



**CYBER-PHYSICAL SYSTEM FINAL PROJECT REPORT**  
**DEPARTMENT OF ELECTRICAL ENGINEERING**  
**UNIVERSITAS INDONESIA**

**SMART HOME AUTOMATION**

**GROUP 11**

<b>Louis Benedict Archie</b>	<b>2206025224</b>
<b>Monica Vierin Pasman</b>	<b>2206029405</b>
<b>Reiki Putra Dermawan</b>	<b>2206062882</b>
<b>Adhelia Putri Maylani</b>	<b>2206814816</b>

## PREFACE

Puji syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa atas rahmat dan karunia-Nya sehingga kami dapat menyelesaikan laporan proyek akhir yang berjudul "Smart Home Automation", tepat pada waktunya. Laporan ini disusun untuk memberikan pemahaman kepada pembaca tentang bagaimana cara kerja dari sistem otomatisasi rumah yang efektif dan efisien dengan memanfaatkan teknologi modern.

Proyek ini merupakan implementasi dari pengetahuan dan keterampilan yang kami peroleh selama mengikuti mata kuliah Sistem Siber-Fisik. Kami juga ingin menyampaikan terima kasih yang sebesar-besarnya kepada dosen pembimbing, F. Astha Ekadiyanto, S.T., M.Sc., serta asisten laboratorium dan teman-teman yang telah membantu dalam proses pembuatan proyek ini.

Meskipun kami telah berusaha menyusun laporan dengan cermat, kami menyadari bahwa masih terdapat kekurangan dan kelemahan yang perlu diperbaiki pada laporan proyek akhir "Smart Home Automation" ini. Oleh karena itu, kami mengharapkan masukan dan saran yang konstruktif dari pembaca sebagai bentuk kritik yang akan membantu kami memperbaiki laporan ini secara optimal. Kami berharap laporan ini dapat memberikan manfaat dalam pengembangan ilmu pengetahuan dan teknologi, khususnya dalam bidang otomatisasi rumah. Terima kasih atas perhatian dan waktu yang diberikan.

Depok, 28 Mei 2004

Group 11

## **TABLE OF CONTENT**

<b>PREFACE</b>	<b>2</b>
<b>TABLE OF CONTENT</b>	<b>3</b>
<b>CHAPTER I INTRODUCTION</b>	<b>4</b>
1.1 Problem Statement	4
1.2 Proposed Solution	4
1.3 Acceptance Criteria	5
1.4 Roles and Responsibilities	5
1.5 Timeline and Milestones	5
<b>CHAPTER II IMPLEMENTATION</b>	<b>6</b>
2.1 Hardware Design and Schematic	6
2.2 Software Development	8
2.3 Hardware and Software Integration	19
<b>CHAPTER III TESTING AND EVALUATION</b>	<b>23</b>
3.1 Testing	23
3.2 Result	24
3.3 Evaluation	24
<b>CHAPTER IV CONCLUSION</b>	<b>25</b>
<b>REFERENCES</b>	<b>26</b>
<b>APPENDICES</b>	<b>27</b>
Appendix A: Project Schematic	27
Appendix B: Documentation	27
Appendix C: Related Link	27

# CHAPTER I

## INTRODUCTION

### 1.1 Problem Statement

Sistem penerangan dan pendingin rumah tradisional seringkali memerlukan intervensi manual untuk menyalakan atau mematikannya. Hal ini tidak hanya menuntut perhatian terus-menerus dari penghuninya tetapi juga dapat menyebabkan penggunaan energi yang tidak efisien. Misalnya, lampu mungkin tetap menyala dalam kondisi penerangan yang baik, atau kipas angin dapat terus menyala meskipun suhu turun, sehingga mengakibatkan konsumsi energi yang tidak diperlukan dan tagihan listrik yang lebih tinggi. Khususnya, di banyak rumah:

- Lampu dibiarkan menyala pada siang hari atau ketika cahaya sekitar cukup.
- Perangkat pendingin seperti kipas bekerja terus menerus, berapapun suhu sebenarnya, sehingga menyebabkan pemborosan energi.

### 1.2 Proposed Solution

Solusi yang dipaparkan adalah sistem *Smart Home* yang dimana dapat menyalakan lampu secara otomatis ketika intensitas cahaya pada ruangan lebih kecil dari parameter yang telah ditetapkan dan juga menyalakan kipas ketika suhu pada ruangan melebihi parameter yang telah ditetapkan. Tentunya sistem ini juga dapat menonaktifkan perangkat yaitu lampu dan juga kipas secara otomatis ketika parameter yang telah ditetapkan tidak dilewati.

Tidak hanya itu informasi suhu dan juga intensitas cahaya juga dapat dipantau dengan LCD yang terdapat pada sistem sehingga pengguna dapat melihat informasi tentang intensitas cahaya dan juga suhu pada ruangan. Hal ini tentunya akan menghemat energi karena sistem akan secara otomatis mengatur lampu dan juga kipas yang dimana akan mengatasi masalah yang ada yaitu pemborosan energi akibat kelalaian dalam menggunakan alat - alat seperti lampu dan kipas.

### **1.3 Acceptance Criteria**

Target kriteria yang ingin dicapai pada proyek berikut adalah:

1. Dapat mengambil data dari suhu ruangan dan juga intensitas cahaya dari sensor - sensor yang digunakan.
2. Menyalakan motor ketika suhu ruangan melebihi batas parameter yang telah diset oleh sistem
3. Menyalakan lampu ketika intensitas cahaya pada ruangan kurang dari batas parameter yang telah di set oleh sistem.
4. Dapat mengimplementasikan *interrupt* dan juga protokol 12C

### **1.4 Roles and Responsibilities**

Roles	Responsibilities	Person
Programmer Arduino	Memprogram Arduino Master	Louis Benedict Archie
	Memprogram Arduino Slave	Monica Vierin Pasman
Pendesain skematik proteus dan rangkaian asli	Membuat rangkaian schematic Arduino, sensor, LED, motor DC agar dapat terhubung	Adhelia Putri Maylani
Mendesain PPT dan Readme	Membuat PPT dan Readme	Reiki Putra Dermawan
Pembuat Laporan	Membuat laporan	<ul style="list-style-type: none"><li>• Louis Benedict Archie</li><li>• Monica Vierin Pasman</li><li>• Reiki Putra Dermawan</li><li>• Adhelia Putri Maylani</li></ul>

## 1.5 Timeline and Milestones

No	Kegiatan	Bulan Mei Tanggal																	
		8	9	10	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
1	Perumusan ide																		
2	Finalisasi ide																		
3	Hardware Desain																		
4	Pengembangan Software																		
5	Menyusun Rangkaian																		
6	Integrasi dan Uji coba Hardware dan Software																		
7	Perakitan dan Pengujian Produk Akhir																		

## CHAPTER II

### IMPLEMENTATION

#### 2.1 Hardware Design and Schematic

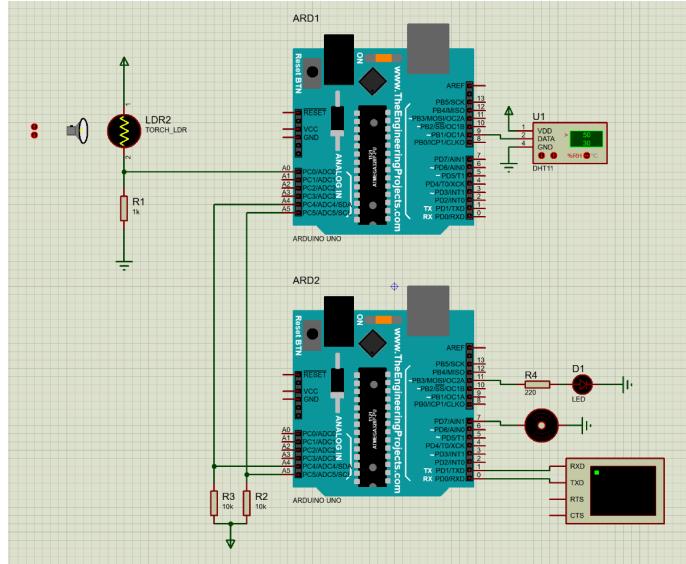


Figure 1. Schema

Pembuatan alat dilakukan dengan terlebih dahulu membuat skema alat atau prototype alat pada software Proteus. Skema ini memudahkan kelompok dalam melakukan perangkaian alat pada rangkaian asli, sehingga meminimalisir kerusakan atau kegagalan komponen asli akibat kesalahan dalam merangkai atau rangkaian yang belum terintegrasi dengan baik.

Komponen yang dibutuhkan dalam mendesain dan merangkai skema beserta fungsinya pada rangkaian antara lain, yaitu:

1. Arduino Uno ATmega328p

Arduino Uno merupakan mikrokontroler berbasis ATmega328p yang berfungsi sebagai otak dari sistem. Dalam proyek ini, digunakan dua unit Arduino Uno, yaitu sebagai master dan slave. Arduino master bertugas membaca data dari sensor DHT11 dan LDR kemudian mengirimkan data tersebut ke Arduino slave. Arduino slave menerima data dari master dan mengendalikan perangkat output seperti LED dan motor DC kipas berdasarkan data yang diterima.

2. Sensor DHT11

DHT11 adalah sensor yang digunakan untuk mengukur suhu dan kelembaban udara. Sensor ini terhubung ke Arduino master, yang akan membaca data suhu dan

kelembaban dari sensor ini. Berdasarkan data yang diperoleh, Arduino master akan mengirimkan informasi tersebut ke Arduino slave untuk mengendalikan motor DC.

### 3. Sensor LDR

Light Dependent Resistor (LDR) adalah sensor yang digunakan untuk mendeteksi intensitas cahaya di lingkungan sekitarnya. Sensor ini juga terhubung ke Arduino master. Arduino master akan membaca resistansi dari LDR yang sudah ditentukan tergantung pada intensitas cahaya. Informasi kemudian dikirimkan ke Arduino slave untuk mengendalikan nyala lampu LED berdasarkan kondisi siang (resistansi rendah) atau malam (resistansi tinggi).

### 4. LED

LED digunakan sebagai indikator visual dalam sistem ini. Arduino slave akan mengontrol LED berdasarkan data yang diterima dari sensor LDR melalui master. LED akan menyala pada malam hari dan mati pada siang hari, sesuai dengan data intensitas cahaya yang dibaca oleh sensor LDR.

### 5. Motor DC

Motor DC digunakan untuk menggerakkan kipas dalam sistem ini. Berdasarkan data suhu yang dibaca oleh sensor DHT11 dan dikirimkan oleh Arduino master, Arduino slave akan mengontrol motor DC untuk menghidupkan atau mematikan kipas. Jika suhu yang terdeteksi melebihi batas, kipas akan dinyalakan untuk membantu mendinginkan ruangan.

### 6. Breadboard

Breadboard adalah papan sirkuit tanpa solder yang digunakan untuk membuat prototipe rangkaian elektronik. Dalam proyek ini, breadboard digunakan untuk menghubungkan berbagai komponen seperti sensor DHT11, sensor LDR, LED, dan resistor ke Arduino tanpa perlu menyolder, sehingga memudahkan pengujian dan memodifikasi rangkaian.

### 7. Resistor ( $10\text{k}\Omega$ , $1\text{k}\Omega$ , $220\Omega$ )

Resistor digunakan untuk membatasi arus yang mengalir ke suatu komponen pada rangkaian. Resistor  $1\text{k}\Omega$  digunakan untuk bekerja dengan sensor LDR guna mengatur sensitivitas pembacaan cahaya, resistor  $10\text{k}\Omega$  digunakan untuk menghubungkan komunikasi antara dua Arduino (master dan slave), memastikan sinyal stabil dan jelas, dan resistor  $220\Omega$  digunakan untuk membatasi arus yang mengalir ke LED, melindunginya dari kerusakan akibat arus yang berlebihan.

## 2.2 Software Development

Guna mempermudah dan pemahaman cara kerja alat, diperlukan kerangka kerja yang telah dibuat flowchart sebagai berikut:

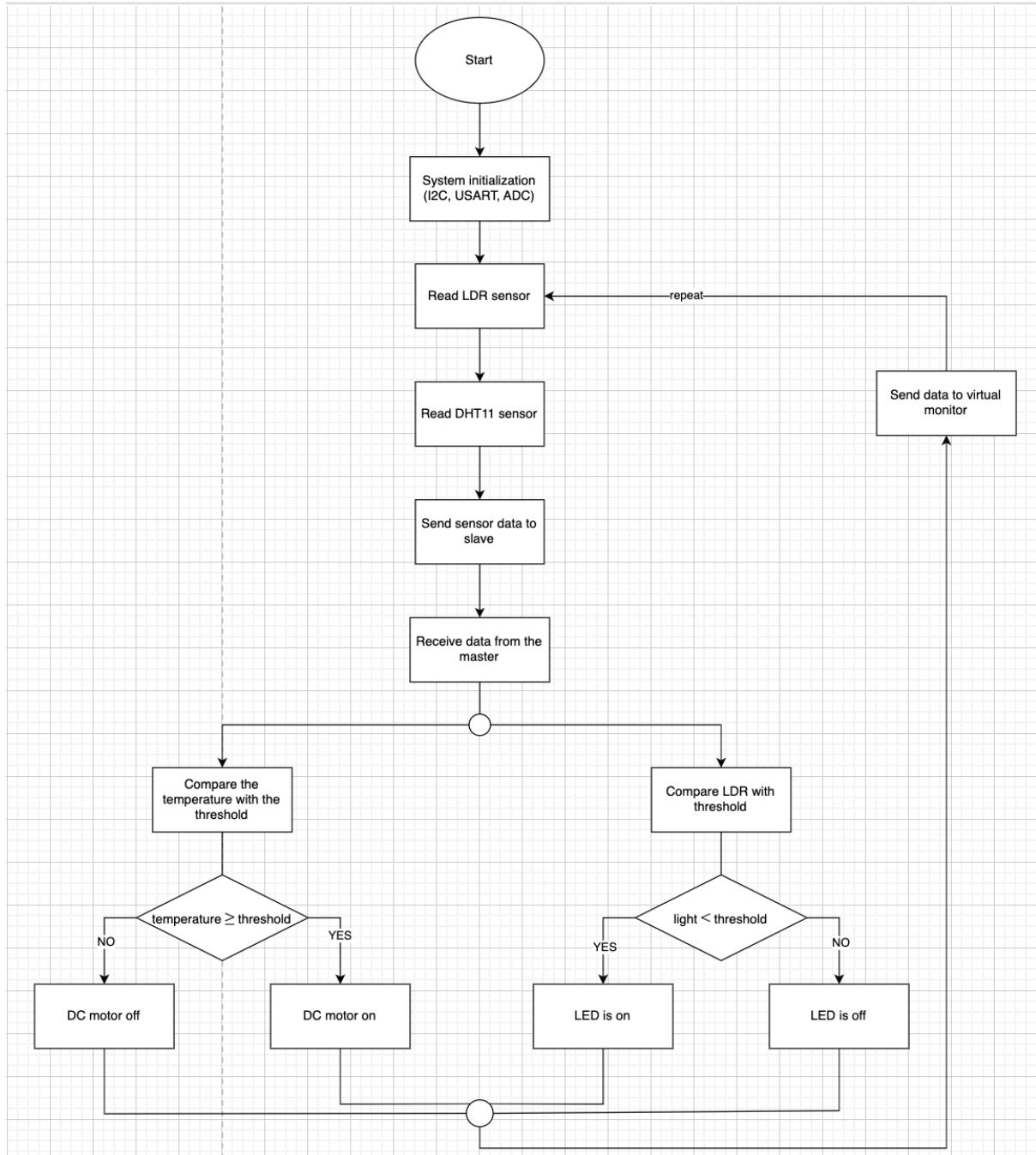


Figure 2. Flowchart

Pengujian dilakukan dengan merakit skema pada komponen asli sesuai desain yang tertera pada *Figure 1*, dan dioperasikan menggunakan kode berikut:

## ● Kode Master

```
#define __SFR_OFFSET 0x00
#include "avr/io.h"

.global main

;=====
main:
    SBI  DDRC, 0          ; Set pin PC0 as input for ADC0
    LDI  R20, 0xC0         ; Internal 2.56V, right-justified data, ADC0
    STS  ADMUX, R20
    LDI  R20, 0x87         ; Enable ADC, ADC prescaler CLK/128
    STS  ADCSRA, R20
    SBI  DDRB, 0           ; Set pin PB0 as output

    LDI  R23, (1<<INT0)   ; Load INT0 bit into R23
    OUT  EIMSK, R23        ; Output R23 to EIMSK register
    LDI  R23, (1<<ISC01)   ; Load ISC01 bit into R23
    STS  EICRA, R23        ; Store R23 into EICRA register

    SEI                      ; Enable global interrupts

    LDI  R21, 1<<TWPS0     ; Set prescaler to 1
    STS  TWSR, R21          ; Store R21 into TWSR register
    LDI  R21, 12              ; Load 12 into R21 (division factor for SCK freq =
400kHz)
    STS  TWBR, R21          ; Store R21 into TWBR register
    LDI  R21, (1<<TWEN)     ; Load TWEN flag into R21
    STS  TWCR, R21          ; Store R21 into TWCR register

    CLR  R25                  ; Initialize temperature register

loop:
    RCALL read_LDR          ; Read data from LDR
    RCALL read_DHT11         ; Read data from DHT11
    RCALL send_data          ; Send data to slave via I2C
    RJMP loop                 ; Repeat the loop

;=====
read_LDR:
    LDI  R20, 0xC7          ; Set ADSC in ADCSRA to start conversion
    STS  ADCSRA, R20
wait_ADC:
    LDS  R21, ADCSRA        ; Check ADIF flag in ADCSRA
    SBRS R21, 4               ; Skip jump when conversion is done (flag set)
```

```

RJMP  wait_ADC          ; Loop until ADIF flag is set
LDI   R17, 0xD7          ; Set ADIF flag again
STS   ADCSRA, R17        ; So that controller clears ADIF
LDS   R18, ADCL          ; Get low-byte result from ADCL
LDS   R19, ADCH          ; Get high-byte result from ADCH
RET

read_DHT11:
    SBI   DDRB, 1           ; Set bit 1 of DDRB (PB1 configured as output)
    CBI   PORTB, 1          ; Clear bit 1 of PORTB (PB1 set low)
    RCALL delay_20ms        ; Call delay_20ms subroutine
    SBI   PORTB, 1          ; Set bit 1 of PORTB (PB1 set high)
    CBI   DDRB, 1           ; Clear bit 1 of DDRB (PB1 configured as input)

w1:
    SBIC  PINB, 1           ; Skip next instruction if bit 1 of PINB is clear
    RJMP  w1                ; Jump to w1 (Wait for DHT11 low pulse)

w2:
    SBIS  PINB, 1           ; Skip next instruction if bit 1 of PINB is set
    RJMP  w2                ; Jump to w2 (Wait for DHT11 high pulse)

w3:
    SBIC  PINB, 1           ; Skip next instruction if bit 1 of PINB is clear
    RJMP  w3                ; Jump to w3 (Wait for DHT11 low pulse)

    RCALL DHT11_reading    ; Read humidity (1st byte)
    RCALL DHT11_reading    ; Read humidity (2nd byte)
    RCALL DHT11_reading    ; Read temperature (3rd byte)

RET

send_data:
    RCALL I2C_start         ; Call I2C_start subroutine
    LDI   R27, 0b10010000   ; Load 0b10010000 into R27
    RCALL I2C_write          ; Call I2C_write subroutine
    MOV   R27, R18            ; Move low byte of LDR data to R27
    RCALL I2C_write          ; Call I2C_write subroutine
    MOV   R27, R19            ; Move high byte of LDR data to R27
    RCALL I2C_write          ; Call I2C_write subroutine
    MOV   R27, R25            ; Move DHT11 data to R27
    RCALL I2C_write          ; Call I2C_write subroutine
    RCALL I2C_stop           ; Call I2C_stop subroutine
RET

I2C_start:

```

```

LDI    R21, (1<<TWINT) | (1<<TWSTA) | (1<<TWEN) ; Load TWINT, TWSTA, and TWEN flags
into R21
STS    TWCR, R21          ; Store R21 into TWCR register
wt1:
LDS    R21, TWCR          ; Load value from TWCR register into R21
SBRS   R21, TWINT         ; Skip next instruction if TWINT flag is set
RJMP   wt1                ; Jump to wt1 (Wait for end of transmission)
RET

I2C_write:
STS    TWDR, R27          ; Store R27 into TWDR register
LDI    R21, (1<<TWINT) | (1<<TWEN) ; Load TWINT and TWEN flags into R21
STS    TWCR, R21          ; Store R21 into TWCR register
wt2:
LDS    R21, TWCR          ; Load value from TWCR register into R21
SBRS   R21, TWINT         ; Skip next instruction if TWINT flag is set
RJMP   wt2                ; Jump to wt2 (Wait for end of transmission)
RET

I2C_stop:
LDI    R21, (1<<TWINT) | (1<<TWSTO) | (1<<TWEN) ; Load TWINT, TWSTO, and TWEN flags
into R21
STS    TWCR, R21          ; Store R21 into TWCR register
RET

DHT11_reading:
LDI    R24, 8              ; Load 8 into R24
CLR    R25                ; Clear data register
w4:
SBIS   PINB, 1             ; Skip next instruction if bit 1 of PINB is set
RJMP   w4                  ; Jump to w4 (Detect data bit - high pulse)
RCALL  delay_timer0        ; Call delay_timer0 subroutine (Wait 50us)
SBIS   PINB, 1             ; Skip next instruction if bit 1 of PINB is set
RJMP   skp                 ; Jump to skp
SEC
ROL    R25                ; Rotate left through carry - Shift in 1 into LSB data
register
RJMP   w5                  ; Jump to w5 (Wait for low pulse)
skp:
LSL    R25                ; Logical shift left - Shift in 0 into LSB data register
w5:
SBIC   PINB, 1             ; Skip next instruction if bit 1 of PINB is clear
RJMP   w5                  ; Jump to w5 (Wait for DHT11 low pulse)
DEC    R24
BRNE  w4                  ; Branch if not equal to w4 (Go back & detect next bit)

```

```

RET

;=====

; Delay subroutines

delay_20ms:           ; Define delay_20ms subroutine
LDI    R21, 255        ; Load 255 into R21
l1:                  ; Label l1
LDI    R22, 210        ; Load 210 into R22
l2:                  ; Label l2
LDI    R23, 2          ; Load 2 into R23
l3:                  ; Label l3
DEC    R23             ; Decrement R23
BRNE  l3              ; Branch to l3 if R23 is not zero
DEC    R22             ; Decrement R22
BRNE  l2              ; Branch to l2 if R22 is not zero
DEC    R21             ; Decrement R21
BRNE  l1              ; Branch to l1 if R21 is not zero
RET

delay_2s:            ; Define delay_2s subroutine
LDI    R21, 255        ; Load 255 into R21
l4:                  ; Label l4
LDI    R22, 255        ; Load 255 into R22
l5:                  ; Label l5
LDI    R23, 164         ; Load 164 into R23
l6:                  ; Label l6
DEC    R23             ; Decrement R23
BRNE  l6              ; Branch to l6 if R23 is not zero
DEC    R22             ; Decrement R22
BRNE  l5              ; Branch to l5 if R22 is not zero
DEC    R21             ; Decrement R21
BRNE  l4              ; Branch to l4 if R21 is not zero
RET

delay_timer0:         ; Define delay_timer0 subroutine
CLR   R20              ; Clear R20
OUT   TCNT0, R20        ; Output R20 to TCNT0 register (Initialize timer0 with
count=0)
LDI   R20, 100          ; Load 100 into R20
OUT   OCR0A, R20         ; Output R20 to OCR0A register (OCR0 = 100)
LDI   R20, 0b00001010   ; Load binary 00001010 into R20
OUT   TCCR0B, R20         ; Output R20 to TCCR0B register (Timer0: CTC mode,
prescaler 8)
cTimer0:               ; Label cTimer0

```

```

IN    R20, TIFR0      ; Read TIFR0 into R20
SBRSS R20, OCF0A      ; Skip next instruction if OCF0A flag is set
RJMP  cTimer0         ; Jump to cTimer0 (Loop back & check OCF0 flag)
CLR   R20             ; Clear R20
OUT   TCCR0B, R20     ; Output R20 to TCCR0B register (Stop timer0)
LDI   R20, (1<<OCF0A) ; Load OCF0A flag into R20
OUT   TIFR0, R20      ; Output R20 to TIFR0 register (Clear OCF0 flag)
RET

```

Program ini ditulis dalam bahasa assembly merupakan program sebagai master dalam sistem komunikasi I2C pada project ini. Program ini bertanggung jawab untuk mengkonfigurasi dan mengendalikan hardware untuk membaca data dari sensor LDR dan DHT11, serta mengirim data melalui I2C. Pertama, pin PC0 diatur sebagai input untuk ADC, dan pin PB0 diatur sebagai output. Konfigurasi ADC diatur untuk menggunakan tegangan referensi internal dan prescaler. Interrupt eksternal INT0 diaktifkan untuk menangani interupsi. Dalam loop utama, program memanggil subrutin untuk membaca nilai LDR dan DHT11, lalu mengirimkan data melalui I2C. Subrutin ‘read\_LDR’ memulai konversi ADC dan menunggu hingga selesai, kemudian membaca nilai ADC. Subrutin ‘read\_DHT11’ menginisialisasi komunikasi dengan sensor DHT11 dan membaca data suhu dan kelembaban. Subrutin ‘send\_data’ mengirimkan data yang dibaca melalui I2C. Subrutin ‘I2C\_start’, ‘I2C\_write’, dan ‘I2C\_stop’ menangani protokol komunikasi I2C. Program juga mencakup rutinitas penundaan untuk mengatur timing dalam komunikasi dengan sensor.

- Kode Slave

```

#define __SFR_OFFSET 0x00
#include "avr/io.h"

.global main

main:
    ; Set PB3 as output (LED)
    SBI DDRB, 3           ; Set PB3 as output for LED
    ; Set PD7 as output (fan motor)
    SBI DDRD, 7           ; Set PD7 as output for fan motor

    ; USART Initialization
    CLR R24               ; Clear R24 register
    STS UCSR0A, R24        ; Store R24 in UCSR0A register
    STS UBRR0H, R24        ; Store R24 in UBRR0H register
    LDI R24, 103            ; Load 103 into R24 (baud rate)

```

```

STS UBRR0L, R24      ; Store R24 in UBRR0L register
LDI R24, (1<<RXEN0) | (1<<TXEN0) ; Load enable flags into R24
STS UCSR0B, R24      ; Store R24 in UCSR0B register
LDI R24, (1<<UCSZ00) | (1<<UCSZ01) ; Load UCSZ flags into R24
STS UCSR0C, R24      ; Store R24 in UCSR0C register

; ADC Initialization
SBI DDRC, 0          ; Set PC0 as input for ADC
LDI R20, 0xC0         ; Load 0xC0 into R20
STS ADMUX, R20        ; Store R20 in ADMUX register (ADC reference and channel
selection)
LDI R20, 0x87         ; Load 0x87 into R20
STS ADCSRA, R20       ; Store R20 in ADCSRA register (ADC enable and prescaler)

main_loop:
RJMP loop1           ; Jump to loop1

loop1:
RCALL delay_20ms     ; Call delay subroutine
; Read temperature from I2C
RCALL I2C_init        ; Call I2C initialization
RCALL I2C_listen       ; Call I2C listen
RCALL I2C_read         ; Call I2C read (low byte of LDR data)
MOV R18, R26           ; Move data from R26 to R18
RCALL I2C_read         ; Call I2C read (high byte of LDR data)
MOV R19, R26           ; Move data from R26 to R19
RCALL I2C_read         ; Call I2C read (temperature data from DHT11)
MOV R27, R26           ; Move data from R26 to R27
RCALL printval         ; Call print value subroutine
RCALL space             ; Call space subroutine
RCALL check_temp        ; Call check temperature subroutine
RJMP loop2             ; Jump to loop2

loop2:
LDI R17, 0x32          ; Load threshold value into R17 (50 in decimal)
; Read and print ADC value
RCALL print_ADC         ; Call print_ADC subroutine
RCALL newline            ; Call newline subroutine
RCALL check_light        ; Call check light subroutine
RJMP loop1             ; Jump to loop1

print_ADC:
; Combine high and low bytes to form the 10-bit ADC value
LDI R25, 0              ; Load 0 into R25
MOV R25, R19             ; Move data from R19 to R25

```

```

SWAP R25          ; Swap nibbles in R25
ANDI R25, 0x03    ; Mask higher bits, keep lower 2 bits
MOV R26, R18      ; Move data from R18 to R26
OR R26, R25       ; OR R26 with R25
RCALL printval   ; Call print value subroutine
RET              ; Return from subroutine

; I2C Initialization
I2C_init:
LDI R21, 0b10010000 ; Load I2C address into R21
STS TWAR, R21        ; Store R21 in TWAR register
LDI R21, (1<<TWEN) ; Load enable flag into R21
STS TWCR, R21        ; Store R21 in TWCR register
LDI R21, (1<<TWINT) | (1<<TWEN) | (1<<TWEA) ; Load control flags into R21
STS TWCR, R21        ; Store R21 in TWCR register
RET                  ; Return from subroutine

; I2C Listen
I2C_listen:
LDS R21, TWCR      ; Load TWCR into R21
SBRS R21, TWINT    ; Skip if TWINT is set
RJMP I2C_listen    ; Jump to I2C_listen
RET                  ; Return from subroutine

; I2C Read
I2C_read:
LDI R21, (1<<TWINT) | (1<<TWEA) | (1<<TWEN) ; Load control flags into R21
STS TWCR, R21        ; Store R21 in TWCR register
wait:
LDS R21, TWCR      ; Load TWCR into R21
SBRS R21, TWINT    ; Skip if TWINT is set
RJMP wait           ; Jump to wait
LDS R26, TWDR      ; Load TWDR into R26
RET                  ; Return from subroutine

delay_20ms:
LDI R21, 255        ; Load 255 into R21
lo1:
LDI R22, 210        ; Load 210 into R22
lo2:
LDI R23, 2           ; Load 2 into R23
lo3:
DEC R23             ; Decrement R23
BRNE lo3            ; Branch if not equal to zero
DEC R22             ; Decrement R22

```

```

    BRNE  lo2          ; Branch if not equal to zero
    DEC   R21          ; Decrement R21
    BRNE  lo1          ; Branch if not equal to zero
    RET               ; Return from subroutine

; Print value subroutine
printval:
    MOV  R16, R26        ; Move data from R26 to R16
    RCALL ASCII_MSD     ; Call ASCII_MSD subroutine
    RCALL LCD_buffer     ; Call LCD_buffer subroutine
    STS  UDR0, R16       ; Store R16 in UDR0 register
    RCALL ASCII_LSD      ; Call ASCII_LSD subroutine
    RCALL LCD_buffer     ; Call LCD_buffer subroutine
    STS  UDR0, R16       ; Store R16 in UDR0 register
    RET               ; Return from subroutine

; USART buffer check
LCD_buffer:
    LDS  R24, UCSR0A    ; Load UCSR0A into R24
    SBRS R24, UDRE0      ; Skip if UDRE0 is set
    RJMP LCD_buffer      ; Jump to LCD_buffer
    RET               ; Return from subroutine

; Convert to ASCII Most Significant Digit
ASCII_MSD:
    MOV  R23, R16        ; Move data from R16 to R23
    ANDI R16, 0xFO       ; Mask lower 4 bits of R16
    SWAP R16             ; Swap nibbles in R16
    SUBI R16, -48         ; Convert to ASCII
    MOV  R28, R16        ; Move data from R16 to R28
    SUBI R28, 58          ; Check if digit is greater than 9
    BRPL A_F_D1          ; Branch if less than or equal to 9

11:
    RET               ; Return from subroutine

A_F_D1:
    SUBI R16, -7          ; Convert to hex ASCII
    RJMP 11              ; Jump to 11

; Convert to ASCII Least Significant Digit
ASCII_LSD:
    MOV  R16, R23        ; Move data from R23 to R16
    ANDI R16, 0x0F       ; Mask higher 4 bits of R16
    SUBI R16, -48         ; Convert to ASCII
    MOV  R28, R16        ; Move data from R16 to R28
    SUBI R28, 58          ; Check if digit is greater than 9

```

```

    BRPL A_F_D0          ; Branch if less than or equal to 9
12:
    RET                  ; Return from subroutine
A_F_D0:
    SUBI R16, -7         ; Convert to hex ASCII
    RJMP 12              ; Jump to 12

; Print space character
space:
    RCALL LCD_buffer    ; Call LCD_buffer subroutine
    LDI R25, 0x20        ; Load space character into R25
    STS UDR0, R25        ; Store R25 in UDR0 register
    RET                  ; Return from subroutine

; Print newline characters
newline:
    RCALL LCD_buffer    ; Call LCD_buffer subroutine
    LDI R24, 0x0A        ; Load newline character into R24
    STS UDR0, R24        ; Store R24 in UDR0 register
    RCALL LCD_buffer    ; Call LCD_buffer subroutine
    LDI R24, 0x0D        ; Load carriage return character into R24
    STS UDR0, R24        ; Store R24 in UDR0 register
    RET                  ; Return from subroutine

; Check temperature
check_temp:
    CPI R27, 30          ; Compare temperature with threshold (30)
    BRSH fan_on          ; Branch if temperature is greater than or equal
    CBI PORTD, 7          ; Clear PD7 (turn off fan)
    RET                  ; Return from subroutine

fan_on:
    SBI PORTD, 7          ; Set PD7 (turn on fan)
    RET                  ; Return from subroutine

; Check light and control LED
check_light:
    CP R17, R18          ; Compare LDR value with threshold
    BRLO LED_off          ; Branch if LDR value is lower
    SBI PORTB, 3          ; Set PB3 (turn on LED)
    RET                  ; Return from subroutine

; Turn off LED
LED_off:
    CBI PORTB, 3          ; Clear PB3 (turn off LED)

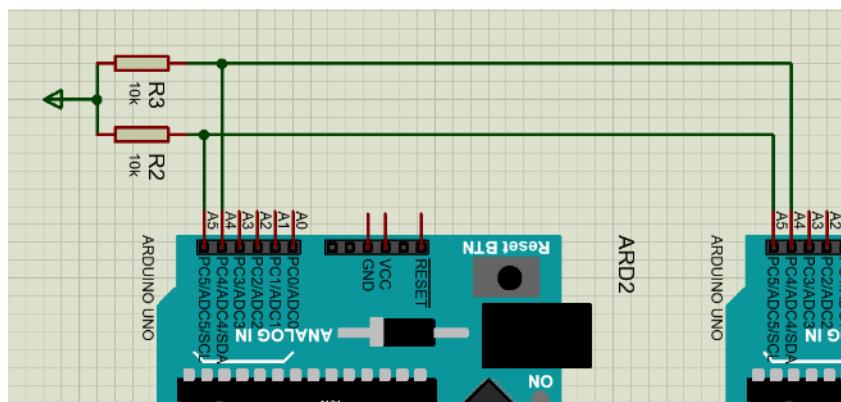
```

```
RET ; Return from subroutine
```

Program slave ini berfungsi untuk menerima data dari master melalui I2C, membaca nilai ADC, dan mengontrol perangkat berdasarkan nilai yang diterima. Program dimulai dengan mengkonfigurasi pin PB3 sebagai output untuk LED dan PD7 sebagai output untuk motor kipas. Selanjutnya, kode menginisialisasi USART untuk komunikasi serial dan ADC untuk membaca nilai dari sensor yang terhubung ke PC0. Dalam loop utama, kode mendapatkan data dari master melalui I2C, membaca nilai sensor LDR dan suhu dan DHT11, serta mencetak nilai ini melalui USART. Selain itu, kode juga memeriksa suhu yang diterima untuk menghidupkan atau mematikan kipas berdasarkan ambang batas (30 derajat), dan memeriksa nilai sensor cahaya untuk menghidupkan atau mematikan LED berdasarkan ambang batas (50 dalam desimal). Subrutin seperti ‘printval’, ‘space’, dan ‘newline’ digunakan untuk mencetak data ke USART, sementara ‘check\_temp’ dan ‘check\_light’ mengatur logika kontrol untuk kipas dan LED.

### 2.3 Hardware and Software Integration

Dalam proyek *smart home automation* ini, berbagai komponen hardware dan software diintegrasikan untuk menciptakan sistem yang fungsional. Berikut adalah deskripsi dari beberapa komponen dan bagaimana mereka berinteraksi:



Komunikasi antara Arduino master dan slave dilakukan melalui protokol I2C. Pada kode master, pin SDA dan SCL diatur untuk mengirimkan data sensor ke slave. Pada bagian kode berikut, inisialisasi I2C dilakukan, dan data dari sensor dikirim ke Arduino slave.

```
RCALL I2C_start      ; Call I2C_start subroutine
LDI   R27, 0b10010000 ; Load 0b10010000 into R27
RCALL I2C_write       ; Call I2C_write subroutine
MOV   R27, R18         ; Move low byte of LDR data to R27
RCALL I2C_write       ; Call I2C_write subroutine
MOV   R27, R19         ; Move high byte of LDR data to R27
```

```

RCALL I2C_write      ; Call I2C_write subroutine
MOV  R27, R25         ; Move DHT11 data to R27
RCALL I2C_write      ; Call I2C_write subroutine
RCALL I2C_stop       ; Call I2C_stop subroutine

```

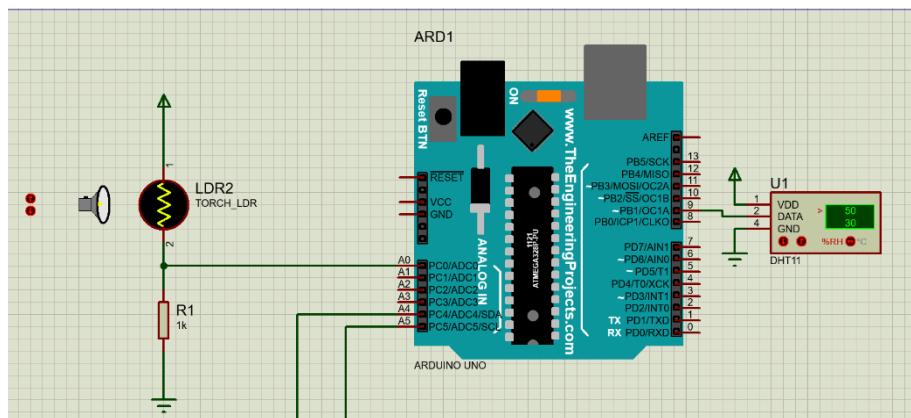
Pada kode di atas, master memulai komunikasi I2C dengan ‘I2C\_start’, mengirim alamat slave, dan kemudian mengirimkan data dari sensor LDR (byte rendah dan tinggi) serta data suhu dari sensor DHT11. Komunikasi I2C dihentikan dengan ‘I2C\_stop’.

```

RCALL I2C_init       ; Call I2C initialization
RCALL I2C_listen     ; Call I2C listen
RCALL I2C_read        ; Call I2C read (low byte of LDR data)
MOV R18, R26          ; Move data from R26 to R18
RCALL I2C_read        ; Call I2C read (high byte of LDR data)
MOV R19, R26          ; Move data from R26 to R19
RCALL I2C_read        ; Call I2C read (temperature data from DHT11)
MOV R27, R26          ; Move data from R26 to R27

```

Pada kode di atas, slave menginisialisasi I2C dengan ‘I2C\_init’, mendengarkan data dari master dengan ‘I2C\_listen’, dan kemudian membaca data yang dikirim oleh master dengan ‘I2C\_read’. Data yang diterima disimpan ke register untuk diproses lebih lanjut.



Sensor LDR terhubung ke pin ADC pada Arduino master untuk mengukur tingkat cahaya, dan sensor DHT11 terhubung untuk membaca suhu dan kelembaban.

```

LDI    R20, 0xC7      ; Set ADSC in ADCSRA to start conversion
STS    ADCSRA, R20
wait_ADC:
LDS   R21, ADCSRA      ; Check ADIF flag in ADCSRA
SBRS  R21, 4            ; Skip jump when conversion is done (flag set)
RJMP  wait_ADC         ; Loop until ADIF flag is set
LDS   R18, ADCL         ; Get low-byte result from ADCL
LDS   R19, ADCH         ; Get high-byte result from ADCH

; Membaca data dari sensor DHT11
RCALL read_DHT11       ; Read data from DHT11

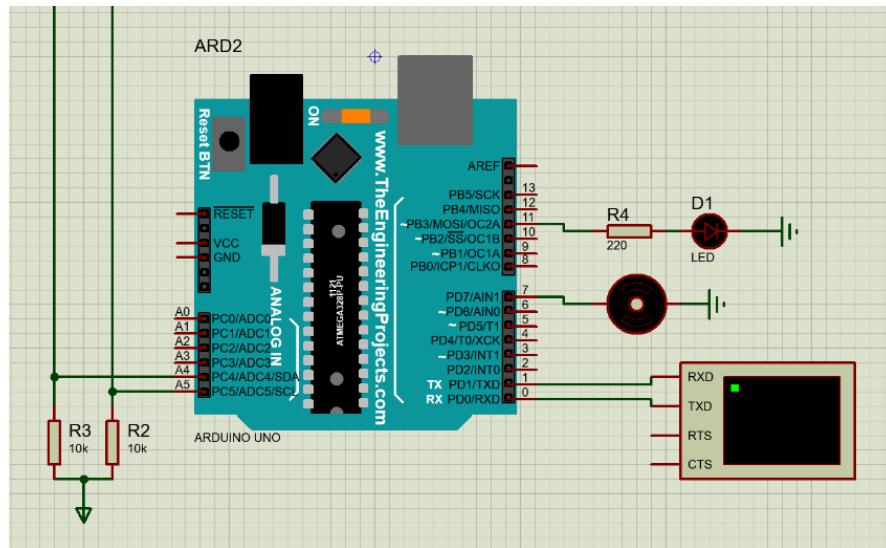
```

```

read_DHT11:
    SBI    DDRB, 1           ; Set bit 1 of DDRB (PB1 configured as output)
    CBI    PORTB, 1          ; Clear bit 1 of PORTB (PB1 set low)
    RCALL delay_20ms        ; Call delay_20ms subroutine
    SBI    PORTB, 1          ; Set bit 1 of PORTB (PB1 set high)
    CBI    DDRB, 1           ; Clear bit 1 of DDRB (PB1 configured as input)

```

Pada kode di atas, master membaca data dari sensor LDR dengan menginisialisasi konversi ADC dan menunggu sampai konversi selesai. Data yang dibaca dari sensor LDR disimpan di register R18 dan R19. Data suhu dari sensor DHT11 dibaca dengan mengambil subroutines ‘read\_DHT11’.



Motor DC dan LED dikendalikan oleh Arduino slave berdasarkan data yang diterima dari master. Jika suhu melebihi ambang batas tertentu ( $\geq 30^\circ$ ), motor kipas akan diaktifkan. LED akan menyala berdasarkan tingkat cahaya yang terdeteksi oleh sensor LDR ( $< 50$ ).

```

; Check temperature
check_temp:
    CPI R27, 30           ; Compare temperature with threshold (30)
    BRSH fan_on            ; Branch if temperature is greater than or equal
    CBI PORTD, 7            ; Clear PD7 (turn off fan)
    RET                     ; Return from subroutine

fan_on:
    SBI PORTD, 7           ; Set PD7 (turn on fan)
    RET                     ; Return from subroutine

; Check light and control LED
check_light:
    CP R17, R18            ; Compare LDR value with threshold
    BRLO LED_off            ; Branch if LDR value is lower

```

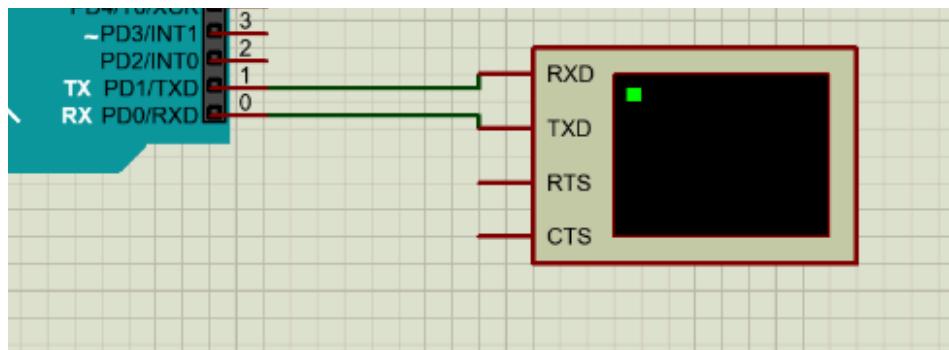
```

SBI PORTB, 3           ; Set PB3 (turn on LED)
RET                   ; Return from subroutine

; Turn off LED
LED_off:
CBI PORTB, 3           ; Clear PB3 (turn off LED)
RET                   ; Return from subroutine

```

Pada kode di atas, slave mengontrol motor berdasarkan suhu yang diterima dari master. Jika suhu lebih dari atau sama dengan 30 derajat Celcius, motor diaktifkan dengan mengatur bit pada PORTD. LED dikendalikan berdasarkan data dari sensor LDR; jika tingkat cahaya kurang dari ambang batas, LED dimatikan dengan clear bit pada PORTB.



Data sensor ditampilkan ke virtual monitor melalui komunikasi serial USART yang diinisialisasi di kode slave. Data dikirim ke monitor untuk visualisasi dan analisis.

```

; USART Initialization
CLR R24                 ; Clear R24 register
STS UCSR0A, R24          ; Store R24 in UCSR0A register
STS UBRROH, R24          ; Store R24 in UBRROH register
LDI R24, 103              ; Load 103 into R24 (baud rate)
STS UBRROL, R24          ; Store R24 in UBRROL register
LDI R24, (1<<RXEN0) | (1<<TXEN0) ; Load enable flags into R24
STS UCSR0B, R24          ; Store R24 in UCSR0B register
LDI R24, (1<<UCSZ00) | (1<<UCSZ01) ; Load UCSZ flags into R24
STS UCSR0C, R24          ; Store R24 in UCSR0C register

; send data to virtual monitor
RCALL printval           ; Call print value subroutine
RCALL space               ; Call space subroutine
RCALL newline             ; Call newline subroutine

```

Pada kode di atas, inisialisasi USART dilakukan dengan mengkonfigurasi register yang diperlukan untuk mengatur baud rate dan mengaktifkan RX dan TX. Data dari sensor kemudian dikirim ke virtual monitor dengan memanggil subroutines ‘printval’, ‘space’, dan ‘newline’ untuk menampilkan nilai sensor dan mengatur format tampilan.

## CHAPTER III

### TESTING AND EVALUATION

#### 3.1 Testing

Setelah pembuatan kode dan rangkaian selesai dilakukan, dilakukan testing untuk menguji apakah program dapat berjalan dengan benar. Akan dilakukan testing pada empat kondisi, yakni ketika suhu di dalam rumah panas (di atas 30°C), ketika suhu di dalam rumah sejuk (di bawah 30°C), ketika cahaya di dalam rumah terang (di atas 50 lux), dan ketika cahaya di dalam rumah gelap (di bawah 50 lux).

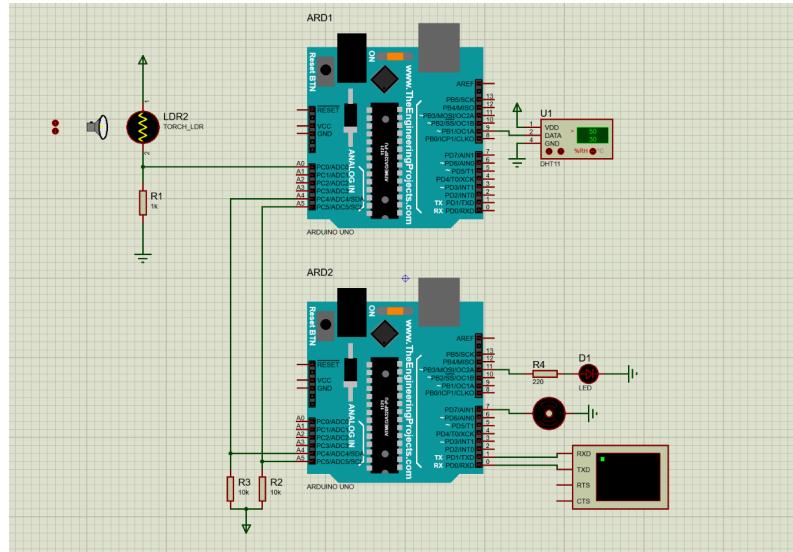


Figure 3. Testing in Proteus

Prediksi hasil yang akan muncul berdasarkan keempat kondisi tersebut secara berturut-turut adalah kipas dalam keadaan hidup, kipas dalam keadaan mati, lampu dalam keadaan mati, dan lampu dalam keadaan hidup. Tanpa mempertimbangkan kondisi yang terjadi di dalam rumah, serial monitor akan terus menyala untuk memberikan pembaharuan informasi mengenai suhu dan tingkat kecerahan di dalam rumah. Pada kolom kiri, serial monitor akan menampilkan hasil pembacaan suhu dari DHT11 dalam satuan °C. Sementara itu, pada kolom kanan, serial monitor akan menampilkan hasil pembacaan tingkat kecerahan dari LDR/photoresistor dalam satuan lux.

### 3.2 Result

Hasil pengetesan rangkaian dapat dilihat pada rangkaian proteus dan rangkaian fisik. Hasil dari pengetesan rangkaian proteus dapat dilihat pada *Figure 4*. Dapat dilihat bahwa kipas dan LED dapat mati/hidup sesuai dengan penjabaran kondisi testing pada bagian 3.1. Serial monitor pun dapat memberikan informasi secara akurat.

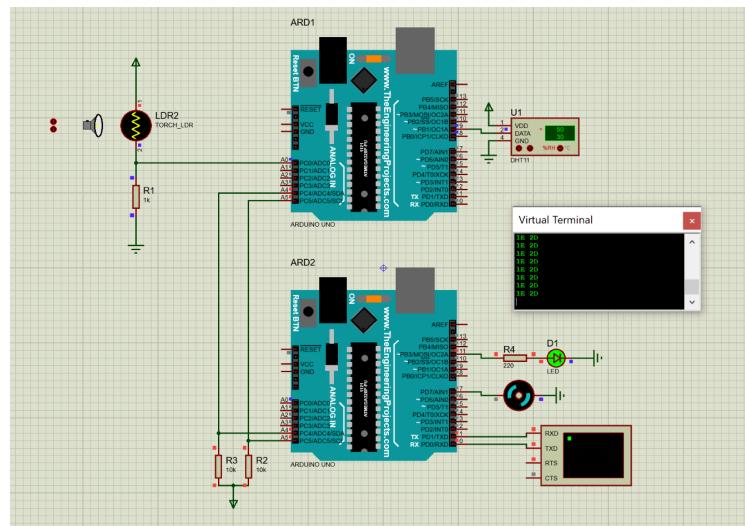


Figure 4. Testing in Proteus

Dari hasil testing rangkaian proteus, dapat dibuktikan bahwa arduino yang menjadi master dapat membaca dan mengirim data hasil pembacaan sensor DHT11 dan LDR/photoresistor dengan akurat. Arduino yang menjadi slave sendiri dapat menerima data dengan baik dan memproses data tersebut dalam pengkondisian menggunakan CPI (Compare with Immediate), sehingga kipas dan LED dapat terprogram dengan baik. Serial monitor sendiri dapat menampilkan update data dengan segera tanpa adanya delay, seperti halnya proses kerja kipas dan LED.

Testing pada rangkaian asli sendiri juga sudah dilakukan. Akan tetapi, alatnya tidak dapat berjalan sebaik jalannya alat pada simulator menggunakan proteus. Terdapat masalah pada salah satu arduino yang membuat transmisi data menggunakan protokol I2C tidak dapat dilakukan. Kedua arduino harus dapat saling terhubung satu sama lain melalui port SDA dan SCL sebagai syarat dilakukannya transmisi data. Akibat tidak berjalannya protokol tersebut, kami pun tidak dapat lanjut untuk mengecek hasil compare dan display data.

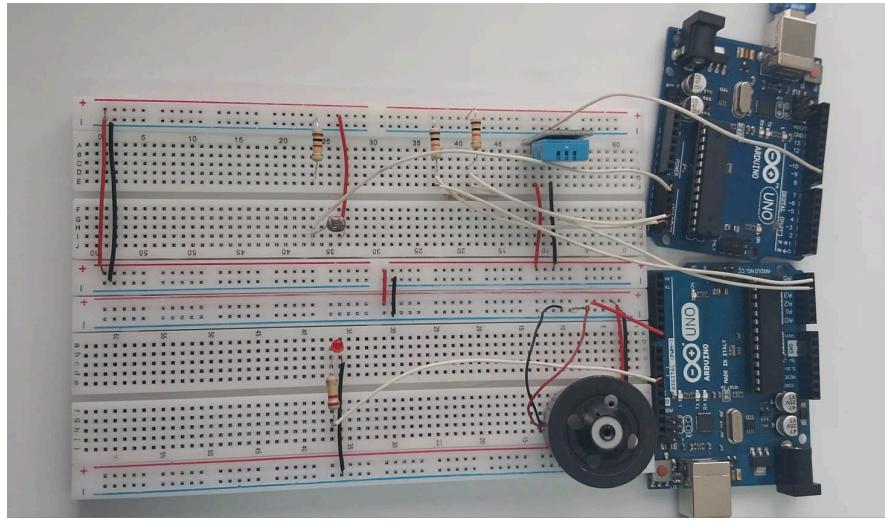


Figure 5. Testing with real components

### 3.3 Evaluation

Dari percobaan dan pengujian yang telah dilakukan, terlihat bahwa proyek ini telah berhasil bekerja sesuai dengan kriteria yang ditentukan. 2 sensor yang digunakan dapat bekerja dengan seharusnya, dimana LDR dapat membaca nilai intensitas cahaya dan menentukan kapan LED akan menyala atau mati serta DHT11 dapat membaca nilai suhu dan menentukan kapan kipas akan berputar atau tidak. Untuk kedepannya, proyek ini dapat dikembangkan dengan menambahkan LCD untuk menampilkan nilai suhu dan intensitas cahaya pada ruangan. Dengan adanya LCD, pengguna alat ini dapat lebih mudah memantau kondisi lingkungan sehingga nantinya dapat dilakukan penyesuaian pada nilai ambang jika diperlukan.

## **CHAPTER IV**

### **CONCLUSION**

Proyek “Smart Home Automation” yang telah kami buat dirancang untuk rumah yang memerlukan otomatisasi untuk menyalakan lampu ketika malam hari dan menyalakan kipas ketika suhu ruangan terasa panas. Pembacaan intensitas cahaya pada alat ini dilakukan dengan menggunakan LDR (Light Dependent Resistor) secara berkala. Untuk menentukan kapan LED akan menyala berdasarkan hasil pembacaan LDR, digunakan nilai ambang yang sesuai dengan kondisi dimana LED perlu dinyalakan. Selain LDR, juga digunakan DHT11 untuk membaca nilai suhu pada ruangan. Penentuan kapan kipas akan menyala dilakukan juga dilakukan dengan menggunakan nilai ambang dimana ketika hasil pembacaan DHT11 lebih besar atau sama dengan nilai ambang yang ditentukan, maka kipas akan berputar. Untuk mengujinya, kami menetapkan nilai ambang 50 lux dan 30 derajat Celcius. Sehingga ketika hasil pembacaan LDR kurang dari 50 lux, LED akan menyala dan ketika hasil pembacaan DHT11 lebih dari atau sama dengan 30 derajat Celcius, kipas akan berputar.

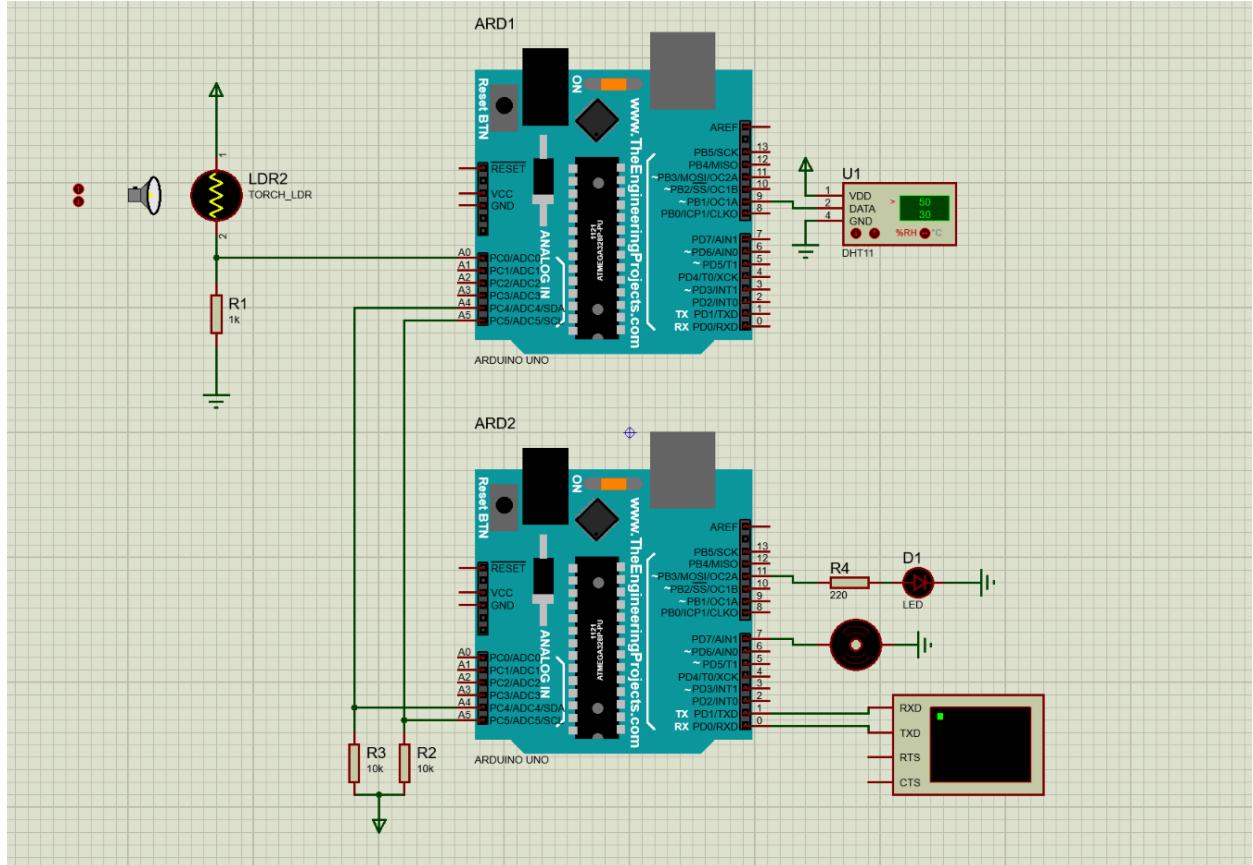
Dari percobaan yang telah dilakukan dapat disimpulkan bahwa proyek “Smart Home Automation” telah terlaksana dengan baik dan dapat bekerja sesuai dengan target dan parameter yang telah ditentukan. Alat ini dapat menyalakan LED ketika nilai lux kurang dari 50 lux dan menyalakan kipas ketika suhu yang dibaca DHT11 lebih dari atau sama dengan 30 derajat Celcius. Semoga dengan proyek yang telah kami buat, implementasi dari “Smart Home Automation” dapat diterapkan untuk menambahkan kenyamanan dan efisiensi energi yang lebih baik.

## REFERENCES

- [1] M. Harditya & M. N. Faza. MODUL 1: ASSEMBLY WITH ARDUINO. emas2.ui.ac.id  
[https://emas2.ui.ac.id/pluginfile.php/4335787/mod\\_resource/content/1/Modul%202%20SSF%20Introduction%20to%20Assembly%20%20I\\_O%20Programming.pdf](https://emas2.ui.ac.id/pluginfile.php/4335787/mod_resource/content/1/Modul%202%20SSF%20Introduction%20to%20Assembly%20%20I_O%20Programming.pdf) (accessed May 21, 2024).
- [2] “Modul 3 SSF : Analog to Digital Converter.” docs.google.com.  
<https://docs.google.com/document/d/1arLt3fqXRw-WgkbqlP1RwYs-XJy9-QAFFEcM21M3u44/edit> (accessed May 21, 2024).
- [3] “Modul 4 SSF : Serial Port.” docs.google.com.  
[https://docs.google.com/document/d/1rRWvBgL3Nsb\\_h10131A-1kiGQkrGGNLedYoeVIGR9zg/edit](https://docs.google.com/document/d/1rRWvBgL3Nsb_h10131A-1kiGQkrGGNLedYoeVIGR9zg/edit) (accessed May 21, 2024).
- [4] “Modul 8 SSF : SPI & I2C.” docs.google.com.  
<https://docs.google.com/document/d/1CsIbwLVUrsKjZ3YhyF0J-gsGWNu1RCQu7JTYMCx3cFY/edit> (accessed May 21, 2024).
- [5] “Modul 9 SSF : Sensor Interfacing.” docs.google.com.  
<https://docs.google.com/document/d/14D8bETDw8x-BbeWWfg2QrjEE1WJA17kAZVDu79iCzCs/edit> (accessed May 21, 2024).
- [6] Anas Kuzechie. Assembly via Arduino (part 8) - ADC Value on Serial Monitor. (Oct. 21, 2021). Accessed: May 21, 2024. [Online Video]. Available: <https://www.youtube.com/watch?v=pmiTszZhPoI>
- [7] Anas Kuzechie. Assembly via Arduino (part 17) - Programming I2. (Nov. 23, 2021). Accessed: May 21, 2024. [Online Video]. Available: <https://www.youtube.com/watch?v=N0tmYhU9sN8>
- [8] Anas Kuzechie. Assembly via Arduino (part 19) - DHT11 Sensor. (Nov. 30, 2021). Accessed: May 21, 2024. [Online Video]. Available: <https://www.youtube.com/watch?v=vnLpzvkCUq8>

## APPENDICES

### Appendix A: Project Schematic



## Appendix B: Documentation



## Appendix C: Related Link

- Pengujian di proteus.mp4
- <https://github.com/monicavierin/Smart-Home>