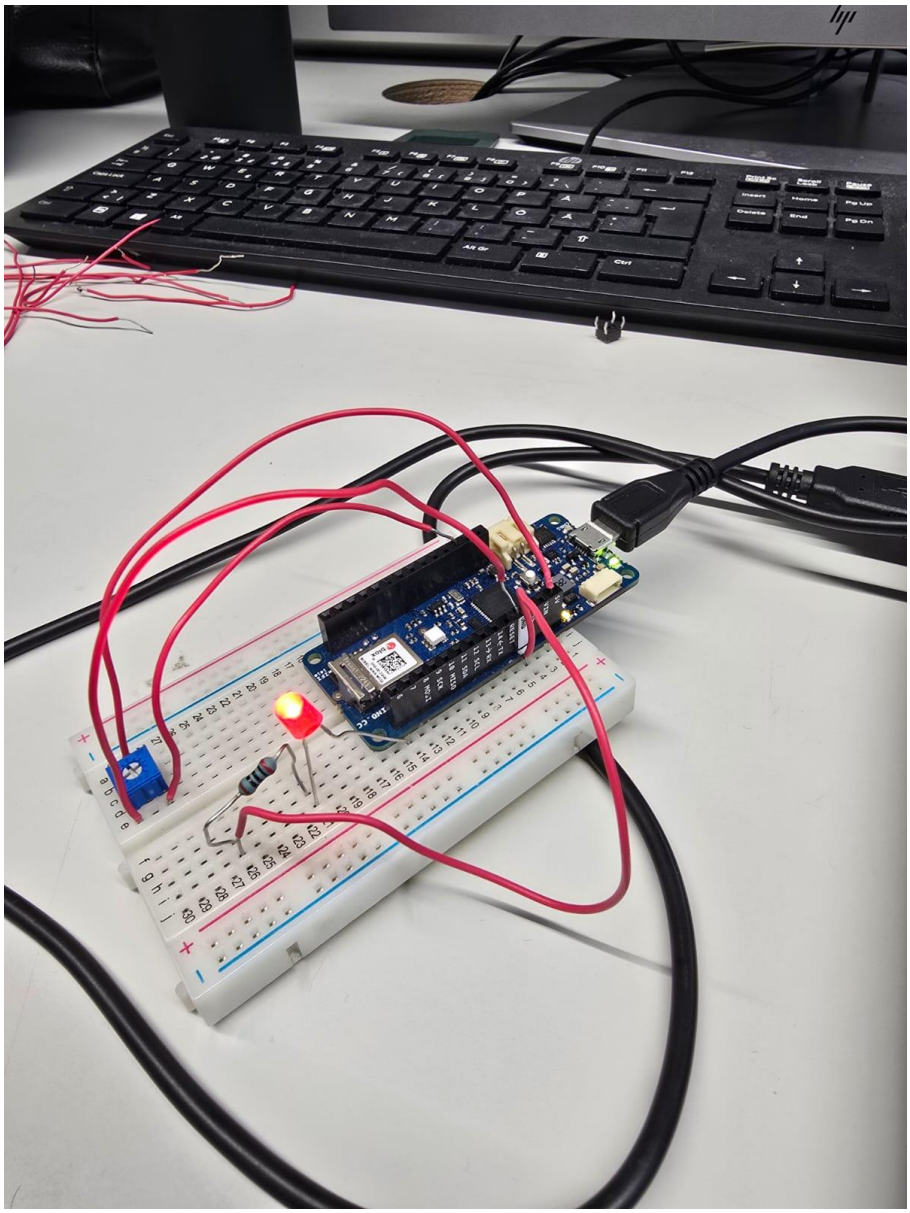
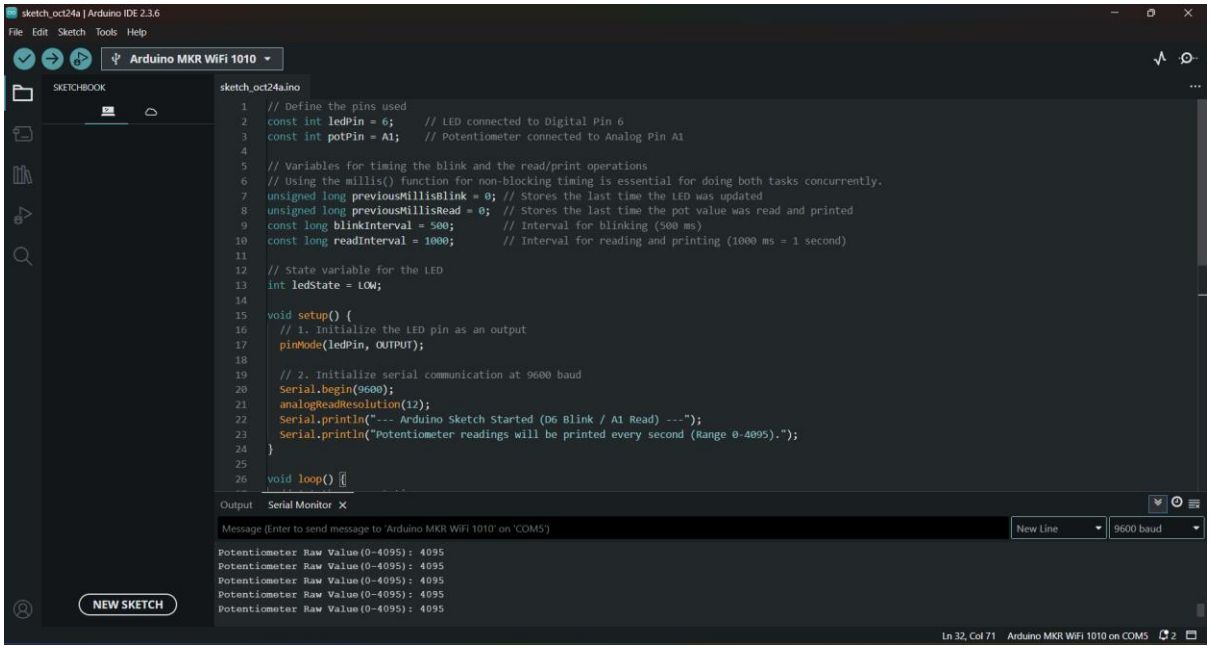
Exercise1:





Exercise 2:

the code:

```
void setup() {
  Serial.begin(9600);


  delay(1000);
  Serial.println("WiFi Network Scanner Starting...");
}


void loop() {


  if (WiFi.status() == WL_NO_MODULE) {
    Serial.println(".");


    while (true);
  }
```

```
Serial.println("\n 📡 Scanning available networks...");

int numNetworks = WiFi.scanNetworks();

if (numNetworks == 0) {
  Serial.println("No networks found.");
} else if (numNetworks > 0) {
  Serial.print(" ");
  Serial.print(numNetworks);
  Serial.println(" networks found:");


  for (int i = 0; i < numNetworks; i++) {
    // Print network details
    Serial.print(" [");
    Serial.print(i + 1);
    Serial.print("] ");
    Serial.print("SSID: ");
    Serial.print(WiFi.SSID(i));
    Serial.print(" | RSSI: ");
    Serial.print(WiFi.RSSI(i));
    Serial.print(" dBm | Encryption: ");
    Serial.println(WiFi.encryptionType(i));
  }
} else {

  Serial.println("Error during network scan.");
}
```

```
  Serial.println("\n--- Next scan in 30 seconds... ---");
  delay(30000);
}
```

Exercise 3:

the code:

```
// Define the variables to hold the two numbers and the operator
float num1 = 0.0;
float num2 = 0.0;
char op = ' ';

void setup() {
  Serial.begin(9600);

  while (!Serial);

  Serial.println("--- Basic Calculator Ready ---");
}

void loop() {

  Serial.println("\n(A) Enter calculation (e.g., 8 * 5 or 1 + 4):");


  while (Serial.available() == 0) {
```

```
}

String input = Serial.readStringUntil('\n');
input.trim();

int matched_items = sscanf(input.c_str(), "%f %c %f", &num1, &op, &num2);

if (matched_items == 3) {
 float result;
 bool valid_operation = true;

 switch (op) {
  case '+':
   result = num1 + num2;
   break;
  case '-':
   result = num1 - num2;
   break;
  case '*':
   result = num1 * num2;
   break;
  case '/':

   if (num2 != 0) {
    result = num1 / num2;
   } else {
    Serial.println("Error: Division by zero is not allowed.");
```

```
        valid_operation = false;
      }
      break;
    default:

      Serial.print("Error: Unknown operator '");
      Serial.print(op);
      Serial.println("'.");
      valid_operation = false;
      break;
  }

  if (valid_operation) {
    Serial.print("Result: ");
    Serial.print(num1);
    Serial.print(" ");
    Serial.print(op);
    Serial.print(" ");
    Serial.print(num2);
    Serial.print(" = ");
    Serial.println(result, 2);
  }
 } else {

  Serial.println("Error: Invalid input format. Please use 'number operator number' (e.g., 8 * 5).");
 }
```

```
}
```

Exercise 4:

the code:

```
#include "sam.h"


#define NEW_LED_PIN 10
#define LED_GROUP 1


#define INPUT_PIN 10


void setup() {

  PM->APBBMASK.reg |= PM_APBBMASK_PORT;


  PORT->Group[LED_GROUP].DIRSET.reg = (1 << NEW_LED_PIN);


  PORT->Group[0].DIRCLR.reg = (1 << INPUT_PIN);
}


void loop() {
  PORT->Group[LED_GROUP].OUTSET.reg = (1 << NEW_LED_PIN);


  delay(1000);


  PORT->Group[LED_GROUP].OUTCLR.reg = (1 << NEW_LED_PIN);


  delay(1000);
```

```
}
```

Lab Procedure and Results Summary

Initial Verification

The initial code successfully demonstrated **bare-metal GPIO control** by toggling the built-in LED (PA20, Group 0) at a **1 Hz frequency** (500 ms ON/OFF). This confirmed the fundamental requirement of enabling the peripheral clock via the **Power Manager (PM)** and setting the pin direction using the **PORT peripheral registers**.

Analysis of Code Modifications

The objective was to re-target the I/O operation from pin PA20 to pin PB10 and change the toggling rate.

- **Pin Redefinition:** The target was switched from **PA20** (Port Group 0, bit 20) to **PB10** (Port Group 1, bit 10). This required updating the code definitions to reference **Port Group 1** and **bit 10**. The SAMD21 groups pins into Port A (Group 0) and Port B (Group 1).
- **Configuration (`setup()`):**
    - The **Power Manager clock enable** line (`PM->APBBMASK.reg |= PM_APBBMASK_PORT;`) remained **unchanged**, as it enables the clock for the entire PORT peripheral, covering both Group 0 and Group 1.
    - The pin direction was set by accessing **Group 1**'s register: `PORT->Group[1].DIRSET.reg = (1 << 10);`. Using **`DIRSET`** ensured that bit 10 was set to '1' (Output) **without altering** the direction status of any other pin on Port B.
- **Operation (`loop()`):**

- The output state was controlled by writing to the **Group 1** registers. **`OUTSET`** was used to set the pin HIGH (ON), and **`OUTCLR`** was used to clear the pin LOW (OFF).
- The **delay** was increased from 500 ms to **1000 ms** for both the ON and OFF states, resulting in a **0.5 Hz** overall toggling frequency (a full cycle every two seconds).

Conclusion

By correctly updating the **CMSIS structure pointers** to target the new **Port Group (1)** and the corresponding **bit position (10)**, the program successfully migrated the GPIO functionality from PA20 to PB10. This process validates the ability to achieve **direct and efficient control** over the SAMD21's hardware registers, offering granular control over individual GPIO pins.

Github link:

https://github.com/Djaber-DB/cyber-physical-systems