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

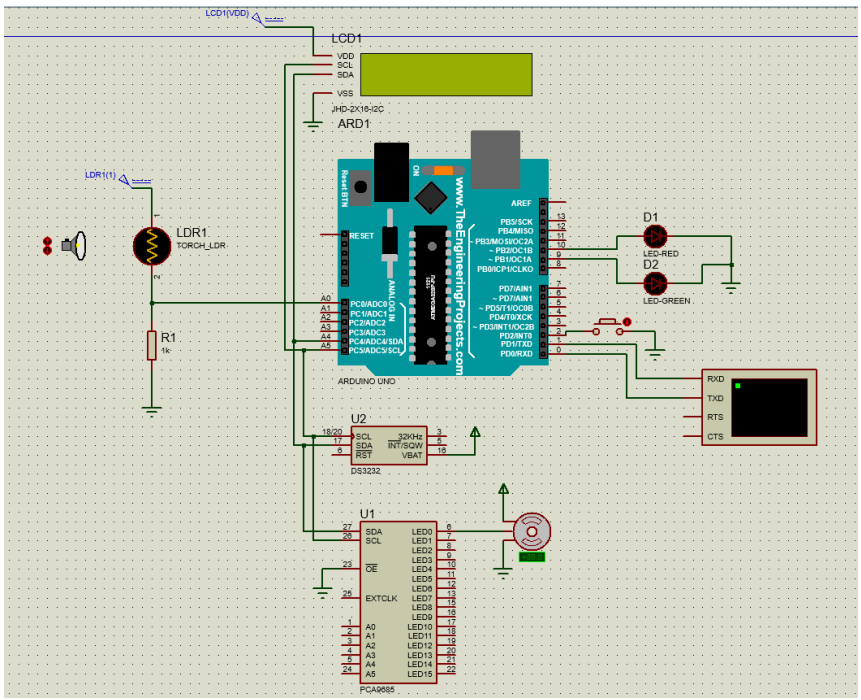


Fig.2 *Proteus Design*

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.

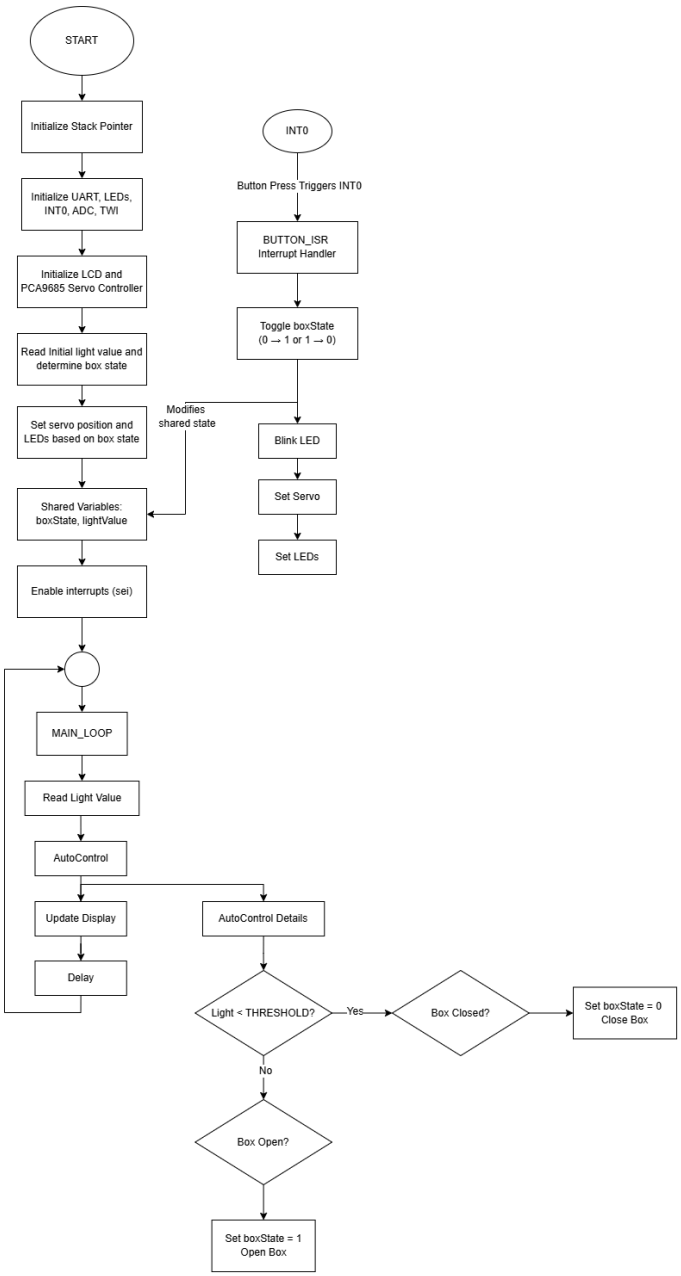
5) Servo PCA9685
   Ini adalah modul PWM 16-channel yang dikontrol melalui I2C yang berfungsi untuk mengatur pergerakan servo motor yang membuka dan menutup penutup tanaman. Karena memungkinkan banyak output PWM dengan dua pin Arduino (SDA/SCL), modul ini sangat efektif.

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

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

// Deklarasi objek
LiquidCrystal_I2C lcd(0x27, 16, 2); // Alamat I2C, kolom, baris
RTC_DS3231 rtc;
Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver(); // Alamat
default 0x40

// Definisi pin
#define LDR_PIN A0        // PC0 (pin 23)
#define BUTTON_PIN 2      // PD2 (pin 4)
#define RED_LED_PIN 10    // PB2 (pin 16)
#define GREEN_LED_PIN 9   // PB1 (pin 15)

// Konstanta
#define SERVO_CHANNEL 0   // Channel pada PCA9685 untuk servo
#define SERVO_OPEN 150    // Nilai PWM untuk posisi terbuka
#define SERVO_CLOSED 450  // Nilai PWM untuk posisi tertutup
#define LIGHT_THRESHOLD 500 // Nilai ambang batas intensitas cahaya
(0-1023)
#define LED_BLINK_INTERVAL 250 // Interval kedip LED saat transisi
status (ms)

// Variabel global
volatile bool boxState = true;        // true = terbuka, false =
tertutup
volatile bool buttonPressed = false;  // Flag untuk button interrupt

// Deklarasi fungsi eksternal ditulis dalam Assembly
extern "C" {
  void setupSystem();
  void readLightIntensity();
  int getLightValue();
  void updateLCD(bool boxState, int lightValue);
  void controlServo(bool openBox);
  void setLEDs(bool isOpen);
  DateTime getCurrentTime();
  void displayDateTime(DateTime now);
}

// Fungsi setup
void setup() {
  // Inisialisasi Serial untuk debugging
  Serial.begin(9600);
  Serial.println(F("Sistem Monitoring Cahaya Fotosintesis"));

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

```
  // Inisialisasi I2C
  Wire.begin();

  // Inisialisasi LCD
  lcd.init();
  lcd.backlight();
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Initializing...");

  // Inisialisasi RTC
  if (!rtc.begin()) {
    Serial.println(F("Couldn't find RTC"));
    lcd.setCursor(0, 1);
    lcd.print("RTC Error!");
    while (1);
  }

  if (rtc.lostPower()) {
    Serial.println(F("RTC lost power, let's set the time!"));
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  }

  // Inisialisasi PCA9685 (driver servo)
  pwm.begin();
  pwm.setPWMFreq(60);  // Frekuensi PWM untuk sebagian besar servos

  // Memanggil fungsi setup sistem dalam assembly
  setupSystem();

  // Inisialisasi LCD dengan pesan
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Initializing...");

  // Ambil nilai cahaya awal
  readLightIntensity();
  int initialLight = getLightValue();

  // Tentukan posisi awal berdasarkan cahaya
  boxState = (initialLight <= LIGHT_THRESHOLD);

  // Set LED dan kontrol servo sesuai status awal
  setLEDs(boxState);
  controlServo(boxState);

  // Tampilkan status awal
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("System Ready");
  lcd.setCursor(0, 1);
  lcd.print("Box: ");
  lcd.print(boxState ? "Open" : "Closed");
  delay(2000);
}

// Interrupt Handler saat tombol ditekan
void buttonInterrupt() {
  // Debouncing sederhana
```

```
  static unsigned long lastInterruptTime = 0;
  unsigned long interruptTime = millis();

  // Jika interupsi terjadi dalam 200ms, abaikan sebagai debouncing
  if (interruptTime - lastInterruptTime > 200) {
    buttonPressed = true;

    // Nyalakan LED merah langsung saat tombol ditekan
    digitalWrite(RED_LED_PIN, HIGH);
    digitalWrite(GREEN_LED_PIN, LOW);
  }

  lastInterruptTime = interruptTime;
}

// Loop utama
void loop() {
  // Kode C hanya menangani interrupt dan event dari interrupt

  // Jika tombol ditekan, toggle status box
  if (buttonPressed) {
    // Toggle status box
    boxState = !boxState;

    // Berkedip sebentar LED sesuai status yang baru
    byte ledPin = boxState ? GREEN_LED_PIN : RED_LED_PIN;
    blinkLED(ledPin, 3);

    // Selanjutnya gerakkan servo
    controlServo(boxState);

    // Reset flag setelah operasi selesai
    buttonPressed = false;

    // Set status LED sesuai status akhir
    setLEDs(boxState);

    // Tampilkan pesan status
    lcd.setCursor(0, 1);
    lcd.print("Box: ");
    lcd.print(boxState ? "Open  " : "Closed");

    Serial.print(F("Box status changed by button: "));
    Serial.println(boxState ? F("Open") : F("Closed"));
  }

  // Semua logika utama ditangani oleh fungsi assembly
  readLightIntensity();
  int lightValue = getLightValue();

  // Logika kontrol box berdasarkan intensitas cahaya
  if (lightValue > LIGHT_THRESHOLD && boxState) {
    // Cahaya cukup terang dan box sedang terbuka, tutup box
    boxState = false;
    controlServo(false);
    setLEDs(false);
    Serial.println(F("Box closed automatically due to sufficient
light"));
  }
  else if (lightValue <= LIGHT_THRESHOLD && !boxState) {
```

```
    // Cahaya tidak cukup terang dan box sedang tertutup, buka box
    boxState = true;
    controlServo(true);
    setLEDs(true);
    Serial.println(F("Box opened automatically due to insufficient
light"));
  }

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

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

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

  delay(500); // Delay untuk stabilitas
}

// Implementasi fungsi C yang digunakan oleh assembly
// Ini adalah fungsi "jembatan" antara C dan assembly

DateTime getCurrentTime() {
  return rtc.now();
}

void controlServo(bool openBox) {
  // Mengontrol servo melalui PCA9685 dengan pergerakan halus
  static int currentPosition = -1;
  int targetPosition = openBox ? SERVO_OPEN : SERVO_CLOSED;

  // Jika posisi saat ini belum diinisialisasi
  if (currentPosition == -1) {
    currentPosition = targetPosition;
    pwm.setPWM(SERVO_CHANNEL, 0, currentPosition);
    return;
  }

  // Gerakkan servo secara bertahap untuk menghindari gerakan
tiba-tiba
  if (currentPosition < targetPosition) {
    for (int pos = currentPosition; pos <= targetPosition; pos += 5)
{
      pwm.setPWM(SERVO_CHANNEL, 0, pos);
      delay(15);  // Delay kecil untuk pergerakan halus
    }
  } else {
    for (int pos = currentPosition; pos >= targetPosition; pos -= 5)
{
```

```
      pwm.setPWM(SERVO_CHANNEL, 0, pos);
      delay(15);  // Delay kecil untuk pergerakan halus
    }
  }

  currentPosition = targetPosition;
}

void setLEDs(bool isOpen) {
  // Atur status LED berdasarkan status box
  digitalWrite(GREEN_LED_PIN, isOpen ? HIGH : LOW);
  digitalWrite(RED_LED_PIN, isOpen ? LOW : HIGH);
}

// Fungsi untuk membuat LED berkedip saat pergantian status
void blinkLED(byte pin, int count) {
  for (int i = 0; i < count; i++) {
    digitalWrite(pin, HIGH);
    delay(LED_BLINK_INTERVAL);
    digitalWrite(pin, LOW);
    delay(LED_BLINK_INTERVAL);
  }
}
```

.S for light monitor system

```
; LightMonitorSystem.S
; Monitoring intensitas cahaya untuk fotosintesis, Assembly AVR
(ATmega328P)

; =========================
; Konstanta
; =========================

.equ BAUD              , 9600
.equ UBRR_VALUE        , 103            ; (F_CPU/16/BAUD)-1

.equ TWBR_VALUE        , 72             ; SCL ≈100kHz
.equ TWI_READ          , 1
.equ TWI_WRITE         , 0

.equ LCD_ADDR          , 0x27
.equ LCD_WRITE_ADDR    , 0x4E            ; (LCD_ADDR << 1) |
TWI_WRITE
.equ LCD_READ_ADDR     , 0x4F            ; (LCD_ADDR << 1) |
TWI_READ

.equ PCA9685_ADDR      , 0x40
.equ PCA9685_WRITE     , 0x80            ;
(PCA9685_ADDR<<1)|TWI_WRITE
.equ PCA9685_READ      , 0x81            ;
(PCA9685_ADDR<<1)|TWI_READ

.equ LED_GREEN         , 1               ; PB1
.equ LED_RED           , 2               ; PB2

.equ LIGHT_THRESHOLD   , 500
```

```
; FIX: Adjust servo positions for better operation
.equ SERVO_OPEN_POS    , 150
.equ SERVO_CLOSED_POS  , 450

; I/O-register (I/O space 0x00-0x3F) & eksternal I/O (>0x3F)
.equ RAMEND            , 0x08FF
.equ SPH               , 0x3E
.equ SPL               , 0x3D

.equ UBRR0H            , 0xC5          ; FIX: Fix extended I/O
addresses
.equ UBRR0L            , 0xC4          ; FIX: Use extended I/O
addresses
.equ UCSR0B            , 0xC1          ; FIX: Use extended I/O
addresses
.equ TXEN0             , 3

.equ DDRB              , 0x04          ; FIX: Use correct I/O
address
.equ PORTB             , 0x05          ; FIX: Use correct I/O
address

.equ MCUCR             , 0x35
.equ ISC01             , 1
.equ GICR              , 0x3B          ; FIX: Use EIMSK instead of
GICR for ATmega328P
.equ EIMSK             , 0x3D          ; FIX: Add EIMSK for
ATmega328P
.equ INT0              , 6

.equ ADMUX             , 0x7C          ; FIX: Use extended I/O
addresses
.equ REFS0             , 6
.equ ADCSRA            , 0x7A          ; FIX: Use extended I/O
addresses
.equ ADEN              , 7
.equ ADPS0             , 0
.equ ADPS1             , 1
.equ ADPS2             , 2
.equ ADSC              , 6
.equ ADCL              , 0x78          ; FIX: Use extended I/O
addresses
.equ ADCH              , 0x79          ; FIX: Use extended I/O
addresses

.equ TWBR              , 0xB8          ; FIX: Use extended I/O
addresses
.equ TWSR              , 0xB9          ; FIX: Add TWI status
register
.equ TWDR              , 0xBB          ; FIX: Use extended I/O
addresses
.equ TWCR              , 0xBC          ; FIX: Use extended I/O
addresses
.equ TWINT             , 7
.equ TWSTA             , 5
.equ TWSTO             , 4
.equ TWEN              , 2

; =========================
; Data Section
```

```
; =========================
.section .data
lightValue:   .byte 2       ; 16-bit ADC
boxState:     .byte 1       ; 0=tutup,1=terbuka
buffer:       .byte 40

; =========================
; Code Section & Vektor
; =========================
.section .text
.org 0x0000
    rjmp RESET
.org 0x0002
    rjmp BUTTON_ISR

; =========================
; RESET & INIT
; =========================
RESET:
    ; Stack pointer
    ldi   r16, hi8(RAMEND)
    out   SPH, r16
    ldi   r16, lo8(RAMEND)
    out   SPL, r16

    ; UART (debug)
    ldi   r16, hi8(UBRR_VALUE)
    sts   UBRR0H, r16             ; FIX: Use STS for extended I/O
    ldi   r16, lo8(UBRR_VALUE)
    sts   UBRR0L, r16             ; FIX: Use STS for extended I/O
    ldi   r16, (1<<TXEN0)
    sts   UCSR0B, r16             ; FIX: Use STS for extended I/O

    ; Port B LED
    ldi   r16, (1<<LED_GREEN)|(1<<LED_RED)
    out   DDRB, r16

    ; FIX: Ensure LEDs are initially off
    ldi   r16, 0
    out   PORTB, r16

    ; INT0 falling edge
    ldi   r16, (1<<ISC01)
    out   MCUCR, r16
    ldi   r16, (1<<INT0)
    out   EIMSK, r16              ; FIX: Use EIMSK instead of GICR

    ; ADC0, AVcc ref, presc=128
    ldi   r16, (1<<REFS0)
    sts   ADMUX, r16              ; FIX: Use STS for extended I/O
    ldi   r16, (1<<ADEN)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0)
    sts   ADCSRA, r16             ; FIX: Use STS for extended I/O

    ; TWI init
    ldi   r16, TWBR_VALUE
    sts   TWBR, r16               ; FIX: Use STS for extended I/O
    ldi   r16, (1<<TWEN)
    sts   TWCR, r16               ; FIX: Use STS for extended I/O

    ; FIX: Add delay after TWI initialization
```

```
    ldi   r19, 50
.TWI_INIT_DELAY:
    rcall Delay_ms
    dec   r19
    brne  .TWI_INIT_DELAY

    ; Init LCD & PCA9685
    rcall LCD_Init
    rcall PCA9685_Init

    ; Tentukan boxState awal
    rcall ReadLight
    lds   r17, lightValue
    lds   r18, lightValue+1       ; FIX: Load high byte too
    ldi   r19, lo8(LIGHT_THRESHOLD)
    ldi   r20, hi8(LIGHT_THRESHOLD) ; FIX: Compare 16-bit properly
    cp    r17, r19
    cpc   r18, r20
    brlt  .L_OPEN0
    ldi   r16, 0
    sts   boxState, r16
    rjmp  .L_INIT_DONE
.L_OPEN0:
    ldi   r16, 1
    sts   boxState, r16
.L_INIT_DONE:
    rcall SetServo
    rcall SetLEDs

    ; FIX: Give some time for initialization to complete
    ldi   r19, 100
.FINAL_INIT_DELAY:
    rcall Delay_ms
    dec   r19
    brne  .FINAL_INIT_DELAY

    sei

MAIN_LOOP:
    rcall ReadLight
    rcall AutoControl
    rcall UpdateDisplay

    ; FIX: Add delay in main loop to prevent too rapid cycling
    ldi   r19, 20
.MAIN_DELAY:
    rcall Delay_ms
    dec   r19
    brne  .MAIN_DELAY

    rjmp MAIN_LOOP

; =========================
; ISR INT0
; =========================
BUTTON_ISR:
    ; FIX: Save used registers in ISR
    push  r16
    push  r17
    in    r16, SREG
```

```
    push  r16

    lds   r16, boxState
    ldi   r17, 1
    eor   r16, r17
    sts   boxState, r16
    rcall BlinkLED
    rcall SetServo
    rcall SetLEDs

    ; FIX: Restore registers
    pop   r16
    out   SREG, r16
    pop   r17
    pop   r16
    reti

; =========================
; ReadLight (ADC)
; =========================
ReadLight:
    ; FIX: Proper ADC reading sequence
    lds   r16, ADCSRA
    ori   r16, (1<<ADSC)
    sts   ADCSRA, r16
.WAIT_ADC:
    lds   r16, ADCSRA
    sbrc  r16, ADSC             ; FIX: Use SBRC (skip if bit
cleared)
    rjmp  .WAIT_ADC

    ; FIX: Read ADCL first, then ADCH
    lds   r17, ADCL
    lds   r18, ADCH
    sts   lightValue, r17
    sts   lightValue+1, r18
    ret

; =========================
; AutoControl
; =========================
AutoControl:
    lds   r17, lightValue
    lds   r18, lightValue+1      ; FIX: Get full 16-bit value
    ldi   r19, lo8(LIGHT_THRESHOLD)
    ldi   r20, hi8(LIGHT_THRESHOLD)
    cp    r17, r19
    cpc   r18, r20               ; FIX: Compare 16-bit properly
    brlt  .LOW
    lds   r18, boxState
    tst   r18
    brne  .CLOSE
    rjmp  .DONE
.LOW:
    lds   r18, boxState
    tst   r18
    breq  .OPEN
    rjmp  .DONE
.OPEN:
    ldi   r16, 1
```

```
    sts   boxState, r16
    rcall SetServo
    rcall SetLEDs
    rjmp  .DONE
.CLOSE:
    ldi   r16, 0
    sts   boxState, r16
    rcall SetServo
    rcall SetLEDs
.DONE:
    ret

; =========================
; SetServo (PCA9685)
; =========================
SetServo:
    ; FIX: Save used registers
    push  r16
    push  r17
    push  r18
    push  r20

    lds   r16, boxState
    cpi   r16, 1
    breq  .SV_OPEN
    ; tutup
    ldi   r20, PCA9685_WRITE
    rcall TWI_Start
    mov   r17, r20
    rcall TWI_Write
    ldi   r17, 0x06    ; Register LED0_ON_L
    rcall TWI_Write
    ldi   r17, 0       ; ON time low byte
    rcall TWI_Write
    ldi   r17, 0       ; ON time high byte
    rcall TWI_Write
    ldi   r17, lo8(SERVO_CLOSED_POS)  ; OFF time low byte
    rcall TWI_Write
    ldi   r17, hi8(SERVO_CLOSED_POS)  ; OFF time high byte
    rcall TWI_Write
    rcall TWI_Stop

    ; FIX: Add delay after servo command
    ldi   r18, 20
.SV_DELAY1:
    rcall Delay_ms
    dec   r18
    brne  .SV_DELAY1

    pop   r20
    pop   r18
    pop   r17
    pop   r16
    ret

.SV_OPEN:
    ; buka
    ldi   r20, PCA9685_WRITE
    rcall TWI_Start
    mov   r17, r20
```

```
    rcall TWI_Write
    ldi   r17, 0x06    ; Register LED0_ON_L
    rcall TWI_Write
    ldi   r17, 0       ; ON time low byte
    rcall TWI_Write
    ldi   r17, 0       ; ON time high byte
    rcall TWI_Write
    ldi   r17, lo8(SERVO_OPEN_POS)    ; OFF time low byte
    rcall TWI_Write
    ldi   r17, hi8(SERVO_OPEN_POS)    ; OFF time high byte
    rcall TWI_Write
    rcall TWI_Stop

    ; FIX: Add delay after servo command
    ldi   r18, 20
.SV_DELAY2:
    rcall Delay_ms
    dec   r18
    brne  .SV_DELAY2

    pop   r20
    pop   r18
    pop   r17
    pop   r16
    ret

; =========================
; SetLEDs
; =========================
SetLEDs:
    ; FIX: Perbaiki logika set LED - PORTB harus diatur ulang setiap
kali
    clr   r17          ; Clear r17 first
    lds   r16, boxState
    cpi   r16, 1
    breq  .LED_GREEN
    ldi   r17, (1<<LED_RED)
    out   PORTB, r17
    ret
.LED_GREEN:
    ldi   r17, (1<<LED_GREEN)
    out   PORTB, r17
    ret

; =========================
; BlinkLED
; =========================
BlinkLED:
    ; FIX: Save used registers
    push  r16
    push  r17
    push  r18
    push  r19

    lds   r16, boxState
    cpi   r16, 1
    breq  .BG_GREEN
    ldi   r17, (1<<LED_RED)
    rjmp  .DO_BLINK
.BG_GREEN:
```

```
    ldi   r17, (1<<LED_GREEN)
.DO_BLINK:
    ldi   r18, 3
.BLINK_LOOP:
    out   PORTB, r17
    rcall Delay_ms
    clr   r16
    out   PORTB, r16
    rcall Delay_ms
    dec   r18
    brne  .BLINK_LOOP

    pop   r19
    pop   r18
    pop   r17
    pop   r16
    ret

; =========================
; UpdateDisplay (LCD)
; =========================
UpdateDisplay:
    ; FIX: Save used registers
    push  r16
    push  r17
    push  r18
    push  r19
    push  r20

    rcall LCD_Clear

    ; Display "Light:" pada LCD
    ldi   r20, LCD_WRITE_ADDR
    rcall TWI_Start
    mov   r17, r20
    rcall TWI_Write
    ldi   r17, 0x40    ; Data mode
    rcall TWI_Write

    ; "L"
    ldi   r17, 'L'
    rcall TWI_Write
    ; "i"
    ldi   r17, 'i'
    rcall TWI_Write
    ; "g"
    ldi   r17, 'g'
```

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.

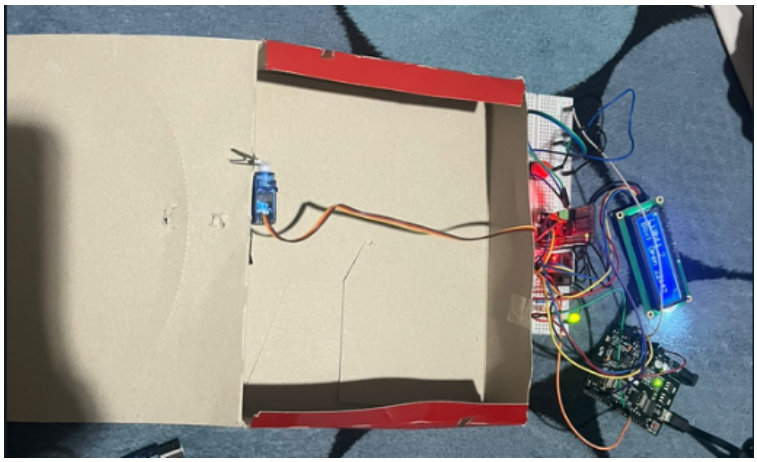
- Tombol fisik dihubungkan ke pin interrupt Arduino (INT0), memungkinkan perubahan status box secara manual dengan respons cepat melalui interrupt handler.

- 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.