Percobaan

1. Menyalakan 1 LED

**Source Code**

int L1 = 15;

void setup() {

pinMode(L1, OUTPUT);

}

void loop() {

digitalWrite(L1,HIGH);

delay(500);

digitalWrite(L1,LOW);

delay(300);

}

1. Menyalakan LED bergiliran

**Source Code**

int L1 = 15;

int L2 = 13;

int L3 = 12;

int L4 = 14;

int L5 = 4;

int L6 = 5;

void setup() {

// put your setup code here, to run once:

pinMode(L1, OUTPUT);

pinMode(L2, OUTPUT);

pinMode(L3, OUTPUT);

pinMode(L4, OUTPUT);

pinMode(L5, OUTPUT);

pinMode(L6, OUTPUT);

}

void loop() {

// put your main code here, to run repeatedly:

digitalWrite(L1,HIGH);

delay(500);

digitalWrite(L1,LOW);

delay(300);

digitalWrite(L2,HIGH);

delay(500);

digitalWrite(L2,LOW);

delay(300);

digitalWrite(L3,HIGH);

delay(500);

digitalWrite(L3,LOW);

delay(300);

digitalWrite(L4,HIGH);

delay(500);

digitalWrite(L4,LOW);

delay(300);

digitalWrite(L5,HIGH);

delay(500);

digitalWrite(L5,LOW);

delay(300);

digitalWrite(L6,HIGH);

delay(500);

digitalWrite(L6,LOW);

delay(300);

}

1. Menyalakan LED menggunakan Button

**Source Code**

const int buttonPin = 4;

const int ledPin = 13;

bool lastButtonState = LOW; // status tombol terakhir

bool ledState = LOW; // status LED

void setup() {

pinMode(buttonPin, INPUT); // tombol sebagai input

pinMode(ledPin, OUTPUT); // LED sebagai output

}

void loop() {

bool buttonState = digitalRead(buttonPin); // cek status tombol

if (buttonState != lastButtonState) { // jika status tombol berubah dari sebelumnya

if (buttonState == LOW) { // jika tombol berubah ke status LOW (dilepas)

if (ledState == HIGH) { // jika status LED hidup

digitalWrite(ledPin, LOW); // maka LED dimatikan

ledState = LOW;

}

else { // jika status LED mati

digitalWrite(ledPin, HIGH); // maka LED dihidupkan

ledState = HIGH;

}

}

delay(50); // debounching

}

lastButtonState = buttonState; // simpan status tombol untuk loop selanjutnya

}

1. Relay

**Sourse Code**

const int relay1 = 5;

const int relay2 = 4;

void setup() {

// put your setup code here, to run once:

Serial.begin(115200);

pinMode(relay1,OUTPUT);

pinMode(relay2,OUTPUT);

}

void loop() {

// put your main code here, to run repeatedly:

digitalWrite(relay1,HIGH);

Serial.println("Relay 1 Hidup");

delay(1000);

digitalWrite(relay2,HIGH);

Serial.println("Relay 1 & 2 Hidup");

delay(1000);

digitalWrite(relay1,LOW);

Serial.println("Relay 1 Mati");

delay(500);

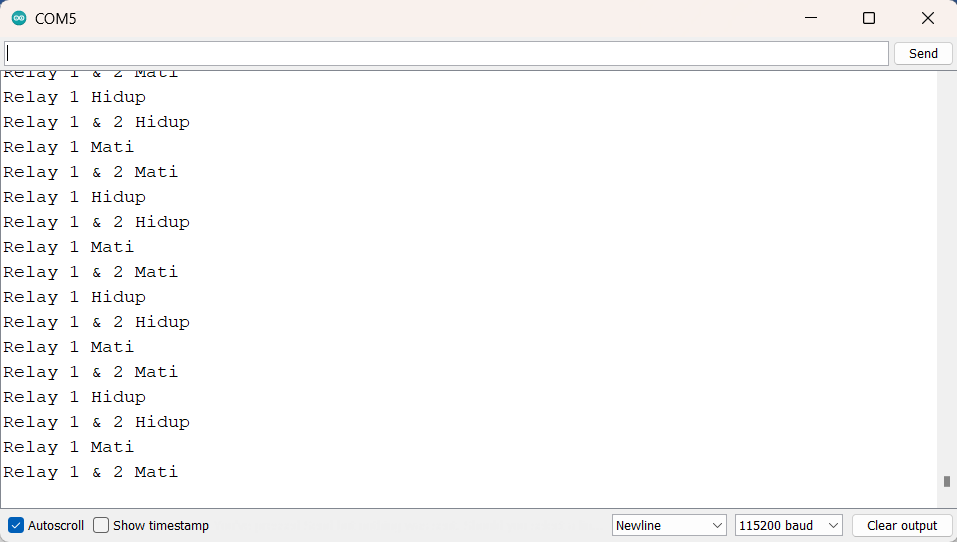
digitalWrite(relay2,LOW);

Serial.println("Relay 1 & 2 Mati");

delay(500);

}

**Hasil:**



a)Relay 1 Nyala b)Relay 1 & 2 Nyala



c)Relay 1 Mati d)Relay 1 & 2 Mati



1. Sensor Ultrasonik

**Sourse Code**

const int TRIGPIN = 5;

const int ECHOPIN = 4;

long timer;

int jarak;

void setup()

{

Serial.begin(115200);

pinMode(ECHOPIN, INPUT);

pinMode(TRIGPIN, OUTPUT);

}

void loop()

{

digitalWrite(TRIGPIN, LOW);

delayMicroseconds(2);

digitalWrite(TRIGPIN, HIGH);

delayMicroseconds(10);

digitalWrite(TRIGPIN, LOW);

timer = pulseIn(ECHOPIN, HIGH);

jarak = timer/58;

delay(500);

Serial.print("Jarak = ");

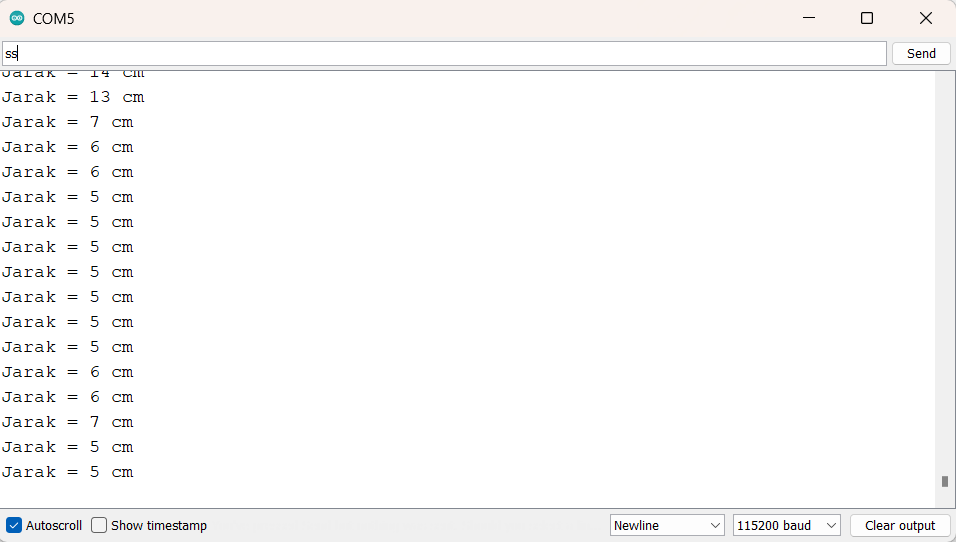
Serial.print(jarak);

Serial.print(" cm");

Serial.println();

}

**Hasil:**

****



1. Sensor DHT11

**Sourse Code**

#include "DHT.h" //library sensor yang telah diimportkan

#define DHTPIN 5 //Pin apa yang digunakan

#define DHTTYPE DHT11 // DHT 11

DHT dht(DHTPIN, DHTTYPE);

void setup() {

Serial.begin(115200); //baud komunikasi serial

Serial.println("Pengujian DHT11!"); //penulisan di serial monitor

dht.begin(); //prosedur memulai pembacaan module sensor

}

void loop() {

delay(2000); //menunggu beberapa detik untuk pembacaan

//pembacaan sensor membutuhkan waktu 250ms

//Pembacaan untuk data kelembaban

float humidity\_1 = dht.readHumidity();

//Pembacaan dalam format celcius (c)

float celcius\_1 = dht.readTemperature();

//pembacaan dalam format Fahrenheit

float fahrenheit\_1 = dht.readTemperature(true);

//mengecek pembacaan apakah terjadi kegagalan atau tidak

if (isnan(humidity\_1) || isnan(celcius\_1) || isnan(fahrenheit\_1)) {

Serial.println("Pembacaan data dari module sensor gagal!");

return;

}

float htof = dht.computeHeatIndex(fahrenheit\_1, humidity\_1);

//Prosedur pembacaaan data indeks panas dalam bentuk fahreheit

float htoc = dht.computeHeatIndex(celcius\_1, humidity\_1, false);

//Prosedur pembacaaan data indeks panas dalam bentuk celcius

//pembacaan nilai pembacaan data kelembaban

Serial.print("Kelembaban: ");

Serial.print(humidity\_1);

Serial.print(" %\t");

//pembacaan nilai pembacaan data suhu

Serial.print("Suhu : ");

Serial.print(celcius\_1); //format derajat celcius

Serial.print("°"); //simbol derajat

Serial.print("C / ");

Serial.print(fahrenheit\_1); //format derajat fahrenheit

Serial.print("°");

Serial.print("F\t");

Serial.print("Indeks Panas: ");

Serial.print(htof);

Serial.print("°");

Serial.print("F / ");

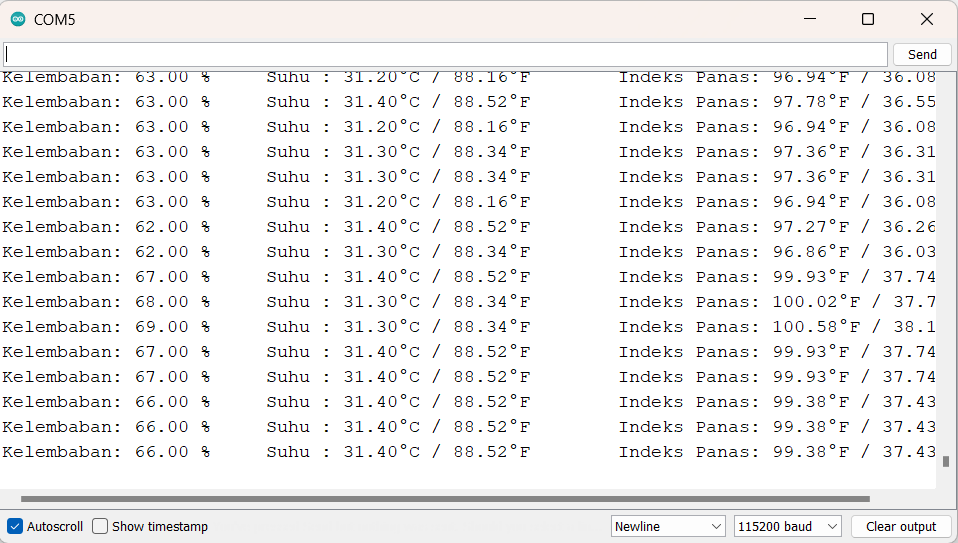
Serial.print(htoc);

Serial.print("°");

Serial.println("C ");

}

**Hasil:**



1. Sensor Hujan

**Sourse Code**

const int sensor\_hujan = 5;

const int LED = 4;

void setup () {

Serial.begin(115200);

pinMode (sensor\_hujan, INPUT);

pinMode (LED, OUTPUT);

}

void loop() {

int kondisi\_sensor = digitalRead(sensor\_hujan);

if (kondisi\_sensor == LOW) {

Serial.println("Cuaca Hujan");

digitalWrite(LED, HIGH);

}

else {

Serial.println("Cuaca Cerah");

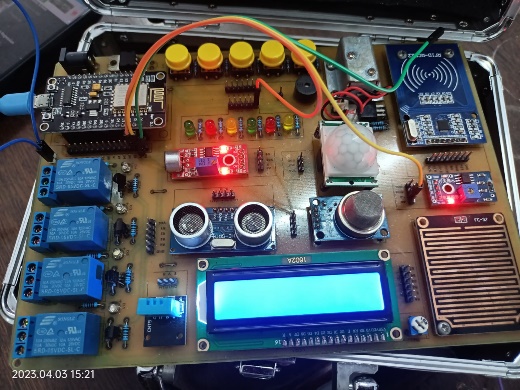
digitalWrite(LED, LOW);

}

delay (1000);

}

**Hasil:**

a)Saat Hujan b)Saat Cerah

1. Module RFID

**Source Code**

#include <SPI.h>

#include <MFRC522.h>

#define SS\_PIN 15

#define RST\_PIN 16

MFRC522 rfid(SS\_PIN, RST\_PIN); // Instance of the class

MFRC522::MIFARE\_Key key;

// Init array that will store new NUID

byte nuidPICC[4];

void setup() {

Serial.begin(115200);

SPI.begin(); // Init SPI bus

rfid.PCD\_Init(); // Init MFRC522

Serial.println();

Serial.print(F("Reader :"));

rfid.PCD\_DumpVersionToSerial();

for (byte i = 0; i < 6; i++) {

key.keyByte[i] = 0xFF;

}

Serial.println();

Serial.println(F("This code scan the MIFARE Classic NUID."));

Serial.print(F("Using the following key:"));

printHex(key.keyByte, MFRC522::MF\_KEY\_SIZE);

}

void loop() {

// Reset the loop if no new card present on the sensor/reader. This saves the entire process when idle.

if ( ! rfid.PICC\_IsNewCardPresent())

return;

// Verify if the NUID has been readed

if ( ! rfid.PICC\_ReadCardSerial())

return;

Serial.print(F("PICC type: "));

MFRC522::PICC\_Type piccType = rfid.PICC\_GetType(rfid.uid.sak);

Serial.println(rfid.PICC\_GetTypeName(piccType));

// Check is the PICC of Classic MIFARE type

if (piccType != MFRC522::PICC\_TYPE\_MIFARE\_MINI &&

piccType != MFRC522::PICC\_TYPE\_MIFARE\_1K &&

piccType != MFRC522::PICC\_TYPE\_MIFARE\_4K) {

Serial.println(F("Your tag is not of type MIFARE Classic."));

return;

}

if (rfid.uid.uidByte[0] != nuidPICC[0] ||

rfid.uid.uidByte[1] != nuidPICC[1] ||

rfid.uid.uidByte[2] != nuidPICC[2] ||

rfid.uid.uidByte[3] != nuidPICC[3] ) {

Serial.println(F("A new card has been detected."));

// Store NUID into nuidPICC array

for (byte i = 0; i < 4; i++) {

nuidPICC[i] = rfid.uid.uidByte[i];

}

Serial.println(F("The NUID tag is:"));

Serial.print(F("In hex: "));

printHex(rfid.uid.uidByte, rfid.uid.size);

Serial.println();

Serial.print(F("In dec: "));

printDec(rfid.uid.uidByte, rfid.uid.size);

Serial.println();

}

else Serial.println(F("Card read previously."));

// Halt PICC

rfid.PICC\_HaltA();

// Stop encryption on PCD

rfid.PCD\_StopCrypto1();

}

/\*\*

Helper routine to dump a byte array as hex values to Serial.

\*/

void printHex(byte \*buffer, byte bufferSize) {

for (byte i = 0; i < bufferSize; i++) {

Serial.print(buffer[i] < 0x10 ? " 0" : " ");

Serial.print(buffer[i], HEX);

}

}

/\*\*

Helper routine to dump a byte array as dec values to Serial.

\*/

void printDec(byte \*buffer, byte bufferSize) {

for (byte i = 0; i < bufferSize; i++) {

Serial.print(buffer[i] < 0x10 ? " 0" : " ");

Serial.print(buffer[i], DEC);

}

}

**Hasil:**

