1. **NO. JOBSHEET** : 2
2. **JUDUL** : PROTOKOL KOMUNIKASI DAN SENSOR
3. **TUJUAN**
    1) Mahasiswa dapat memahami cara kerja protokol komunikasi yang terdapat pada ESP32, seperti UART, I2C, OneWire, SPI.
    2) Mahasiswa dapat menggunakan protokol komunikasi data seperti UART, I2C, OneWire, dan SPI untuk mengakses sensor.
    3) Mahasiswa dapat memanfaatkan transducer sensor dan actuator untuk membuat sebuah perangkat IoT.
4. **ALAT DAN BAHAN**
    1) ESP32
    2) Breadboard
    3) Kabel jumper
    4) Sensor DHT 11, RFID
    5) LED (5) dan Push Button (3)
    6) Servo
    7) Resistor 330,1K, 10K Ohm (@ 3)
5. **TEORI SINGKAT**

ESP32 adalah nama dari mikrokontroler yang dirancang oleh perusahaan yang berbasis di Shanghai, China yakni Espressif Systems. ESP32 menawarkan solusi jaringan WiFi dan BLE. ESP32 menggunakan prosesor dual core yang berjalan di instruksi Xtensa LX16. Selain itu, ESP32 telah mendukung protokol komunikasi seperti I2C, UART dan SPI.

5.1. *Universal Asynchronous Receiver Transmitter* (UART)

UART adalah protokol komunikasi yang umum digunakan dalam pengiriman data serial antara device satu dengan yang lainnya. Sebagai contoh komunikasi antara sesama mikrokontroler atau mikrokontroler ke PC. Dalam pengiriman data, clock antara pengirim dan penerima harus sama karena paket data dikirim tiap bit mengandalkan clock tersebut. Inilah salah satu keuntungan model asynchronous dalam pengiriman data karena dengan hanya satu kabel transmisi maka data dapat dikirimkan. Berbeda dengan model synchronous yang terdapat pada protokol SPI dan I2C, karena protokol membutuhkan minimal dua kabel dalam transmisi data, yaitu transmisi clock dan data. Namun kelemahan model asynchronous adalah dalam hal kecepatannya dan jarak transmisi. Karena semakin cepat dan jauhnya

jarak transmisi membuat paket-paket bit data menjadi terdistorsi sehingga data yang dikirim atau diterima bisa mengalami error.

Asynchronous memungkinkan transmisi mengirim data tanpa sang pengirim harus mengirimkan sinyal detak ke penerima. Sebaliknya, pengirim dan penerima harus mengatur parameter waktu di awal dan bit khusus ditambahkan untuk setiap data yang digunakan untuk mensinkronkan unit pengiriman dan penerimaan. Saat sebuah data diberikan kepada UART untuk transmisi Asynchronous, "Bit Start" ditambahkan pada setiap awal data yang akan ditransmisikan. Bit Start digunakan untuk memperingatkan penerima yang kata data akan segera dikirim, dan memaksa bit-bit sinyal di receiver agar sinkron dengan bit-bit sinyal di pemancar. Kedua bit ini harus akurat agar tidak memiliki penyimpangan frekuensi dengan lebih dari 10% selama transmisi bit-bit yang tersisa dalam data. (Kondisi ini ditetapkan pada zaman teleprinter mekanik dan telah dipenuhi oleh peralatan elektronik modern. Setelah Bit Start, bit individu dari data yang dikirim, dengan sinyal bit terkecil yang pertama dikirim. Setiap bit dalam transmisi ditransmisikan serupa dengan jumlah bit lainnya, dan penerima mendeteksi jalur di sekitar pertengahan periode setiap bit untuk menentukan apakah bit adalah 1 atau 0. Misalnya, jika dibutuhkan dua detik untuk mengirim setiap bit, penerima akan memeriksa sinyal untuk menentukan apakah itu adalah 1 atau 0 setelah satu detik telah berlalu, maka akan menunggu dua detik dan kemudian memeriksa nilai bit berikutnya, dan seterusnya.

5.2. *Serial Peripheral Interface* (SPI)

SPI adalah protokol data serial sinkron digunakan oleh mikrokontroler untuk berkomunikasi dengan satu atau lebih perangkat periferal cepat jarak pendek. Hal ini juga dapat digunakan untuk komunikasi antara dua mikrokontroler. Dengan koneksi SPI selalu ada perangkat satu master (biasanya mikrokontroler) yang mengontrol perangkat periferal.

SPI merupakan salah satu mode komunikasi serial synchrounous kecepatan tinggi yang dimiliki oleh Atmega 328. Komunikasi SPI membutuhkan 3 jalur yaitu MOSI, MISO, dan SCK. Melalui komunikasi ini data dapat saling dikirimkan baik antara mikrokontroller maupun antara mikrokontroller dengan

peripheral lain di luar mikrokontroller. Penjelasan 3 jalur utama dari SPI adalah sebagai berikut.

1. **MOSI** : Master Output Slave Input Artinya jika dikonfigurasi sebagai master maka pin MOSI sebagai output tetapi jika dikonfigurasi sebagai slave maka pin MOSI sebagai input.
2. **MISO** : Master Input Slave Output Artinya jika dikonfigurasi sebagai master maka pin MISO sebagai input tetapi jika dikonfigurasi sebagai slave maka pin MISO sebagai output.
3. **CLK** : Clock Jika dikonfigurasi sebagai master maka pin CLK berlaku sebagai output tetapi jika dikonfigurasi sebagai slave maka pin CLK berlaku sebagai input.

5.3. *Inter Integrated Circuit* (I2C),

I2C adalah sebuah protokol untuk komunikasi serial antar IC, dan sering disebut juga Two Wire Interface (TWI). Bus yang digunakan untuk komunikasi antara mikrokontroler dan device lainnya seperi sensor, dll.

Komunikasi dilakukan melalui dua jalur: SDA (serial data) dan SCL (serial clock). Setiap device I2C memiliki 7-bit alamat yang unik. MSB adalah fix dan ditujukan untuk kategori device. Sebagai contoh, 1010 biner ditujukan untuk serial EEPROM. Tiga bit berikutnya memungkinkan 8 kombinasi alamat I2C, yang berarti, dimungkinkan 8 device dengan tipe yang sama, beroperasi pada bus I2C yang sama. Pengalamatan 7-bit memungkinkan 128 device pada bus yang sama. Alamat I2C dikirim dalam byte pertama. LSB dari byte ini digunakan untuk menunjukkan bila master akan melakukan penulisan (0) atau pembacaan (0) terhadap slave.

Device yang mengirim data sepanjang bus disebut master. Sementara device yang menerima data disebut slave. Master memulai transmisi dengan sebuah sinyal start, dan menghentikan transmisi dengan sebuah sinyal stop pada jalur SDA. Selama sinyal start dan stop, jalur SCL harus dalam keadaan high. Setelah master memulai pengiriman data dengan sebuah sinyal start, master menulis satu byte alamat device kepada slave. Setiap byte data harus memiliki panjang 8-bit. Slave harus memberikan konfirmasi dari byte data yang diterimanya dengan sebuah bit acknowledge (ACK).

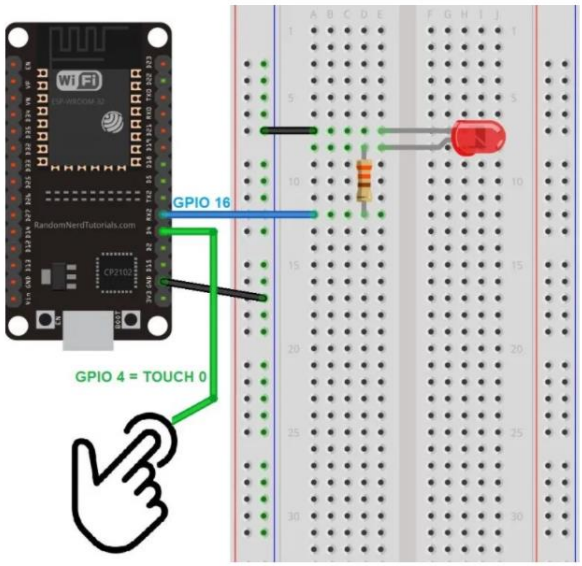
6. LANGKAH PERCOBAAN

A. ESP32 Capacitive Touch Sensor

1. Hubungkan kabel jumper Male-to-Female pada Pin D4 ESP32.
2. Buka Arduino IDE dan upload script program berikut ke ESP32.

```
// ESP32 Touch Test
// Just test touch pin - Touch0 is T0 which is on GPIO 4.

void setup() {
  Serial.begin(115200);
  delay(1000); // give me time to bring up serial monitor
  Serial.println("ESP32 Touch Test");
}

void loop() {
  Serial.println(touchRead(4));  // get value of Touch 0 pin = GPIO 4
  delay(1000);
}
```

3. Buka serial monitor untuk melihat raw data. Ubah tampilan serial monitor menjadi Serial Plotter pada menu **Tools > Serial Plotter**.
4. Sentuh ujung kabel jumper dan amati yang terjadi, kemudian dokumentasikan hasilnya.
5. Buatlah rangkaian seperti gambar berikut ini.

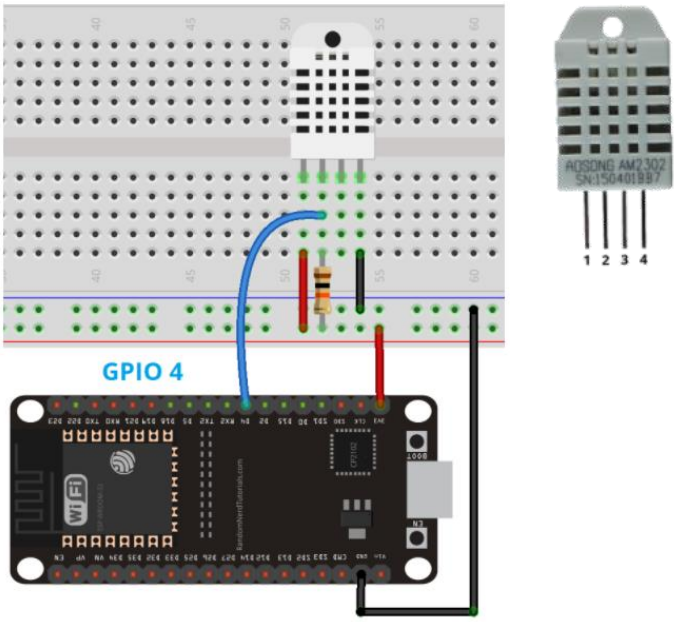


6. Buatlah program agar LED menyala ketika sensor disentuh, dan LED akan mati ketika sensor tidak disentuh.
7. Buatlah program agar ketika sensor disentuh, LED menyala Blink.

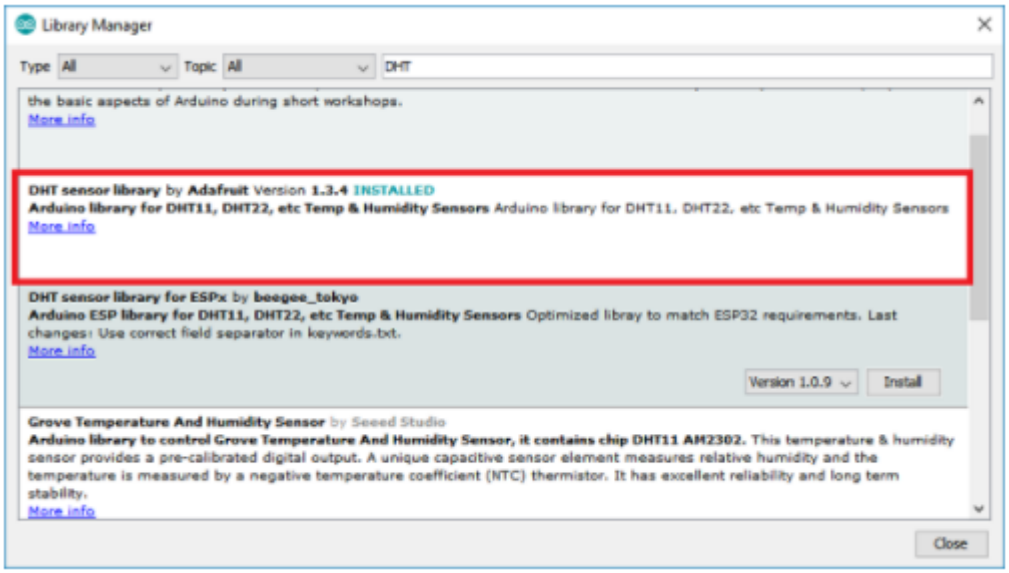
8. Buatlah program agar ketika LED menyala, maka pada Serial Monitor akan menampilkan angka yang akan bertambah setiap kali sensor disentuh.
9. Tambahkan 2 LED sehingga pada rangkaian terdapat 3 LED. Buatlah program agar ketika sensor disentuh, LED menyala menjadi running LED. Nyala running LED tersebut adalah bergerak dari kiri ke kanan, kemudian kanan ke kiri secara kontinyu.

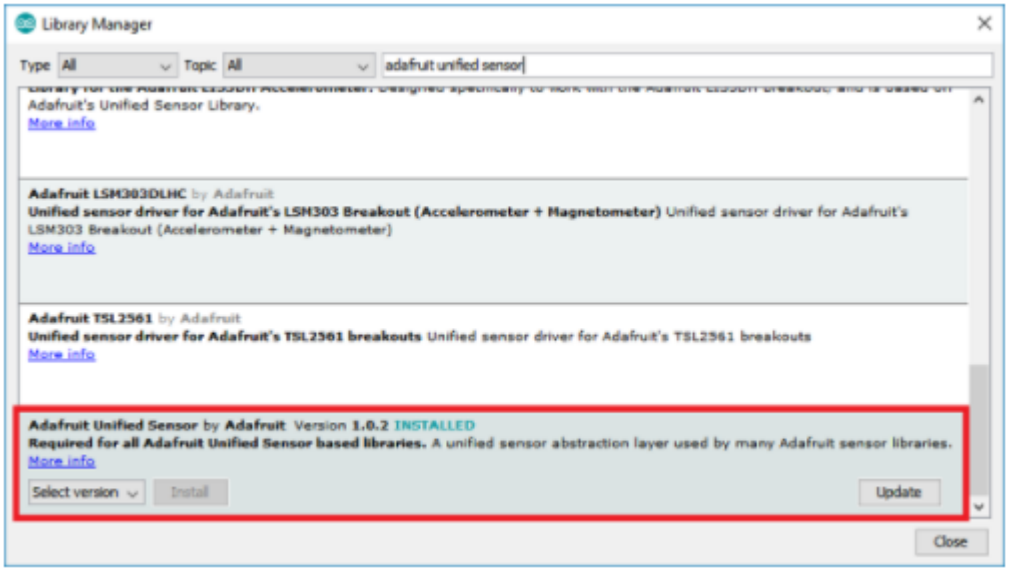
B. Mengakses Sensor DHT 11 (Single Wire / BUS)

1. Buatlah rangkaian seperti pada Gambar di bawah ini.



2. Install library sensor DHT 11 melalui **Sketch > Include Library > Manage Libraries**. Ketikkan **DHT** pada kolom pencarian, pilih library yang akan diinstall seperti pada Gambar berikut ini. Kemudian install juga **Adafruit Unified Sensor** menggunakan cara yang sama.





3. Buatlah program seperti pada script di bawah ini untuk mengakses sensor DHT11. Kemudian upload program tersebut pada ESP32 dan dokumentasikan hasilnya.

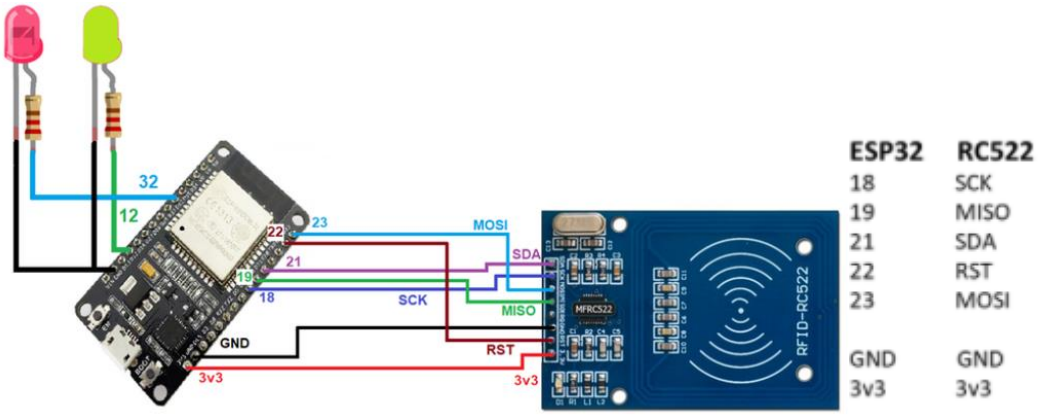
```
// REQUIRES the following Arduino libraries:
// - DHT Sensor Library: https://github.com/adafruit/DHT-sensor-library
// - Adafruit Unified Sensor Lib: https://github.com/adafruit/Adafruit_Sensor

#include "DHT.h"

#define DHTPIN 4     // Digital pin connected to the DHT sensor

// Uncomment whatever type you're using!
#define DHTTYPE DHT11   // DHT 11
//#define DHTTYPE DHT22   // DHT 22  (AM2302), AM2321
//#define DHTTYPE DHT21   // DHT 21 (AM2301)
```

```
DHT dht(DHTPIN, DHTTYPE);

void setup() {
 Serial.begin(9600);
 Serial.println(F("DHT11 Embedded System Test!"));

 dht.begin();
}

void loop() {
 // Wait a few seconds between measurements.
 delay(2000);

 // Reading temperature or humidity takes about 250 milliseconds!
 // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
 float h = dht.readHumidity();
 // Read temperature as Celsius (the default)
 float t = dht.readTemperature();
 // Read temperature as Fahrenheit (isFahrenheit = true)
 float f = dht.readTemperature(true);

 // Check if any reads failed and exit early (to try again).
 if (isnan(h) || isnan(t) || isnan(f)) {
  Serial.println(F("Failed to read from DHT sensor!"));
  return;
 }

 // Compute heat index in Fahrenheit (the default)
 float hif = dht.computeHeatIndex(f, h);
 // Compute heat index in Celsius (isFahreheit = false)
 float hic = dht.computeHeatIndex(t, h, false);

 Serial.print(F("Humidity: "));
 Serial.print(h);
 Serial.print(F("%  Temperature: "));
 Serial.print(t);
 Serial.print(F("°C "));
 Serial.print(f);
 Serial.print(F("°F  Heat index: "));
 Serial.print(hic);
 Serial.print(F("°C "));
 Serial.print(hif);
 Serial.println(F("°F"));
}
```

4. Buatlah program agar ketika suhu rungan mencapai 30 derajat celcius, maka ESP32 akan menyalakan LED Merah dan buzzer secara beep (blink). Apabila suhu dibawah 30 derajat, ESP32 akan mematikan buzzer dan menyalakan LED berbentuk running LED seperti pada **Percobaan A**. Kemudian dokumentasikan hasilnya.

C. Mengakses Sensor RFID (SPI Communication)

1. Buatlah rangkaian seperti pada gambar di bawah ini.



2. Install Library MFRC522 dari Library Manager.
3. Kemudian buatlah program seperti pada script berikut ini.

```
#include <SPI.h>
#include <MFRC522.h>

#define SS_PIN    21  // ESP32 pin GIO21
#define RST_PIN   22 // ESP32 pin GIO22

MFRC522 rfid(SS_PIN, RST_PIN);

byte keyTagUID[4] = {0xFF, 0xFF, 0xFF, 0xFF};

void setup() {
  Serial.begin(9600);
  SPI.begin(); // init SPI bus
  rfid.PCD_Init(); // init MFRC522

  Serial.println("Tap RFID/NFC Tag on reader");
}

void loop() {
  if (rfid.PICC_IsNewCardPresent()) { // new tag is available
    if (rfid.PICC_ReadCardSerial()) { // NUID has been readed
      MFRC522::PICC_Type piccType = rfid.PICC_GetType(rfid.uid.sak);

      if (rfid.uid.uidByte[0] == keyTagUID[0] &&
          rfid.uid.uidByte[1] == keyTagUID[1] &&
          rfid.uid.uidByte[2] == keyTagUID[2] &&
          rfid.uid.uidByte[3] == keyTagUID[3] ) {
        Serial.println("Access is granted");
      }
      else
      {
        Serial.print("Access denied for user with UID:");
        for (int i = 0; i < rfid.uid.size; i++) {
```

```
      Serial.print(rfid.uid.uidByte[i] < 0x10 ? " 0" : " ");
      Serial.print(rfid.uid.uidByte[i], HEX);
     }
     Serial.println();
    }

    rfid.PICC_HaltA(); // halt PICC
    rfid.PCD_StopCrypto1(); // stop encryption on PCD
   }
  }
}
```

4. Dekatkan kartu atau Tag RFID ke RFID Reader. Amati dan analisis cara kerja programnya.
5. Buatlah program agar Tag RFID yang terbaca sebelumya dapat digunakan untuk hak akses. Apabila Tag RFID didekatkan pada Reader, maka LED Hijau akan menyala, servo akan bergerak ke kanan (lalu kembali ke posisi semula setelah 3 detik) dan di Serial Monitor akan tertampil pesan “Akses Diterima, Silahkan Masuk”. Apabila Tag RFID tidak dikenali, maka LED Merah akan menyala, servo tidak bergerak dan di Serial Monitor akan tertampil pesan “Akses Ditolak”. Gunakan Tag RFID lain untuk mencoba.
6. Amati yang terjadi, analisis dan dokumentasikan hasilnya.

7. PERTANYAAN DAN TUGAS