



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

ARDUINO RUSSIAN ROULETTE

GROUP 20

Evandita Wiratama Putra	2206059572
Muhammad Billie Elian	2206059446
Valentino Farish Adrian	2206825896
Ajriya Muhammad Arkan	2206031826

PREFACE

Puji syukur kami panjatkan kehadirat Allah SWT Tuhan Yang Maha Esa yang telah menganugerahkan banyak nikmat sehingga kami dapat menyusun laporan proyek akhir praktikum Sistem Siber Fisik ini dengan baik. Laporan ini berisi tentang uraian hasil pembuatan proyek akhir kami mengenai pengimplementasian mikrokontroler ATMega328P pada Arduino UNO dengan menggunakan bahasa Assembly, yaitu Arduino Russian Roulette.

Laporan ini kami susun dengan bantuan dan dukungan berbagai pihak diantaranya; Bapak F. Astha Ekadiyanto, ST., M.Sc. selaku dosen mata kuliah Sistem Siber Fisik dan Bang Aldrian Raffi Wicaksono selaku asisten lab pendamping untuk proyek akhir Sistem Siber Fisik Kelompok 20. Oleh karena itu kami sampaikan terima kasih atas waktu, tenaga dan pikirannya yang telah diberikan.

Kami menyadari bahwa dalam penyusunan laporan proyek akhir ini, masih terdapat berbagai kekurangan dan ketidak sempurnaan. Oleh karena itu, kami sangat mengharapkan saran dan kritik yang membangun dari para pembaca demi perbaikan di masa mendatang. Kami sebagai penyusun laporan ini terbuka untuk segala bentuk masukan yang dapat meningkatkan kualitas laporan ini. Akhir kata, kami berharap laporan proyek akhir ini dapat memberikan manfaat, terutama bagi kelompok kami, serta bagi para pembaca secara umum

Depok, Mei 24, 2024

Group 20

TABLE OF CONTENTS

CHAPTER 1.....	4
INTRODUCTION.....	4
1.1 PROBLEM STATEMENT.....	4
1.3 ACCEPTANCE CRITERIA.....	5
1.4 ROLES AND RESPONSIBILITIES.....	5
1.5 TIMELINE AND MILESTONES.....	5
CHAPTER 2.....	7
IMPLEMENTATION.....	7
2.1 HARDWARE DESIGN AND SCHEMATIC.....	7
2.2 SOFTWARE DEVELOPMENT.....	7
2.3 HARDWARE AND SOFTWARE INTEGRATION.....	8
CHAPTER 3.....	9
TESTING AND EVALUATION.....	9
3.1 TESTING.....	9
3.2 RESULT.....	9
3.3 EVALUATION.....	10
CHAPTER 4.....	11
CONCLUSION.....	11

CHAPTER 1

INTRODUCTION

1.1 PROBLEM STATEMENT

Dalam era digital ini, hiburan dan permainan terus berkembang dengan menggabungkan teknologi terbaru untuk menciptakan pengalaman yang lebih menarik dan interaktif. Salah satu tantangan yang dihadapi oleh pengembang permainan adalah menciptakan game yang tidak hanya menghibur tetapi juga mendidik dan aman untuk dimainkan. Banyak permainan fisik tradisional yang dapat berbahaya jika dilakukan tanpa pengawasan yang tepat atau dengan perangkat yang tidak aman. Salah satu contoh klasik adalah permainan Russian Roulette, yang berisiko tinggi dan berbahaya dalam bentuk aslinya.

Adapun permasalahan yang dihadapi adalah:

- Bagaimana menciptakan sebuah permainan yang dapat memberikan sensasi ketegangan dan keseruan yang mirip dengan Russian Roulette tetapi tanpa risiko cedera atau bahaya nyata?
- Bagaimana menggabungkan komponen elektronik sederhana dengan pemrograman untuk menciptakan permainan interaktif yang mendidik dalam pemrograman mikrokontroler?
- Bagaimana cara mengimplementasikan konsep permainan ini dalam bentuk proyek yang edukatif bagi pelajar atau hobiis yang ingin belajar lebih lanjut tentang pemrograman Assembly pada platform Arduino?

1.2 PROPOSED SOLUTION

Untuk mengatasi permasalahan yang telah diidentifikasi, kami mengusulkan pembuatan proyek "Arduino Russian Roulette" menggunakan mikrokontroler AVR ATMega328P pada platform Arduino, diprogram dengan bahasa Assembly. Proyek ini dirancang untuk memberikan pengalaman bermain yang menegangkan namun aman, serta memberikan kesempatan bagi pengguna untuk belajar dan memahami dasar-dasar pemrograman mikrokontroler dan interaksi dengan perangkat keras elektronik.

Permainan ini mengubah konsep berbahaya Russian Roulette menjadi sebuah permainan elektronik yang sepenuhnya aman. Dalam proyek ini, "tembakan" diwakili oleh sinyal elektronik yang tidak membahayakan pemain, dengan sebuah LED dan buzzer digunakan untuk menandakan tembakan yang berhasil. Pada awal permainan, juri akan memutar potensiometer yang terhubung dengan ADC (Analog to Digital Converter) pada mikrokontroler. Nilai yang diperoleh dari ADC akan menentukan posisi slot peluru secara acak, dengan empat slot yang berbeda. Ini mensimulasikan pengacakan dalam permainan asli namun dengan metode yang aman dan terkendali.

Setiap ronde permainan, pemain akan menekan sebuah tombol untuk "menembak" dirinya. Jika peluru tidak berada pada slot saat ini, pemain akan selamat dan permainan berlanjut ke ronde berikutnya. Jika peluru berada pada slot saat ini, pemain akan kalah dan permainan berakhir. Sebelum ronde berikutnya dimulai, pemain harus menekan photoresistor untuk memutar cylinder revolver ke slot berikutnya, sebelum Arduino (revolver) diserahkan pada pemain selanjutnya. Hal ini memastikan bahwa permainan berjalan dengan baik dan menambah unsur interaktif dalam permainan.

Indikator seperti LED dan buzzer memberikan umpan balik instan mengenai hasil tembakan, sementara hasil setiap ronde juga ditampilkan melalui Serial Monitor, memberikan informasi lebih lanjut dan meningkatkan interaktivitas permainan. Selain memberikan hiburan, proyek ini juga memberikan nilai edukatif bagi pelajar atau hobiis yang ingin belajar lebih dalam tentang pemrograman Assembly pada mikrokontroler AVR. Implementasi ini memungkinkan pengguna untuk mendapatkan pengalaman praktis yang nyata dalam memprogram dan mengoperasikan mikrokontroler serta komponen elektronik terkait.

1.3 ACCEPTANCE CRITERIA

The acceptance criteria of this project are as follows:

1. Membaca input dari potensiometer untuk menentukan posisi awal peluru dalam silinder. Modul Analog-to-Digital Converter (ADC) digunakan untuk membaca input analog dari potensiometer. Potentiometer berfungsi sebagai silinder revolver, yang menentukan posisi peluru pada awal permainan berdasarkan nilai ADC yang terbaca.

2. Membaca input dari photoresistor untuk mensimulasikan aksi hammer, memajukan silinder ke slot berikutnya. ADC juga digunakan untuk membaca input dari photoresistor, yang bertindak sebagai hammer. Sensor interfacing memastikan photoresistor dapat mendeteksi aksi pemain untuk memajukan silinder ke slot berikutnya.
3. Menggunakan komunikasi serial untuk menampilkan hasil setiap ronde, menunjukkan apakah pemain masih hidup atau sudah mati. Komunikasi serial digunakan untuk menampilkan hasil setiap ronde pada Serial Monitor. Hasil ini memberitahu pemain apakah mereka masih dalam permainan (hidup) atau telah kalah (mati), sehingga meningkatkan interaktivitas dan transparansi permainan.
4. Melakukan operasi aritmatika untuk memetakan nilai ADC ke slot peluru tertentu dalam silinder. Operasi aritmatika digunakan untuk memetakan nilai ADC dari potensiometer ke slot peluru dalam silinder. Nilai konversi berkisar dari 0 hingga 1024, yang dibagi menjadi empat segmen yang sama untuk mewakili empat slot, sehingga menentukan lokasi peluru secara akurat.
5. Menggunakan interrupt untuk mereset permainan untuk ronde baru atau permainan baru. Modul interrupt digunakan untuk menangani reset permainan. Ini memungkinkan sistem merespons input pengguna dengan cepat untuk memulai permainan atau ronde baru, memastikan permainan dapat direset dengan cepat dan efisien.

1.4 ROLES AND RESPONSIBILITIES

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

Roles	Responsibilities	Person
Role 1	Software Development, Hardware Implementation	Evandita Wiratama Putra
Role 2	Laporan Akhir, Hardware Design, Hardware Implementation	Muhammad Billie Elian

Role 3	Laporan Akhir, PPT, .md, Hardware Implementation	Valentino Farish Adrian
Role 4	Role 4 responsibilities	Ajriya Muhammad Arkan

Table 1. Roles and Responsibilities

1.5 TIMELINE AND MILESTONES

Insert Gantt Chart here. The Gantt Chart should consist of date interval for:

- a) Hardware Design completion: A milestone indicating the date when the hardware design for the embedded system is finalized, including schematic.
- b) Software Development: The date when the development of the user-created assembly code (software) begins, focusing on specific tasks and functionalities.
- c) Integration and Testing of Hardware and Software: A milestone indicating when the hardware and software components are integrated and tested together to ensure proper functionality.
- d) Final Product Assembly and Testing: A milestone marking when the final system product is assembled, tested, and verified to meet the acceptance criteria.

TASK	21 May	22 May	23 May	24 May	25 May	26 May	27 May
Hardware Design Completion							
Software Development							
Integration & Testing							
Final Product & Testing							

CHAPTER 2

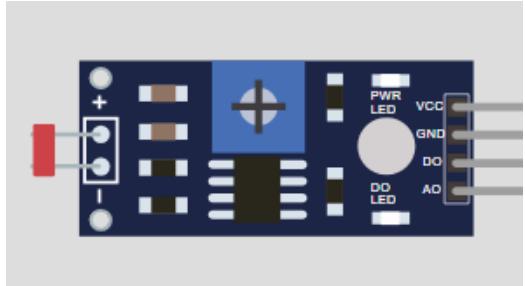
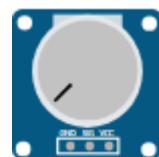
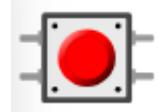
IMPLEMENTATION

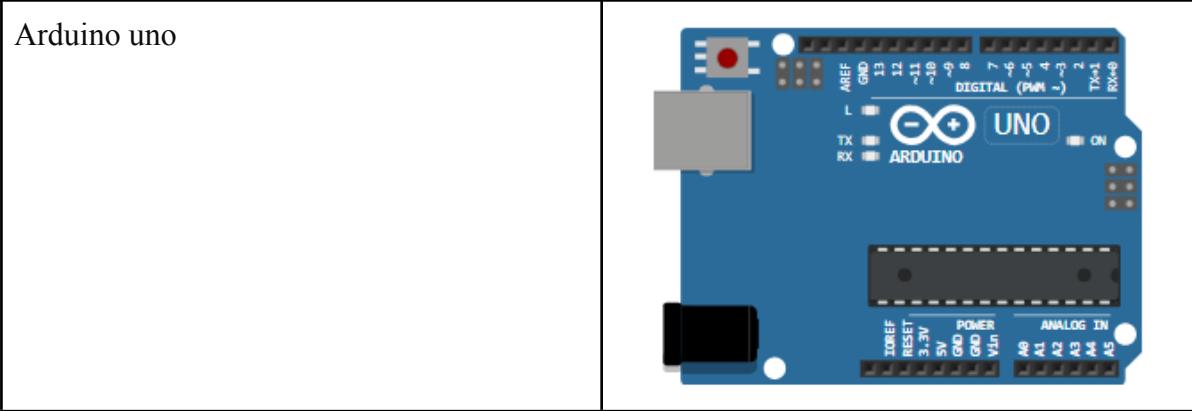
2.1 HARDWARE DESIGN AND SCHEMATIC

Desain perangkat keras untuk proyek ini melibatkan penggunaan beberapa komponen utama :

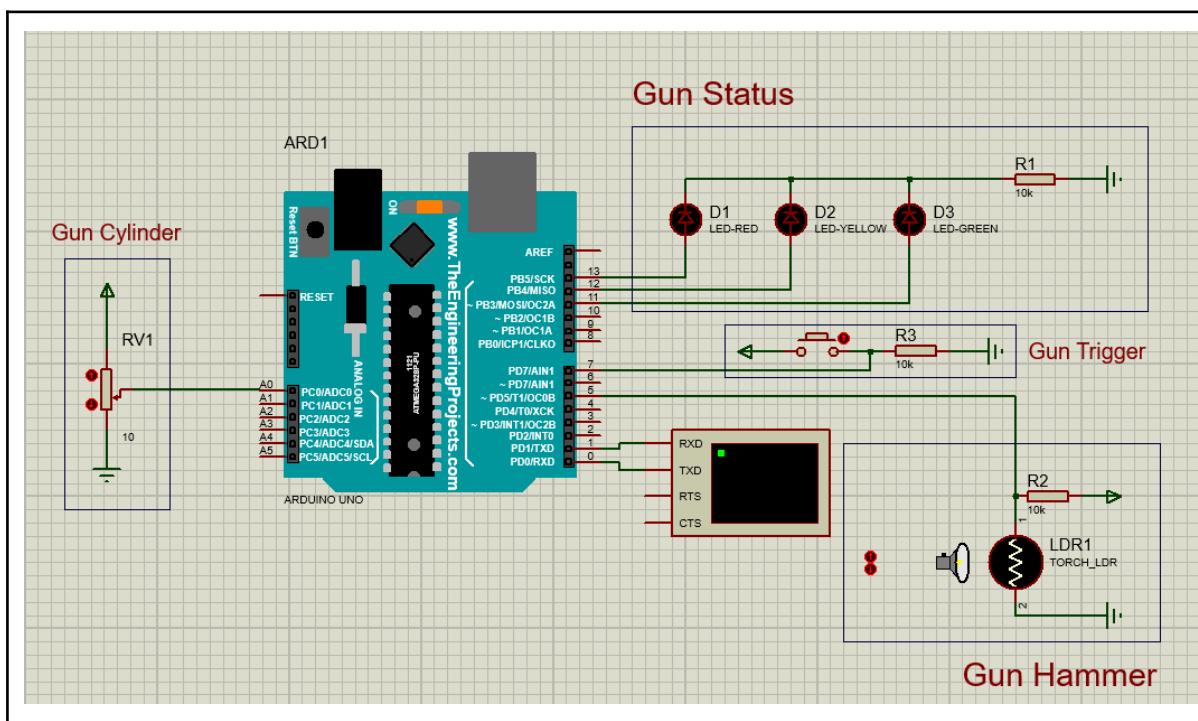
- Photoresistor sebagai sensor cahaya,
- Potensiometer untuk mengatur tingkat kecerahan LED,
- LED sebagai indikator visual,
- Pushbutton sebagai tombol tembak
- Mikrokontroler Arduino Uno sebagai otak utama dari sistem

Hardware Design

Photoresistor-sensor	
Potentiometer	
Push button	
Led	



Schematic



2.2 SOFTWARE DEVELOPMENT

Perangkat lunak untuk proyek ini melibatkan implementasi sebuah permainan menembak real-time menggunakan mikrokontroler Atmel ATmega328P. Perangkat lunak ditulis dalam bahasa Assembly AVR, dengan menggunakan AVR Studio IDE untuk pengembangan. Loop program utama dirancang untuk menangani inisialisasi permainan, tindakan pemain, dan progresi ronde.

Komponen-komponen kunci dari perangkat lunak ini termasuk inisialisasi komunikasi serial untuk tujuan debugging, pengaturan input analog untuk membaca status fotoresistor dan tombol tembak, serta mengintegrasikan input ini dengan logika permainan.

Selain itu, perangkat lunak mengelola ronde permainan, memeriksa kondisi penembakan peluru, memperbarui status pemain (hidup atau mati), dan beralih antar ronde berdasarkan tindakan pemain.

Secara keseluruhan, proses pengembangan perangkat lunak difokuskan pada menciptakan pengalaman permainan yang responsif dan efektif menggunakan mikrokontroler ATmega328P.

Code :

```
#define __SFR_OFFSET 0x00
#include "avr/io.h"
#include <avr/interrupt.h>
;-----
.global main

;=====
.org 0x0000          ; Reset Interrupt Handler
    rjmp main_loop
;=====

;=====
main:
;-----
; Uses R16 temporarily
; Setup for main program
;-----
    RCALL init_serial      ; Initialize Serial Communication
    LDI    R16, 0xff
    OUT    DDRB, R16        ; Set all pin B for output
    CBI    DDRD, 7          ; Set pin pD7 Input for Shoot
;=====
```

```

;=====
main_loop:
;-----
; Getting all data needed to start the game
;-----
    RCALL print_intro          ; Print game is about to start in 5
Second

    RCALL delay_sec           ; 5 second delay
    RCALL delay_sec
    RCALL delay_sec
    RCALL delay_sec
    RCALL delay_sec

    RCALL init_ADC0           ; Setting up analog input for PC0
    RCALL read_ADC            ; Read input for PC0
    RCALL get_counter         ; Get Counter value from analog read
;=====

;=====
rounds:
;-----
; Uses R16 and R25 temporarily
; Each round, a player will try to shoot
;-----
    LDI    R16, 0b00010000
    OUT    PORTB, R16          ; Turn on Yellow LED to indicate a
bullet is ready to be fired
    RCALL shoot                ; Wait for the player to shoot
    CPI    R25, 0               ; Check if a bullet iss fired
    BRNE  no_bullet
;=====
```

```

;=====
;bullet:
;-----
; Uses R16 temporarily
; If a bullet is fired, the player will lose
;-----
        RCALL delay_sec
        RCALL print_death      ; Announce the player is dead

        LDI    R16, 0b00100000
        OUT    PORTB, R16       ; Turn on Red LED to indicate a
bullet is fired

        RCALL delay_sec          ; 3 Second delay before game ends
        RCALL delay_sec
        RCALL delay_sec

        RJMP   main_loop
;=====

;=====
;no_bullet:
;-----
; Uses R16 temporarily
; If a bullet is not fired, the player will survive
;-----
        RCALL delay_sec
        RCALL print_alive        ; Announce the player is dead

        LDI    R16, 0b00001000
        OUT    PORTB, R16       ; Turn on Green LED to indicate a
bullet is not fired

        RCALL init_ADC1          ; Setting up for analog input in PC1
;=====
```

```

;=====
next_round:
;-----
; Uses R19 temporarily
; Enter next round if photoresistor is pressed
;-----
    RCALL read_ADC          ; Read analog input in PC1
    CPI    R19, 3            ; Check if the photoresistor is dark
    BREQ   rounds           ; If the photoresistor is dark, then
enter next round
    RJMP   next_round       ; If not, hold in this round
;=====

;=====
init_ADC0:
;-----
; Uses R20 temporarily
; Input in pin PC0
;-----
    SBI    DDRC, 0          ;set pin PC0 as i/p for ADC0
;-----
    LDI    R20, 0x40          ;internal 2.56V, right-justified data, ADC0
    STS    ADMUX, R20
    LDI    R20, 0x87          ;enable ADC, ADC prescaler CLK/128
    STS    ADCSRA, R20
    RET
;=====

;=====
init_ADC1:

```

```

;-----  

; Uses R20 temporarily  

; Input in pin PC1  

;  

    SBI    DDRC, 1      ;set pin PC1 as i/p for ADC1  

;  

    LDI    R20, 0x41    ;internal 2.56V, right-justified data, ADC1  

    STS    ADMUX, R20  

    LDI    R20, 0x87    ;enable ADC, ADC prescaler CLK/128  

    STS    ADCSRA, R20  

    RET  

;=====

;  

;=====

read_ADC:  

;  

; Uses R17, R20, R21 temporarily  

; R18 is Output low  

; R19 is Output high  

;  

    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  

;=====
```

```

;=====
;init_serial:
;-----
; Uses R24 temporarily
;-----
    CLR    R24
    STS    UCSR0A, R24          ;clear UCSR0A register
    STS    UBRR0H, R24          ;clear UBRR0H register
    LDI    R24, 51              ;& store in UBRRL 51 value
    STS    UBRRL, R24           ;to set baud rate 19200
    LDI    R24, 1<<RXEN0 | 1<<TXEN0 ;enable RXB & TXB
    STS    UCSR0B, R24
    LDI    R24, 1<<UCSZ00 | 1<<UCSZ01 ;asynch, no parity, 1 stop, 8
bits
    STS    UCSR0C, R24
    RET
;=====

;print_serial:
;-----
; Uses R16, R23, R27, R28 temporarily
; This is used for debugging purposes only
;-----
    MOV    R23, R19
    MOV    R16, R19
    RCALL ASCII_LSD
    RCALL LCD_buffer
    STS    UDR0, R16      ;print digit1 of 10-bit result
;
    MOV    R16, R18
    RCALL ASCII_MSD
    RCALL LCD_buffer
    STS    UDR0, R16      ;print digit2 of 10-bit result
;
    MOV    R16, R18

```

```

    RCALL ASCII_LSD
    RCALL LCD_buffer
    STS UDR0, R16      ;print digit3 of 8-bit result
;
;-----+
LDI R16, 0x0A
RCALL LCD_buffer
STS UDR0, R16      ;Enter in Serial

;
;-----+
LDI R16, 0x0D
RCALL LCD_buffer
STS UDR0, R16      ;Finish Statement

;
;-----+
RET                  ;return to loop()
;=====

LCD_buffer:
    LDS R27, UCSR0A
    SBRS R27, UDRE0    ;test data buffer if data can be sent
    RJMP LCD_buffer
    RET
;=====

ASCII_MSD:
    MOV R23, R16      ;save copy of result
    ANDI R16, 0xF0     ;extract & swap high-nibble
    SWAP R16
    SUBI R16, -48      ;R16 = R16 - (48) = R16 + 48
    MOV R28, R16      ;save a copy of high-byte result
    SUBI R28, 58       ;if +ve
    BRPL A_F_D1        ;branch & add 7 to get ASCII A to F

adc_loop1: RET
;=====

ASCII_LSD:
    MOV R16, R23      ;restore copy of result
    ANDI R16, 0XF      ;extract low-nibble
    SUBI R16, -48      ;R16 = R16 - (48) = R16 + 48
    MOV R28, R16      ;save a copy of high-byte result
    SUBI R28, 58       ;if +ve
    BRPL A_F_D0        ;branch & add 7 to get ASCII A to F

adc_loop2: RET
;=====

A_F_D1:

```

```

        SUBI  R16, -7           ;R16 = R16 - (7) = R16 + 7
        RJMP  adc_loop1

;-----
A_F_D0:
        SUBI  R16, -7           ;R16 = R16 - (7) = R16 + 7
        RJMP  adc_loop2
=====

=====
delay_sec:
;-----
; Gives 1 second delay
;-----
        RCALL delay_timer1
        RCALL delay_timer1
        RET
=====

=====
delay_timer1:
;-----
; Uses R20 temporarily
; Create 0.5 sec delay
;-----
.EQU value, 57724          ;value to give 0.5 sec delay
        LDI    R20, hi8(value)
        STS    TCNT1H, R20
        LDI    R20, lo8(value)
        STS    TCNT1L, R20      ;initialize counter TCNT1 = value
;
        LDI    R20, 0b00000000
        STS    TCCR1A, R20
        LDI    R20, 0b00000101
        STS    TCCR1B, R20      ;normal mode, prescaler = 1024
;
timer_loop:
        IN     R20, TIFR1       ;get TIFR1 byte & check
        SBRS  R20, TOV1         ;if TOV1=1, skip next instruction
        RJMP  timer_loop       ;else, loop back & check TOV1 flag
;

```

```

LDI    R20, 1<<TOV1
OUT    TIFR1, R20      ;clear TOV1 flag
;
;-----
LDI    R20, 0b00000000
STS    TCCR1B, R20      ;stop timer0
RET
;=====

;

;=====
get_counter:
;-----
; Uses result from R19 to get counter
; Counter is saved in R25
;-----
LDI    R25, 1
counter_loop:
SUBI  R19, 1
BRLO  done_counter
INC   R25
RJMP  counter_loop
done_counter: RET
;=====

;=====
====

shoot:
;-----
; Hold until button in pin PD7 is clicked
; Once clicked, decrement Counter (R25)
;-----
SBIS  PIND, 7
RJMP  shoot
DEC   R25
RET
;=====

;=====
====

print_alive:
;-----
; Uses R17, R18, R30, R31 temporarily
; Announce the player survived the round
;-----

```

```

LDI    R30, lo8(msg_alive)
LDI    R31, hi8(msg_alive)          ;Z points to string
message

alive_agn:
    LPM    R18, Z+                ;load char of string onto
R18
    CPI    R18, 0                ;check if R18=0 (end of
string)
    BREQ   alive_exit           ;if yes, exit

;-----
alive_loop:
    LDS    R17, UCSR0A
    SBRS   R17, UDRE0          ;test data buffer if data
can be sent
    RJMP   alive_loop
    STS    UDR0, R18            ;send char in R18 to
serial monitor

;-----
    RJMP   alive_agn           ;loop back & get next
character

;-----
alive_exit:
    RET
;-----
msg_alive:
    .ascii "Alive"
    .byte 10,13,0
=====

;-----
print_death:
; -----
; Uses R17, R18, R30, R31 temporarily
; Announce the player did not survive the round
; -----
    LDI    R30, lo8(msg_death)
    LDI    R31, hi8(msg_death)      ;Z points to string
message

death_agn:

```

```

        LPM    R18, Z+                      ;load char of string onto
R18
        CPI    R18, 0                       ;check if R18=0 (end of
string)
        BREQ   death_exit                 ;if yes, exit

;-----
death_loop:
        LDS    R17, UCSR0A
        SBRS   R17, UDRE0                  ;test data buffer if data
can be sent
        RJMP   death_loop
        STS    UDR0, R18                  ;send char in R18 to
serial monitor

;-----
        RJMP   death_agn                ;loop back & get next
character

;-----
death_exit:
        RET
;-----
msg_death:
        .ascii "Death"
        .byte 10,13,0
;=====
;=====

print_intro:
;-----
; Uses R17, R18, R30, R31 temporarily
; Announce the game is about to start
;-----
        LDI    R30, lo8(msg_intro)
        LDI    R31, hi8(msg_intro)          ;Z points to string
message
intro_agn:
        LPM    R18, Z+                  ;load char of string onto
R18
        CPI    R18, 0                   ;check if R18=0 (end of
string)

```

```

BREQ  intro_exit           ;if yes, exit

;-----
intro_loop:
    LDS   R17, UCSR0A
    SBRS  R17, UDRE0          ;test data buffer if data
can be sent
    RJMP  intro_loop
    STS   UDR0, R18           ;send char in R18 to
serial monitor

;-----
    RJMP  intro_agn           ;loop back & get next
character

;-----
intro_exit:
    RET
;-----
msg_intro:
    .ascii "Game starts in 5sec..."
    .byte 10,13,0
=====

```

2.3 HARDWARE AND SOFTWARE INTEGRATION

Dalam kode yang kami buat, integrasi *hardware* dan *software* digunakan untuk membuat permainan sederhana. *Hardware* yang digunakan meliputi LED, fotoresistor, sedangkan untuk *software* adalah program yang mengatur logika permainan.

Berikut integrasi yang kami gunakan pada proyek kali ini :

- Inisialisasi : dilakukan di awal program untuk mengatur pengaturan awal perangkat keras dan perangkat lunak.
- Loop permainan : bagian utama yang mengatur alur permainan. Mulai dari memperkenalkan permainan, menunggu beberapa detik sebelum mulai, membaca nilai potensiometer untuk menentukan yang menembak, dan memulai ronde permainan.
- Ronde permainan : bagian ini mengatur logika untuk setiap ronde permainan.

- Akhir ronde : Berdasarkan apakah peluru ditembakkan atau tidak, permainan akan mengumumkan apakah pemain kalah atau berhasil melewati ronde, dan melanjutkan ke ronde berikutnya atau mengakhiri permainan.

Kode untuk integrasi ke *hardware*

Inisialisasi

```
main:
; Inisialisasi
CALL init_serial ; Inisialisasi Komunikasi Serial
RCALL init_ADC0 ; Pengaturan input analog untuk PC0
LDI R16, 0xff
OUT DDRB, R16 ; Set semua pin B sebagai output

CBI DDRD, 7 ; Set PD7 dan PD5 sebagai input
CBI DDRD, 5

SBI DDRD, 0 ; Set PD0 dan PD1 sebagai output
SBI DDRD, 1
SBI PORTD, 0
SBI PORTD, 1
```

Loop Utama

```
main_loop:
; Menampilkan pesan perkenalan
CALL print_intro ; Menampilkan pesan bahwa permainan akan
segera dimulai dalam 5 detik

; Delay 5 detik sebelum mulai
RCALL delay_sec
RCALL delay_sec
RCALL delay_sec
RCALL delay_sec
RCALL delay_sec
```

```
; Membaca nilai dari potensiometer (ADC0) dan memulai permainan
RCALL read_ADC
RCALL read_ADC
CALL print_serial
RCALL get_counter
```

Integrasi Tombol

```
shoot:
; Menunggu sampai tombol pada pin PD7 ditekan
SBIS PIND, 7
RJMP shoot
DEC R25           ; Mengurangi nilai counter
RET
```

CHAPTER 3

TESTING AND EVALUATION

3.1 TESTING

Testing pertama, kami lakukan melalui rangkaian yang telah dibuat pada Proteus 8, lalu menjalankan program yang dibuat yang sudah diekspor pada sketch arduino. Disini kami melakukan testing, untuk membuktikan program berjalan dengan baik melalui simulasi Proteus, dengan mengubah nilai Potensiometer sebagai silinder. Pengaturan Potensiometer akan di testing melalui range nya, yaitu;

- Slot 1: 0 - 255
- Slot 2: 255 - 511
- Slot 3: 512 - 767
- Slot 4: 768 - 1024

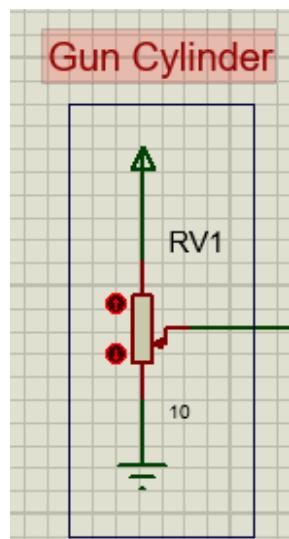
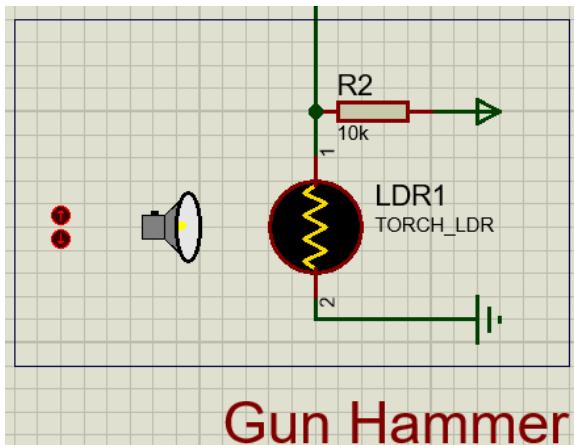


Fig 1. Potensiometer sebagai Gun Cylinder

Kami melakukan testing untuk pengaturan Gun Hammer dengan Photoresistor, yang mengganti slot ke slot selanjutnya, dan kami melakukan testing untuk Gun Status dengan 3 LED (Merah, Kuning, Hijau), yang masing-masing merepresentasikan keadaan revolver ketika permainan dimulai. Selanjutnya, kami juga menghubungkan Arduino ke Serial Monitor, yang akan menampilkan hasil dari permainan.



Gun Hammer

Fig 2. Gun Hammer dengan Photoresistor

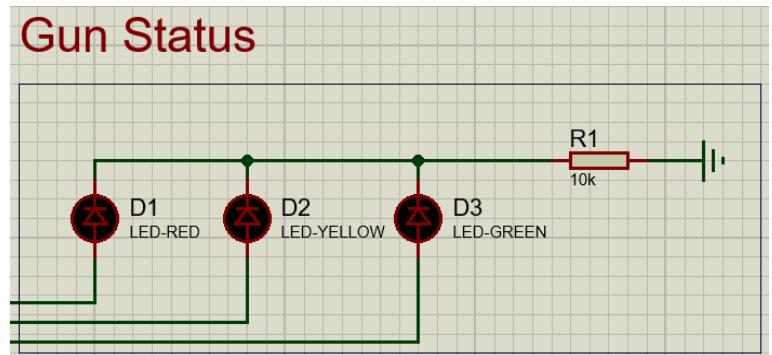


Fig 3. Gun Status dengan 3 LED

Pertama, kami melakukan mengatur Potensiometer, untuk mengatur nilai ke dalam range yang masih didalam slot silinder 1. Disini kami mengatur nilai Potensiometer hingga 62 dalam hexadecimal, atau 98 dalam bilangan decimal.



Fig 5. Slot 1

Kemudian, kami melakukan mengatur Potensiometer, untuk mengatur nilai ke dalam range yang masih didalam slot silinder 2. Disini kami mengatur nilai Potensiometer hingga 116 dalam hexadecimal, atau 278 dalam bilangan decimal.



Fig 6. Slot 2

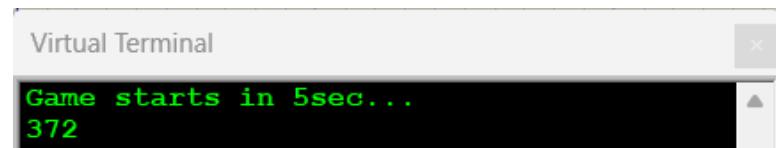
Selanjutnya, kami melakukan mengatur Potensiometer, untuk mengatur nilai ke dalam range yang masih didalam slot silinder 3. Disini kami mengatur nilai Potensiometer hingga 225 dalam hexadecimal, atau 549 dalam bilangan decimal.



```
Virtual Terminal
Game starts in 5sec...
225
```

Fig 7. Slot 3

Terakhir, kami melakukan mengatur Potensiometer, untuk mengatur nilai ke dalam range yang masih didalam slot silinder 4. Disini kami mengatur nilai Potensiometer hingga 372 dalam hexadecimal, atau 882 dalam bilangan decimal.

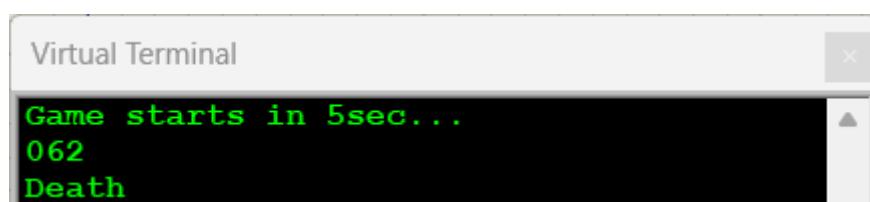


```
Virtual Terminal
Game starts in 5sec...
372
```

Fig 8. Slot 4

3.2 RESULT

Setelah input dari Potensiometer dibaca oleh mikrokontroler dan nilainya dikonversikan dengan ADC (get_counter) pada pin PC0 yang diindikasikan dengan delay 5 detik, LED akan menunjukkan keadaan dari revolver dengan warna kuning, yang menandakan button, pin PD7, siap untuk ditembakkan. Ketika Player menekan button, counter yang menyimpan posisi peluru akan diperiksa. Jika counter menunjukkan nilai 1, maka peluru berada pada Slot 1 dan pemain akan kalah. Hal ini terindikasi dengan LED merah yang menyala, dan pesan “*Death*” ditampilkan melalui serial monitor.



```
Virtual Terminal
Game starts in 5sec...
062
Death
```

Fig 9. Hasil testing Slot 1

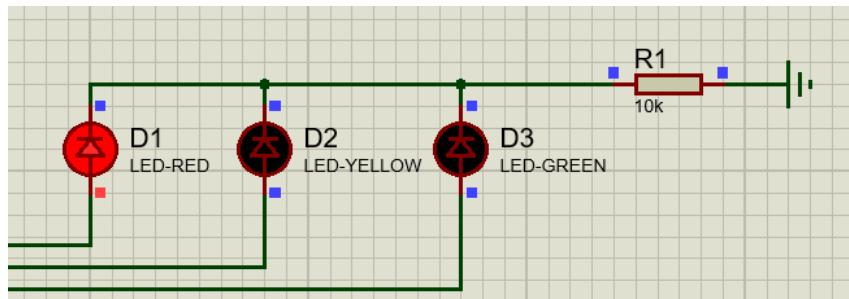


Fig 10. Hasil testing jika Player kalah

Jika nilai yang terbaca dari potensiometer berada dalam rentang 256 - 511, peluru akan berada di Slot 2. Ketika Player menekan button untuk menembak, counter akan diperiksa kembali. Jika counter menunjukkan nilai 2, peluru ada di Slot 2, sehingga Player kalah.

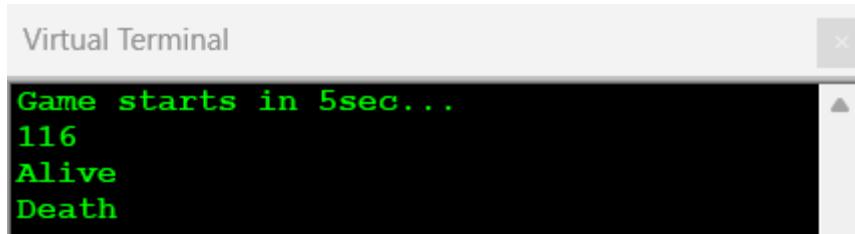


Fig 11. Hasil testing Slot 2

Untuk nilai potensiometer dalam rentang 512 - 767, peluru ditempatkan di Slot 3. Saat Player menekan button untuk menembak, counter diperiksa. Jika counter menunjukkan nilai 3, peluru ada di Slot 3, dan Player akan kalah.



Fig 12. Hasil testing Slot 3

Nilai potensiometer dalam rentang 768 - 1023 menempatkan peluru di Slot 4. Player menekan button untuk menembak dan sistem memeriksa counter. Jika counter menunjukkan nilai 4, peluru ada di Slot 4 dan Player kalah.

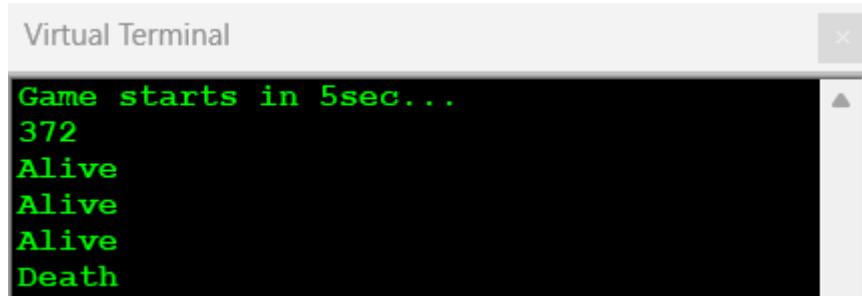


Fig 13. Hasil testing Slot 4

Dari hasil testing yang didapatkan, hasilnya sudah sesuai dengan konsep permainan Russian Roulette yang diinginkan, dan dapat disimpulkan bahwa integrasi dari program yang telah dibuat dengan rangkaian Proteus, sudah berfungsi dengan baik. Penggunaan testing melalui Proteus merupakan bagian yang penting untuk memvalidasi fungsi program, sebelum implementasi ke mikrokontroler ATMega328P dengan Arduino, dan juga sensor yang digunakan yaitu Photoresistor. Dengan testing ini juga kami dapat memastikan seluruh skenario pengujian dapat beroperasi dengan benar di semua kondisi yang kami harapkan pula.

3.3 EVALUATION

Dalam bagian evaluasi, kami akan menjelaskan lebih lanjut proses yang terjadi ketika program dari rangkaian ini berjalan, dari awal inisialisasi, hingga Player kalah. Program memulai dengan inisialisasi komunikasi serial dan pengaturan port I/O, serta menampilkan pesan "Game starts in 5 sec..." sambil menunggu selama 5 detik. ADC kemudian membaca nilai potensiometer di PC0 untuk menentukan slot peluru, yang dikonversi menjadi posisi peluru menggunakan fungsi `get_counter`. Pada awal ronde permainan, LED kuning menyala untuk menunjukkan bahwa permainan siap dimulai. Program menunggu Player menekan button (PD7) untuk menembak. Ketika button ditekan, program memeriksa counter yang menyimpan posisi peluru. Jika counter sesuai dengan posisi peluru, Player kalah. Indikasi kekalahan melibatkan LED merah yang menyala dan pesan "Death" yang ditampilkan melalui serial monitor, diikuti oleh beberapa detik penundaan sebelum permainan dimulai ulang.

Jika Player selamat, LED hijau menyala dan pesan "Alive" ditampilkan melalui serial monitor. Untuk melanjutkan ke ronde berikutnya, Player harus menutup photoresistor (PC1). Program membaca photoresistor untuk memastikan Player siap melanjutkan ke ronde berikutnya. Jika photoresistor cukup gelap, program melanjutkan ke ronde berikutnya dan mengulangi proses di atas. Player kalah ketika posisi peluru sesuai dengan nilai counter yang menunjukkan slot peluru. Indikasi kekalahan melibatkan menyalakan LED merah dan menampilkan pesan "Death" melalui serial monitor. Permainan kemudian berakhir untuk Player tersebut.

CHAPTER 4

CONCLUSION

Proyek ini mengimplementasikan kontrol permainan sederhana menggunakan mikrokontroler ATmega328P (Arduino Uno). Program ini memanfaatkan berbagai fitur mikrokontroler seperti ADC untuk membaca nilai sensor, interrupt untuk menangani input tombol, dan kontrol output untuk mengendalikan LED.

Penggunaan ADC digunakan untuk membaca nilai dari sensor photoresistor dan potensiometer yang kemudian diproses dalam permainan. Sementara itu, input dari push button digunakan untuk mengecek saat pemain menembak.

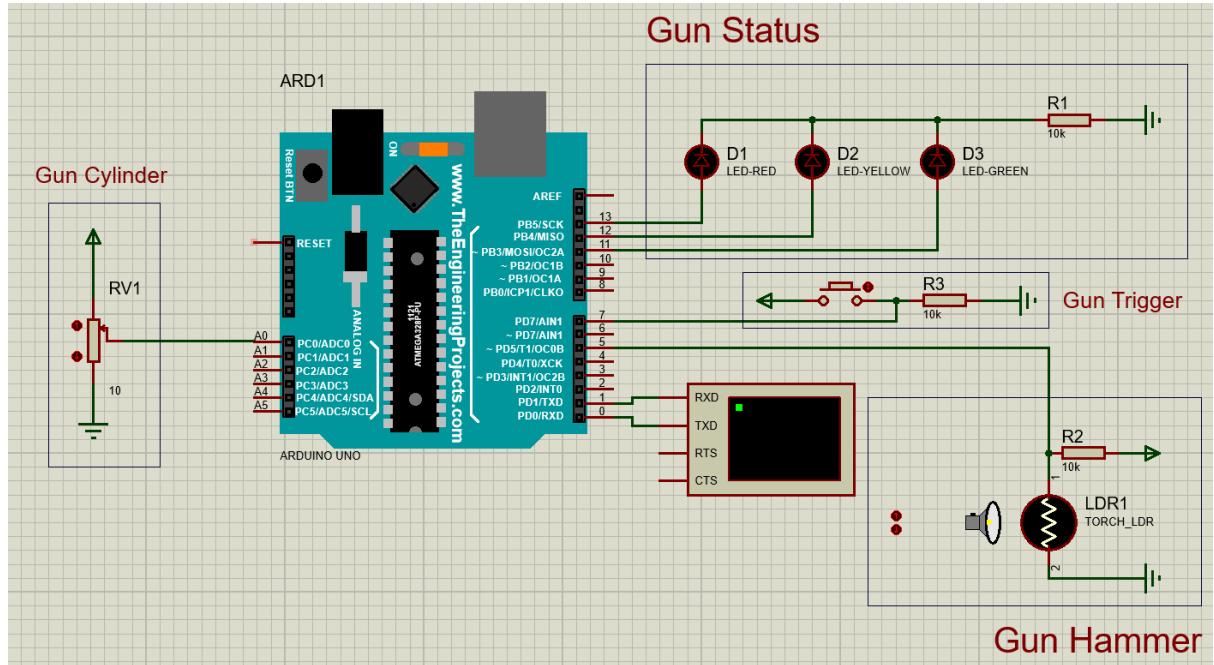
Proyek ini kami harapkan dapat memberikan gambaran tentang bagaimana mengintegrasikan perangkat dengan keras dengan program untuk menciptakan fungsi kontrol. Dengan sedikit modifikasi atau penambahan fitur, proyek ini dapat dibuat menjadi permainan yang lebih kompleks atau aplikasi kontrol yang dapat diaplikasikan dalam kehidupan sehari-hari.

REFERENCES

- [1] N. Shevchenko, "The story behind 'Russian roulette', the Infamous Deadly Game," Russia Beyond, <https://www.rbth.com/history/332732-history-russian-roulette-game-origins> (accessed May 25, 2024).
- [2] Digilab. "Assembly with Arduino.", 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 25, 2024).
- [3] Digilab. "Analog to Digital Converter.", <https://docs.google.com/document/d/1arLt3fqXRw-WgkbqlP1RwYs-XJy9-QAFFEcM21M3u44/edit>. (accessed May 25, 2024).
- [4] Digilab. "Serial Port.", https://docs.google.com/document/d/1rRWvBgL3Nsb_h10131A-1kiGQkrGGNLedYeVIGR9zg/edit. (accessed May 25, 2024).
- [5] Digilab. "Arithmetic.", https://emas2.ui.ac.id/pluginfile.php/4411914/mod_resource/content/1/Modul%205%20SSF%20_Arithmetic.pdf. (accessed May 25, 2024).
- [6] Digilab. "Timer.", https://docs.google.com/document/d/1eYX23D9J5vi9YKXsKkMGc2uem-RJoQITHA_5TkcDfn4/edit. (accessed May 25, 2024).
- [7] Digilab. "External Hardware Interrupts.", https://docs.google.com/document/d/1VW7j3k_scKyOlzo72scSMFU58EgTTLoiM0shQ48TUA/edit. (accessed May 25, 2024).
- [8] Digilab. "Sensor Interfacing.", <https://docs.google.com/document/d/14D8bETDw8x-BbeWWfg2QrjEE1WJA17kAZVDu79iCzCs/edit>. (accessed May 25, 2024).

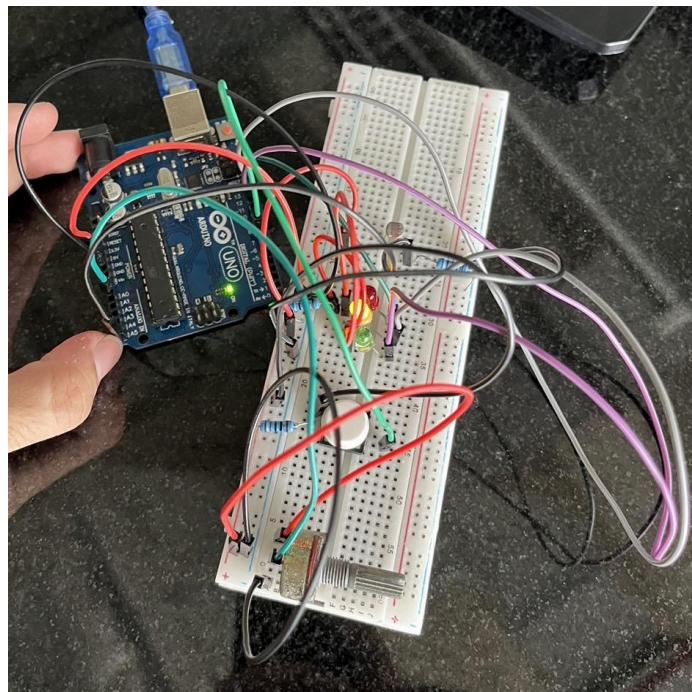
APPENDICES

Appendix A: Project Schematic

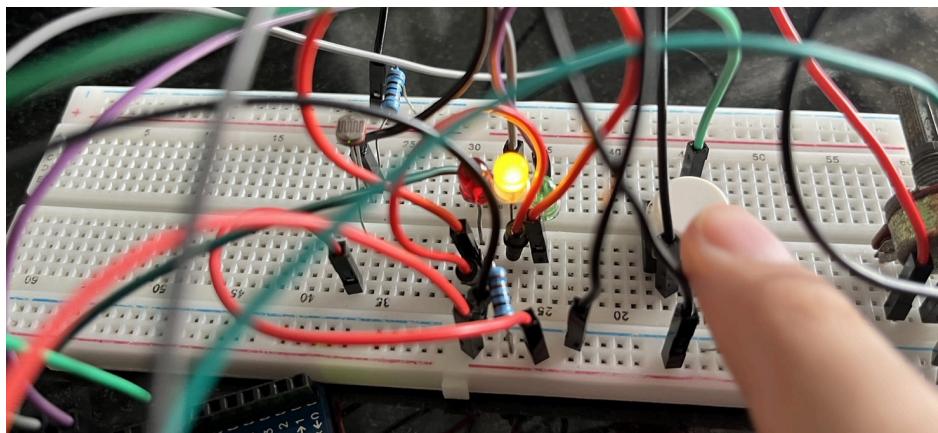


Appendix B: Documentation

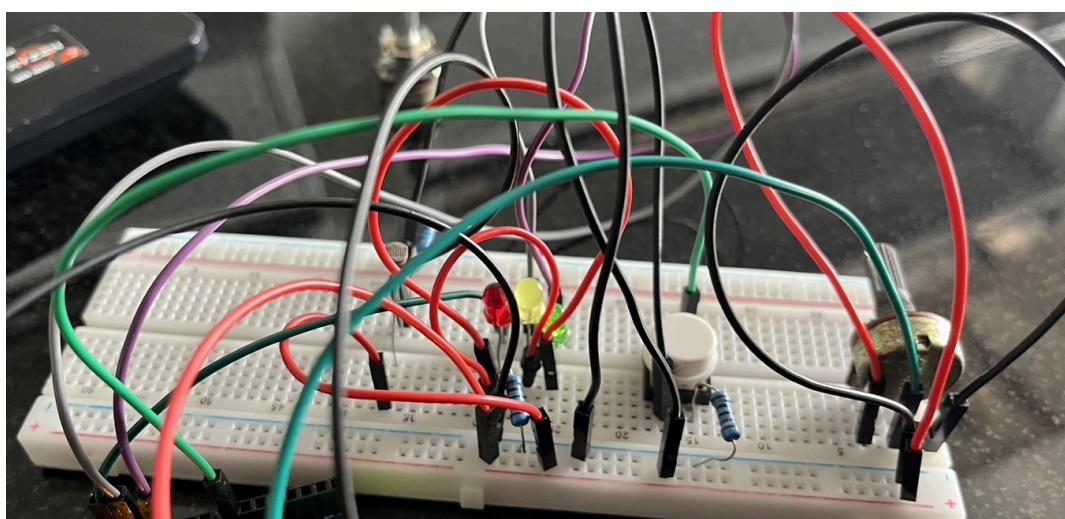
Rangkaian:



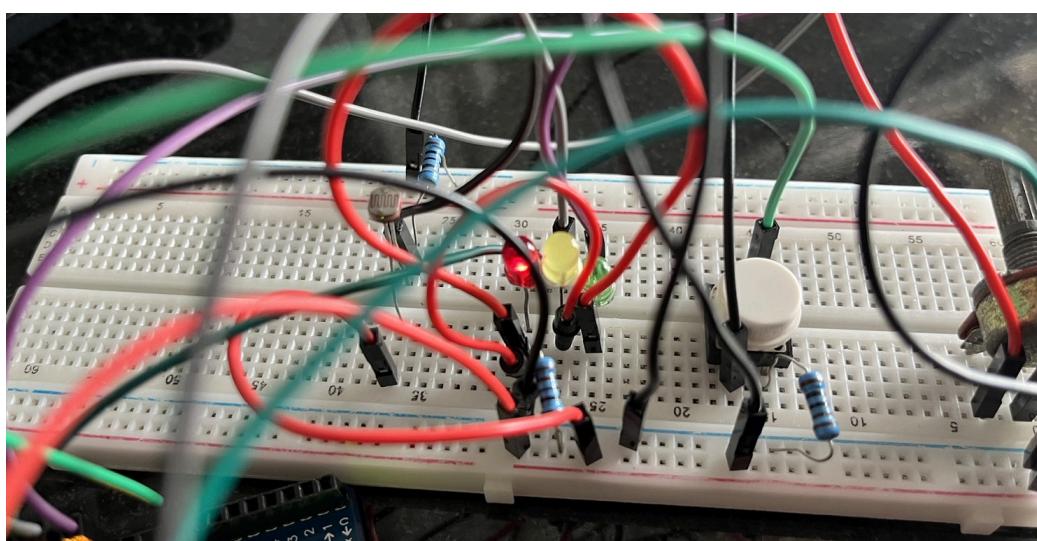
Kondisi Ready:



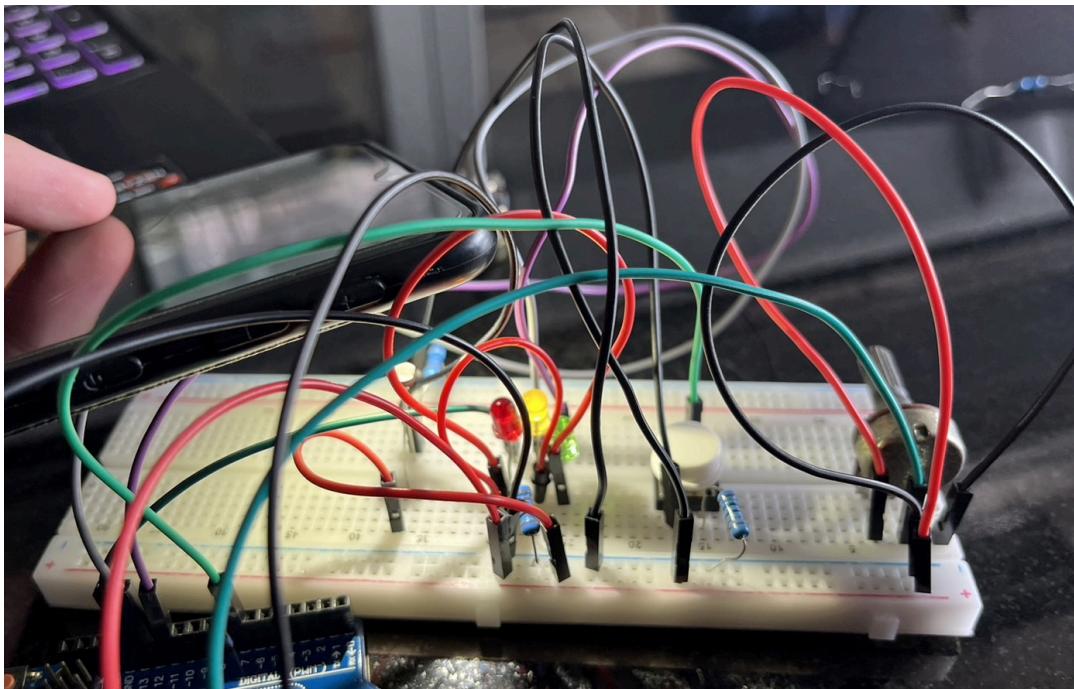
Kondisi Alive:



Kondisi Death:



Hammer Photoresistor:



Cylinder Potentiometer:

