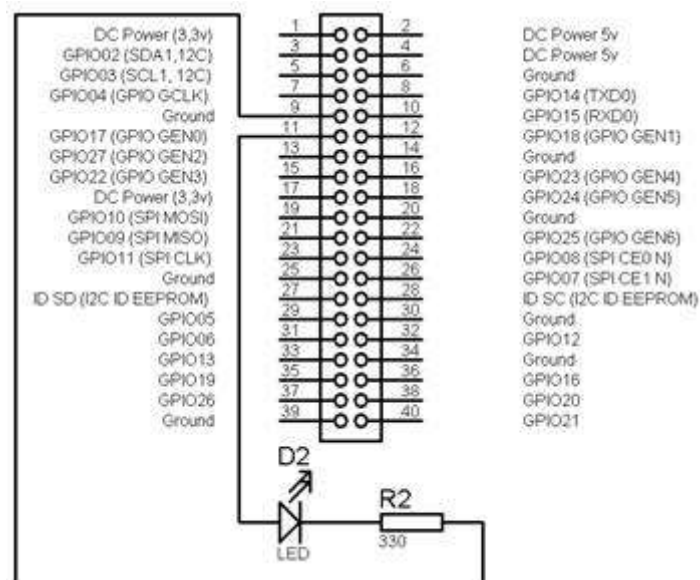
Setelah program diatas dijalankan maka kondisi dari push button yang terhubung dengan pin GPIO akan ditampilkan setiap 0.5 detik sebagaimana gambar berikut



Gambar 1.3 Tampilan terminal Antarmuka GPIO Push Button

Antarmuka LED Sederhana

LED merupakan sebuah indikator yang biasa digunakan untuk memberikan sebuah tanda kepada pengguna. LED Merupakan salah satu contoh aktuator sederhana karena sifatnya yang dapat merubah tegangan menjadi besaran alam (cahaya). Pada subbab ini akan dibahas bagaimana cara menghidupkan dan mematikan LED dengan PIN GPIO yang terdapat pada raspberry pi. Berikut gambar skematiknya



Gambar 1.4 Antarmuka LED Sederhana

Langkah-langkah:

- Buka terminal, ketikkan command “sudo idle” pada terminal
- Pilih File >> New Window.
- Ketik list program berikut.


```
# -----
# Program Antarmuka GPIO - Simple LED
# File: gpio_LED.py
# -----

import RPi.GPIO as GPIO      //pemanggilan library GPIO
import time                  //pemanggilan library delay pada raspberry
GPIO.setmode(GPIO.BOARD)
GPIO.setup(11, GPIO.OUT)     //mengkonfigurasi GPIO pada raspberry
                              sebagai output
while True:
    GPIO.output(11, True)
    sleep(1)
    GPIO.output(11, False)
    Sleep(1)
```

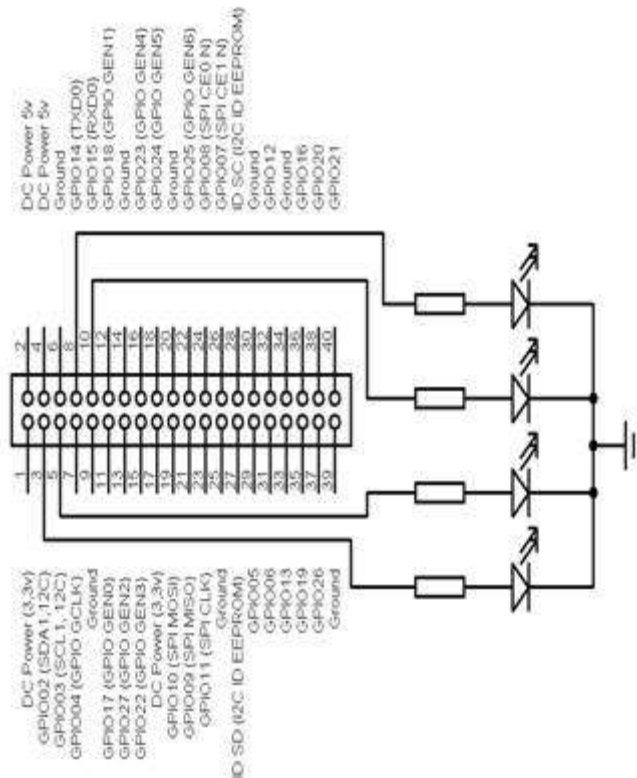
Simpan program diatas dengan nama file *gpio_led.py* lalu jalankan dengan mengetikan perintah berikut pada terminal

```
pi@raspberrypi ~ $ #Menjalankan program gpio_led.py
pi@raspberrypi ~ $ sudo python gpio_led.py
```

Setelah program diatas dijalankan maka LED yang terhubung dengan pin GPIO 11 akan menyala dan hidup dengan interval masing-masing satu detik.

Antarmuka LED Berjalan

Antarmuka ini merupakan pengembangan dari antarmuka sebelumnya yaitu mengendalikan beberapa LED yang terhubung dengan pin GPIO agar dapat mati dan hidup secara bergantian. Berikut gambar skematik dari antarmuka ini



Gambar 1.5 Skematik Antarmuka LED Berjalan

Adapun bahan yang dibutuhkan berdasarkan skematik diatas adalah:

- | | |
|---|----|
| • Raspberry Pi 2 single board circuit | 1x |
| • Kabel UTP (<i>Cross Connection</i>) | 1x |
| • PC Windows based | 1x |
| • Power Supply 2A | 1x |
| • LED | 4x |
| • Resistor 330Ω | 4x |

Langkah-langkah:

- Buka terminal, ketikan command “sudo idle” pada terminal
- Pilih File >> New Window.
- Ketik list program berikut.

```
import RPi.GPIO as GPIO          //pemanggilan library GPIO
import time as sleep             //pemanggilan library delay pada raspberry
GPIO.setmode(GPIO.BOARD)
GPIO.setup(3, GPIO.OUT)          //mengkonfigurasi GPIO pada raspberry sebagai
output
GPIO.setup(5, GPIO.OUT)
GPIO.setup(8, GPIO.OUT)
GPIO.setup(10, GPIO.OUT)
while True:
    GPIO.output(3, True)
```

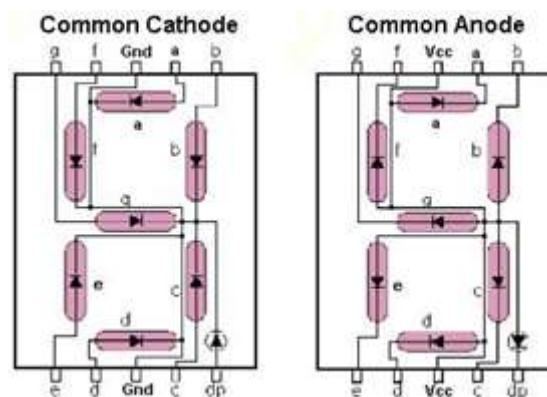
```

sleep(1)
GPIO.output(3, False)
GPIO.output(5, True)
sleep(1)
GPIO.output(5, False)
GPIO.output(8, True)
sleep(1)
GPIO.output(8, False)
GPIO.output(10, True)
sleep(1)
GPIO.output(10, False)

```

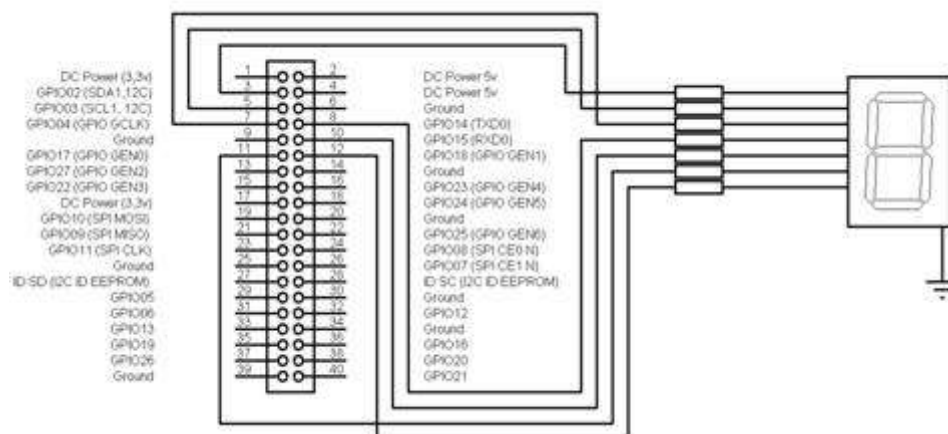
Antarmuka Seven Segment

Seven segment sebenarnya sama halnya dengan LED karena seven segment tersusun dari 7 LED. Akan tetapi LED pada seven segment disusun sedemikian rupa sehingga jika dihidupkan dengan komposisi tertentu akan melambangkan sebuah angka dari 0 – 9. Untuk lebih memahaminya berikut gambar dari seven segment yang akan kita gunakan



Gambar 1.6 Struktur Seven Segment

Setelah mengetahui struktur susunan LED pada seven segment, kemudian hubungkan seven segment dengan raspberry berdasarkan skematik berikut.



Gambar 1.7 Skematik Antarmuka Seven Segment

Dari gambar skematik diatas dapat diketahui bahan yang diperlukan adalah sebagai berikut

- | | |
|--|----|
| 1. Raspberry Pi 2 single board circuit | 1x |
| 2. Kabel UTP (<i>Cross Connection</i>) | 1x |
| 3. PC Windows based | 1x |
| 4. Power Supply 2A | 1x |
| 5. Push Button | 1x |
| 6. Resistor 330 Ω | 1x |
| 7. Seven segment | 1x |

Setelah selesai dengan rangkaian diatas tahap selanjutnya adalah menuliskan program. Berikut program *python* pada raspberry untuk menampilkan angka 0 – 9 secara bergantian dengan interval satu detik

```
# -----  
# Program Antarmuka GPIO - Seven Segment  
# File: gpio_7seg.py  
# -----  
  
from RPi import GPIO  
from time import sleep  
GPIO.setmode(GPIO.BOARD)  
  
ledA=3  
ledB=5  
ledC=7  
ledD=8  
ledE=10  
ledF=11  
ledG=12  
  
GPIO.setup(ledA, GPIO.OUT)  
GPIO.setup(ledB, GPIO.OUT)  
GPIO.setup(ledC, GPIO.OUT)  
GPIO.setup(ledD, GPIO.OUT)  
GPIO.setup(ledE, GPIO.OUT)  
GPIO.setup(ledF, GPIO.OUT)  
GPIO.setup(ledG, GPIO.OUT)  
GPIO.setup(ledDOT, GPIO.OUT)  
  
  
def satu():  
  
    GPIO.output(ledF, True)  
    GPIO.output(ledE, True)
```

```
def dua() :
```

```
    GPIO.output(ledA, True)
    GPIO.output(ledB, True)
    GPIO.output(ledG, True)
    GPIO.output(ledE, True)
    GPIO.output(ledD, True)
```

```
def tiga() :
```

```
    GPIO.output(ledA, True)
    GPIO.output(ledF, True)
    GPIO.output(ledG, True)
    GPIO.output(ledE, True)
    GPIO.output(ledD, True)
```

```
def empat() :
```

```
    GPIO.output(ledB, True)
    GPIO.output(ledF, True)
    GPIO.output(ledG, True)
    GPIO.output(ledC, True)
```

```
def lima() :
```

```
    GPIO.output(ledA, True)
    GPIO.output(ledF, True)
    GPIO.output(ledG, True)
    GPIO.output(ledC, True)
    GPIO.output(ledD, True)
```

```
def enam() :
```

```
    GPIO.output(ledA, True)
    GPIO.output(ledF, True)
    GPIO.output(ledG, True)
    GPIO.output(ledC, True)
    GPIO.output(ledD, True)

    GPIO.output(ledE, True)
```

```

def tujuh():

    GPIO.output(ledA, True)
    GPIO.output(ledB, True)
    GPIO.output(ledC, True)

def delapan():

    GPIO.output(ledA, True)
    GPIO.output(ledB, True)
    GPIO.output(ledC, True)
    GPIO.output(ledD, True)
    GPIO.output(ledE, True)
    GPIO.output(ledF, True)
    GPIO.output(ledG, True)

while(1):
    satu()
    sleep(1)
    dua()
    sleep(1)
    tiga()
    sleep(1)
    empat()
    sleep(1)
    lima()
    sleep(1)
    enam()
    sleep(1)
    tujuh()
    sleep(1)
    delapan()
    sleep(1)

```

Simpan program diatas dengan nama file *gpio_pwm.py* lalu jalankan dengan mengetikan perintah berikut pada terminal

```

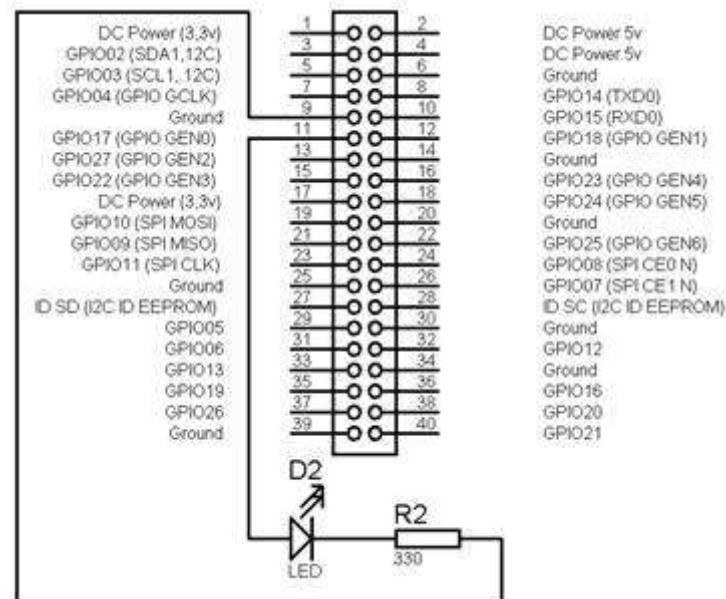
pi@raspberrypi ~ $ #Menjalankan program gpio_7seg.py
pi@raspberrypi ~ $ sudo python gpio_7seg.py

```

Setelah program diatas dijalankan maka seven segment akan menampilkan angka 1 – 8 secara bergantian dengan interval 1 detik.

Antarmuka LED Dimmer dengan PWM

Untuk mengendalikan tingkat kecerahan atau intensitas cahaya yang dikeluarkan oleh LED, kita dapat menggunakan pin GPIO dengan mode PWM. PIN GPIO yang telah diatur dengan mode output ini kemudian dihubungkan dengan LED sehingga dapat diatur tingkat intensitas cahayanya. Berikut gambar skematik dari antarmuka ini



Gambar 1.8 Skematik Antarmuka LED Dimmer

```
# -----
# Program Antarmuka GPIO - LED Dimmer
# File: gpio_pwm.py
#-----

from RPi.GPIO import GPIO          //pemanggilan library GPIO
from time import sleep             //pemanggilan library delay pada raspberry
GPIO.setmode(GPIO.BOARD)

GPIO.setup(11, GPIO.OUT)           //konfigurasi GPIO pada raspberry sebagai output

merah = GPIO.PWM(11, 100) //inisialisasi "merah" untuk PWM port 7 dengan frekuensi 100Hz
merah.start(0)                    //dimulai dari 0% dutycycle = off
while True:
    for i in range(0,101):         //looping dilakukan dari 1% hingga 100%
        merah.ChangeDutyCycle(i)
```

```

sleep(0.05)
for i in range(100,-1,-1)    //looping dilakukan dari 99% hingga 0%
    merah.ChangeDutyCycle(i)
sleep(0.05)

```

Simpan program diatas dengan nama file *gpio_pwm.py* lalu jalankan dengan mengetikan perintah berikut pada terminal

```

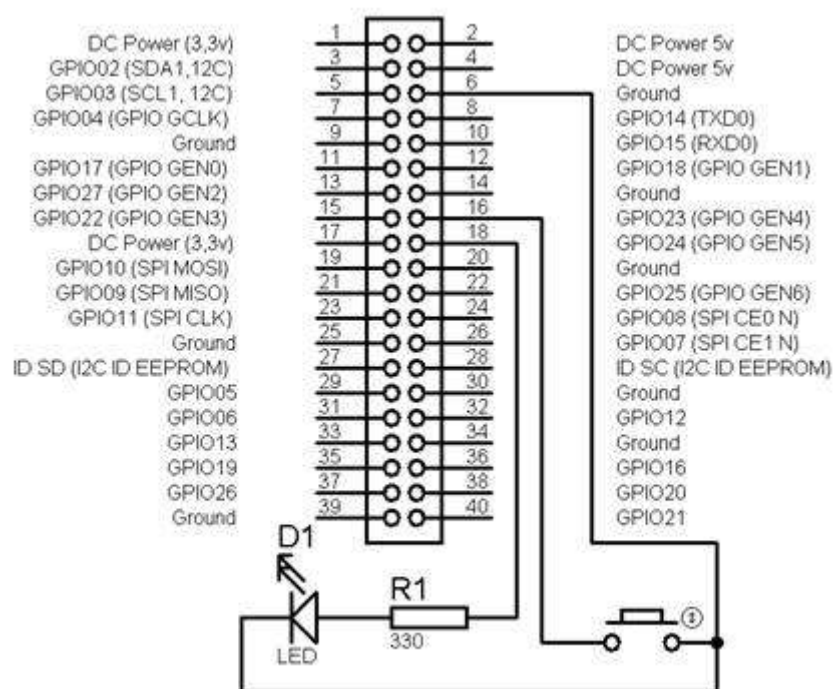
pi@raspberrypi ~ $ #Menjalankan program gpio_pwm.py
pi@raspberrypi ~ $ sudo python gpio_pwm.py

```

Setelah program diatas dijalankan maka data dari sensor LED akan menyala dari terang ke redup dan sebaliknya.

Antarmuka Push Button dan LED

Sensor yang akan digunakan pada bab ini berupa *push button* yang akan mengeluarkan sinyal HIGH/LOW ketika ditekan atau dilepas. Kondisi dari push button ini akan digunakan untuk mengendalikan sebuah LED yang sama-sama terhubung dengan pin GPIO raspberry serta menampilkan pesan “LED ON / OFF” pada terminal. Berikut skematik dari antarmuka ini



Gambar 1.9 Skematik Antarmuka Sensor dan Aktuator

Setelah selesai dengan rangkaian diatas tahap selanjutnya adalah menuliskan program. Berikut program *python* pada raspberry untuk mengendalikan aktutor berupa LED dengan sensor sederhana berupa *push button*.

```

# -----
# Program Antarmuka GPIO - Push Button & LED
# File: gpio_btn_led.py

```



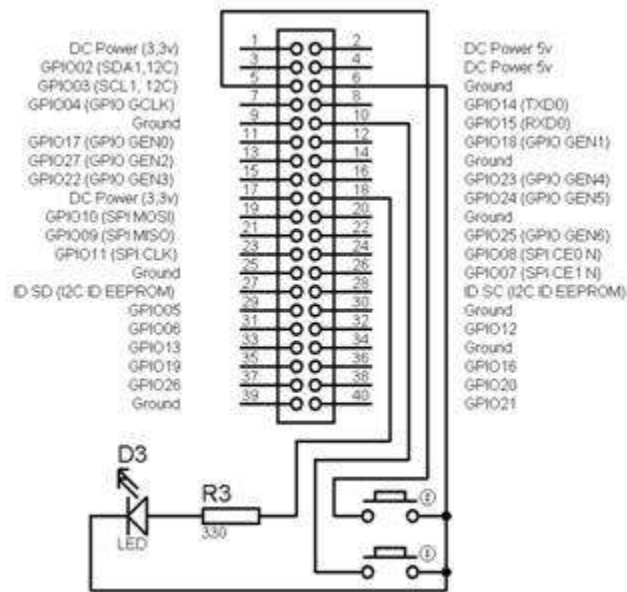
```
# -----

from RPi import GPIO
from time import sleep
GPIO.setmode(GPIO.BOARD)
button=16
LED=18
GPIO.setup(button, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(LED, GPIO.OUT)
BS1=False
while(1):
    if GPIO.input(button)==0:
        print"LED ON"
        if BS1==False:
            GPIO.output(LED, True)
            BS1=True
            sleep(0.5)
        else:
            print"LED OFF"
            GPIO.output(LED, False)
            BS1=False
            Sleep(0.5)
```

Simpan program diatas dengan nama file *gpio_btn_led.py* lalu jalankan dengan mengetikan perintah berikut pada terminal

```
pi@raspberrypi ~ $ #Menjalankan program gpio_btn_led.py
pi@raspberrypi ~ $ sudo python gpio_btn_led.py
```

Setelah program diatas dijalankan maka data dari sensor LM35 akan ditampilkan setiap 0.5 detik sebagaimana gambar berikut



Gambar 1.11 Skematik Antarmuka Pengatur LED Dimmer

Dari gambar skematik diatas, maka dapat diketahui bahan yang dibutuhkan adalah

- | | |
|--|----|
| 1. Raspberry Pi 2 single board circuit | 1x |
| 2. Kabel UTP (<i>Cross Connection</i>) | 1x |
| 3. PC Windows based | 1x |
| 4. Power Supply 2A | 1x |
| 5. Push Button | 1x |
| 6. Resistor 330Ω | 1x |
| 7. LED | 1x |

Setelah selesai dengan rangkaian diatas, maka tahap selanjutnya adalah

```
# -----
# Program Antarmuka GPIO - Pengatur LED Dimmer
# File: gpio_btn_pwm.py
# -----

from RPi.GPIO import GPIO          //pemanggilan library GPIO
from time import sleep             //pemanggilan library delay pada raspberry
GPIO.setmode(GPIO.BOARD)

button1=5                          //inisialisasi PIN GPIO
button2=10

LED=18

GPIO.setup(button1,GPIO.IN,pull_up_down=GPIO.PUD_UP)
GPIO.setup(button2,GPIO.IN,pull_up_down=GPIO.PUD_UP)
GPIO.setup(LED1,GPIO.OUT)

pwm1=GPIO.PWM(LED1,500)

pwm1.start(0)

bright=0                          // set nilai kecerahan 1%
while(1):                        //perulangan terus menerus
```

```

        if GPIO.input(button1)==0:
            print "Button 1 ditekan"
            bright=bright-10                //set nilai kecerahan dibagi 2
            pwm1.ChangeDutyCycle(bright)
            sleep(0.25)
            print "Nilai kecerahan: ",bright
        if GPIO.input(button2)==0:
            print "Button 2 ditekan"
            bright=bright+10                //nilai kecerahan dikali 2
            if bright>100:                  //kondisi ketika kecerahan lebih dari
100
                bright=100
                print "Keccerahan maksimal"
            pwm1.ChangeDutyCycle(bright)
            sleep(.25)
            print "Nilai Keccerahan: ",bright

```

Simpan program diatas dengan nama file *gpio_btn_led.py* lalu jalankan dengan mengetikan perintah berikut pada terminal

```

pi@raspberrypi ~ $ #Menjalankan program gpio_btn_pwm.py
pi@raspberrypi ~ $ sudo python gpio_btn_pwm.py

```

Setelah program diatas dijalankan, tekan tombol 1 yang terhubung dengan pin GPIO 5 untuk mengurangi intensitas cahaya dengan menurunkan nilai PWM. Begitu juga sebaliknya, tekan tombol 2 yang terhubung dengan pin GPIO 10 untuk meningkatkan intensitas cahaya LED dengan menaikkan nilai PWM. Berikut hasil monitoring dari terminal setelah program dijalankan

```

pi@raspberrypi ~ $
Button 2 ditekan
Nilai kecerahan 10
Button 2 ditekan
Nilai kecerahan 20
Button 2 ditekan
Nilai kecerahan 30
Button 2 ditekan
Nilai kecerahan 40
Button 2 ditekan
Nilai kecerahan 50
Button 1 ditekan
Nilai kecerahan 40
Button 1 ditekan
Nilai kecerahan 30
Button 1 ditekan
Nilai kecerahan 20
Button 1 ditekan
Nilai kecerahan 10
Button 1 ditekan
Nilai kecerahan 0

```

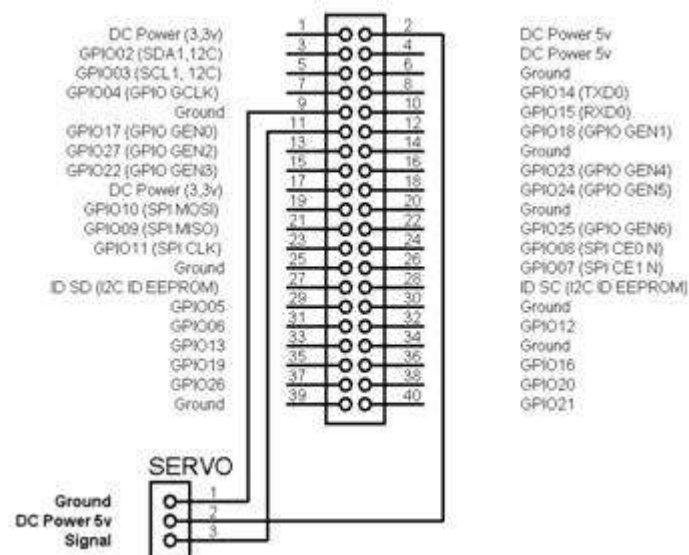
Gambar 1.12 Monitoring PWM melalui Terminal

Pada hasil monitoring melalui terminal diatas dapat dilihat bahwa ketika tombol 2 ditekan maka nilai kecerahan yang dalam hal ini sebenarnya adalah PWM bertambah 10. Begitu juga sebaliknya, apabila tombol 1 ditekan maka nilai kecerahan akan berkurang 10. Sehingga tingkat intensitas cahaya yang dikeluarkan LED mengikuti nilai kecerahan.

Antarmuka Motor Servo

Servo merupakan sebuah perangkat yang terdiri dari motor DC, gear transmisi, potensiometer, dan sebuah IC kontroller. Berbeda dengan motor DC yang terus menerus berputar jika dialiri arus, servo hanya bergerak berdasarkan sudut yang diinginkan dan akan berhenti setelah sudut tujuan tercapai serta mempertahankan sudut tujuan. Hal ini terjadi, karena terdapat umpan balik dari potensiometer yang terhubung dengan motor DC secara mekanik. Umpan balik dari potensiometer inilah yang dijadikan sebagai sistem kontrol oleh IC kontroller pada servo.

Pada antarmuka ini akan dibahas bagaimana cara menggerakkan servo pada sudut-sudut tertentu yang dikendalikan oleh raspberry melalui pin GPIO. Berikut gambar skematiknya



Gambar 1.13 Skematik Antarmuka Servo

Dari gambar skematik diatas, dapat diketahui bahwa bahan yang dibutuhkan adalah

- | | |
|--|----|
| 1. Raspberry Pi 2 single board circuit | 1x |
| 2. Kabel UTP (<i>Cross Connection</i>) | 1x |
| 3. PC Windows based | 1x |
| 4. Power Supply 2A | 1x |
| 5. Motor servo | 1x |

Setelah selesai menyusun rangkain sesuai dengan skematik diatas, selanjutnya adalah membuat program untuk menggerakkan servo ke sudut 0 derajat, 90 derajat, dan 180 derajat secara berurutan. Berikut program yang harus dituliskan pada editor anda

```
# -----
```

```
# Program Antarmuka GPIO - Motor Servo
# File: gpio_servo.py
# -----
from RPi import GPIO
from time import sleep
GPIO.setmode(GPIO.BOARD)
GPIO.setup(11, GPIO.OUT)
pwm=GPIO.PWM(11,50)
pwm.start(5)
while True:
    pwm.ChangeDutyCycle(2.5) #zero degrees
    sleep(1)
    pwm.ChangeDutyCycle(6.2) #90 degrees
    sleep(1)
pwm.ChangeDutyCycle(11)
    sleep(1)
```

Simpan program diatas dengan nama file *gpio_btn_led.py* lalu jalankan dengan mengetikan perintah berikut pada terminal

```
pi@raspberrypi ~ $ #Menjalankan program gpio_servo.py
pi@raspberrypi ~ $ sudo python gpio_servo.py
```

Setelah program diatas dijalankan maka servo akan bergerak sesuai dengan yang diinginkan

Antarmuka Sensor

Pada subbab ini akan dibahas mengenai antarmuka raspberry dengan arduino untuk mengakses data sensor yang terhubung dengan arduino. Sensor adalah sebuah perangkat yang sering digunakan untuk mendeteksi atau menangkap sinyal listrik atau cahaya. Sensor merubah besaran fisik (contoh: temperatur, tekanan darah, kelembapan, kecepatan, dan lain-lain) menjadi sinyal yang dapat diukur secara elektrik. Adapun sensor yang akan dibahas adalah sensor-sensor yang termasuk dalam kategori sensor analog dan digital

Antarmuka Sensor Analog

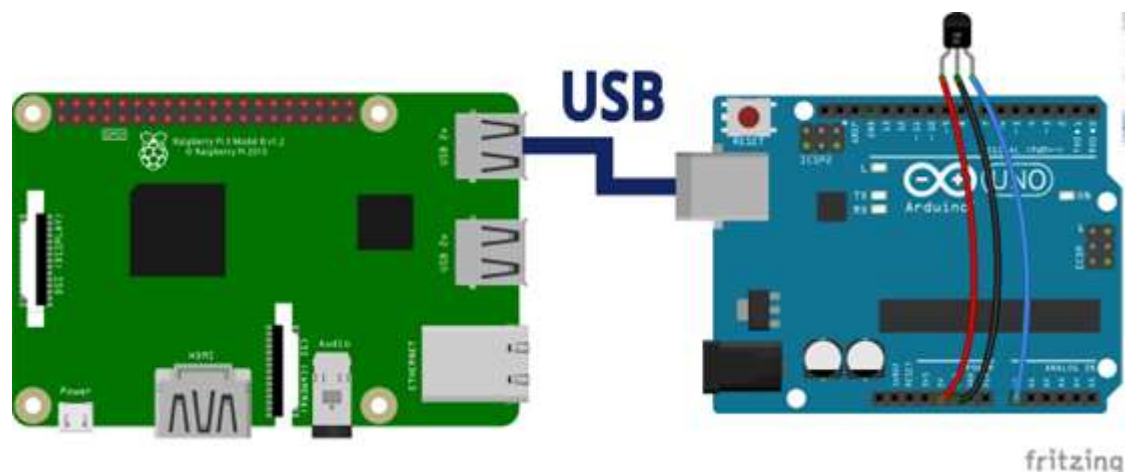
Sensor analog merupakan sensor yang secara terus menerus mengeluarkan sinyal analog. Sinyal analog tidak dapat dibaca langsung oleh arduino sehingga harus diubah menjadi digital terlebih dahulu dengan komponen *Analog Digital Converter (ADC)*. *Analog to Digital Converter* atau dapat disingkat dengan ADC merupakan sebuah metode untuk merubah sinyal analog ke sinyal digital. Sinyal analog perlu diubah menjadi digital dikarenakan semua mikrokontroller termasuk ATmega yang digunakan pada arduino hanya dapat mengolah

sinyal digital. Sehingga untuk dapat membaca sinyal analog, data keluaran dari sensor analog harus dihubungkan dengan PIN analog arduino.

Antarmuka Thermostat (LM35)

LM35 adalah sebuah IC pembaca temperatur dengan output tegangan yang linear dengan satuan derajat celcius. Sensor ini memiliki tingkat akurasi $\pm 1/4^{\circ}\text{C}$ pada suhu ruangan dan $\pm 3/4^{\circ}\text{C}$ pada suhu yang melebihi rentang $-55^{\circ}\text{C} - 150^{\circ}\text{C}$. Sensor LM35 dapat digunakan dengan satu *power supply* atau bahkan tanpa *power supply*, hal ini karena sensor LM35 hanya membutuhkan arus sebesar $60\ \mu\text{A}$. Pada saat beroperasi, sensor ini menghasilkan panas yang sangat kecil yaitu kurang dari 0.1°C .

Pada antarmuka thermostat ini, sensor akan terhubung dengan arduino pada pin analog A0. Data dari sensor LM35 akan dibaca oleh raspberry menggunakan protokol firmata, sehingga sebelumnya harus sudah diupload program *Standard Firmata* pada arduino. Arduino terhubung dengan raspberry dengan USB Serial melalui kabel downloader bawaan arduino. Berikut gambar skematik dari antarmuka thermostat pada arduino



Gambar x Antarmuka Thermostat Setelah selesai dengan rangkaian diatas tahap selanjutnya adalah menuliskan program. Berikut program *python* pada raspberry untuk mengakses data pada arduino dengan *library firmata* dan menampilkannya pada terminal.

```
# -----  
# Program Antarmuka LM35  
# File: lm35.py  
# -----  
  
import pyfirmata  
from time import sleep  
  
#PORT USB Arduino  
PORT = '/dev/ttyACM0'
```

```

BOARD = pyfirmata.Arduino(PORT)

#PIN Analog A0 untuk LM35
ITER = pyfirmata.util.Iterator(BOARD)
ITER.start()
LM35_PIN = BOARD.get_pin('a:0:1')

while True:
    dataLM35 = LM35_PIN.read()
    print "Data Thermostat ", dataLM35, " Celcius"
    sleep(0.5)

```

*Catatan: Variabel PORT merupakan port usb arduino yang terdeteksi oleh raspberry yang pada umumnya adalah **ttyACM0** atau **ttyUSB0**. Untuk mengetahui pada device mana arduino terhubung, kita dapat menggunakan perintah **dmesg**.*

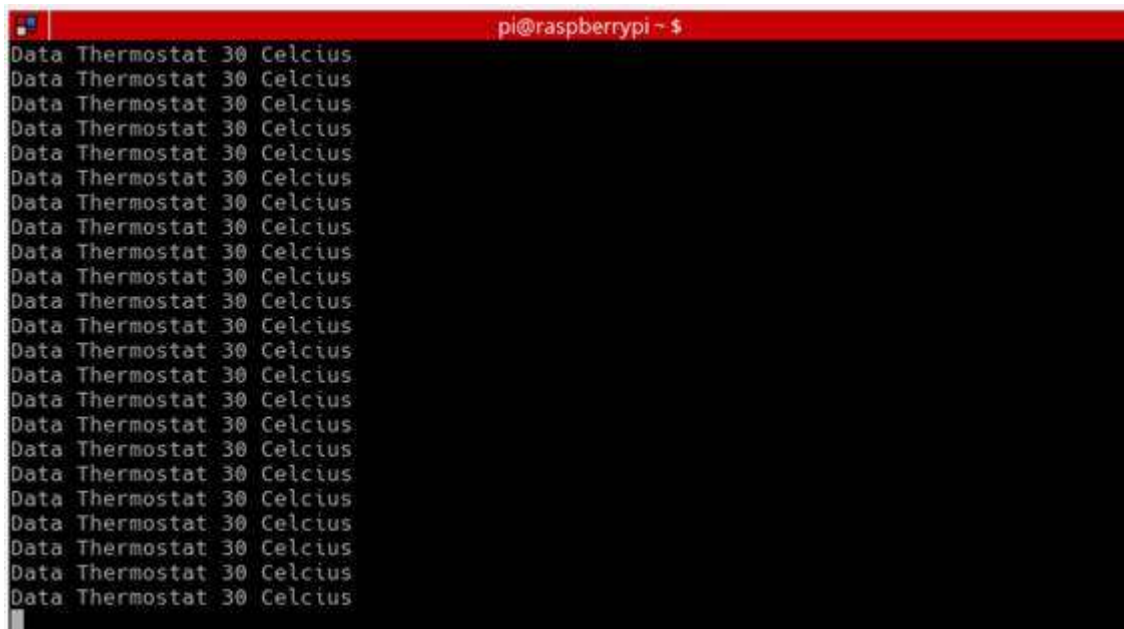
Simpan program diatas dengan nama file *lm35.py* lalu jalankan dengan mengetikan perintah berikut pada terminal

```

pi@raspberrypi ~ $ #Menjalankan program lm35.py
pi@raspberrypi ~ $ sudo python lm35.py

```

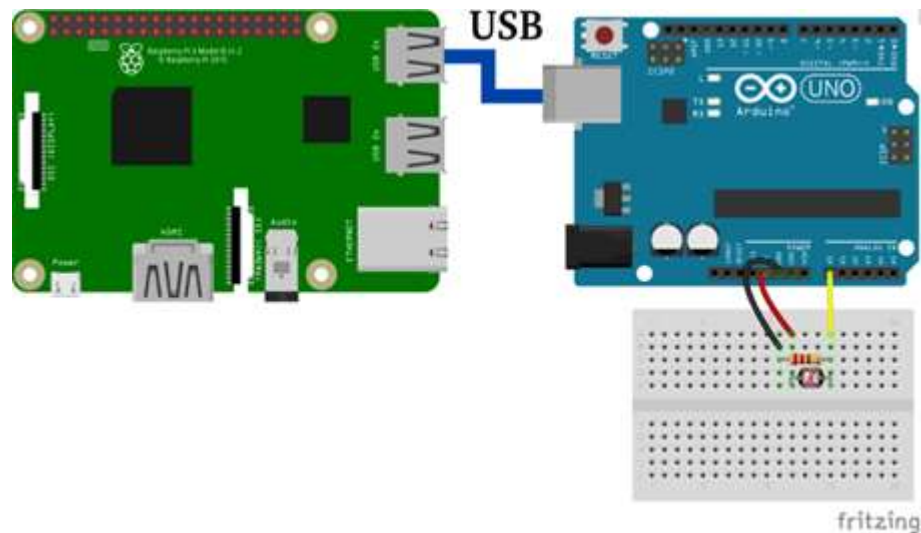
Setelah program diatas dijalankan maka data dari sensor LM35 akan ditampilkan setiap 0.5 detik sebagaimana gambar berikut



Gambar 2.1 Tampilan Data LM35 pada Terminal

Antarmuka Light Depend Resistance (LDR)

Sensor LDR terhubung dengan arduino pada pin analog A0 sementara raspberry terhubung dengan arduino melalui kabel USB. Data sensor akan dibaca oleh raspberry dengan memanfaatkan protokol firmata sebagaimana yang telah dibahas sebelumnya. Berikut gambar skematiknya



Gambar 2.2 Skematik Antarmuka LDR

Pada skematik diatas terdapat tambahan resistor yang berfungsi sebagai *pull-down* rangkaian. Setelah selesai menyusun rangkaian seperti diatas, kemudian tulis program dibawah ini untuk membaca data sensor LDR pada arduino.

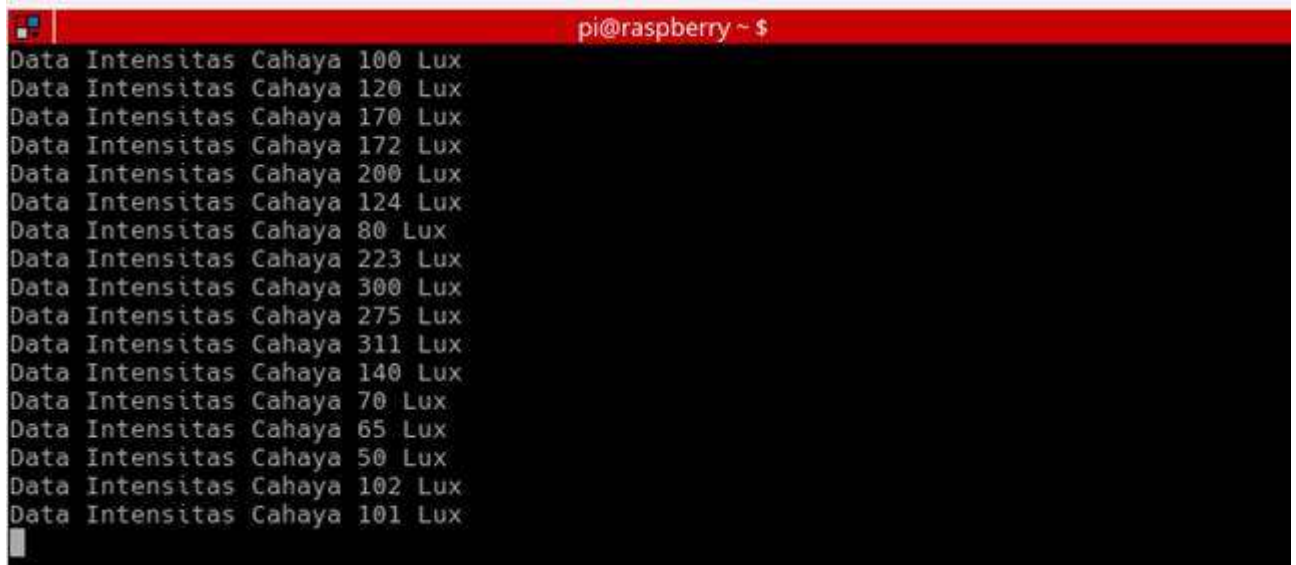
```
# -----  
# Program Antarmuka LDR  
# File: ldr.py  
# -----  
  
import pyfirmata  
from time import sleep  
  
#PORT USB Arduino  
PORT = '/dev/ttyACM0'  
BOARD = pyfirmata.Arduino(PORT)  
  
#PIN Analog A0 untuk LM35  
ITER = pyfirmata.util.Iterator(BOARD)  
ITER.start()  
LDR_PIN = BOARD.get_pin('a:0:1')  
  
while True:  
    dataLDR = LDR_PIN.read()  
    print "Data Intensitas Cahaya ", dataLDR, " Lux"
```

```
sleep(0.5)
```

Simpan program diatas dengan nama file *ldr.py* lalu jalankan dengan mengetikan perintah berikut pada terminal

```
pi@raspberrypi ~ $ #Menjalankan program lm35.py  
pi@raspberrypi ~ $ sudo python lm35.py
```

Setelah program diatas dijalankan maka data dari sensor LDR akan ditampilkan setiap 0.5 detik sebagaimana gambar berikut



```
pi@raspberrypi ~ $  
Data Intensitas Cahaya 100 Lux  
Data Intensitas Cahaya 120 Lux  
Data Intensitas Cahaya 170 Lux  
Data Intensitas Cahaya 172 Lux  
Data Intensitas Cahaya 200 Lux  
Data Intensitas Cahaya 124 Lux  
Data Intensitas Cahaya 80 Lux  
Data Intensitas Cahaya 223 Lux  
Data Intensitas Cahaya 300 Lux  
Data Intensitas Cahaya 275 Lux  
Data Intensitas Cahaya 311 Lux  
Data Intensitas Cahaya 140 Lux  
Data Intensitas Cahaya 70 Lux  
Data Intensitas Cahaya 65 Lux  
Data Intensitas Cahaya 50 Lux  
Data Intensitas Cahaya 102 Lux  
Data Intensitas Cahaya 101 Lux
```

Gambar 2.3 Tampilan data LDR pada terminal

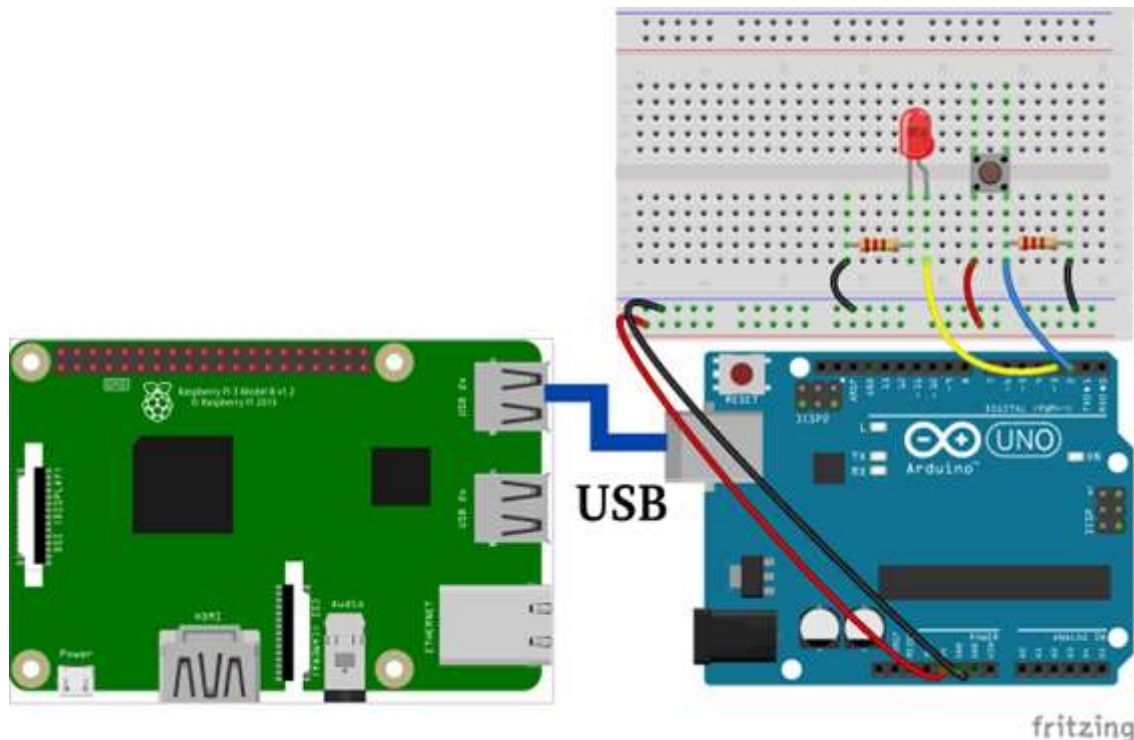
Antarmuka Sensor Digital

Sensor digital merupakan sebuah sensor yang memiliki keluaran tegangan HIGH atau LOW (sinyal digital). Hal ini tentunya berbeda dengan sensor analog yang memiliki keluaran berupa sinyal analog secara terus menerus. Dengan demikian sensor digital dapat dibaca langsung oleh arduino tanpa proses ADC. Semua sensor digital dapat dihubungkan pada PIN analog maupun digital arduino.

Antarmuka Push Button

Push button merupakan sakelar sederhana yang berfungsi untuk mengendalikan sebuah sistem atau proses tertentu. Tombol pada *push button* biasanya terbuat dari bahan keras seperti logam atau besi yang permukaannya disesuaikan dengan bentuk jari atau tangan manusia. Push button termasuk kategori sensor karena berfungsi untuk merubah sebuah tekanan mekanik pada tombol menjadi sinyal elektrik. Pada antarmuka ini *push button* akan digunakan untuk mengontrol sebuah LED yang terhubung dengan Arduino dimana prosesnya dikendalikan oleh raspberry melalui protokol firmata. LED digunakan sebagai indikator tambahan, jadi LED bukanlah sensor digital!!.

Push button terhubung dengan arduino yaitu pada PIN Digital 3, sedangkan LED terhubung dengan arduino pada PIN Digital 2. Pin Digital nomor 1 dan 2 pada arduino tidak digunakan karena PIN tersebut merupakan PIN Serial yang telah digunakan untuk komunikasi serial Firmata antara arduino dengan raspberry. Berikut gambar skematik yang digunakan



Gambar 2.4 Skematik antarmuka Push Button & LED

Pada skematik diatas terdapat tambahan resistor pada *push button* yang berfungsi sebagai *pull-down* sehingga jika tombol ditekan maka sinyal HIGH akan masuk pada pin digital 2 sebaliknya, jika tombol tidak ditekan maka sinyal LOW akan mengalir pada pin digital 2. Resistor yang berfungsi sebagai *pull down* ini digunakan karena pada protokol *firmata* tidak terdapat perintah untuk mengatur pin digital dengan *mode PULL_UP*. Setelah selesai menyusun rangkaian seperti diatas, kemudian tulis program dibawah ini untuk membaca data sensor *push button* untuk menghidupkan LED yang keduanya terhubung dengan arduino.

```
# -----
# Program Antarmuka Push Button dan LED
# File: btn_led.py
# -----

import pyfirmata
from time import sleep
#PORT USB Arduino
PORT = '/dev/ttyACM0'
LED_PIN = 3 #PIN LED pada arduino
BOARD = pyfirmata.Arduino(PORT)
```

```
#PIN D2 untuk Push Button
ITER = pyfirmata.util.Iterator(BOARD)
BTN_PIN = BOARD.get_pin('d:0:2')
BOARD.digital[LED_PIN].mode = OUTPUT # Mode Output
ITER.start()
while True:
    dataBTN = BTN_PIN.read()
    if dataBTN == 1:
        print "Tombol ditekan, LED ON"
        BOARD.digital[LED_PIN].write(1)
    else:
        print "Tombol dilepas, LED OFF"
        BOARD.digital[LED_PIN].write(0)
    sleep(0.5)
```

Simpan program diatas dengan nama file *ldr.py* lalu jalankan dengan mengetikan perintah berikut pada terminal

```
pi@raspberrypi ~ $ #Menjalankan program btn_led.py
pi@raspberrypi ~ $ sudo python btn_led.py
```

Setelah program diatas dijalankan maka kondisi dari sensor *push button* akan ditampilkan setiap 0.5 detik dan LED akan menyala atau mati sesuai dengan kondisi push button



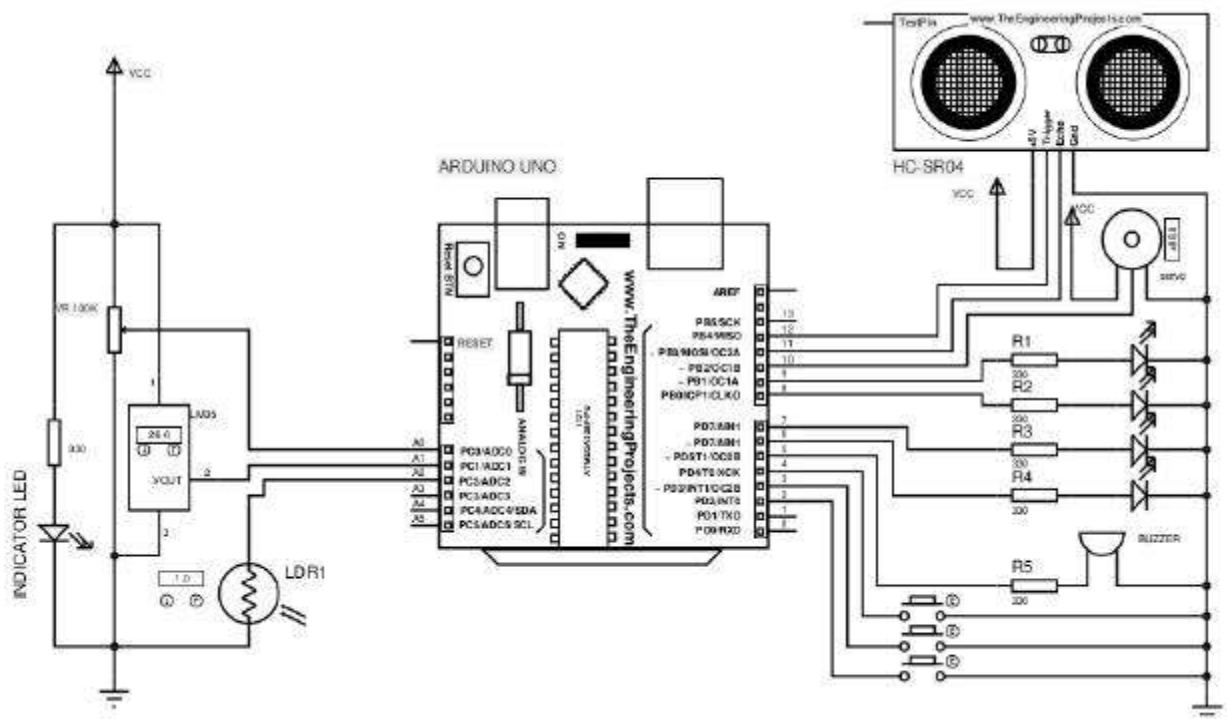
Gambar 2.5 Tampilan kondisi Push Button dan LED pada Terminal

Antarmuka Sensor

Pada antarmuka sensor ini akan dibahas bagaimana cara membaca data sensor yang terdapat pada arduino dalam hal ini sensor berupa potensiometer. Data hasil pembacaan ini nantinya akan ditampilkan pada sebuah widget label yang terdapat pada tkinter.

Widget Label dan Potensiometer

Sebelumnya sudah mempelajari membaca data analog dari arduino menggunakan pyfirmata. Untuk program ini akan menampilkan data yang dikirim dari pin Analog arduino pada GUI. Sesuai pada rangkaian tersedia pin analog yang digunakan adalah pin A0 yang tersambung pada potensiometer sebagai input analog. Berikut gambar skematiknya



Gambar 3.1 Skematik Antarmuka

Catatan: Skematik diatas akan digunakan pada program-program antarmuka berikutnya pada bab enam.

Berikut program yang digunakan untuk membangun program GUI, membaca data potensiometer, dan menampilkannya pada widget label.

Program Monitoring Pin I/O dengan GUI Label

```
import Tkinter
from pyfirmata import Arduino,util
from time import sleep
#Configuration Arduino
port = '/dev/ttyACM0'
board = Arduino(port)
```

```

#Configurasi pin Analog pada arduino
it=util.Iterator(board)
it.start()
#Pin yang digunakan A0
a0=board.get_pin('a:0:i')
#fungsi untuk membaca data analog dan menampilkan di Label
def onStartButtonPress():
    while True:
        if flag.get():
            analogReadLabel.config(text=str(a0.read()*255))
            analogReadLabel.update_idletasks()
            top.update()
        else:
            break
    board.exit()
    top.destroy()
#Fungsi untuk menutup jendela GUI
def onStopButtonPress():
    flag.set(False)

#Membangun GUI
top = Tkinter.Tk()
top.title("data potensio")
flag=Tkinter.BooleanVar(top)
flag.set(True)
#
descriptionLabel = Tkinter.Label(top, text="Potentiometer input:- ")
descriptionLabel.grid(column=1, row=1)
analogReadLabel = Tkinter.Label(top, text="Press Start..")
analogReadLabel.grid(column=2, row=1)
startButton= Tkinter.Button(top,
                             text="Start",
                             command=onStartButtonPress)
startButton.grid(column=1,row=3)
stopButton = Tkinter.Button(top,
                             text="Exit",
                             command=onStopButtonPress)
stopButton.grid(column=2,row=3)
top.mainloop()

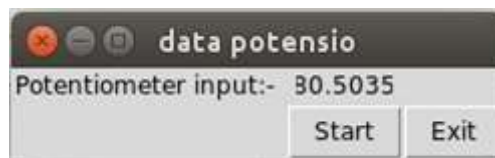
```

Jalankan program maka output GUI yang akan tampil seperti gambar berikut:



Gambar 3.2 Hasil Program

Tekan Button start untuk mulai membaca data analog dari potensio maka hasil data yang ditampilkan pada Label seperti gambar berikut:



Gambar 3.3 Hasil Monitoring

Antarmuka Aktuator

Aktuator yang digunakan pada antarmuka aktuator ini untuk mewakili berbagai macam aktuator yang tersedia adalah LED. Hal ini karena semua aktuator memiliki prinsip pemrograman yang sama hanya saja penggunaan aktuatornya yang berbeda.

Tkinter Widget Button dan LED

Pada praktek selanjutnya adalah membuat program GUI untuk on/off LED antarmuka dengan arduino. Sebelumnya sudah disediakan Arduino dan modul shield Arduino. Untuk gambar skematik yang digunakan adalah skematik pada gambar 6.1.

Didalam Tkinter terdapat berbagai fungsi salah satunya adalah Button. Button merupakan suatu tombol yang sering digunakan sebagai input data. Pada program ini Button akan menghidupkan Led yang ada pada modul arduino. Untuk programnya tulis program pada text editor:

Program Blink LED GUI Tkinter

```
from Tkinter import *
import pyfirmata
from time import sleep
port = '/dev/ttyACM0'
board = pyfirmata.Arduino(port)
print 'loading...'
sleep(5)
def onStartButtonPress():
    print 'run'
    startButton.config(state=DISABLED)
    ledPin.write(1)
    #LED is on for fix amount of time specified below
    sleep(5)
```

```

        ledPin.write(0)
        startButton.config(state=ACTIVE)
ledPin=board.get_pin('d:8:o')
top=Tk()
top.title("Blink LED using Button")
top.minsize(300,30)
startButton=Button(top,text="Start",command=onStartButtonPress)
startButton.pack()
top.mainloop()

```

Setelah menulis program diatas maka jalankan program. Maka akan tampil GUI seperti gambar berikut:



Gambar 3.4 Hasil Program Blink LED GUI

Cara kerja dari program ini dengan menekan Button Start maka LED yang ada pada modul Arduino akan hidup selama 5 detik.

Tkinter Widget Entry dan LED (Variabel Tunda)

Setelah belajar program menghidupkan LED menggunakan Button program selanjutnya adalah menggunakan widget entry. Pada program ini sama seperti program sebelumnya yaitu menghidupkan LED namun waktu jeda(delay) di input dari keyboard secara manual menggunakan entry yang ada pada Tkinter. Tulis program berikut menggunakan text editor:

Program GUI GUI Entry time LED on

```

from Tkinter import *
import pyfirmata
from time import sleep
port = '/dev/ttyACM0'
board = pyfirmata.Arduino(port)
print 'loading...'
sleep(5)
def onStartButtonPress():
    print 'run'
    startButton.config(state=DISABLED)
    ledPin.write(1)
    #LED is on for fix amount of time specified below
    sleep(5)
    ledPin.write(0)
    startButton.config(state=ACTIVE)

```

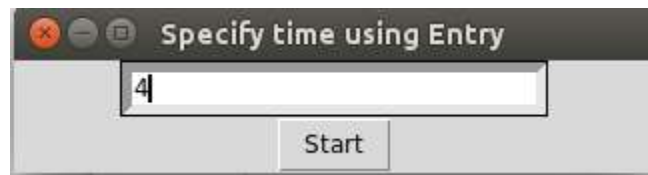


```

ledPin=board.get_pin('d:8:o')
top=Tk()
top.title("Blink LED using Button")
top.minsize(300,30)
startButton=Button(top,text="Start",command=onStartButtonPress)
startButton.pack()
top.mainloop()

```

Run program diatas maka output GUI seperti gambar berikut:



Gambar 3.5 GUI Entry time LED on

Apabila program berhasil di jalankan maka akan seperti gambar diatas. Pada gambar diatas angka yang di entry dari keyboard angka 4. Maka setelah Button start ditekan maka Led pada arduino akan menyala selama 4 detik.

Tkinter Widget Scale dan LED dengan PWM

PWM (pulse width modulation) adalah salah satu teknik modulasi dengan mengubah lebar pulsa (duty cycle) dengan nilai amplitudo dan frekuensi yang tetap. Satu siklus pulsa merupakan kondisi high kemudian berada di zona transisi ke kondisi low. Lebar pulsa PWM berbanding lurus dengan amplitudo sinyal asli yang belum termodulasi. Duty Cycle merupakan representasi dari kondisi logika high dalam suatu periode sinyal dan dinyatakan dalam bentuk (%) dengan range 0% sampai 100%, sebagai contoh jika sinyal berada dalam kondisi high terus menerus artinya memiliki duty cycle sebesar 100%. Jika waktu sinyal keadaan high sama dengan keadaan low maka sinyal mempunyai duty cycle sebesar 50%.



Gambar 3.6 Duty cycle pada PWM

Pada praktek antarmuka akan membuat program GUI mengatur nilai duty cycle pada pwm untuk mengatur tingkat kecerahan pada LED. Sebelumnya LED tersambung pada pin pwm yang ada pada arduino uno adalah 3,5,6,9,10,11. Input dari yang digunakan adalah data yang

ada pada widget scale yang ada pada Tkinter. Range data yang dikirim dari widget scale 0-100. Nilai ini akan menjadi data duty cycle PWM untuk mengatur tingkat kecerahan pada LED.

Program Mengatur PWM LED dengan scale

```
import Tkinter
import pyfirmata
from pyfirmata import PWM
from time import sleep
port = '/dev/ttyACM0'
board = pyfirmata.Arduino(port)
#PIN pwm arduino yang digunakan pin 9
pin=9
board.digital[pin].mode=PWM
#fungsi untuk menghidupkan LED menggunakan PWM.
Def onStartButtonPress():
    timePeriod = timePeriodEntry.get()
    timePeriod = float(timePeriod)
    ledBrightness = brightnessScale.get()
    ledBrightness = float(ledBrightness)
    startButton.config(state=Tkinter.DISABLED)
    board.digital[pin].write(ledBrightness/100.0)
    sleep(timePeriod)
    board.digital[pin].write(0)
    startButton.config(state=Tkinter.ACTIVE)

#Membangun GUI.
top =Tkinter.Tk()
top.title("Brigness using scale")
top.minsize(320,240)
#Menambah widget etry pada GUI
timePeriodEntry =Tkinter.Entry(top,bd=5,width=25)
timePeriodEntry.pack()
timePeriodEntry.focus_set()
#Menambah widget Scale pada GUI dengan range min-max 0-100
brightnessScale =Tkinter.Scale(top,                                from_=0,to=100,
orient=Tkinter.HORIZONTAL)
brightnessScale.pack()
#Manambah widget Button start pada GUI
startButton=Tkinter.Button(top,text="Start",command=onStartButtonPress)
startButton.pack()
```

```
top.mainloop()
```

Kemudian jalankan program, maka hasil GUI seperti gambar berikut:



Gambar 3.7 GUI entry time dan scale PWM LED

Pada GUI diatas terdapat scale untuk mengirim data duty cicle pwm untuk tingkat kecerahan LED sedangkan entry untuk mengatur jeda waktu LED hidup.

Tkinter Widget Check Button dan Multi LED

Pada program sebelumnya program untuk menghidupkan LED menggunakan scale,entry. Pada program selanjutnya kita akan membuat program pemilihan untuk menyalakan LED menggunakan widget Check Button yang ada pada Tkinter. Untuk penulisan program menambah widget pada Gui seperti gambar berikut:

Program menambah widget Check Button pada Gui

```
import Tkinter
import pyfirmata
from time import sleep
port = '/dev/ttyACM0'
board = pyfirmata.Arduino(port)
#Menambah 2 PIN arduino 7,8 untuk LED
redPin=board.get_pin('d:8:o')
greenPin=board.get_pin('d:7:o')

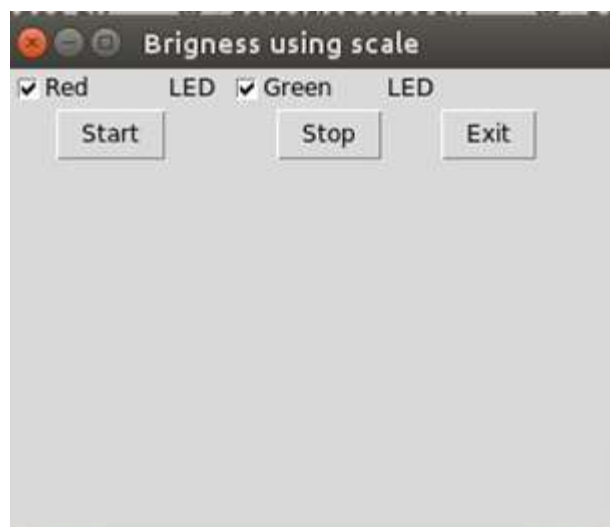
#fungsi untuk menghidupkan LED
def onStartButtonPress():
    redPin.write(redVar.get())
    greenPin.write(greenVar.get())
#fungsi untuk mematikan LED
def onStopButtonPress():
    redPin.write(0)
    greenPin.write(0)
#Membangun GUI
top = Tkinter.Tk()
top.title("Brigness using scale")
```

```

top.minsize(320,240)
redVar = Tkinter.IntVar()
# Inisialisasi widget checkbutton
redCheckBox= Tkinter.Checkbutton(top,
                                text="Red   LED",
                                variable=redVar)
redCheckBox.grid(column=1,   row=1)
greenVar = Tkinter.IntVar()
greenCheckBox = Tkinter.Checkbutton(top,
                                    text="Green LED",
                                    variable=greenVar)
greenCheckBox.grid(column=2,row=1)
#Inisialisasi widget Button sebagai tombol start stop
startButton= Tkinter.Button(top,
                             text="Start",
                             command=onStartButtonPress)
startButton.grid(column=1,   row=3)
stopButton = Tkinter.Button(top,
                             text="Stop",
                             command=onStopButtonPress)
stopButton.grid(column=2,   row=3)
exitButton = Tkinter.Button(top,
                             text="Exit",
                             command=top.quit)
exitButton.grid(column=3,   row=3)
top.mainloop()

```

Jalankan programnya maka GUI yang akan tampil seperti gambar berikut:



Gambar 3.8 Check Button control LED

Pilih salahsatu atau kedua check button pada GUI . Lalu tekan tombol start maka kedua Led yang ada pada modul arduino Pin 7 dan 8 akan hidup jika kedua check button dipilih semua. Tombol stop untuk meng OFF kan LED sedangkan button exit untuk menutup jendela GUI.

Dasar mengolah data di Python

Pada bab ini akan dijelaskan tentang prosedur membuat dan memodifikasi file menggunakan *python*. Modul yang akan digunakan merupakan modul yang telah disediakan *python*. Berikut ini beberapa metode yang dapat digunakan untuk dapat bekerja dengan file menggunakan *python*.

Metode open()

Metode open() merupakan metode default yang disediakan python untuk membaca ataupun menulis file. Untuk contoh dasar program membuka file dengan open seperti pada contoh program berikut: Berikut adalah contoh sederhana membuat data txt dengan python:

Contoh Membuat file txt *Python*

```
#Menulis dan membaca isi data dengan python
f = open('test.txt','w')
f.write("Teknik Antarmuka II, Pemograman Python")
```

Pada metode open diatas terdapat parameter 'test.txt' dan 'w'. 'test.txt' merupakan nama file yang akan dibuat sedangkan 'w' merupakan metode menulis. Text yang akan ditulis pada file tersebut adalah 'Teknik Antarmuka II, Pemrograman Python'. Untuk melihat hasil file yang sudah dibuat dengan program python diatas, buka folder dimana script diatas dibuat maka akan terdapat file dengan nama *test.txt* yang berisi "*Teknik Antarmuka II, Pemograman Python*" seperti pada berikut:

```
pi@raspberrypi ~ $ cat test.txt
Teknik Antarmuka II, Pemograman Python
pi@raspberrypi ~ $
```

Gambar 4.1 Hasil pembuatan text

Mode lain yang dapat digunakan untuk memanipulasi file adalah:

Tabel 4.1Daftar Fungsi memanipulasi data

Mode	1. Keterangan
w	2. Membuka atau membuat sebuah file untuk penulisan saja. Ini akan menimpa file dengan nama yang sama.
w+	3. Membuka atau membuat file untuk penulisan dan pembacaan file. Ini akan menimpa file dengan nama yang sama.
r	4. Membuka file hanya untuk pembacaan saja.
r+	5. Membuka file untuk penulisan dan pembacaan.
a	6. Membuka file untuk menambahkan isi diakhir dokumen.
a+	7. Membuka file untuk menambahkan isi diakhir dokumen dan membacanya.

Menyimpan data dari Arduino dalam bentuk CSV

Setelah mempelajari program python memanipulasi data berupa text, pada program ini adalah menyimpan data sensor dari arduino menggunakan protocol Firmata. Untuk contoh programnya seperti berikut:

ProgramMembaca data Arduino dan menyimpan CSV

```
import csv
import pyfirmata
from time import sleep
```

```

port = '/dev/ttyACM0'
board = pyfirmata.Arduino(port)

it = pyfirmata.util.Iterator(board)
it.start()

a1 = board.get_pin('a:1:i')
a0 = board.get_pin('a:0:i')

with open('DataSensor.csv', 'w') as f:
    w = csv.writer(f)
    w.writerow(["No", "Potensiometer", "LM 35"])
    i = 0
    lm35Data = a1.read()
    potData = a0.read()
    while i<25:
        lm35Data = a1.read()
        potData = a0.read()
        print(i, potData, lm35Data)
        print i
        sleep(1)
        i += 1
        row = [i, potData, lm35Data]
        w.writerow(row)
    print "File CSV Telah Tersedia!"

board.exit()

```

Pada program diatas data sensor yang disimpan adalah data analog dari potensio dan data suhu dari LM 35. Masing-masing data akan disimpan sebanyak 25 data dengan durasi setiap detik. Untuk hasil program diatas seperti pada tabel Berikut:

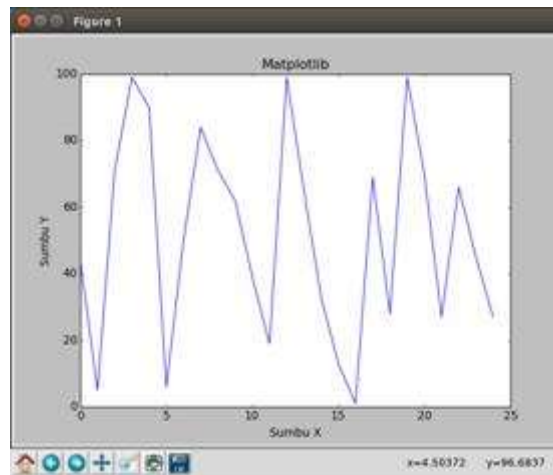
Tabel 4.2 Data CSV sensor

No	Potensiometer	LM 35
1		
2	0.4995	0.0596
3	0.4985	0.0596
4	0.4995	0.0596
5	0.4985	0.0596
6	0.4985	0.0596
7	0.4995	0.0596
8	0.4985	0.0596
9	0.5005	0.0596
10	0.3539	0.0557
11	0.1642	0.0547
12	0	0.0547
13	0	0.0547
14	0.2698	0.0547
15	0.4897	0.0587
16	0.7527	0.0577
17	1	0.0508
18	1	0.0518
19	0.9062	0.0538
20	0.7634	0.0567
21	0.2219	0.0547
22	0.3118	0.0538
23	0.7312	0.0577
24	0.7302	0.0567
25	0.7302	0.0577

Plotting Data dengan *Library Matplotlib*

Apa itu *matplotlib* ?

Library *matplotlib* adalah salah satu library plotting yang populer dan di support oleh Python. Library ini terinspirasi dari MATLAB, namun library ini berbeda. Library *matplotlib* merupakan *library* open source yang dapat membantu membuat plot 2D dari baris code sederhana dan banyak digunakan pada aplikasi Python untuk visualisasi dan analisis. Dalam penggunaannya *library* ini membutuhkan penggunaan library *numPy* atau *SciPy* yang digunakan untuk perhitungan matematis untuk analisis.



Gambar 4.2 Contoh Plotting Matplotlib

Konfigurasi Matplotlib

Untuk meng-*install* library matplotlib pada Raspberry pi sangat mudah tanpa menggunakan Setuptools. Berikut perintah sederhana untuk meng-*install* matplotlib pada *Raspberry pi*:

Contoh perintah terminal *install matplotlib*

```
pi@raspberrypi:~ $ sudo apt-get install python-matplotlib
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

Untuk library tambahan mengolah data plotting maka install packet Numpy dan scipy. Untuk installnya seperti berikut:

Contoh perintah *install numpy*

```
pi@raspberrypi:~ $ sudo apt-get install python-numpy
Reading package lists... Done
Building dependency tree
Reading state information... Done
python-numpy is already the newest version (1:1.12.1-3).
python-numpy set to manually installed.
```

Berikut adalah program sederhana untuk plotting data.

Contoh Program plotting data

```
from matplotlib import pyplot
import random

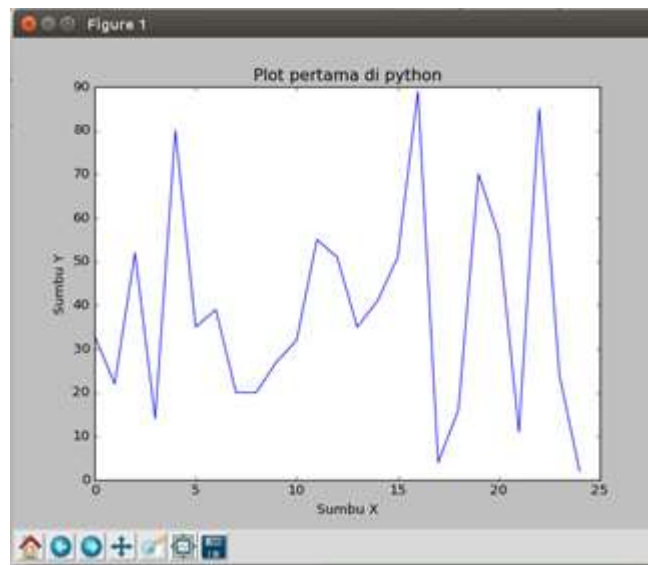
x = range (0,25)
y = [random.randint(0,100) for r in range (0,25)]
fig1 = pyplot.figure()
```

```

pyplot.plot(x, y, '-')
pyplot.title('Plot pertama di python')
pyplot.xlabel('Sumbu X')
pyplot.ylabel('Sumbu Y')
pyplot.show()

```

Jalankan program diatas maka hasilnya seperti pada gambar berikut:



Gambar 4.3 Hasil plotting data random

Plotting Data dari File CSV

Setelah mempelajari dasar mengolah file CSV dan Plotting data dengan Matplotlib, pada bab ini akan mencoba menampilkan plotting data dari file CSV yang sebelumnya sudah diambil dari data sensor. Untuk contoh programnya seperti berikut:

```

import csv
from matplotlib import pyplot
i=[]
mValues = []
pValues = []

with open('DataSensor.csv','r') as f:
    reader = csv.reader(f)
    header = next(reader,None)
    for row in reader:
        i.append(int(row[0]))
        pValues.append(float(row[0]))
        if row[2]=='True':

```

```

        mValues.append(1)
    else:
        mValues.append(0)

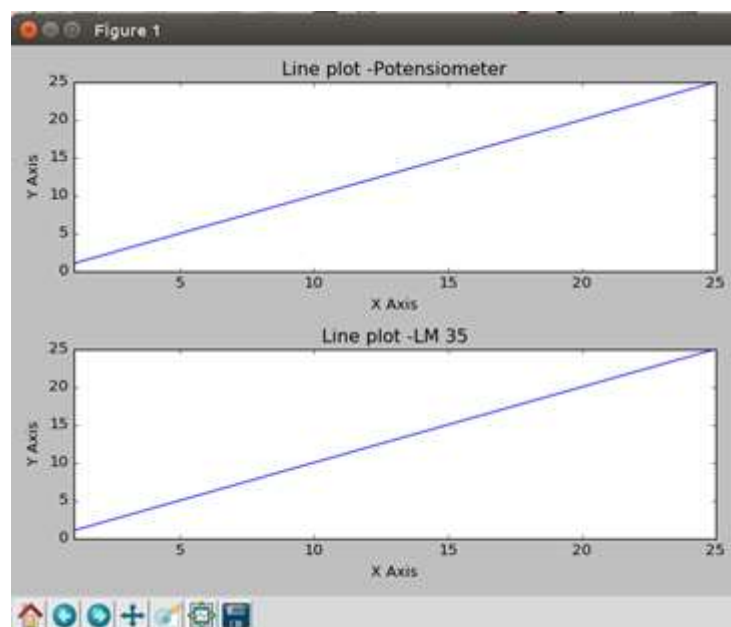
pyplot.subplot(2,1,1)
pyplot.plot(i,pValues,'-')
pyplot.title('Line plot -'+header[1])
pyplot.xlim(1,25)
pyplot.xlabel('X Axis')
pyplot.ylabel('Y Axis')

pyplot.subplot(2,1,2)
pyplot.plot(i,pValues,'-')
pyplot.title('Line plot -'+header[2])
pyplot.xlim(1,25)
pyplot.xlabel('X Axis')
pyplot.ylabel('Y Axis')

pyplot.tight_layout()
pyplot.show()

```

Untuk hasil program diatas seperti pada gambar berikut:



Gambar 4.4 Plotting data CSV

Plotting Data Realtime Python

Setelah sebelumnya mempelajari plotting data dari file CSV menggunakan matplotlib, pada sub bab ini yang akan dipelajari adalah plotting data secara realtime dari data sensor pada Arduino menggunakan protokol firmata. Untuk program plotting data realtime dari arduino seperti berikut:

Program realtime plotting Arduino

```
import sys, csv
from matplotlib import pyplot
import pyfirmata
from time import sleep

#port untuk pyfirmata arduino
port='/dev/ttyACM0'
board=pyfirmata.Arduino(port)

#menggunakan PIN analog Untuk arduino
it=pyfirmata.util.Iterator(board)
it.start()

#variable untuk pin analog
a0=board.get_pin('a:0:1')

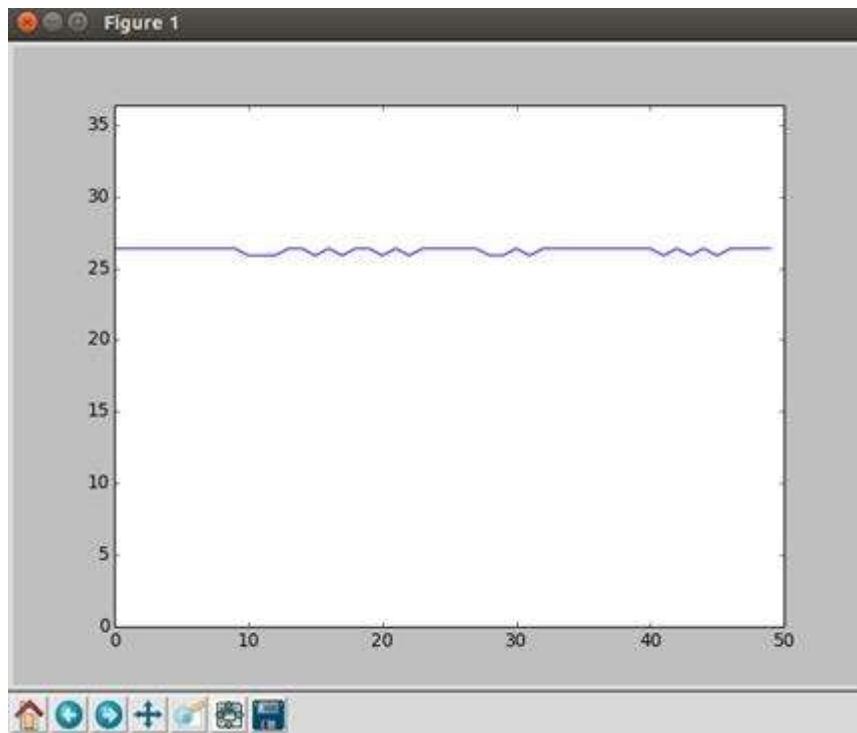
#
pyplot.ion()

pData=[0]*25
fig=pyplot.figure()
pyplot.title('Data realtime sensor')
ax1=pyplot.axes()
l1,=pyplot.plot(pData)
pyplot.ylim([0,1])

#real_time value
while True:
    try:
        sleep(0.1)
        pData.append(float(a0.read()))
        print a0.read()
        pyplot.set_ylim([0,1])
        del pData[0]
        l1.set_xdata([i for i in xrange(25)])
        l1.set_ydata(pData)
```

```
pyplot.show()  
except KeyboardInterrupt:  
    board.exit()  
    break
```

Untuk hasil plotting realtime program diatas seperti berikut:



Gambar 4.5 Plotting data Real Time

g. Rubrik Penilaian