FUNDAMENTAL OF EMBEDDED SYSTEM FINAL PROJECT REPORT

DEPARTMENT OF ELECTRICAL ENGINEERING

UNIVERSITAS INDONESIA

LUMA – Light Utilization Monitoring for Agriculture

GROUP 17

Andhika Fadhlan Wijanarko	**2306267164**
Ekananda Zhafif Dean	**2306264420**
Muhammad Iqbal Alfajri	**2306250705**
Reyhan Ahnaf Deannova	**2306267100**
Zhafira Zahra Alfarisy	**2306250636**

PREFACE

Segala puji dan syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa atas limpahan rahmat dan karunia-Nya sehingga kami dapat menyelesaikan laporan proyek akhir Praktikum Sistem Embedded ini dengan baik dan tepat waktu. Laporan ini disusun sebagai bagian dari pemenuhan tugas akhir dalam praktikum yang telah kami jalani selama satu semester.

Proyek akhir yang kami kerjakan mengangkat topik *LUMA – Light Utilization Monitoring for Agriculture*, sebuah sistem monitoring dan pengaturan intensitas cahaya yang diterima tanaman guna mendukung proses fotosintesis secara optimal. Sistem ini memanfaatkan Arduino sebagai pengendali utama dan sensor LDR untuk mendeteksi tingkat pencahayaan. Apabila intensitas cahaya melebihi *threshold* yang ditentukan, Arduino akan mengaktifkan motor (servo atau DC) untuk menutup box pelindung tanaman. Data intensitas cahaya serta status box ditampilkan melalui LCD yang terhubung menggunakan antarmuka I2C. Pencatatan waktu dilakukan oleh modul RTC DS3231, sementara pengukuran intensitas cahaya berjalan secara periodik dengan bantuan Timer, dan Interrupt digunakan untuk menangani kejadian seperti penekanan tombol secara real-time. Sistem ini dirancang untuk memastikan tanaman memperoleh pencahayaan sesuai dengan kebutuhannya.

Selama proses penyusunan laporan ini, kami mendapatkan banyak dukungan dan bantuan dari berbagai pihak. Oleh karena itu, kami mengucapkan terima kasih yang sebesar-besarnya kepada para asisten laboratorium dan rekan-rekan tim atas kerja sama, semangat, dan kontribusi yang telah diberikan demi kelancaran proyek ini.

Kami menyadari bahwa laporan ini masih memiliki keterbatasan dan mungkin terdapat kekurangan dalam penyajiannya. Untuk itu, kami terbuka terhadap segala kritik dan saran yang membangun demi perbaikan dan peningkatan kualitas laporan di masa yang akan datang.

Akhir kata, kami berharap laporan ini dapat memberikan manfaat bagi para pembaca serta turut berkontribusi dalam pengembangan ilmu pengetahuan, khususnya di bidang sistem embedded. Semoga Tuhan Yang Maha Esa senantiasa memberikan bimbingan dan keberkahan kepada kita semua.

Depok, 14 May 2025

Group 17

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 PROBLEM STATEMENT

Pencahayaan yang optimal merupakan salah satu faktor penting dalam mendukung proses fotosintesis dan pertumbuhan tanaman secara maksimal. Namun, kondisi pencahayaan alami yang tidak menentu seringkali menjadi kendala, karena tanaman bisa saja menerima cahaya berlebih atau justru kekurangan cahaya pada waktu-waktu tertentu. Pengaturan intensitas cahaya secara manual pun kurang efisien dan tidak selalu responsif terhadap perubahan kondisi lingkungan. Oleh karena itu, dibutuhkan sebuah sistem otomatis yang mampu memantau dan mengatur intensitas cahaya secara real-time agar tanaman tetap berada dalam kondisi ideal.

Proyek *LUMA – Light Utilization Monitoring for Agriculture* hadir sebagai solusi berbasis Arduino yang menggabungkan sensor LDR untuk pendeteksian cahaya, motor (servo atau DC) untuk mengatur buka tutup box pelindung tanaman, serta modul RTC DS3231 dan LCD I2C untuk pencatatan waktu dan tampilan data secara real-time. Sistem ini juga memanfaatkan Timer untuk pengukuran berkala dan Interrupt untuk menangani input pengguna seperti tombol. Dengan rancangan ini, LUMA diharapkan mampu menjadi sistem monitoring yang efektif, efisien, dan mudah diterapkan dalam praktik pertanian modern.

1.2 ACCEPTANCE CRITERIA

Kriteria yang menyatakan proyek berhasil adalah sebagai berikut

- Dapat membaca intensitas cahaya menggunakan sensor LDR
- Menampilkan status intensitas cahaya pada LCD 16x20
- Mengoperasikan motor servo menggunakan PWM
- Menggunakan data intensitas cahaya serta timer untuk mengatur buka tutup jendela LUMA
- Mewujudkan sinergi antara komponen untuk mendukung laju fotosintesis tanaman

1.3 ROLES AND RESPONSIBILITIES

The roles and responsibilities assigned to the group members are as follows:

Roles	Responsibilities	Person
Anggota	- Membeli 9x resistor 220 ohm, 3x led merah, 3x led hijau, dan 3x led biru, 3x servo, 1 modul pwm 16bit, 1 lcd 20x16, 2x kabel usb3 ke micro, 1 kabel usb3 ke usb type b, 2x push button, dan 1 LDR. - Menyediakan 1 breadboard, arduino sebanyak 3 buah, kabel-kabel jumper, dan potensiometer untuk tes program. - Merangkai rangkaian sesuai proteus dari awal sampai akhir kecuali untuk pergantian LDR usang. - melakukan tes fungsi program dengan potensiometer. - Membuat mekanisme link servo motor versi awal.	Andhika Fadhlan Wijanarko
Role 2	- Membantu Perbaikan Kode :	Ekananda Zhafif Dean

	- UpdateDisplay tidak menampilkan informasi lengkap (nilai cahaya dan status) - Alamat I/O register tidak benar - TWI (I2C) tidak memiliki deteksi error - Beberapa bagian implementasi servo tidak optimal - Membuat Penjelasan Proyek yang dibuat di readme github.	
Role 3	- Membeli alat2 yang akan digunakan dalam proyek - Merakit rangkaian asli dengan panduan rangkaian proteus -	Muhammad Iqbal Alfajri
Role 4	- Mengajukan Ide - Membuat Rangkaian Proteus - Membuat Keseluruhan Kode	Reyhan Ahnaf Deannova

	- Membantu mengoreksi rangkaian	
Role 5	- Membuat Laporan - Membuat Timeline - Membuat Flowchart - Membuat PPT	Zhafira Zahra Alfarisy

Table 1. Roles and Responsibilities

1.5 TIMELINE AND MILESTONES

Task	May 2025													
	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Brainstorming Ide dan Konsep														
Membuat Kode														
Design Hardware dan Schematic														
Menggabungkan Kode dengan Hardware Testing														
Assembly Produk Final dan Pengumpulan														

Fig.1 *Gantt Chart*

CHAPTER 2

IMPLEMENTATION

2.1 EQUIPMENT

Komponen-komponen yang dibutuhkan dalam proyek ini adalah:

- Arduino Uno
- LCD I2C 16 x 20
- LED
- Button
- Photoresistor
- RTC DS3231
- LDR
- Servo PCA9685
- Resistor
- BreadBoard
- Jumper Cable
- Batre 9V

2.2 IMPLEMENTATION

2.2.1 HARDWARE DESIGN AND SCHEMATIC

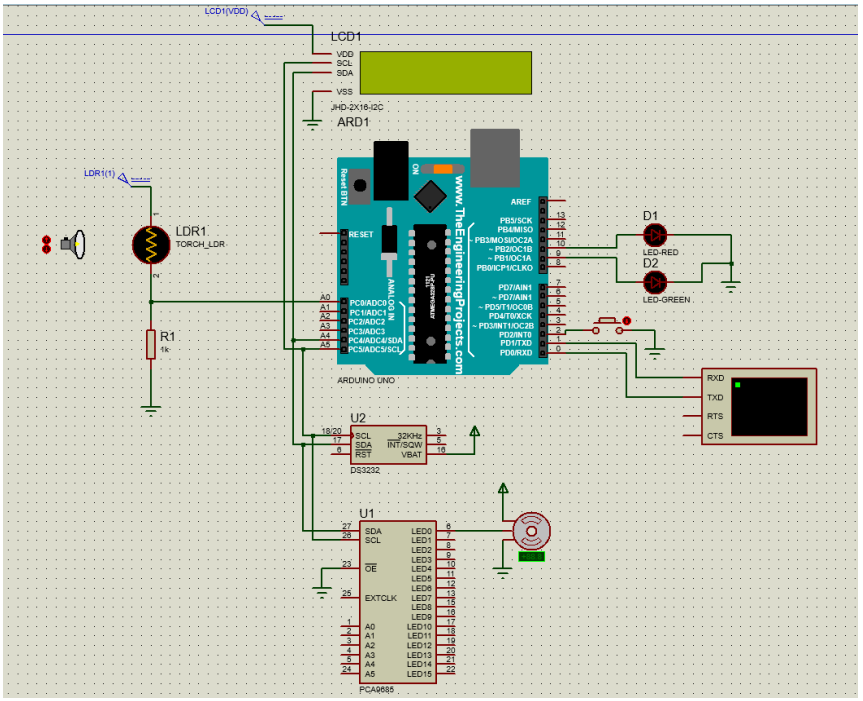


Fig.2 *Proteus Design*

Untuk membantu kelompok merakit alat secara virtual sebelum merakit komponen asli, proses pembuatan alat dimulai dengan merancang plan atau prototipe menggunakan software Proteus. Tujuannya adalah untuk mengurangi kemungkinan kerusakan pada komponen asli yang dapat terjadi karena kesalahan rangkaian atau integrasi yang tidak sempurna.

Antara lain, komponen yang diperlukan untuk mendesain dan merangkai skema alat, serta fungsinya dalam rangkaian, adalah:

1) Arduino UNO
   Berfungsi sebagai pusat kendali utama sistem. Arduino membaca data dari sensor LDR melalui pin analog (A0), menampilkan informasi pada LCD I2C, mengontrol servo motor melalui driver PCA9685, dan membaca waktu dari modul RTC DS3231. Selain itu, Arduino juga menangani interrupt dari tombol manual untuk mengubah status kontrol secara manual.

2) Sensor LDR
   Sensor LDR digunakan untuk mendeteksi intensitas cahaya di lingkungan sekitar. Nilai resistansi LDR berubah sesuai dengan tingkat cahaya yang diterima, dan data ini dibaca oleh Arduino melalui pin A0 menggunakan ADC. Nilai ini digunakan untuk memutuskan apakah penutup tanaman harus dibuka atau ditutup.

3) Modul LCD I2C
   Digunakan untuk menampilkan informasi sistem seperti: Intensitas cahaya saat ini, waktu aktual dari RTC, status penutup tanaman (terbuka/tertutup) LCD ini terhubung ke Arduino melalui jalur I2C (SDA ke A4, SCL ke A5), mengurangi penggunaan pin digital.

4) RTC DS3231
   Modul waktu nyata yang memberikan waktu dan tanggal akurat kepada Arduino. Komunikasi menggunakan I2C (SDA/SCL). RTC penting untuk mencatat waktu pembukaan dan penutupan kotak berdasarkan siklus cahaya alami tanaman.

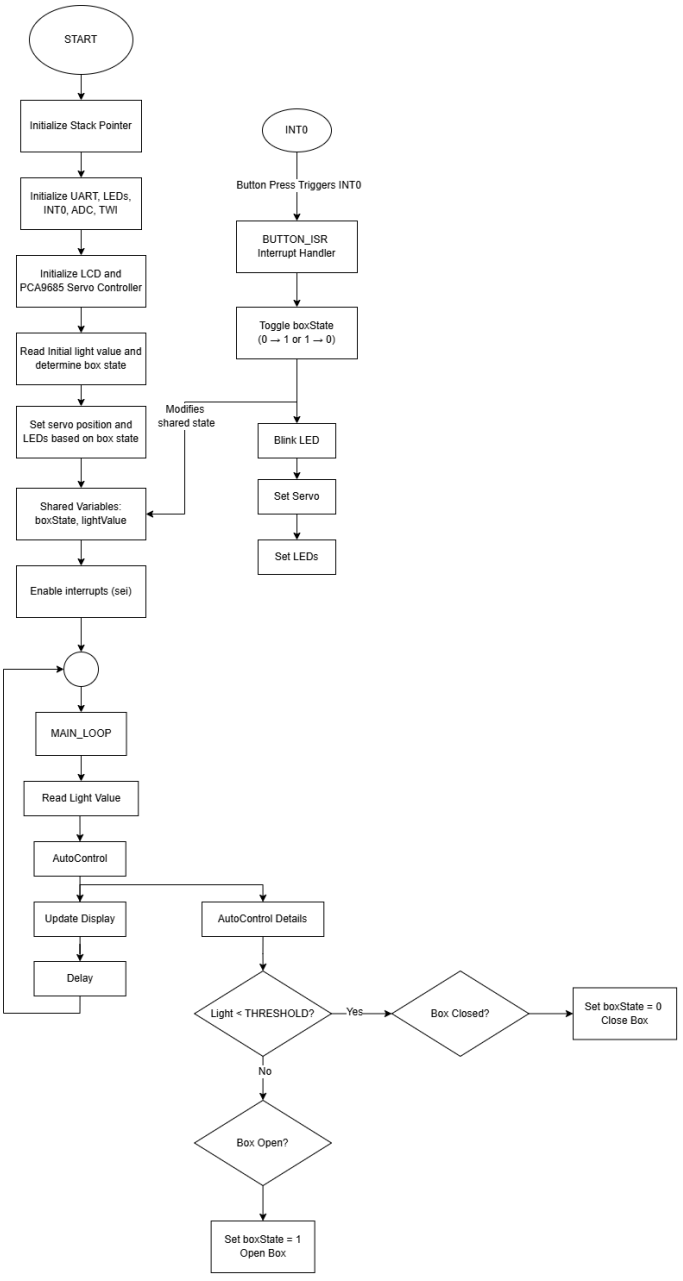
5) Servo PCA9685
   Ini adalah modul PWM 16-channel yang dikontrol melalui I2C yang berfungsi untuk mengatur pergerakan servo motor yang membuka dan menutup penutup tanaman. Karena memungkinkan banyak output PWM dengan dua pin Arduino (SDA/SCL), modul ini sangat efektif.

6) LED Merah dan LED Hijau
   - LED Merah menyala saat penutup tanaman dalam kondisi tertutup (terlalu terang).
   - LED Hijau menyala saat penutup tanaman terbuka (terlalu gelap).

7) Push Button
   Push button terhubung ke pin digital (INT0 / pin 2) dan digunakan untuk mengubah status kontrol secara manual. Ketika ditekan, akan terjadi interrupt yang memicu perubahan status penutup dan mengabaikan mode otomatis untuk sementara waktu. Juga menyebabkan LED berkedip sebagai indikator transisi status.

2.2.2 SOFTWARE DEVELOPMENT

Untuk membuat penyusunan cara kerja alat lebih mudah, flowchart dibuat menggunakan website draw.io sebagai berikut:



Dengan demikian terciptalah dua buah kode yaitu pertama .ino dan kedua .S untuk Arduino.

.ino for interrupt

```
/*
 * LightMonitorSystem.ino
 */

#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <RTClib.h>
#include <Adafruit_PWMServoDriver.h>

// Objek global
LiquidCrystal_I2C lcd(0x27, 16, 2);
RTC_DS3231 rtc;
Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver();

// Pin definitions
#define BUTTON_PIN 2
#define SERVO_CHANNEL 0
#define SERVO_OPEN 150
#define SERVO_CLOSED 450

// Variabel global
volatile bool boxState = true;
volatile bool buttonPressed = false;

// Deklarasi fungsi Assembly
extern "C" {
  void setupPins();
  void setupADC();
  int readLightSensor();
  void setLEDs(bool greenOn, bool redOn);
  void blinkLEDTimes(int pin, int count);
  int getThreshold();
}

void setup() {
  Serial.begin(9600);
  Serial.println(F("Light Monitor System"));

  // Setup interrupt
  pinMode(BUTTON_PIN, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(BUTTON_PIN),
buttonInterrupt, FALLING);

  // I2C dan perangkat
  Wire.begin();
  lcd.init();
  lcd.backlight();

  if (!rtc.begin()) {
    Serial.println(F("RTC Error!"));
    while(1);
  }
```

```
  if (rtc.lostPower()) {
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  }

  pwm.begin();
  pwm.setPWMFreq(60);

  // Setup Assembly functions
  setupPins();
  setupADC();

  // Status awal
  int lightValue = readLightSensor();
  boxState = (lightValue <= getThreshold());
  controlServo(boxState);
  setLEDs(boxState, !boxState);

  lcd.clear();
  lcd.print("System Ready");
  delay(2000);
}

void buttonInterrupt() {
  static unsigned long lastTime = 0;
  unsigned long currentTime = millis();

  if (currentTime - lastTime > 200) {
    buttonPressed = true;
    setLEDs(false, true); // Red ON saat tombol ditekan
  }
  lastTime = currentTime;
}

void loop() {
  // Handle button press
  if (buttonPressed) {
    boxState = !boxState;

    // Blink LED sesuai status baru
    int blinkPin = boxState ? 9 : 10; // Green : Red
    blinkLEDTimes(blinkPin, 3);

    controlServo(boxState);
    setLEDs(boxState, !boxState);
    buttonPressed = false;

    Serial.print(F("Manual toggle: "));
    Serial.println(boxState ? F("Open") : F("Closed"));
  }

  // Auto control berdasarkan cahaya
  int lightValue = readLightSensor();
  int threshold = getThreshold();

  if (lightValue > threshold && boxState) {
    boxState = false;
    controlServo(false);
    setLEDs(false, true);
    Serial.println(F("Auto closed"));
  }
```

```
  else if (lightValue <= threshold && !boxState) {
    boxState = true;
    controlServo(true);
    setLEDs(true, false);
    Serial.println(F("Auto opened"));
  }

  // Update LCD
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(F("Light: "));
  lcd.print(lightValue);

  lcd.setCursor(0, 1);
  lcd.print(F("Box: "));
  lcd.print(boxState ? F("Open") : F("Closed"));

  // Tampilkan waktu
  DateTime now = rtc.now();
  lcd.setCursor(10, 1);
  if (now.hour() < 10) lcd.print('0');
  lcd.print(now.hour());
  lcd.print(':');
  if (now.minute() < 10) lcd.print('0');
  lcd.print(now.minute());

  delay(500);
}

// Servo control - tetap di Arduino karena menggunakan library
void controlServo(bool openBox) {
  static int currentPos = -1;
  int targetPos = openBox ? SERVO_OPEN : SERVO_CLOSED;

  if (currentPos == -1) {
    currentPos = targetPos;
    pwm.setPWM(SERVO_CHANNEL, 0, currentPos);
    return;
  }

  if (currentPos < targetPos) {
    for (int pos = currentPos; pos <= targetPos; pos += 5) {
      pwm.setPWM(SERVO_CHANNEL, 0, pos);
      delay(15);
    }
  } else {
    for (int pos = currentPos; pos >= targetPos; pos -= 5) {
      pwm.setPWM(SERVO_CHANNEL, 0, pos);
      delay(15);
    }
  }
  currentPos = targetPos;
}
```

.S for light monitor system

```
; LightMonitorSystem.S
```

```
#include <avr/io.h>

; Pin definitions
.equ GREEN_LED, 1    ; PB1 (Digital 9)
.equ RED_LED, 2      ; PB2 (Digital 10)
.equ LDR_CHANNEL, 0  ; ADC0 (PC0)

; Constants
.equ LIGHT_THRESHOLD_LOW, 244   ; 500 & 0xFF (low byte)
.equ LIGHT_THRESHOLD_HIGH, 1    ; 500 >> 8 (high byte)

.section .text

; ========================================
; setupPins - Setup pin I/O
; ========================================
.global setupPins
setupPins:
    push r16
    push r17

    ; Set PB1 (Green LED) dan PB2 (Red LED) sebagai output
    in r16, _SFR_IO_ADDR(DDRB)
    ori r16, (1 << GREEN_LED) | (1 << RED_LED)
    out _SFR_IO_ADDR(DDRB), r16

    ; Matikan kedua LED initially
    in r16, _SFR_IO_ADDR(PORTB)
    andi r16, ~((1 << GREEN_LED) | (1 << RED_LED))
    out _SFR_IO_ADDR(PORTB), r16

    pop r17
    pop r16
    ret

; ========================================
; setupADC - Setup ADC untuk sensor cahaya
; ========================================
.global setupADC
setupADC:
    push r16

    ; Set ADMUX: AVCC reference, ADC0 channel
    ldi r16, (1 << REFS0)  ; AVCC reference, channel 0
    sts _SFR_MEM_ADDR(ADMUX), r16

    ; Set ADCSRA: Enable ADC, prescaler 128
    ldi r16, (1 << ADEN) | (1 << ADPS2) | (1 << ADPS1) | (1 <<
ADPS0)
    sts _SFR_MEM_ADDR(ADCSRA), r16

    pop r16
    ret

; ========================================
; readLightSensor - Baca sensor LDR
; Return: r24-r25 (16-bit ADC value)
; ========================================
.global readLightSensor
readLightSensor:
```

```
    push r16
    push r17

    ; Start ADC conversion
    lds r16, _SFR_MEM_ADDR(ADCSRA)
    ori r16, (1 << ADSC)  ; Set ADSC bit
    sts _SFR_MEM_ADDR(ADCSRA), r16

    ; Wait for conversion complete
wait_adc:
    lds r16, _SFR_MEM_ADDR(ADCSRA)
    sbrc r16, ADSC        ; Skip if ADSC = 0
    rjmp wait_adc

    ; Read ADC result (ADCL first, then ADCH)
    lds r24, _SFR_MEM_ADDR(ADCL)      ; Low byte to r24
    lds r25, _SFR_MEM_ADDR(ADCH)      ; High byte to r25

    pop r17
    pop r16
    ret

; ======================================
; setLEDs - Kontrol LED Green dan Red
; Parameters: r24 = greenOn (0/1), r22 = redOn (0/1)
; ======================================
.global setLEDs
setLEDs:
    push r16
    push r17

    ; Baca status PORTB saat ini
    in r16, _SFR_IO_ADDR(PORTB)

    ; Clear LED bits
    andi r16, ~((1 << GREEN_LED) | (1 << RED_LED))

    ; Set Green LED jika r24 != 0
    tst r24
    breq check_red
    ori r16, (1 << GREEN_LED)

check_red:
    ; Set Red LED jika r22 != 0
    tst r22
    breq write_portb
    ori r16, (1 << RED_LED)

write_portb:
    ; Write ke PORTB
    out _SFR_IO_ADDR(PORTB), r16

    pop r17
    pop r16
    ret

; ======================================
; blinkLEDTimes - Blink LED beberapa kali
; Parameters: r24 = pin (9 atau 10), r22 = count
; ======================================
```

```
.global blinkLEDTimes
blinkLEDTimes:
    push r16
    push r17
    push r18
    push r19
    push r20

    ; Convert Arduino pin to bit position
    ; Pin 9 = PB1 (bit 1), Pin 10 = PB2 (bit 2)
    mov r18, r24
    ldi r19, 1          ; Default bit 1 (Green LED)
    cpi r18, 10
    brne blink_loop
    ldi r19, 2          ; bit 2 (Red LED)

blink_loop:
    ; Check if count > 0
    tst r22
    breq blink_done

    ; Turn LED ON
    in r16, _SFR_IO_ADDR(PORTB)
    mov r17, r19
    ldi r20, 1
blink_shift_on:
    dec r17
    breq set_led_on
    lsl r20
    rjmp blink_shift_on
set_led_on:
    or r16, r20
    out _SFR_IO_ADDR(PORTB), r16

    ; Delay ~250ms (approximate)
    rcall delay_250ms

    ; Turn LED OFF
    in r16, _SFR_IO_ADDR(PORTB)
    com r20             ; Invert mask
    and r16, r20
    out _SFR_IO_ADDR(PORTB), r16

    ; Delay ~250ms
    rcall delay_250ms

    ; Decrement count
    dec r22
    rjmp blink_loop

blink_done:
    pop r20
    pop r19
    pop r18
    pop r17
    pop r16
    ret

; ======================================
; getThreshold - Return light threshold value
```

```
; Return: r24-r25 = 500
; ======================================
.global getThreshold
getThreshold:
    ldi r24, LIGHT_THRESHOLD_LOW
    ldi r25, LIGHT_THRESHOLD_HIGH
    ret

; ======================================
; delay_250ms - Delay approximately 250ms
; Assumes 16MHz clock
; ======================================
delay_250ms:
    push r16
    push r17
    push r18

    ; Outer loop count for ~250ms
    ldi r18, 250       ; ~1ms per iteration

delay_outer:
    ldi r17, 200       ; Inner loop
delay_middle:
    ldi r16, 80        ; Inner-inner loop
delay_inner:
    dec r16
    brne delay_inner
    dec r17
    brne delay_middle
    dec r18
    brne delay_outer

    pop r18
    pop r17
    pop r16
    ret
```

2.2.3 HARDWARE AND SOFTWARE INTEGRATION

Pada proyek ini, integrasi antara perangkat keras (rangkaian) dan perangkat lunak (program) dilakukan untuk menciptakan sistem pemantauan dan pengatur intensitas cahaya bagi tanaman secara otomatis. Sistem ini menggunakan Arduino Uno sebagai mikrokontroler utama yang mengendalikan seluruh proses.

- Proses Inisialisasi

Pada awal program, stack pointer inisialisasi, kemudian berbagai perangkat seperti UART (untuk komunikasi serial), LED indikator, interrupt eksternal (INT0), ADC (untuk pembacaan sensor analog), dan protokol TWI (I2C) inisialisasi. Modul LCD dan driver servo PCA9685 juga diinisialisasi untuk siap digunakan. Setelah itu, nilai awal cahaya dibaca dari sensor LDR untuk menentukan status awal box (terbuka atau tertutup).

- Main Loop

Dalam loop utama, Arduino secara terus-menerus membaca nilai cahaya dari LDR. Berdasarkan nilai tersebut, program menjalankan fungsi AutoControl yang memutuskan apakah box harus terbuka atau tertutup. Jika nilai cahaya di bawah ambang batas (THRESHOLD) dan box sedang tertutup, maka box akan dibuka. Sebaliknya, jika cahaya melebihi ambang dan box terbuka, maka box akan ditutup. Perubahan posisi box diatur dengan menggerakkan servo melalui driver PCA9685. Status dan nilai cahaya terbaru juga diperbarui pada LCD. Program memberikan jeda waktu (delay) untuk menghindari pembacaan terlalu cepat yang dapat menyebabkan fluktuasi.

- Penanganan Interrupt

Sistem menggunakan interrupt eksternal (INT0) yang dipicu oleh penekanan tombol fisik. Ketika tombol ditekan, interrupt handler akan dijalankan yang mengubah status box (toggle antara terbuka dan tertutup), menyalakan LED indikator sesuai status, dan mengatur posisi servo untuk menggerakkan box.

Dalam proses integrasi ini, pada rangkaian harus disesuaikan input dan outputnya dengan kode yang sudah dibuat:

- Sensor LDR mengirimkan data analog ke pin A0 Arduino, kemudian ADC mengkonversi sinyal tersebut menjadi nilai digital untuk diproses.

- Modul RTC DS3231 diakses melalui I2C untuk mendapatkan informasi waktu yang akurat, sehingga sistem dapat melakukan pencatatan waktu dan penjadwalan tugas secara tepat.

- Driver PCA9685 menerima perintah servo dari Arduino melalui I2C untuk mengatur posisi motor servo membuka dan menutup box.

- LCD I2C menerima data dari Arduino untuk menampilkan nilai cahaya dan status box secara user-friendly.

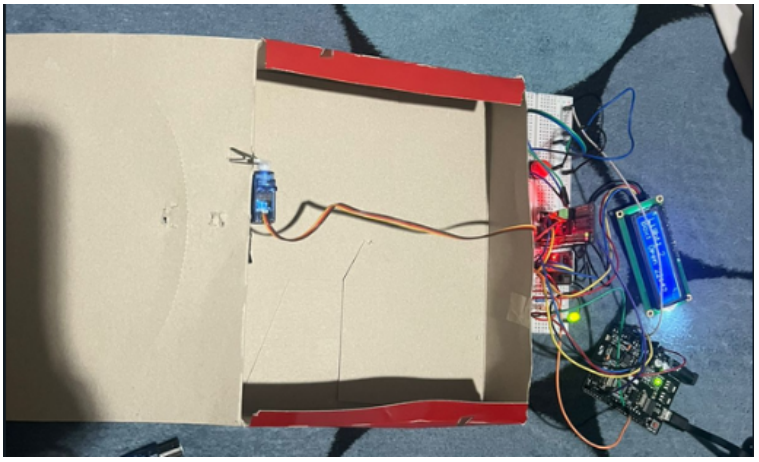
- Tombol fisik dihubungkan ke pin interrupt Arduino (INT0), memungkinkan perubahan status box secara manual dengan respons cepat melalui interrupt handler.

- LED indikator disambungkan ke pin digital Arduino sebagai output status visual.

CHAPTER 3

TESTING AND ANALYSIS

3.1 TESTING & RESULT



Pengujian sistem dilakukan melalui dua tahap utama, yaitu simulasi menggunakan perangkat lunak Proteus dan perakitan rangkaian fisik.

Pada tahap simulasi di Proteus, seluruh rangkaian diuji dan berjalan dengan lancar. Semua komponen, termasuk sensor LDR, LCD I2C, motor servo, serta modul RTC DS3231, berfungsi sesuai rancangan. Sensor memberikan respons yang akurat, aktuator bergerak sesuai perintah, dan tampilan pada LCD menunjukkan data yang tepat. Keberhasilan simulasi ini memberikan kepastian bahwa desain rangkaian dan program sudah benar dan siap untuk tahap implementasi fisik.

Setelah simulasi berhasil, tim melanjutkan dengan merakit rangkaian secara fisik. Namun, dalam pengujian fisik ditemukan beberapa kendala, antara lain: sensor LDR yang tidak responsif dengan baik, motor servo yang tidak selalu bergerak, serta tumpuan box pelindung yang kurang kuat. Meskipun demikian, sebagian komponen seperti LCD dan modul RTC bekerja dengan baik dan sesuai harapan.

3.2 ANALYSIS

Berdasarkan hasil pengujian fisik, tim melakukan analisis terhadap permasalahan yang muncul. Respons sensor LDR yang kurang stabil diduga disebabkan oleh penempatan dan pengkabelan yang belum optimal. Motor servo yang tidak berfungsi sempurna kemungkinan terkait dengan suplai daya dan sambungan konektor yang kurang baik. Kelemahan tumpuan mekanis box memengaruhi kestabilan pergerakan motor sehingga perlu dilakukan penguatan struktur.

Setelah dilakukan perbaikan berupa pemasangan ulang sensor dengan posisi dan kabel yang lebih baik, pengecekan koneksi dan catu daya motor, serta penguatan mekanik tumpuan box, sistem dapat berfungsi dengan baik hingga tahap akhir perakitan. Meskipun pemasangan LDR belum sepenuhnya rapi, integrasi antara program dan perangkat keras secara keseluruhan sudah berhasil dan berjalan lancar.

Kesimpulannya, simulasi digital memberikan validasi awal yang baik terhadap desain, sementara proses perakitan fisik mengungkap dan memungkinkan penyelesaian masalah yang krusial agar sistem dapat beroperasi secara optimal di dunia nyata.

CHAPTER 4

4.1 CONCLUSION

Baik rangkaian digital (simulasi) maupun fisik berhasil dibuat dan beroperasi dengan lancar. Pengujian simulasi memberikan jaminan terhadap kebenaran desain, sedangkan tahap perakitan fisik memberikan pengalaman troubleshooting dan penyempurnaan yang penting untuk menghasilkan sistem yang handal dan siap digunakan.

REFERENCES

- [1]I. to, “Modul 2 SSF: Introduction to Assembly & I/O Programming,” *Google Docs*, 2019. https://docs.google.com/document/d/1s84Y1xrGyJwWXQJgdBQL6bTaFFZtiOsA4_3YGouP1CY/edit?tab=t.0
- [2]A. to, “Modul 3 SSF : Analog to Digital Converter,” *Google Docs*, 2020. https://docs.google.com/document/d/1arLt3fqXRw-WgkbqlP1RwYs-XJy9-QAFfEcM21M3u44/edit?tab=t.0
- [3]S. Port, “Modul 4 SSF : Serial Port,” *Google Docs*, 2019. https://docs.google.com/document/d/1rRWvBgL3Nsb_h10131A-1kiGQkrGGNLedYoeVIGR9zg/edit?tab=t.0
- [4]Pi My Life Up, “Arduino Light Sensor: Learn to Setup a Photoresistor (LDR),” *YouTube*, Jan. 12, 2025. https://www.youtube.com/watch?v=XtldqC3dzXU
- [5]“EMAS2: Log in to the site,” *Ui.ac.id*, 2025. https://emas2.ui.ac.id/pluginfile.php/5033027/mod_resource/content/2/Modul%206%20SSF_%20Timer.pdf
- [6]Modul 7 SSF: Interrupt, “Modul 7 SSF: Interrupt,” *Google Docs*, 2019. https://docs.google.com/document/d/1VW7j3k_scKyOlzo72scSMFU58EgT-TLoiMOshQ48TUA/edit?tab=t.0
- [7]Modul 8 SSF : SPI & I2C, “Modul 8 SSF : SPI & I2C,” *Google Docs*, 2020. https://docs.google.com/document/d/1CsIbwLVUrsKjZ3YhyF0J-gsGWNU1RCQu7JTYMCx3cFY/edit?tab=t.0
- [8]Sensor, “Modul 9 SSF : Sensor Interfacing,” *Google Docs*, 2019. https://docs.google.com/document/d/14D8bETDw8x-BbeWWfg2QrjEE1WJA17kAZVDu79iCzCs/edit?tab=t.0
- [9] A. Rahman, S. Kumar, and M. T. Hossain, "Precision Agriculture Technologies for Smart Farming: Sensors, IoT, and Control Systems," New York, NY, USA: Springer, 2024, pp. 127-156.
- [10] M. A. Mazidi and S. Chen, "Embedded Systems Design with AVR Microcontrollers: Interrupts, Timers, I2C, LCD, and Modern C++," 3rd ed., Boca Raton, FL, USA: CRC Press, 2023, pp. 215-278.
- [11] L. Taiz, E. Zeiger, I. M. Møller, and A. Murphy, "Plant Physiology and Development: Light Response Optimization in Controlled Agricultural Environments," 7th ed., Sunderland, MA, USA: Sinauer Associates, 2023, pp. 342-389.

- [12] S. Navulur and M. Geetha, "Agricultural IoT Systems: From Sensors to Smart Decisions," IEEE Transactions on Industrial Electronics, vol. 71, no. 3, pp. 2851-2863, Mar. 2024, doi: 10.1109/TIE.2023.3246589.
- [13] R. H. Barnett and L. O'Cull, "Microcontroller Programming: The Microchip AVR," 4th ed., Boca Raton, FL, USA: CRC Press, 2023, pp. 189-232.