EMBEDDED SYSTEM FINAL PROJECT REPORT
DEPARTMENT OF ELECTRICAL ENGINEERING
UNIVERSITAS INDONESIA

HOLO (HORIZON-ORIENTED LASER OUTPUT)

KELOMPOK PA23

BRYAN HERDIANTO	**2306210885**
FILAGA TIFIRA MUTHI	**2306208445**
JESIE TENARDI	**2306162002**
MUHAMMAD RIYAN SATRIO WIBOWO	**2306229323**

UNIVERSITAS INDONESIA
2025

PREFACE

Laporan ini berisi hasil dokumentasi pengerjaan dari proyek dengan judul HOLO (Horizon-Oriented Laser Output) yang dikembangkan sebagai proyek akhir dari Praktikum Embedded System. Tujuan proyek kami adalah merancang dan menerapkan sistem penunjukan arah sebagai bagian yang penting dalam berbagai bidang seperti astronomi, sistem kendali, hingga teknologi informasi. Sistem ini kita buat dengan mengimplementasikan berbagai komponen listrik yang telah kami pelajari selama semester ini dalam praktikum.

Pendekatan kami melibatkan riset yang mendalam mengenai sistem koordinat horizon dan azimuth serta cara mengontrol sudut dari sebuah servo agar bisa akurat. Dengan menggunakan Arduino sebagai basis, kami merancang proyek kami dengan mengintegrasikan keypad, servo, MAX7219, dan Laser Module. Melalui pengujian yang ketat, kami memastikan keberhasilan dan efektivitas dari solusi yang kami rancang, dan laporan proyek ini pun berfungsi sebagai dokumentasi singkat tentang perkembangan proyek kami, tantangan yang dihadapi, pembagian tugas dan tanggung jawab, serta solusi yang telah ditemukan.

Terakhir, kami ingin mengucapkan rasa terima kasih sebesar-besarnya terhadap seluruh tim asisten Digital Laboratory dan teman-teman kami atas bimbingan dan dukungan yang telah mereka berikan. Proyek ini merupakan hasil kerjasama dan komitmen kami dalam mengembangkan sistem penunjukkan arah secara tiga dimensi.

TABLE OF CONTENTS

BAB I.....4
PENDAHULUAN.....4
1.1 LATAR BELAKANG.....4
1.2 SOLUSI.....4
1.3 KRITERIA KEBERHASILAN.....5
1.4 PERAN DAN TANGGUNG JAWAB.....5
BAB 2.....6
IMPLEMENTASI.....6
2.1 DESAIN HARDWARE.....6
2.2 SOFTWARE.....7
2.3 INTEGRASI HARDWARE DAN SOFTWARE.....9
BAB 3.....24
EVALUASI DAN TESTING.....24
3.1 TESTING PROTEUS.....24
3.2 TESTING PHYSICAL CIRCUIT.....27
BAB 4.....29
KESIMPULAN.....29

BAB I

PENDAHULUAN

1.1 LATAR BELAKANG

Saat ini, sistem penunjukan arah menjadi bagian yang penting dalam berbagai bidang seperti astronomi, sistem kendali, hingga teknologi informasi. Sistem ini biasanya memanfaatkan sistem koordinat Azimuth (horizontal) dan Altitude (vertikal) sebagai acuan untuk menunjuk arah secara akurat. Akan tetapi, penunjukan arah secara manual dengan tenaga manusia tidak efisien dan rawan terhadap *human error*, dimana penunjukan arah seringkali tidak akurat.

Selain itu, seringkali sistem penunjukan arah dibutuhkan di tempat yang berbahaya atau sulit dijangkau manusia. Dalam kasus tersebut, tentunya sulit jika sistem penunjukan arah dilakukan secara manual. Dibutuhkan suatu sistem untuk mengotomasi proses penunjukan arah sehingga proses tersebut dapat berjalan secara efisien, aman, dan akurat.

Melihat hal tersebut, kami memutuskan untuk menyelesaikan masalah tersebut menggunakan sistem otomatis menggunakan dua Arduino, keypad, serial display, serta servo dan laser. Dengan sistem ini, penunjukan arah dapat dilakukan oleh sistem secara otomatis sesuai dengan keinginan pengguna, meningkatkan efisiensi dan akurasi penunjukan arah.

1.2 SOLUSI

Sebagai solusi dari masalah yang telah disebutkan, kami mengimplementasikan Arduino untuk melakukan otomasi proses penunjukan arah. Terdapat dua Arduino yang kami gunakan yaitu Arduino master yang berfungsi untuk menerima input pengguna, dan Arduino slave yang berfungsi untuk menggerakkan servo. Kami menggunakan *keypad* untuk menerima input derajat azimuth dan altitude dari pengguna. Nilai input dari pengguna ditampilkan dengan *serial display*.

Data input dari pengguna dikirimkan dari master ke slave untuk mengendalikan servo yang terhubung dengan PWM. Kalibrasi dilakukan untuk memastikan sudut perputaran servo sesuai dengan input dari pengguna. Kami juga mengintegrasikan *reset button* untuk mengembalikan servo ke posisi awal. Implementasi seluruh fungsionalitas ini diharapkan dapat menghasilkan sistem penunjukan arah yang akurat dan efisien, serta minim intervensi.

1.3 KRITERIA KEBERHASILAN

Kriteria keberhasilan proyek ini ditentukan oleh beberapa faktor, antara lain:

1. Arduino master dapat menerima input *keypad* dengan benar.
2. Mode input (azimuth/altitude) dapat diatur dengan baik menggunakan *keypad*.
3. Mode dan input *keypad* ditampilkan dengan benar pada *serial display*.
4. Arduino master dapat mengirimkan data kepada Arduino slave.
5. Arduino slave dapat menggerakkan servo dengan sudut yang sesuai berdasarkan input user.

1.4 PERAN DAN TANGGUNG JAWAB

Peran dan tanggung jawab masing-masing anggota kelompok terbagi menjadi berikut:

Roles	Responsibilities	Person
Brainstorming and Design	Memikirkan ide desain, membuat prototype, dan meng-spesifikasi komponen yang dibutuhkan pada proyek	Bryan, Riyan
Kode and Software	Menulis kode	Bryan
Proteus	Mengimplementasikan desain rangkaian pada software Proteus	Bryan
Laporan dan Dokumentasi	Membuat laporan yang lengkap dan terstruktur serta dokumentasi pengerjaan rangkaian	Bryan, Riyan, Jesie, Filaga
Final Assembly	Membuat rangkaian fisik dan melakukan testing	Bryan, Riyan

BAB 2

IMPLEMENTASI

2.1 DESAIN HARDWARE

Dalam membuat sistem penunjukan arah "HOLO," dibutuhkan sejumlah komponen yang perlu diselaraskan. Terdapat 2 buah Arduino yang digunakan dimana satu Arduino berperan sebagai master dan Arduino lainnya berperan sebagai slave. Pada Arduino master, terdapat sebuah *keypad* yang berfungsi untuk menerima input derajat dari pengguna. *Keypad* tersebut sekaligus berfungsi untuk mengganti mode (azimuth/altitude) dan men-*trigger* pergerakan servo. Input tersebut ditampilkan dengan *serial display* MAX7219 yang dihubungkan dengan protokol *Serial Peripheral Interface* (SPI) dengan format "Azm/Alt (spasi) Derajat."

Gambar 1. MAX7219 berupa 8-digit 7-segment

Pada Arduino slave, terdapat 2 servo 180° untuk azimuth dan altitude, serta *button* untuk menyalakan/mematikan modul laser KY-008. Untuk menghubungkan kedua Arduino, digunakan protokol *Inter-Integrated Circuit* (I2C). Arduino master akan mengirimkan input pengguna kepada Arduino slave setiap pengguna menekan tombol ❇ atau #. Arduino slave kemudian mengolah input pengguna untuk menggerakkan servo sesuai derajat yang diinput oleh pengguna.

Gambar 2. Servo SG90 yang digunakan pada proyek ini

2.2 SOFTWARE

Dalam membuat proyek ini, kode yang kami buat menggunakan bahasa Assembly AVR yang terbagi menjadi beberapa bagian utama, antara lain:

a. Arduino Master

Pada program Arduino master, terdapat label ***main*** yang berisi inisialisasi bagian penting yang akan digunakan oleh program, seperti setup protokol SPI yang digunakan untuk komunikasi dengan *serial display* MAX7219, mengatur I2C untuk komunikasi dengan slave, dan mengatur pembacaan *keypad*. Selain itu, dilakukan setup input/output port dengan mengubah nilai register DDRx.

Program kemudian berlanjut ke konfigurasi *serial display* untuk menampilkan mode. Mode azimuth ditampilkan sebagai "AZI" dengan subroutine ***MAX7219_disp_azi***, sedangkan mode altitude ditampilkan sebagai "ALt" dengan subroutine ***MAX7219_disp_alt***. Selanjutnya, program masuk ke subroutine utama, yaitu ***wait_keypress*** yang akan dijalankan hingga *keypad* ditekan oleh pengguna.

b. Keypad Input

Keypad diinisialisasi dengan mengatur register DDRx, dimana kolom pada *keypad* diatur sebagai input dan baris pada *keypad* diatur sebagai output. Program menerima input *keypad* dengan dengan melakukan *grounding* (meng-*clear* bit pada PORTx) pada setiap baris *keypad* secara sekuensial. Ketika *keypad* ditekan, dilakukan *debouncing* terlebih dahulu sebelum memproses input. Input *keypad* dikombinasikan untuk menghasilkan bilangan multidigit, dimana bilangan input tersebut disimpan di register. Di samping itu, *key* 11 digunakan untuk men-*trigger* komunikasi I2C dengan slave, serta untuk mengubah mode input (azimuth/altitude).

c. Display MAX7219

Serial display diinisialisasi dengan protokol SPI, dimana kecerahan, mode decoding, dan scan limit diatur pada *main*. *Display* tersebut menampilkan value azimuth/altitude tergantung pada mode yang dipilih pengguna. Setelah program menerima input, subroutine ***send_bytes*** akan dipanggil untuk mengirimkan data kepada MAX7219 sebagai slave, dimana byte *command* dikirimkan pertama, diikuti oleh byte data.

d. Arduino Slave

Pada program Arduino slave, terdapat label ***main*** yang berisi inisialisasi input/output untuk servo serta button sebagai interrupt eksternal. Selanjutnya, terdapat inisialisasi I2C pada subroutine ***I2C_init***, dimana address dan *handling* TWI interrupt diatur. Program kemudian masuk ke main loop ***agn***, dimana Arduino slave menunggu command dari Arduino master dengan memanggil subroutine ***I2C_listen***. Jika command dari Arduino master diterima, program akan memanggil subroutine ***I2C_read*** dan memproses hasil pembacaan untuk menggerakkan servo.

e. Kontrol Servo

Setelah menerima data dari Arduino master melalui I2C, Arduino slave akan memproses nilai input user dengan subroutine ***angle_to_timer***. Subroutine ini mengubah range nilai 0-180° dari input user ke nilai konstanta PWM (40-180) menggunakan *piecewise linear interpolation* untuk menghasilkan sudut yang tepat pada servo. Hasil perhitungan kemudian digunakan oleh subroutine ***rotate_servo1*** dan ***rotate_servo2*** untuk menggerakkan servo menggunakan sinyal PWM dari Timer0. Sinyal PWM tersebut mempunyai panjang gelombang 1-2ms (menggunakan ***delay_timer0***) dengan periode 20 ms (menggunakan ***delay_20ms***) yang merupakan standar untuk kontrol servo.

f. Interrupt Handling

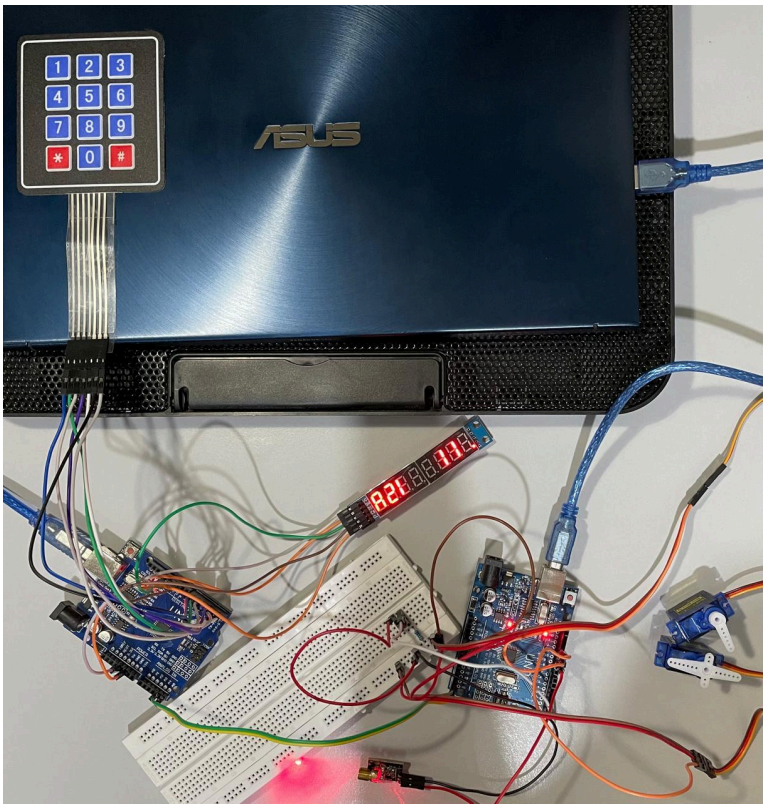
Pada rangkaian, terdapat *button* yang terhubung dengan INT0. *Button* ini akan menjadi *trigger* untuk interrupt eksternal yang diatur pada program, dimana interrupt ini diatur untuk mendeteksi *falling edge*. Ketika *button* ditekan, maka interrupt handler akan berjalan dan akan dilakukan *toggling* pada port PB2 untuk menyalakan/mematikan Laser Module KY-008..

g. Komunikasi Master-Slave

Arduino master berkomunikasi dengan Arduino slave menggunakan protokol I2C. Komunikasi dimulai dengan inisialisasi interface TWI dengan frekuensi 400kHz. Dalam pengiriman data ke Arduino slave, Arduino master mengikuti standar protokol I2C, yaitu dimulai dengan mengirimkan kondisi START, dan dilanjut dengan address slave dengan *write bit*, byte data, dan kondisi STOP. Program kami juga mempunyai *error handling* dengan mengecek flag setiap komunikasi dilakukan.

2.3 INTEGRASI HARDWARE DAN SOFTWARE

Spesifikasi *hardware* dan *software* yang telah dibuat kemudian diintegrasikan dalam satu rangkaian asli. Kami menggunakan Arduino IDE untuk mengupload program ke kedua Arduino yang digunakan. Berikut adalah rangkaian final dari proyek ini:

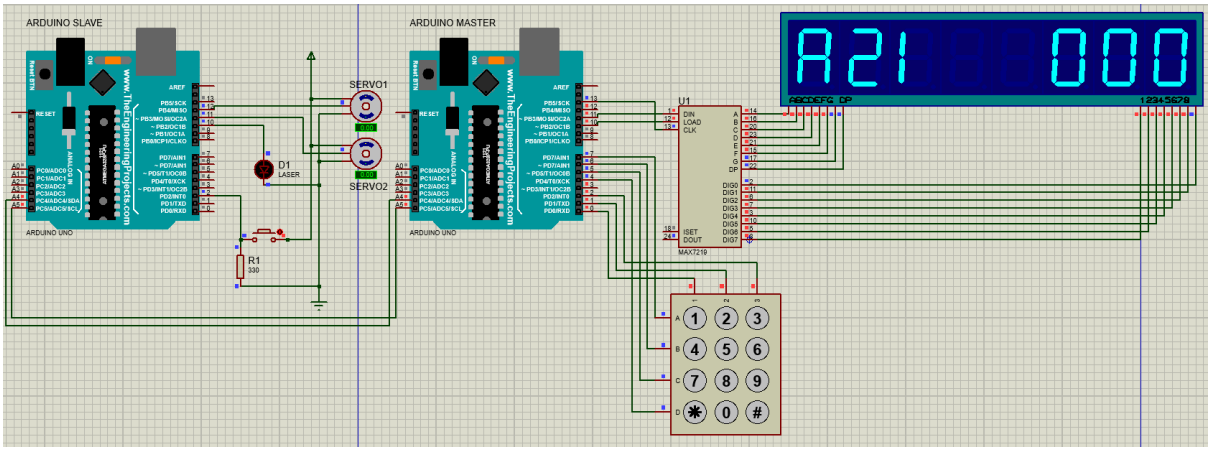


Gambar 3. Rangkaian Fisik Final

Dalam rangkaian tersebut, dapat dilihat bahwa Arduino master terhubung langsung dengan *keypad* untuk menerima input user, dan *serial display* MAX7219 untuk menampilkan mode dan sudut yang diinputkan user. Arduino master dihubungkan dengan Arduino slave, dimana komunikasi dengan protokol I2C berjalan. Pada breadboard, terdapat *button* yang

terhubung dengan Arduino slave. *Button* ini digunakan untuk menyalakan/mematikan modul laser. Selain itu, Arduino slave juga terhubung dengan servo, dimana pergerakan servo akan disesuaikan dengan input dari user.

Untuk mengurangi kemungkinan terjadinya error akibat kecacatan *hardware* yang digunakan, kami juga mensimulasikan rangkaian dengan aplikasi Proteus:



Gambar 4. Rangkaian pada Proteus, dengan Arduino master di kanan dan Arduino slave di kiri.

Berikut adalah kode assembly yang diupload pada Arduino master:

```
;-----------------------
; HOLO (Horizon-Oriented Laser Output)
;-----------------------
#define __SFR_OFFSET 0x00
#include "avr/io.h"
;-----------------------
.global main
;=============================================================
===
main:
;------------
.equ  SCK, 5
.equ  MOSI, 3
.equ  SS, 2
;-------------------------------------------------------------
--
    LDI   R17, (1<<MOSI)|(1<<SCK)|(1<<SS)
    OUT   DDRB, R17       ;set MOSI, SCK, SS as o/p
    ;-------------------------------------------------------
    LDI   R17, (1<<SPE)|(1<<MSTR)|(1<<SPR0)
```

```
    OUT   SPCR, R17       ;enable SPI as master, fsck=fosc/16
    ;-----------------------------------------------------
    LDI   R17, 0x0A       ;set segment intensity (0 to 15)
    LDI   R18, 8          ;intensity level = 8
    RCALL send_bytes      ;send command & data to MAX7219
    ;-----------------------------------------------------
    LDI   R17, 0x09       ;set decoding mode command
    LDI   R18, 0b00000111 ;decoding byte
    RCALL send_bytes      ;send command & data to MAX7219
    ;-----------------------------------------------------
    LDI   R17, 0x0B       ;set scan limit command
    LDI   R18, 0x07       ;8 digits connected to MAX7219
    RCALL send_bytes      ;send command & data to MAX7219
    ;-----------------------------------------------------
    LDI   R17, 0x0C       ;set turn ON/OFF command
    LDI   R18, 0x01       ;turn ON MAX7219
    RCALL send_bytes      ;send command & data to MAX7219
    ;-----------------------------------------------------
    LDI R16, 3
    CLR R25
    RCALL I2C_init
    RJMP  MAX7219_disp_azi
;-------------------------------------------------------------
---
send_bytes:
    CBI   PORTB, SS       ;enable slave device MAX7219
    OUT   SPDR, R17       ;transmit command
    ;-----------------------------------------------------
l2: IN    R19, SPSR
    SBRS  R19, SPIF       ;wait for byte transmission
    RJMP  l2              ;to complete
    ;-----------------------------------------------------
    OUT   SPDR, R18       ;transmit data
    ;-----------------------------------------------------
l3: IN    R19, SPSR
    SBRS  R19, SPIF       ;wait for byte transmission
    RJMP  l3              ;to complete
    ;-----------------------------------------------------
    SBI   PORTB, SS       ;disable slave device MAX7219
    RET
;-------------------------------------------------------------
---
I2C_init:
    LDI   R21, 0
    STS   TWSR, R21       ;prescaler = 0
```

```
    LDI   R21, 12          ;division factor = 12
    STS   TWBR, R21        ;SCK freq = 400kHz
    LDI   R21, (1<<TWEN)
    STS   TWCR, R21        ;enable TWI
    RET
;============================================================
==
MAX7219_disp_azi:
;-----------------
    LDI   R17, 0x08        ;select digit 7
    LDI   R18, 0x77        ;data = A
    RCALL send_bytes       ;send command & data to MAX7219
    ;-------------------------------------------------------
    LDI   R17, 0x07        ;select digit 6
    LDI   R18, 0x6D        ;data = Z
    RCALL send_bytes       ;send command & data to MAX7219
    ;-------------------------------------------------------
    LDI   R17, 0x06        ;select digit 5
    LDI   R18, 0x06        ;data = I
    RCALL send_bytes       ;send command & data to MAX7219
    ;-------------------------------------------------------
    LDI   R17, 0x05        ;select digit 4
    LDI   R18, 0x00        ;data = space
    RCALL send_bytes       ;send command & data to MAX7219
    ;-------------------------------------------------------
    LDI   R17, 0x04        ;select digit 3
    LDI   R18, 0x00        ;data = space
    RCALL send_bytes       ;send command & data to MAX7219
    ;-------------------------------------------------------
    RJMP  keypad
MAX7219_disp_alt:
;-----------------
    LDI   R17, 0x08        ;select digit 7
    LDI   R18, 0x77        ;data = A
    RCALL send_bytes       ;send command & data to MAX7219
    ;-------------------------------------------------------
    LDI   R17, 0x07        ;select digit 6
    LDI   R18, 0x0E        ;data = L
    RCALL send_bytes       ;send command & data to MAX7219
    ;-------------------------------------------------------
    LDI   R17, 0x06        ;select digit 5
    LDI   R18, 0x0F        ;data = t
    RCALL send_bytes       ;send command & data to MAX7219
    ;-------------------------------------------------------
    LDI   R17, 0x05        ;select digit 4
```

```
    LDI   R18, 0x00       ;data = space
    RCALL send_bytes      ;send command & data to MAX7219
    ;-------------------------------------------------------
    LDI   R17, 0x04       ;select digit 3
    LDI   R18, 0x00       ;data = space
    RCALL send_bytes      ;send command & data to MAX7219
;=============================================================
===
keypad:
    LDI   R20, 0xF0       ;low nibble port D i/p (column
lines)
    OUT   DDRD, R20       ;high nibble port D o/p (row lines)

;-------------------------------------------------------------
gnd_rows:
    LDI   R20, 0x0F       ;send 0 to high nibble port D
    OUT   PORTD, R20      ;to ground all rows

;-------------------------------------------------------------
wait_release:
    NOP
    IN    R21, PIND       ;read key pins
    ANDI  R21, 0x07       ;mask unsed bits
    CPI   R21, 0x07       ;equal if no keypress
    BRNE  wait_release    ;do again until keys released

;-------------------------------------------------------------
wait_keypress:
    NOP
    IN    R21, PIND       ;read key pins
    ANDI  R21, 0x07       ;mask unsed bits
    CPI   R21, 0x07       ;equal if no keypress
    BREQ  wait_keypress   ;keypress? no, go back & check

;-------------------------------------------------------------
    RCALL my_delay        ;delay to cancel switch bounce

;-------------------------------------------------------------
    IN    R21, PIND       ;2nd check for keypress
    ANDI  R21, 0x07       ;which ensures that 1st keypress
    CPI   R21, 0x07       ;was not erroneous due to spike
noise
    BREQ  wait_keypress

;-------------------------------------------------------------
```

```
    LDI   R21, 0b01111111 ;ground row 1
    OUT   PORTD, R21
    NOP
    IN    R21, PIND       ;read all columns
    ANDI  R21, 0x07       ;mask unsed bits
    CPI   R21, 0x07       ;equal if no key
    BRNE  row1_col        ;row 1, find column

;-----------------------------------------------------
    LDI   R21, 0b10111111 ;ground row 2
    OUT   PORTD, R21
    NOP
    IN    R21, PIND       ;read all columns
    ANDI  R21, 0x07       ;mask unsed bits
    CPI   R21, 0x07       ;equal if no key
    BRNE  row2_col        ;row 2, find column

;-----------------------------------------------------
    LDI   R21, 0b11011111 ;ground row 3
    OUT   PORTD, R21
    NOP
    IN    R21, PIND       ;read all columns
    ANDI  R21, 0x07       ;mask unsed bits
    CPI   R21, 0x07       ;equal if no key
    BRNE  row3_col        ;row 3, find column

;-----------------------------------------------------
    LDI   R21, 0b11101111 ;ground row 4
    OUT   PORTD, R21
    NOP
    IN    R21, PIND       ;read all columns
    ANDI  R21, 0x07       ;mask unsed bits
    CPI   R21, 0x07       ;equal if no key
    BRNE  row4_col        ;row 4, find column
;-----------------------------------------------------
---
row1_col:
    LDI   R30, lo8(row1_digits)
    LDI   R31, hi8(row1_digits)
    RJMP  find
row2_col:
    LDI   R30, lo8(row2_digits)
    LDI   R31, hi8(row2_digits)
    RJMP  find
row3_col:
```

```
    LDI   R30, lo8(row3_digits)
    LDI   R31, hi8(row3_digits)
    RJMP  find
row4_col:
    LDI   R30, lo8(row4_digits)
    LDI   R31, hi8(row4_digits)
    RJMP  find
;-------------------------------------------------------------
---
find:
    LSR   R21           ;logical shift right
    BRCC  match
    LPM   R20, Z+
    RJMP  find
match:
    LPM   R20, Z

    ; Check if the key value is 11 (decimal)
    CPI   R20, 11
    BRNE  store         ; If not 11, store

    RCALL I2C_start       ;transmit START condition
    LDI   R27, 0b10010000 ;SLA(1001000) + W(0)
    RCALL I2C_write       ;write slave address SLA+W
    MOV   R27, R24        ;data byte to be transmitted
    RCALL I2C_write       ;write data byte
    MOV   R27, R25        ;servo selection byte (0 or 1)
    RCALL I2C_write       ;write data byte
    RCALL I2C_stop        ;transmit STOP condition

    CLR   R24           ; Clear low byte
    LDI   R16, 3

    LDI   R18, 0        ; Value to display (0)
    LDI   R17, 1        ; Digit 1
    RCALL send_bytes
    LDI   R17, 2        ; Digit 2
    RCALL send_bytes
    LDI   R17, 3        ; Digit 3
    RCALL send_bytes

    TST R25
    BRNE jump_to_azi
    LDI R25, 1
    RJMP MAX7219_disp_alt
```

```
jump_to_azi:
    CLR R25
    RJMP MAX7219_disp_azi
store:
    LSL  R24            ; Multiply by 2
    MOV  R30, R24       ; Save intermediate result
    LSL  R24            ; Multiply by 4 (2 * 2 = 4)
    LSL  R24            ; Multiply by 8 (2 * 2 * 2 = 8)
    ADD  R24, R30       ; Add original x2 and x8 to x10
    ADD  R24, R20       ; Add previous number

    MOV  R17, R16
    DEC  R16
    MOV   R18, R20
    RCALL send_bytes      ;send command & data to MAX7219
    RJMP  gnd_rows
;=============================================================
==
I2C_start:
    LDI   R21, (1<<TWINT)|(1<<TWSTA)|(1<<TWEN)
    STS   TWCR, R21       ;transmit START condition

;-------------------------------------------------------------
wt1:LDS   R21, TWCR
    SBRS  R21, TWINT      ;TWI interrupt = 1?
    RJMP  wt1             ;no, wait for end of transmission

;-------------------------------------------------------------
    RET
;=============================================================
==
I2C_write:
    STS   TWDR, R27       ;copy SLA+W into data register
    LDI   R21, (1<<TWINT)|(1<<TWEN)
    STS   TWCR, R21       ;transmit SLA+W

;-------------------------------------------------------------
wt2:LDS   R21, TWCR
    SBRS  R21, TWINT
    RJMP  wt2             ;wait for end of transmission

;-------------------------------------------------------------
    RET
;=============================================================
==
```

```
I2C_stop:
    LDI   R21, (1<<TWINT)|(1<<TWSTO)|(1<<TWEN)
    STS   TWCR, R21       ;transmit STOP condition
    RET
;-------------------------------------------------------------
---
row1_digits: .byte  1,2,3
row2_digits: .byte  4,5,6
row3_digits: .byte  7,8,9
row4_digits: .byte  11,0,11
;=============================================================
===
my_delay:
    LDI   R21, 255
l6: LDI   R22, 255
l7: LDI   R23, 10
l8: DEC   R23
    BRNE  l8
    DEC   R22
    BRNE  l7
    DEC   R21
    BRNE  l6
    RET
```

Berikut adalah kode assembly yang diupload pada Arduino slave:

```
;------------------------
; HOLO (Horizon-Oriented Laser Output)
;------------------------
#define __SFR_OFFSET 0x00
#include "avr/io.h"
#include "avr/interrupt.h"
;------------------------
.global main
.global __vector_1
;=============================================================
===
main:
;-----------
    SBI   DDRB, 4         ;pin PB4 o/p for servo control
    SBI   DDRB, 0         ;pin PB0 o/p for servo control

    SBI   DDRB, 2           ; PB2 = output
    CBI   DDRD, 2           ; PD2 = input
```

```
    LDI   R16, (1 << ISC01) ; Falling edge
    STS   EICRA, r16

    LDI   R16, (1 << INT0)  ; Enable INT0
    OUT   EIMSK, r16

    LDI   R31, 0

    SEI                     ; Enable global interrupts
;-------------------------------------------------------------
---
agn:CLR   R27
    RCALL I2C_init          ;initialize TWI module
    RCALL I2C_listen        ;listen to bus to be addressed

    ; First, check if this is a valid message for us
    LDS   R21, TWSR         ;get TWI status
    ANDI  R21, 0xF8         ;mask prescaler bits
    CPI   R21, 0x60         ;SLA+W received, ACK sent?
    BRNE  agn               ;if not, restart

    ; Now read the angle byte
    RCALL I2C_read          ;read angle byte
    MOV   R26, R27

    ; Read the second byte (servo selection)
    RCALL I2C_read          ;read servo selection byte
    MOV   R31, R27          ;store in R31 (0=servo1,
1=servo2)

    ; Wait for STOP condition
    LDI   R21, (1<<TWINT)|(1<<TWEN)|(1<<TWEA)
    STS   TWCR, R21

    RCALL angle_to_timer

    CPI   R31, 0
    BRNE  rotate_servo2
    RJMP  rotate_servo1
;=============================================================
==
rotate_servo2:
;------------
    LDI   R20, 10         ;count to give enough cycles of PWM
```

```
l2: SBI   PORTB, 0
    RCALL delay_timer0
    CBI   PORTB, 0        ;send msec pulse to rotate servo
    RCALL delay_20ms      ;wait 20ms before re-sending pulse
    DEC   R20
    BRNE  l2              ;go back & repeat PWM signal
    CLR   R26
    RJMP  agn
rotate_servo1:
;------------
    LDI   R20, 10         ;count to give enough cycles of PWM
l1: SBI   PORTB, 4
    RCALL delay_timer0
    CBI   PORTB, 4        ;send msec pulse to rotate servo
    RCALL delay_20ms      ;wait 20ms before re-sending pulse
    DEC   R20
    BRNE  l1              ;go back & repeat PWM signal
    CLR   R26
    RJMP  agn
/*
; FOR PROTEUS CIRCUIT
angle_to_timer:
    ; Check specific angle cases first
    CPI   R26, 0
    BRNE  not_0
    LDI   R26, 61
    RET
not_0:
    CPI   R26, 45
    BRNE  not_45
    LDI   R26, 77
    RET
not_45:
    CPI   R26, 90
    BRNE  not_90
    LDI   R26, 93
    RET
not_90:
    CPI   R26, 135
    BRNE  not_135
    LDI   R26, 108
    RET
not_135:
    CPI   R26, 180
    BRNE  interpolate
```

```
    LDI   R26, 127
    RET
;-------------------------------------------------------
interpolate:
    ; Determine the angle range
    CPI   R26, 45
    BRLT  range_0_45

    CPI   R26, 90
    BRLT  range_45_90

    CPI   R26, 135
    BRLO  range_90_135

    CPI   R26, 180
    BRLO  range_135_180

    LDI   R26, 127
    RET
;------------------- 0° to 45° : 61 to 77
range_0_45:
    ; PWM = 61 + angle * 16 / 45
    MOV   R20, R26        ; angle
    LDI   R21, 16
    MUL   R20, R21
    LDI   R22, 45
    MOV   R18, R0
    MOV   R19, R1
    RCALL simple_div
    LDI   R26, 61
    ADD   R26, R18
    RET

;------------------- 45° to 90° : 77 to 93
range_45_90:
    SUBI  R26, 45         ; angle - 45
    ; PWM = 77 + (angle * 16 / 45)
    MOV   R20, R26
    LDI   R21, 16
    MUL   R20, R21
    LDI   R22, 45
    MOV   R18, R0
    MOV   R19, R1
    RCALL simple_div
    LDI   R26, 77
```

```
    ADD   R26, R18
    RET

;------------------- 90° to 135° : 93 to 108
range_90_135:
    SUBI  R26, 90
    ; PWM = 93 + (angle * 15 / 45)
    MOV   R20, R26
    LDI   R21, 15
    MUL   R20, R21
    LDI   R22, 45
    MOV   R18, R0
    MOV   R19, R1
    RCALL simple_div
    LDI   R26, 93
    ADD   R26, R18
    RET
;------------------- 135° to 180° : 108 to 127
range_135_180:
    SUBI  R26, 135
    ; PWM = 108 + (angle * 19 / 45)
    MOV   R20, R26
    LDI   R21, 19
    MUL   R20, R21
    LDI   R22, 45
    MOV   R18, R0
    MOV   R19, R1
    RCALL simple_div
    LDI   R26, 108
    ADD   R26, R18
    RET
*/
; FOR PHYSICAL CIRCUIT
angle_to_timer:
    ; Direct matches first
    CPI   R26, 0
    BRNE  not_0
    LDI   R26, 35
    RET
not_0:
    CPI   R26, 45
    BRNE  not_45
    LDI   R26, 65
    RET
not_45:
```

```
    CPI   R26, 90
    BRNE  not_90
    LDI   R26, 95
    RET
not_90:
    CPI   R26, 135
    BRNE  not_135
    LDI   R26, 130
    RET
not_135:
    CPI   R26, 180
    BRNE  interpolate
    LDI   R26, 158
    RET
; ------------------------------------------------------------
interpolate:
    CPI   R26, 45
    BRLT  range_0_45
    CPI   R26, 90
    BRLT  range_45_90
    CPI   R26, 135
    BRLO  range_90_135
    CPI   R26, 180
    BRLO  range_135_180
    LDI   R26, 158
    RET
; 0°-45°: 35-65
range_0_45:
    MOV   R20, R26        ; R20 = angle
    LDI   R21, 30         ; slope numerator
    MUL   R20, R21
    LDI   R22, 45         ; slope denominator
    MOV   R18, R0
    MOV   R19, R1
    RCALL simple_div
    LDI   R26, 35
    ADD   R26, R18
    RET
; 45°-90°: 65-95
range_45_90:
    SUBI  R26, 45
    MOV   R20, R26
    LDI   R21, 30
    MUL   R20, R21
    LDI   R22, 45
```

```
    MOV   R18, R0
    MOV   R19, R1
    RCALL simple_div
    LDI   R26, 65
    ADD   R26, R18
    RET
; 90°-135°: 95-130
range_90_135:
    SUBI  R26, 90
    MOV   R20, R26
    LDI   R21, 35
    MUL   R20, R21
    LDI   R22, 45
    MOV   R18, R0
    MOV   R19, R1
    RCALL simple_div
    LDI   R26, 95
    ADD   R26, R18
    RET
; 135°-180°: 130-158
range_135_180:
    SUBI  R26, 135
    MOV   R20, R26
    LDI   R21, 28
    MUL   R20, R21
    LDI   R22, 45
    MOV   R18, R0
    MOV   R19, R1
    RCALL simple_div
    LDI   R26, 130
    ADD   R26, R18
    RET
; Input: R19:R18 = 16-bit dividend, R22 = 8-bit divisor
; Output: R18 = quotient
simple_div:
    CLR   R17           ; Clear remainder
    LDI   R16, 16       ; 16 bits to process
div_loop:
    LSL   R18           ; Shift low byte left, bit 7 to carry
    ROL   R19           ; Shift high byte left with carry
    ROL   R17           ; Shift remainder left with carry

    CP    R17, R22      ; Compare remainder with divisor
    BRLO  skip_sub      ; If remainder < divisor, skip
subtraction
```

```
    SUB   R17, R22     ; Subtract divisor from remainder
    INC   R18          ; Set result bit to 1
skip_sub:
    DEC   R16          ; Decrement bit counter
    BRNE  div_loop     ; If not done, continue loop
    RET
I2C_init:
    LDI   R21, 0b10010000
    STS   TWAR, R21         ;store slave address 0b10010000
    LDI   R21, (1<<TWEN)
    STS   TWCR, R21         ;enable TWI
    LDI   R21, (1<<TWINT)|(1<<TWEN)|(1<<TWEA)
    STS   TWCR, R21         ;enable TWI & ACK
    RET
;=============================================================
==
I2C_listen:
    LDS   R21, TWCR
    SBRS  R21, TWINT
    RJMP  I2C_listen        ;wait for slave to be addressed
    RET
;=============================================================
==
I2C_read:
    LDI   R21, (1<<TWINT)|(1<<TWEA)|(1<<TWEN)
    STS   TWCR, R21         ;enable TWI & ACK

;-------------------------------------------------------
wt: LDS   R21, TWCR
    SBRS  R21, TWINT
    RJMP  wt                ;wait for data byte to be read

;-------------------------------------------------------
    LDS   R27, TWDR         ;store received byte
    RET
;=============================================================
===
__vector_1:
    sbi PINB, 2
    reti
;=============================================================
===
;delay subroutines
;=============================================================
```

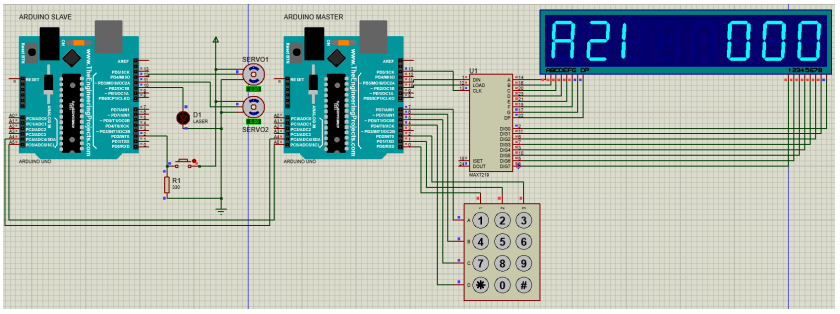
```
===
delay_timer0:              ;delay via Timer0

;-----------------------------------------------------------
    CLR   R21
    OUT   TCNT0, R21       ;initialize timer0 with count=0
    MOV   R21, R26
    OUT   OCR0A, R21
    LDI   R21, 0b00001100
    OUT   TCCR0B, R21      ;timer0: CTC mode, prescaler 256

;-----------------------------------------------------------
l3: IN    R21, TIFR0       ;get TIFR0 byte & check
    SBRS  R21, OCF0A       ;if OCF0=1, skip next instruction
    RJMP  l3               ;else, loop back & check OCF0 flag

;-----------------------------------------------------------
    CLR   R21
    OUT   TCCR0B, R21      ;stop timer0

;-----------------------------------------------------------
    LDI   R21, (1<<OCF0A)
    OUT   TIFR0, R21       ;clear OCF0 flag
    RET
;=============================================================
===
delay_20ms:                ;delay 20ms
    LDI   R21, 255
l4: LDI   R22, 210
l5: LDI   R23, 2
l6: DEC   R23
    BRNE  l6
    DEC   R22
    BRNE  l5
    DEC   R21
    BRNE  l4
    RET
```

BAB 3

EVALUASI DAN TESTING

3.1 TESTING PROTEUS

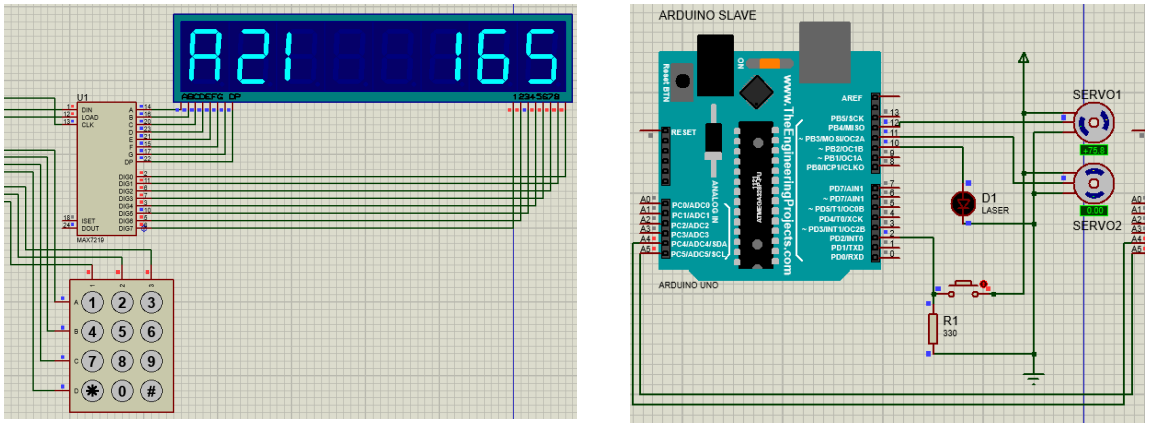
Sebelumnya, perlu dipahami bahwa dalam simulasi ini, servo motor yang digunakan adalah positional servo motor 180 derajat. Komponen servo pada Proteus menampilkan -90 derajat sebagai posisi 0 derajat. Lalu, 90 derajat menunjukkan posisi 180 derajat. Berikut adalah hasil run awal-awal dari sistem secara simulasi di Proteus:



Gambar 5. Rangkaian Proteus Keseluruhan

Dari gambar di atas, dapat dilihat bahwa sistem menunjukkan mode azimuth terlebih dulu dengan tampilan 0 derajat. Untuk memasukkan derajat yang diinginkan, user dapat menekan tombol 1-9 pada keypad. Setelah itu, user dapat menekan tombol ❇ untuk mengirimkan data ke Arduino slave.

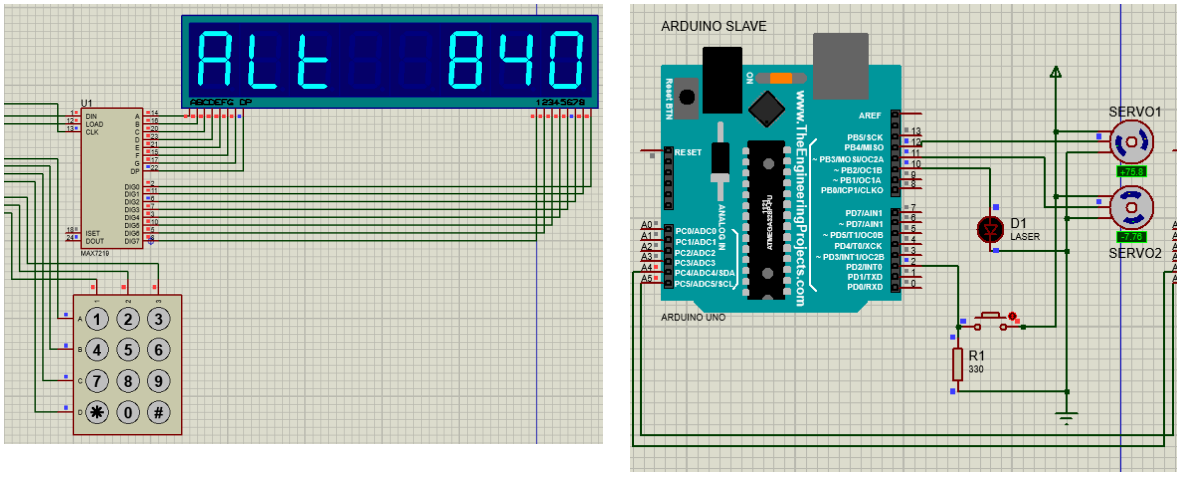
Berikut adalah hasil memasukkan derajat azimuth untuk menggerakkan servo:



Gambar 6. Testing Memasukkan Derajat Azimuth #1

Dari gambar di atas, kita bisa melihat bahwa user telah memasukkan derajat azimuth sebesar 165 derajat. Display MAX7219 menampilkan mode azimuth dengan sudut 165 derajat. Setelah itu, user menekan tombol ❇ untuk mengirimkan data ke Arduino slave. Setelah itu, SERVO1 bergerak ke sudut 165 derajat (-90 + 165 = 75). SERVO1 menunjukkan sudut +75.8 derajat yang tidak benar-benar pas dengan sudut 165 derajat. Hal ini karena ketidakakuratan dari kalibrasi PWM yang dibuat. Meskipun gitu, hal ini masih dalam batas toleransi yang dapat diterima.

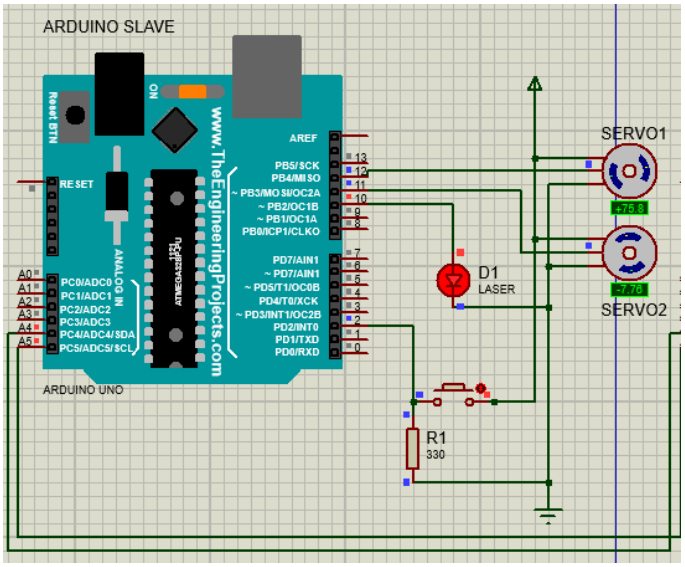
Berikut adalah hasil memasukkan derajat altitude untuk menggerakkan servo:



Gambar 7. Testing Memasukkan Derajat Altitude #1

Dari gambar di atas, kita bisa melihat bahwa user telah memasukkan derajat altitude sebesar 84 derajat. Display MAX7219 menampilkan mode altitude dengan sudut 84 derajat. Setelah itu, user menekan tombol ❇ untuk mengirimkan data ke Arduino slave. Setelah itu, SERVO2 bergerak ke sudut 84 derajat (-90 + 84 = -6). Sama seperti sebelumnya, kalibrasi belum sepenuhnya akurat, tetapi masih dalam batas toleransi yang dapat diterima.

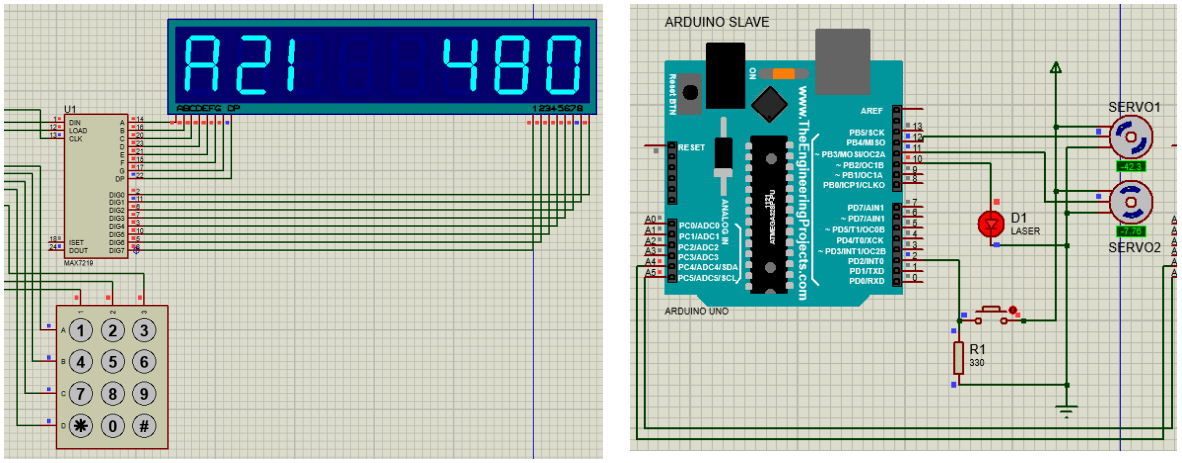
Berikut adalah hasil menekan push button menyalakan Laser Module KY-008:



Gambar 8. Testing Menyalakan Laser Module

Karena menggunakan interrupt, maka ketika tombol ditekan, Laser Module KY-008 akan menyala secara real-time. Ketika tombol dilepas, Laser Module KY-008 akan mati secara real-time. Interrupt ini memungkinkan sistem untuk mengontrol Laser Module KY-008 tanpa ada delay.

Mode akan berubah dari azimuth ke altitude dan balik ke azimuth lagi. Maka, setelah melakukan input pada mode altitude, user dapat kembali ke mode azimuth. Berikut adalah hasilnya:

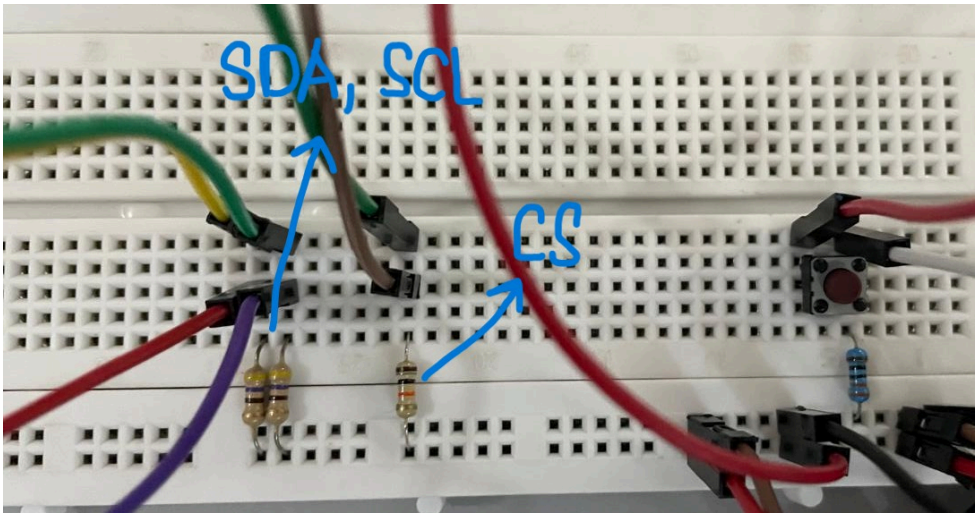


Gambar 9. Testing Memasukkan Derajat Azimuth #2

Dari gambar di atas, kita bisa melihat bahwa user telah memasukkan derajat azimuth sebesar 48 derajat. Display MAX7219 menampilkan mode azimuth dengan sudut 48 derajat. Setelah itu, user menekan tombol ❇ untuk mengirimkan data ke Arduino slave. Setelah itu, SERVO1 bergerak ke sudut 48 derajat (-90 + 48 = -42). Sama seperti sebelumnya, kalibrasi belum sepenuhnya akurat, tetapi masih dalam batas toleransi yang dapat diterima.

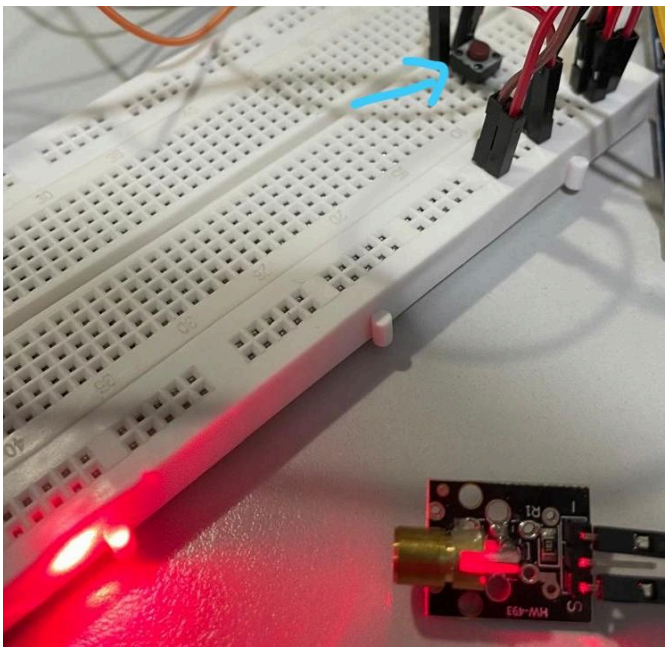
3.2 TESTING PHYSICAL CIRCUIT

Kalibrasi yang dilakukan pada sistem ini adalah kalibrasi manual. Servo motor yang digunakan pada proyek ini ada dua, yaitu servo motor fisik dan servo motor simulasi. Kalibrasi yang dilakukan berbeda untuk kedua servo itu. Oleh karena itu, terdapat dua kode dengan kalibrasi yang berbeda (silakan melakukan comment / uncomment pada bagian kalibrasi sesuai dengan servo yang digunakan). Selain itu, terdapat juga perbedaan sedikit pada rangkaian fisik. Rangkaian fisik menggunakan resistor pull-up 400 ohm pada pin SDA dan SCL serta resistor pull-up 10k ohm pada pin CS untuk display MAX7219. Kedua resistor berfungsi agar sistem dapat berjalan dengan baik dan lancar. Berikut adalah foto rangkaian fisik dan resistor pull-up yang digunakan:



Gambar 10. Resistor pull-up yang digunakan di rangkaian

Rangkaian ini juga ada push button yang berguna untuk menyalakan / mematikan Laser Module. Push button ini bekerja bersamaan dengan interrupt agar bekerja secara real-time. Berikut adalah gambar letak push button pada rangkaian dan Laser Module menyala:



Gambar 11. Gambar push button dan Laser Module menyala

BAB 4

KESIMPULAN

Proyek ini telah berhasil melaksanakan fungsi-fungsi yang sesuai baik pada rangkaian fisik maupun simulasi di Proteus. Sistem mampu mengintegrasikan komunikasi I2C antara Arduino master dan slave, mengontrol sudut servo secara PWM, serta memproses input dari keypad dan menampilkannya pada tampilan MAX7219. Selain itu, servo dapat digerakkan secara presisi melalui sinyal PWM hasil interpolasi sudut, dan sistem juga dilengkapi dengan fitur interrupt untuk mengontrol Laser Module KY-008 secara real-time.

Namun demikian, terdapat beberapa aspek yang masih perlu diperbaiki dan ditingkatkan. Salah satunya adalah dengan menambahkan kamera sebagai alternatif Laser Module KY-008 untuk memungkinkan sistem melakukan targeting visual secara tiga dimensi. Selain itu, akurasi kontrol motor servo dapat ditingkatkan dengan pemrosesan PWM dalam bentuk desimal. Penggunaan motor servo 360° untuk sumbu azimuth juga memungkinkan rotasi penuh, serta sistem perlu mampu memproses data sudut lebih dari 255 derajat baik melalui komunikasi I2C maupun saat menggerakkan servo.

DAFTAR PUSTAKA

Harditya, Michael and Muhammad Naufal Faza. “Modul 2 SSF: Introduction to Assembly & I/O Programming,” DIGILAB FTUI. [Online] Available: https://docs.google.com/document/d/1s84Y1xrGyJwWXQJgdBQL6bTaFFZtiOsA4_3YGouP1CY/edit?tab=t.0

Harditya, Michael and Muhammad Naufal Faza. “Modul 5 SSF: Aritmatika,” DIGILAB FTUI. [Online] Available: https://docs.google.com/document/d/1cNCJwh6nwIxk909Xpov8moYjrxezg19j5JIndWUcZQU/edit?tab=t.0

Harditya, Michael and Muhammad Naufal Faza. “Modul 6 SSF: Timer,” DIGILAB FTUI. [Online] Available: https://emas2.ui.ac.id/pluginfile.php/5033027/mod_resource/content/2/Modul%206%20SSF_%20Timer.pdf

Harditya, Michael and Muhammad Naufal Faza. “Modul 7 SSF: Interrupt,” DIGILAB FTUI. [Online] Available: https://docs.google.com/document/d/1VW7j3k_scKyOlzo72scSMFU58EgT-TLoiMOshQ48TUA/edit?tab=t.0

Harditya, Michael and Muhammad Naufal Faza. “Modul 8 SSF : SPI & I2C,” DIGILAB FTUI. [Online] Available: https://docs.google.com/document/d/1CsIbwLVUrsKjZ3YhyF0J-gsGWNU1RCQu7JTYMCx3cFY/edit?tab=t.0

Harditya, Michael and Muhammad Naufal Faza. “Modul 9 SSF : Sensor Interfacing,” DIGILAB FTUI. [Online] Available: https://docs.google.com/document/d/14D8bETDw8x-BbeWWfg2QrjEE1WJA17kAZVDu79iCzCs/edit?tab=t.0