ESP32 BASIC BOARD PRACTICE

A. Objective

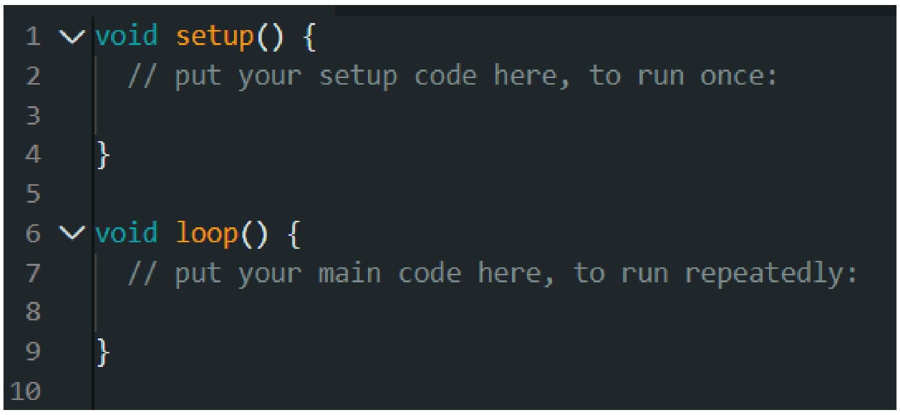
1. Understand the basics of microcontrollers.
2. Using the Serial Monitor feature in the Arduino IDE.
3. Apply basic programming on microcontrollers to control simple electronic devices.
4. Analyze how the control of electronic devices can be applied in daily life.

B. Basic Theory

Microcontrollers have a number of GPIO pins that can be selected and functioned as inputs or outputs. In this chapter, we will practice accessing GPIO pins as outputs to control LEDs. Use the Serial Monitor feature in the Arduino IDE to display virtual outputs with programmed messages and control LEDs via Serial Monitor.

The basic programming structure of Arduino is as shown below. There are two main differences between the two, as follows.

- `void setup()` used for initialization (pins, communication, sensors), here the program is executed only once.
- `void loop()` used to write the main logic of the program, here the program is executed repeatedly as long as the device is active.



```
void setup() {
  // put your setup code here, to run once:

}

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

}

```

C. Tools & Materials

a. ESP32
b. LED 5mm

c. Breadboard

d. Jumper wire

e. Laptop installed Arduino IDE

f. USB Type C cable

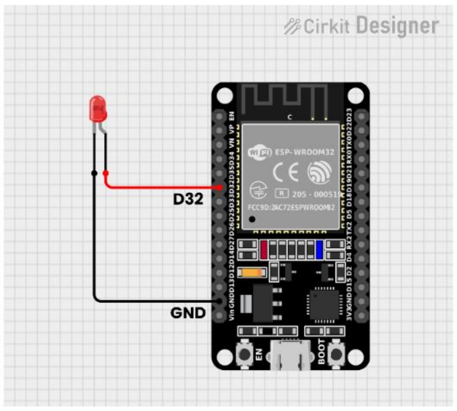
D. Work Safety

1. Conduct, or carry out the practicum in a clean and dry place.
2. Use a laptop, or computer properly, and correctly.
3. Do not touch the electrified parts of the lab kit directly.
4. Ensure that the power supply of the lab kit is off during the installation or assembly of the lab kit.
5. Always pay attention to the lab kit ports. Do not confuse, reverse, or misplace them in the installation, or assembly of the lab kit.
6. Use the components in the lab kit as intended.
7. If a short circuit occurs, immediately turn off the power supply to the lab kit.
8. Follow the lab steps as stated in the work steps and pay attention to safety.

E. Work Steps

5.1 *Blinking* LED

a. Create a circuit as shown below. Make sure to use GPIO pins that support output.

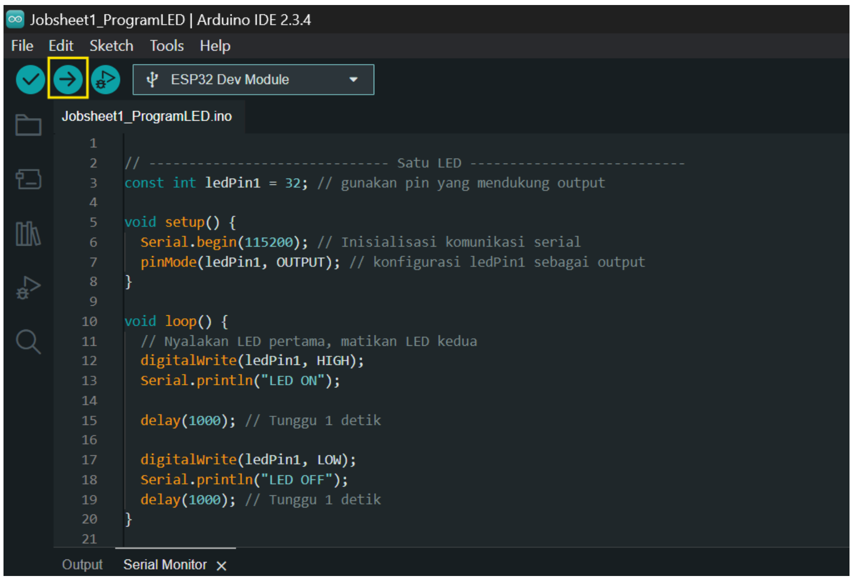


b. Enter the following code.

```
const int ledPin1 = 32; // inisialisasi led di pin GPIO 32

void setup() {
  Serial.begin(115200); // inisialisasi komunikasi serial
  pinMode(ledPin1, OUTPUT); // konfigurasi ledPin1 sebagai
output
}

void loop() {
  // Nyalakan LED
  digitalWrite(ledPin1, HIGH);
  Serial.println("LED ON"); // Menampilkan pesan
  delay(1000); // Tunggu 1 detik

  // Matikan LED
  digitalWrite(ledPin1, LOW);
  Serial.println("LED OFF"); // Menampilkan pesan
  delay(1000); // Tunggu 1 detik
}
```

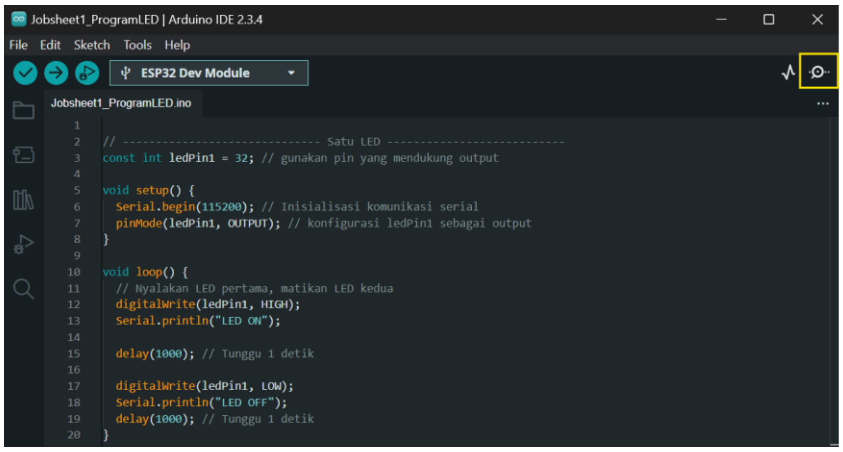
c. Connect the ESP32 to a laptop, before Verify or Compile code make sure the ESP32 Board is connected in Arduino IDE.

d. If the compile process has no errors, continue to upload the program as shown below.


Jobsheet1_ProgramLED | Arduino IDE 2.3.4
File Edit Sketch Tools Help
ESP32 Dev Module
Jobsheet1_ProgramLED.ino
Output
Serial Monitor

```

// ---------------------------- Satu LED -------------------------
const int ledPin1 = 32; // gunakan pin yang mendukung output

void setup() {
  Serial.begin(115200); // Inisialisasi komunikasi serial
  pinMode(ledPin1, OUTPUT); // konfigurasi ledPin1 sebagai output
}

void loop() {
  // Nyalakan LED pertama, matikan LED kedua
  digitalWrite(ledPin1, HIGH);
  Serial.println("LED ON");

  delay(1000); // Tunggu 1 detik

  digitalWrite(ledPin1, LOW);
  Serial.println("LED OFF");
  delay(1000); // Tunggu 1 detik
}

```

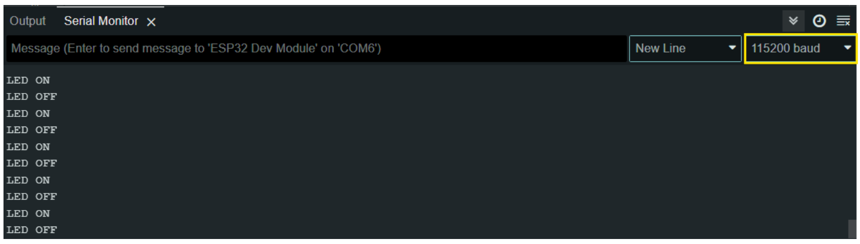
e. The upload process is successful, we can see the results of the program through the Serial Monitor window, the top right icon.


Jobsheet1_ProgramLED | Arduino IDE 2.3.4
File Edit Sketch Tools Help
Jobsheet1_ProgramLED.ino

ESP32 Dev Module

```
// ----------------------------- Satu LED --------------------------
const int ledPin1 = 32; // gunakan pin yang mendukung output

void setup() {
  Serial.begin(115200); // Inisialisasi komunikasi serial
  pinMode(ledPin1, OUTPUT); // konfigurasi ledPin1 sebagai output
}

void loop() {
  // Nyalakan LED pertama, matikan LED kedua
  digitalWrite(ledPin1, HIGH);
  Serial.println("LED ON");

  delay(1000); // Tunggu 1 detik

  digitalWrite(ledPin1, LOW);
  Serial.println("LED OFF");
  delay(1000); // Tunggu 1 detik
}
```

f. The Serial Monitor window requires synchronizing the baud rate for serial communication. Change the baud rate to 115200 in order to display the message as programmed.



Message (Enter to send message to 'ESP32 Dev Module' on 'COM6')

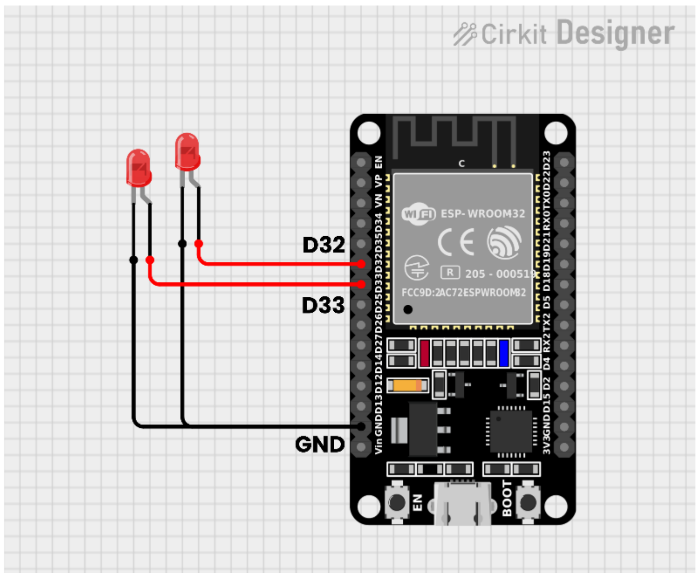
New Line

115200 baud

```
LED ON
LED OFF
LED ON
LED OFF
LED ON
LED OFF
LED ON
LED OFF
LED ON
LED OFF
```

5.2 *Blinking* Two LED

a. Create a circuit as shown below. Make sure to use GPIO pins that support output.



b. Enter the following code.

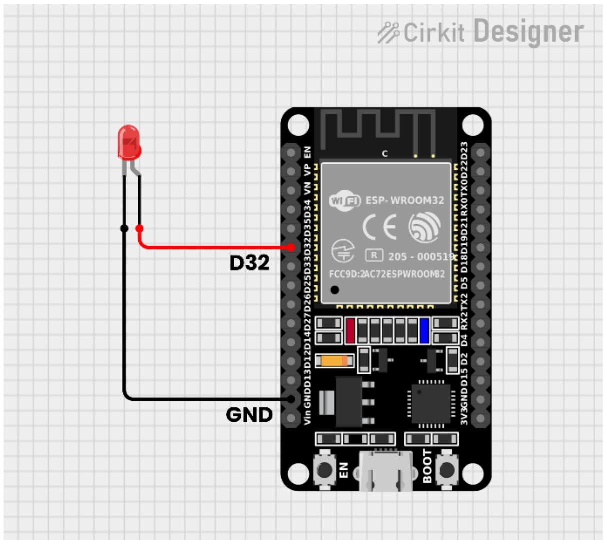
```
const int ledPin1 = 33; // inisialisasi LED di pin GPIO 33
const int ledPin2 = 32; // inisialisasi LED di pin GPIO 32

void setup() {
  pinMode(ledPin1, OUTPUT); // led 1 sebagai output
  pinMode(ledPin2, OUTPUT); // led 2 sebagai output
}

void loop() {
  // Nyalakan LED pertama, matikan LED kedua
  digitalWrite(ledPin1, HIGH);
  digitalWrite(ledPin2, LOW);
  delay(1000); // Tunggu 1 detik

  // Matikan LED pertama, nyalakan LED kedua
  digitalWrite(ledPin1, LOW);
  digitalWrite(ledPin2, HIGH);
  delay(1000); // Tunggu 1 detik
}
```

c. Connect the ESP32 to a laptop, before Verify or Compile code make sure the ESP32 Board is connected in Arduino IDE.

d. If the compile process is complete and there are no errors, then upload the program, then wait for it to finish.

5.3 LED Control from Serial Monitor

a. Create a circuit as shown below. Make sure to use GPIO pins that support output.



b. Enter the following code.

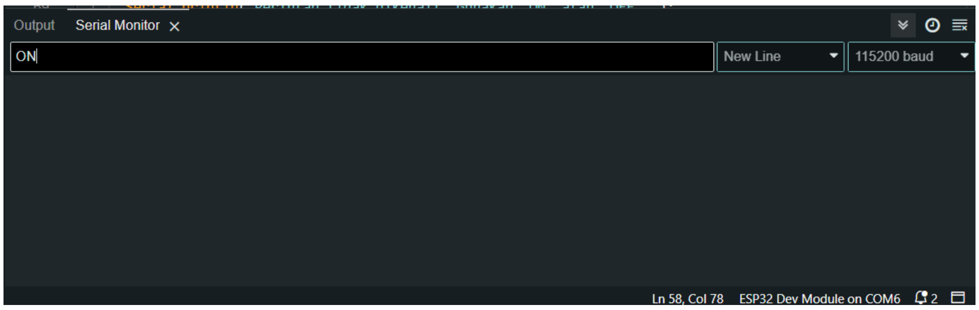
```
const int ledPin1 = 32; // Inisialisasi LED di pin GPIO 32

void setup() {
  pinMode(ledPin1, OUTPUT); // Konfigurasi LED sebagai output
  Serial.begin(115200); // Inisialisasi komunikasi serial
  Serial.println("Ketik 'ON' untuk menyalakan LED, 'OFF' untuk
mematikan LED.");
}

void loop() {
  if (Serial.available() > 0) { // Jika ada data dari Serial
Monitor
    String command = Serial.readStringUntil('\n'); // Baca input
hingga enter
    command.trim(); // Hapus karakter spasi atau newline

    if (command == "ON") {
      digitalWrite(ledPin1, HIGH); // Nyalakan LED
      Serial.println("LED ON");
    }
    else if (command == "OFF") {
      digitalWrite(ledPin1, LOW); // Matikan LED
      Serial.println("LED OFF");
    }
    else {
      Serial.println("Perintah tidak dikenali. Gunakan 'ON' atau
'OFF'.");
    }
  }
}
```

c. Connect the ESP32 to a laptop, before Verify or Compile code make sure the ESP32 Board is connected in Arduino IDE.

d. If the compile process is complete and there are no errors, then upload the program, then wait for it to finish.

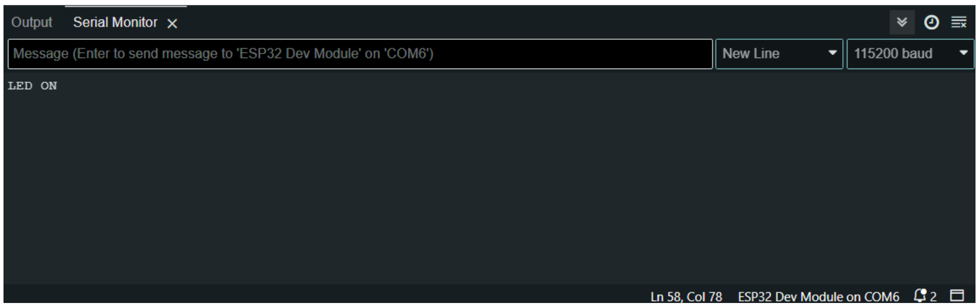
e. Enter the Serial Monitor, then change the baud rate as programmed.

f. Type the "ON" or "OFF" command as configured, to turn the LED on or off. After typing the command click enter.



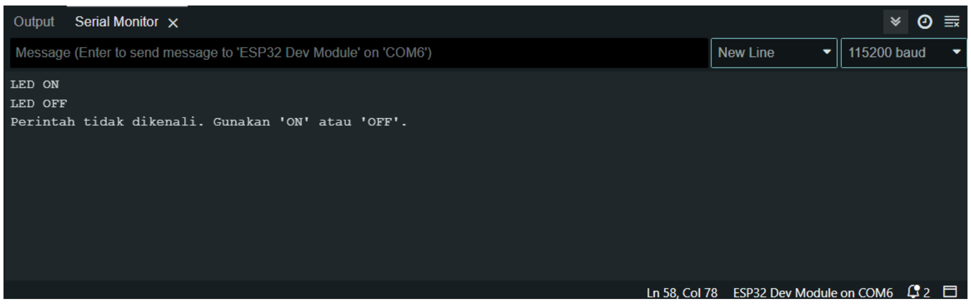
```
ON
```

g. When you have finished sending the command, the message "LED ON" will appear and the LED will light up. Vice versa when typing "OFF" it will turn off the LED and display the message "LED OFF".



```
LED ON
```

h. If there is a typo, the command cannot be executed, so the program will display an error message.



```
LED ON
LED OFF
Perintah tidak dikenali. Gunakan 'ON' atau 'OFF'.
```

F. Questions

1. Display the output based on the Work Step instructions! Add analysis and working principle!
2. Create a program to control an LED with an on or off duration that can be set via Serial Monitor!
3. Make a program like the traffic concept using 3 LEDs, for example:
   - LED 1 on for 3 seconds, LEDs 2 and 3 off.
   - LED 2 on for 3 seconds, LEDs 1 and 3 off.
   - LED 3 on for 3 seconds, LEDs 1 and 2 off.
4. Document the results on the Practicum Worksheet. Include a video of the practicum results and the source code!
5. Upload the Practicum Report worksheet (in PDF format) and video documentation on Google Drive.
   https://drive.google.com/drive/folders/1I2Dsi14Gmf9mdjtKauldQ7rw9M7A9IMr?usp=drive_link
6. Terms of drive collection as follows:
   - Use your UNNES student account.
   - Create a folder with the format "NIM_Full Name".
   - Upload Practicum Report worksheets and video documentation.