

UNIVERSITAS TELKOM
FAKULTAS TEKNIK ELEKTRO
PROGRAM STUDI TEKNIK KOMPUTER

LAPORAN TUGAS BESAR
MIKROPROSESOR DAN ANTARMUKA

**PERANCANGAN DAN IMPLEMENTASI PERMAINAN “REFLEX
MASTER” BERBASIS BLACKPILL DENGAN SISTEM FEEDBACK
VISUAL DAN AUDIO**



Kelompok: 2

1. Cheyza Amanda Syafira (101032300081) - System & Block Diagram
2. Monica Febyana (101032330068) - Implementasi I2C, UART & Testing
3. Ali Nafis Hamdi (101032330202) - Hardware design & Wiring

Dosen: Hasbi Ash Shiddieqy, S.T., M.T

SEMESTER 5 - TAHUN AKADEMIK 2025/2026

ABSTRAK

Laporan ini membahas perancangan dan implementasi permainan refleks sederhana berbasis mikrokontroler STM32F411 Blackpill sebagai pemenuhan CLO 3. Permainan ini dirancang untuk melatih kecepatan respon pemain dengan menekan tombol yang sesuai dengan LED yang menyala dalam batas waktu tertentu. Sistem dikembangkan menggunakan pemrograman berbasis CMSIS dengan pengaturan peripheral secara langsung melalui register.

Pada implementasinya, sistem memanfaatkan timer TIM1 dan TIM2 untuk menghasilkan sinyal PWM pada lima LED sebagai indikator visual permainan. SysTick timer digunakan sebagai acuan waktu untuk pengaturan animasi LED, delay, serta batas waktu permainan. Input dari lima tombol dibaca menggunakan metode polling GPIO, sedangkan informasi skor dan status permainan ditampilkan melalui LCD 16×2 dan serial monitor menggunakan komunikasi UART. Selain itu, ADC internal digunakan untuk menghasilkan nilai acak sebagai seed pada proses pemilihan LED target.

Sistem terdiri dari beberapa subsistem utama, yaitu subsistem input (tombol), subsistem timing, subsistem output visual dan audio (LED, LCD, dan buzzer), serta subsistem komunikasi UART. Hasil pengujian menunjukkan bahwa sistem dapat berjalan dengan baik dan responsif sesuai dengan rancangan, serta menunjukkan pemahaman mahasiswa terhadap penggunaan peripheral mikrokontroler STM32 pada level register.

Kata Kunci: CMSIS, STM32F411, UART, Timer Interrupt, Register Programming.

DAFTAR ISI

ABSTRAK.....	1
DAFTAR ISI.....	2
BAB I: PENDAHULUAN.....	5
1.1 Latar Belakang.....	5
1.2 Rumusan Masalah.....	5
1.3 Tujuan.....	6
1.4 Batasan Masalah.....	6
1.5 Manfaat.....	7
BAB II: LANDASAN TEORI.....	9
2.1 Mikrokontroler STM32F411.....	9
2.2 Serial Communication Protocol UART.....	9
2.3 Serial Communication Protocol I2C.....	10
2.4 Timer dan Counter.....	11
2.5 Interrupt System.....	12
2.6 GPIO Configuration.....	13
2.7 ADC.....	13
BAB III: REQUIREMENT DAN SPESIFIKASI SISTEM.....	15
3.1 System Requirements.....	15
3.2 Functional Requirements.....	16
3.3 System Specifications.....	17
3.3.1 Hardware Specifications.....	17
3.3.2 Software Specifications.....	19
BAB IV: PERANCANGAN SISTEM.....	20
4.1 Arsitektur Sistem.....	20
4.2 Desain Hardware.....	20
4.2.1 Schematic Diagram.....	20
4.2.2 Bill of Materials (BOM).....	21
4.3 Desain Software.....	23
4.3.1 Flowchart Sistem.....	23
4.3.2 State Machine Diagram.....	24
4.4 Desain Subsistem.....	24
4.4.1 Subsistem Input (EXTI & GPIO).....	24
4.4.2 Subsistem Timing (Timer Dynamics).....	25
4.4.3 Subsistem Display (UART Communication).....	25
4.4.4 Subsistem Display (I2C Communication).....	27
4.4.5 Subsistem ADC.....	28
BAB V: IMPLEMENTASI.....	30
5.1 Implementasi Hardware.....	30

5.2 Implementasi Software (Register-Based).....	32
5.2.1 RCC Configuration.....	32
5.2.2 GPIO Configuration.....	34
5.2.3 UART Configuration.....	35
5.2.4 I2C Configuration.....	36
5.2.5 Timer Configuration.....	37
5.2.6 External Interrupt Configuration.....	38
5.2.7 Interrupt Service Routines.....	40
5.3 Pembagian Tugas Kelompok.....	40
BAB VI: PENGUJIAN DAN ANALISIS.....	42
6.1 Metodologi Pengujian.....	42
6.2 Pengujian Sistem Keseluruhan.....	46
6.3 Troubleshooting.....	48
BAB VII: KESIMPULAN DAN SARAN.....	50
7.1 Kesimpulan.....	50
7.2 Saran.....	51
AI Editing Statement.....	52
DAFTAR PUSTAKA.....	53
LAMPIRAN.....	57

BAB I: PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi sistem tertanam membuat mikrokontroler banyak digunakan pada aplikasi yang membutuhkan respon cepat dan pengolahan input–output secara real-time. Salah satu penerapannya adalah pada permainan interaktif, di mana sistem harus mampu membaca input pengguna dengan cepat dan memberikan umpan balik secara langsung. Dalam hal ini, mikrokontroler berperan sebagai pengendali utama yang mengatur alur permainan, membaca tombol, serta mengendalikan output seperti LED, buzzer, dan tampilan.

Proyek Reflex Master Game menggunakan mikrokontroler STM32F411 Blackpill untuk menguji kecepatan reaksi pemain dalam menekan tombol yang sesuai dengan LED target dalam batas waktu tertentu. Sistem ini dirancang agar dapat bekerja secara presisi dengan memanfaatkan timer dan interrupt. TIM3 digunakan sebagai countdown timer untuk membatasi waktu respons pemain, sedangkan external interrupt (EXTI) digunakan agar pembacaan tombol dapat dilakukan secara responsif tanpa polling.

Selain itu, komunikasi UART diterapkan untuk mengirimkan skor, level, dan waktu reaksi ke komputer melalui serial monitor, sehingga memudahkan proses pengujian dan evaluasi hasil permainan. Penggunaan PWM dimanfaatkan untuk mengatur tingkat kecerahan LED pada setiap level, sementara ADC internal digunakan sebagai sumber nilai acak untuk menentukan LED target.

Proyek ini dikembangkan menggunakan pemrograman berbasis register (CMSIS) sehingga mahasiswa dapat memahami cara kerja peripheral STM32 secara langsung. Dengan demikian, permainan Reflex Master tidak hanya berfungsi sebagai hiburan, tetapi juga sebagai media pembelajaran dalam penerapan GPIO, timer, interrupt, UART, ADC, dan PWM pada sistem embedded.

1.2 Rumusan Masalah

1. Bagaimana mengimplementasikan komunikasi serial UART berbasis register pada mikrokontroler Blackpil?

2. Bagaimana mengkonfigurasi timer (TIM3) untuk menghasilkan countdown timer per level serta timer interrupt untuk mendeteksi timeout dalam permainan secara presisi?
3. Bagaimana mengimplementasikan sistem interrupt menggunakan EXTI untuk empat tombol input dan timer interrupt sehingga respons tombol dapat dibaca secara cepat tanpa polling?
4. Bagaimana mengintegrasikan peripheral UART, TIM3, EXTI, ADC ke dalam satu sistem permainan *Reflex Master* yang stabil, sinkron, dan berfungsi sesuai tujuan?

1.3 Tujuan

- **Umum:** Merancang dan mengimplementasikan permainan *Reflex Master* berbasis mikrokontroler Blackpill yang mampu mengukur waktu reaksi pemain secara akurat melalui pemanfaatan UART, timer, interrupt, serta pemrograman tingkat register pada sistem tertanam.
- **Khusus:**
 - a. Menggunakan UART untuk mengirim skor dan statistik permainan ke PC melalui serial monitor.
 - b. Mengonfigurasi TIM3 sebagai countdown timer per level dan memanfaatkan interrupt timer untuk mendeteksi timeout permainan.
 - c. Mengatur register GPIO, TIM, USART, NVIC, ADC, dan PWM untuk mengendalikan input tombol, timer, komunikasi serial, serta karakteristik LED dan buzzer.
 - d. Menggabungkan EXTI untuk lima tombol input, ADC untuk random number generation, PWM untuk pengaturan brightness LED per level, serta seluruh modul sistem menjadi permainan yang stabil dan responsif.

1.4 Batasan Masalah

- **Hardware:**
 - a. Mikrokontroler yang digunakan adalah Blackpill STM32F411.
 - b. Input menggunakan 5 tombol yang terhubung ke EXTI.

- c. Output menggunakan LED (dikendalikan dengan PWM) dan buzzer.
 - d. ADC digunakan untuk menghasilkan angka acak sederhana.
 - e. Komunikasi eksternal hanya melalui UART ke PC.
- Software:
 - a. Pemrograman dilakukan dalam bahasa C dengan pendekatan register-level programming.
 - b. Konfigurasi terbatas pada peripheral GPIO, UART, TIM3, EXTI, ADC, dan PWM.
 - c. Timer yang digunakan hanya TIM3 sebagai countdown timer dan sumber interrupt timeout.
 - d. Tidak menggunakan library HAL atau STM32CubeMX, hanya akses register langsung.
 - e. Data yang diproses terbatas pada waktu reaksi, countdown, pembacaan input, dan pengiriman skor.
- Lingkup:
 - a. Fokus pada implementasi permainan respons cepat dengan stimulus LED/buzzer dan input tombol.
 - b. Tidak mencakup penyimpanan data jangka panjang, layar LCD, komunikasi wireless, atau multiplayer.
 - c. Pengujian dilakukan hanya pada satu perangkat Blackpill dan satu pemain.
 - d. Pengembangan fitur lanjutan seperti level kompleks, efek suara, atau UI komputer tidak dibahas dalam proyek ini.

1.5 Manfaat

- Memberikan pemahaman langsung mengenai konsep dasar sistem tertanam, seperti pengolahan input-output, pemanfaatan timer, interrupt, UART, ADC, dan PWM, sehingga membantu mahasiswa memahami cara kerja mikrokontroler secara nyata dan terstruktur.
- Meningkatkan keterampilan dalam pemrograman tingkat register pada STM32, konfigurasi peripheral seperti GPIO, TIM, EXTI, USART, ADC,

dan PWM, serta kemampuan debugging melalui komunikasi serial, sehingga memperkuat kompetensi teknis dalam embedded system.

- Menghasilkan permainan respons cepat yang dapat digunakan untuk melatih waktu reaksi, sekaligus menjadi contoh implementasi sistem embedded yang sederhana, mudah diuji, dan potensial dikembangkan untuk aplikasi pelatihan, edukasi, atau demonstrasi teknologi.

BAB II: LANDASAN TEORI

2.1 Mikrokontroler STM32F411

Blackpill STM32F411CEU6 merupakan papan pengembangan yang menggunakan mikrokontroler STM32F411 berbasis arsitektur ARM Cortex-M4 32-bit. Inti prosesor Cortex-M4 dirancang untuk aplikasi real-time karena memiliki performa komputasi yang cukup tinggi, didukung oleh Floating Point Unit (FPU) dan instruksi DSP. Mikrokontroler ini mampu bekerja hingga frekuensi 100 MHz serta dilengkapi dengan 512 KB Flash dan 128 KB SRAM, sehingga memadai untuk aplikasi dengan kebutuhan respon cepat dan pengolahan data yang tidak terlalu kompleks.

Dari sisi arsitektur internal, STM32F411 menggunakan struktur bus AHB dan APB untuk menghubungkan CPU dengan berbagai peripheral seperti GPIO, timer, UART, ADC, dan modul lainnya. Mekanisme interrupt dikelola oleh Nested Vectored Interrupt Controller (NVIC) yang memungkinkan penanganan interupsi secara cepat dan terstruktur. Hal ini sangat penting pada sistem yang membutuhkan respon segera terhadap kejadian tertentu, seperti penekanan tombol atau terjadinya timeout pada timer.

Dengan karakteristik tersebut, Blackpill STM32F411 dinilai sesuai untuk implementasi permainan Reflex Master. Mikrokontroler ini mampu menjalankan logika permainan secara real-time, menangani input tombol melalui external interrupt (EXTI), menghasilkan pengaturan waktu yang presisi menggunakan timer, serta mengendalikan output berupa LED dan buzzer. Selain itu, komunikasi data ke komputer dapat dilakukan melalui UART. Pendekatan pemrograman berbasis register yang digunakan pada proyek ini juga memberikan pemahaman lebih dalam mengenai cara kerja peripheral dan arsitektur internal mikrokontroler secara langsung.

2.2 Serial Communication Protocol UART

Universal Asynchronous Receiver/Transmitter (UART) merupakan salah satu metode komunikasi serial yang banyak digunakan pada sistem tertanam

karena sederhana dan mudah diimplementasikan. UART bekerja tanpa sinyal clock bersama dan menggunakan dua jalur utama, yaitu TX untuk pengiriman data dan RX untuk penerimaan data. Pada proyek permainan Reflex Master berbasis Blackpill STM32F411, UART dimanfaatkan untuk mengirim informasi skor, level, dan status permainan ke PC sehingga hasil permainan dapat dipantau melalui serial monitor secara real-time.

Kecepatan komunikasi UART ditentukan oleh baud rate, yang menyatakan jumlah bit yang dikirim per detik. Pada mikrokontroler STM32, pengaturan baud rate dilakukan dengan menghitung nilai pembagi berdasarkan frekuensi clock USART, kemudian nilai tersebut dimasukkan ke dalam register USART_BRR. Kesamaan baud rate antara pengirim dan penerima sangat penting agar data dapat diterima dengan benar.

Data pada komunikasi UART dikirim dalam bentuk frame yang terdiri dari start bit, data bit, dan stop bit, dengan konfigurasi yang umum digunakan adalah 8 data bit tanpa parity dan 1 stop bit (8N1). Pengendalian UART pada STM32 dilakukan melalui beberapa register utama, seperti USART_CR1 untuk mengaktifkan modul dan transmitter, USART_SR untuk memantau status pengiriman data, serta USART_DR sebagai register data. Dengan konfigurasi ini, UART dapat bekerja secara andal sebagai media komunikasi sederhana dalam sistem embedded.

2.3 Serial Communication Protocol I2C

Inter-Integrated Circuit (I2C) merupakan salah satu protokol komunikasi serial sinkron yang banyak digunakan pada sistem tertanam karena hanya memerlukan dua jalur komunikasi, yaitu Serial Data Line (SDA) dan Serial Clock Line (SCL). Protokol ini memungkinkan beberapa perangkat periferal terhubung dalam satu bus yang sama dengan konsep master dan slave. Pada sistem berbasis mikrokontroler seperti Blackpill STM32F411, I2C umum digunakan untuk berkomunikasi dengan perangkat eksternal seperti LCD, sensor, dan modul memori, sehingga dapat menghemat penggunaan pin input/output.

Pada komunikasi I2C, perangkat master berperan dalam menghasilkan sinyal clock serta menginisialisasi proses komunikasi, sedangkan perangkat slave merespons berdasarkan alamat unik yang dimilikinya. Setiap proses komunikasi diawali dengan kondisi start, diikuti pengiriman alamat slave beserta bit Read/Write, kemudian transfer data dalam bentuk byte. Setelah setiap byte data dikirimkan, slave akan memberikan sinyal acknowledge (ACK) sebagai tanda bahwa data diterima dengan benar. Proses komunikasi diakhiri dengan kondisi stop yang menandakan berakhirnya transfer data pada bus I2C.

Mikrokontroler STM32F411 menyediakan peripheral I2C internal yang dapat dikonfigurasi melalui beberapa register utama, seperti I2C_CR1 untuk mengaktifkan modul I2C, I2C_CCR untuk pengaturan kecepatan clock, serta I2C_DR sebagai register data. Kecepatan komunikasi I2C dapat disesuaikan sesuai kebutuhan sistem, umumnya menggunakan Standard Mode (100 kHz) atau Fast Mode (400 kHz). Dengan mekanisme ini, I2C mampu bekerja secara stabil untuk komunikasi jarak pendek antarperangkat dalam sistem embedded.

Dalam proyek permainan Reflex Master, protokol I2C digunakan sebagai media komunikasi antara mikrokontroler dan modul LCD berbasis I2C untuk menampilkan informasi permainan seperti skor, level, dan status permainan. Penggunaan I2C memungkinkan tampilan data dilakukan secara efisien tanpa membebani banyak pin mikrokontroler. Selain itu, penerapan komunikasi I2C pada proyek ini memberikan pemahaman praktis mengenai konsep komunikasi serial sinkron serta cara kerja peripheral I2C pada mikrokontroler STM32 secara langsung.

2.4 Timer dan Counter

Timer dan counter merupakan salah satu modul penting pada mikrokontroler yang digunakan untuk pengaturan waktu dan perhitungan secara presisi. Timer bekerja berdasarkan sumber clock internal mikrokontroler, sehingga frekuensi clock sangat memengaruhi ketelitian pengukuran waktu. Pada permainan Reflex Master, timer dimanfaatkan sebagai countdown timer untuk membatasi waktu respons pemain pada setiap level permainan.

Pengaturan timer dilakukan melalui prescaler dan Auto-Reload Register (ARR). Prescaler berfungsi membagi frekuensi clock agar sesuai dengan kebutuhan sistem, sedangkan ARR menentukan batas hitungan maksimum timer sebelum kembali ke nilai awal. Dengan pengaturan kedua parameter ini, sistem dapat menghasilkan delay dan batas waktu yang stabil sesuai dengan alur permainan.

Mikrokontroler STM32 menyediakan beberapa mode timer, di antaranya mode counter dan PWM. Mode counter digunakan sebagai dasar perhitungan waktu pada countdown permainan, sedangkan mode PWM dimanfaatkan untuk mengatur tingkat kecerahan LED pada setiap level. Pemanfaatan timer dengan beberapa mode ini membuat sistem permainan dapat berjalan secara real-time dan responsif.

2.5 Interrupt System

Sistem interrupt merupakan mekanisme pada mikrokontroler yang memungkinkan prosesor merespons suatu kejadian tanpa harus menunggu alur program utama selesai. Dengan mekanisme ini, sistem dapat memberikan respons secara cepat dan lebih efisien dibandingkan metode polling. Pada proyek Reflex Master, interrupt digunakan untuk mendeteksi penekanan tombol pemain serta menangani batas waktu permainan, sehingga respon sistem dapat berjalan secara real-time.

Pada arsitektur ARM Cortex-M, pengelolaan interrupt dilakukan oleh Nested Vectored Interrupt Controller (NVIC). NVIC berfungsi mengatur aktivasi interrupt dan menentukan prioritas setiap sumber interrupt. Fitur nested interrupt memungkinkan interrupt dengan prioritas lebih tinggi untuk dijalankan terlebih dahulu. Dalam proyek ini, NVIC digunakan untuk mengelola interrupt dari timer dan tombol input agar sistem tetap sinkron.

Sementara itu, External Interrupt (EXTI) digunakan untuk mendeteksi perubahan logika pada pin input, seperti saat tombol ditekan. EXTI dapat dikonfigurasi berdasarkan jenis tepi sinyal, misalnya falling edge atau rising edge.

Pada permainan Reflex Master, EXTI dimanfaatkan sebagai pemicu input pemain sehingga sistem dapat langsung merespons tanpa keterlambatan. Pemanfaatan interrupt ini mendukung kinerja permainan agar lebih responsif dan stabil.

2.6 GPIO Configuration

GPIO (General Purpose Input/Output) merupakan peripheral dasar pada mikrokontroler yang berfungsi sebagai penghubung antara sistem dengan perangkat eksternal melalui pin digital. Setiap pin GPIO dapat dikonfigurasikan sebagai input atau output sesuai kebutuhan aplikasi. Pada mode input, GPIO dapat diatur sebagai floating, pull-up, atau pull-down untuk menjaga kestabilan logika, serta sebagai mode analog ketika pin digunakan oleh ADC. Pada permainan Reflex Master, GPIO input digunakan untuk membaca penekanan tombol pemain, sedangkan mode analog dimanfaatkan sebagai masukan ADC untuk menghasilkan variasi nilai acak.

Pada mode output, GPIO dapat dikonfigurasikan sebagai push-pull atau open-drain. Konfigurasi push-pull memungkinkan pin menghasilkan logika tinggi dan rendah secara aktif, sehingga cocok digunakan untuk mengendalikan perangkat seperti LED. Sementara itu, mode open-drain lebih umum digunakan pada aplikasi tertentu seperti komunikasi berbasis bus. Dalam proyek ini, GPIO output dikonfigurasikan dalam mode push-pull untuk mengontrol LED sebagai indikator visual selama permainan berlangsung.

2.7 ADC

Analog to Digital Converter (ADC) merupakan salah satu peripheral pada mikrokontroler yang berfungsi mengubah sinyal analog berupa tegangan menjadi data digital agar dapat diolah oleh sistem. Nilai tegangan yang masuk akan dikonversi menjadi bilangan digital berdasarkan resolusi ADC yang digunakan. Pada permainan Reflex Master, ADC dimanfaatkan sebagai sumber nilai acak dengan memanfaatkan fluktuasi tegangan pada pin analog, sehingga alur permainan menjadi lebih bervariasi dan tidak bersifat monoton.

ADC menyediakan beberapa mode konversi, seperti *single conversion* yang melakukan konversi satu kali setiap perintah, serta *continuous conversion* yang bekerja secara terus-menerus. Selain itu, terdapat *scan mode* yang memungkinkan pembacaan beberapa channel secara berurutan. Mode-mode tersebut dipilih sesuai kebutuhan sistem, yaitu untuk mengambil sampel tegangan analog sebagai dasar pembangkitan nilai acak.

Parameter penting lainnya adalah *sampling time*, yaitu waktu yang dibutuhkan ADC untuk mengambil sampel sebelum proses konversi dilakukan. Pengaturan sampling time berpengaruh terhadap kestabilan dan keakuratan data. Sampling yang terlalu singkat dapat menghasilkan data yang kurang stabil, sedangkan sampling yang lebih lama cenderung memberikan hasil yang lebih akurat meskipun dengan kecepatan pembacaan yang lebih rendah.

BAB III: REQUIREMENT DAN SPESIFIKASI SISTEM

3.1 System Requirements

Sistem Reflex Master dibuat untuk mengukur sekaligus melatih waktu reaksi pemain dengan memanfaatkan mikrokontroler Blackpill STM32F411. Sistem ini dirancang agar mampu membaca input dengan cepat dan memberikan respons berupa tampilan visual maupun suara secara langsung, sehingga pemain dapat memahami kondisi permainan secara jelas. Persyaratan utama sistem adalah sebagai berikut:

a. Input Handling

Input berasal dari lima buah push button yang digunakan sebagai tombol mulai dan tombol respons pemain. Seluruh tombol dihubungkan ke jalur external interrupt (EXTI) agar penekanan tombol dapat terdeteksi secara cepat. Waktu respons sistem ditargetkan tidak lebih dari 10 ms sehingga pengukuran reaksi pemain tetap akurat tanpa menggunakan metode polling.

b. Output/Display

Output sistem terdiri dari LED, LCD, dan buzzer. LED berfungsi sebagai indikator visual utama yang menyala sesuai dengan level dan kondisi permainan. LCD digunakan untuk menampilkan informasi seperti skor, level, dan waktu reaksi pemain. Sementara itu, buzzer digunakan sebagai umpan balik suara, misalnya saat permainan dimulai, jawaban benar atau salah, serta ketika waktu habis. Seluruh output diperbarui mengikuti jalannya permainan secara real-time.

c. Communication

Sistem komunikasi pada mikrokontroler menggunakan dua protokol, yaitu UART dan I2C. UART mengirimkan data ke PC secara serial dan asinkron, mencakup skor, level, waktu reaksi, serta status permainan seperti start, hit, miss, dan timeout, sehingga data dapat langsung dipantau melalui serial monitor. Sementara I2C digunakan untuk mengontrol LCD, mengirim perintah dan menampilkan informasi permainan secara

real-time, sehingga pemain dapat melihat skor dan level langsung di layar tanpa harus bergantung pada PC.

3.2 Functional Requirements

Tabel 3.1 Tabel *Functional Requirements*

ID	Requirement	Priority
FR-01	Sistem harus dapat mendeteksi penekanan tombol Start melalui mekanisme external interrupt (EXTI) untuk memulai permainan dari level 1 dengan skor awal nol.	High
FR-02	Sistem harus menghasilkan pola LED acak menggunakan ADC sebagai <i>random seed</i> dan menyalakan LED sesuai pola tersebut pada setiap level.	High
FR-03	Jika pemain berhasil menekan tombol yang benar sebelum waktu habis, sistem harus menambah skor pemain, menampilkan respon pada LCD, menghasilkan bunyi pada buzzer dengan pola suara pendek melalui pengaturan delay berbasis timer sistem.	Medium
FR-04	Jika pemain menekan tombol yang salah atau waktu habis, sistem harus menghentikan permainan, menampilkan teks Game Over pada LCD, mengaktifkan buzzer dengan pola suara panjang.	Medium
FR-05	Sistem harus mengirimkan skor permainan, waktu reaksi, dan status akhir level melalui UART ke PC untuk dipantau melalui serial monitor.	Medium
FR-06	Sistem harus menampilkan informasi permainan secara lokal melalui LCD yaitu status permainan, skor dan level pemain.	Low

3.3 System Specifications

3.3.1 Hardware Specifications

Komponen Utama:

Tabel 3.2 Tabel *Hardware Specification*

Komponen	Spesifikasi	Jumlah	Fungsi
Mikrokontroler	STM32F411CEU6	1	Komponen Processing
LCD	16x2 Green UART	1	Output Tampilan
Button	Push button	5	Input Pengguna
LED	5mm	5	Indikator Tampilan + Target
Buzzer	Passive	1	Indikator Suara
Resistor	1/4W	5	Pembatasi Arus
Potentiometer	10K	1	Pembatasi Arus
USB to TTL	CH340	1	Komunikasi Serial ke PC

Pin Mapping:

Tabel 3.3 Tabel Pin Mapping

Pin STM32	Connect To	Fungsi	Peripheral
PA0	LED 1 (+ resistor)	PWM LED	TIM2_CH1 (AF1)
PA1	LED 2 (+ resistor)	PWM LED	TIM2_CH2 (AF1)
PA2	LED 3 (+ resistor)	PWM LED	TIM2_CH3 (AF1)

Pin STM32	Connect To	Fungsi	Peripheral
PA3	LED 4 (+ resistor)	PWM LED	TIM2_CH4 (AF1)
PA8	LED 5 (+ resistor)	PWM LED	TIM1_CH1 (AF1)
PB0	Push Button 1	External Interrupt	EXTI0
PB1	Push Button 2	External Interrupt	EXTI1
PB12	Push Button 3	External Interrupt	EXTI3
PB3	Push Button 4	External Interrupt	EXTI4
PB4	Push Button 5	External Interrupt	EXTI12
PB10	Buzzer	Output Suara	GPIO Output
PB8	LCD SCL	I2C Clock	GPIO Output Open-Drain
PB9	LCD SDA	I2C Data	GPIO Output Open-Drain
PA9	UART1 TX	Serial Output	USART1 AF7
PA7	ADC Input (pot/knob)	Random number / entropy source	ADC1 Channel7

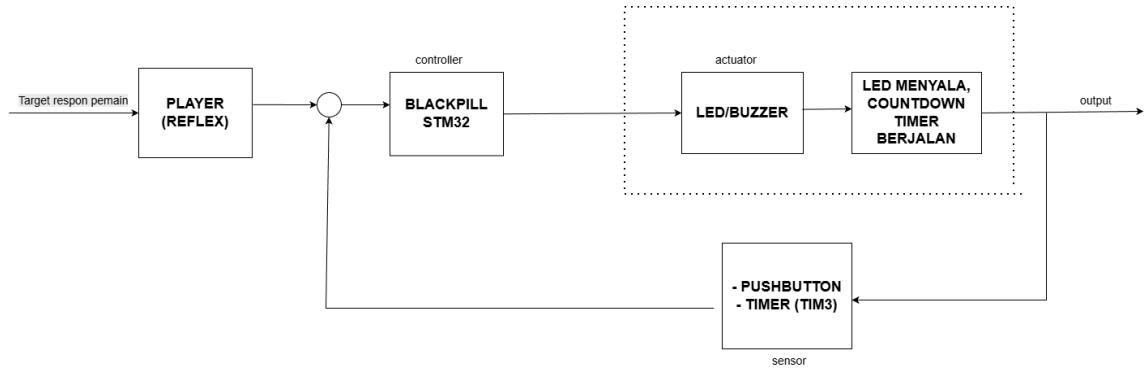
3.3.2 Software Specifications

- Development Tool: VSCode + PlatformIO + CMSIS
- Programming: Bahasa C
- Clock: 16 Mhz (HSI)
- Communication: UART (9600 baud) + I2C (100kHz)
- Timer: 1MHz (1 μ s resolution)

BAB IV: PERANCANGAN SISTEM

4.1 Arsitektur Sistem

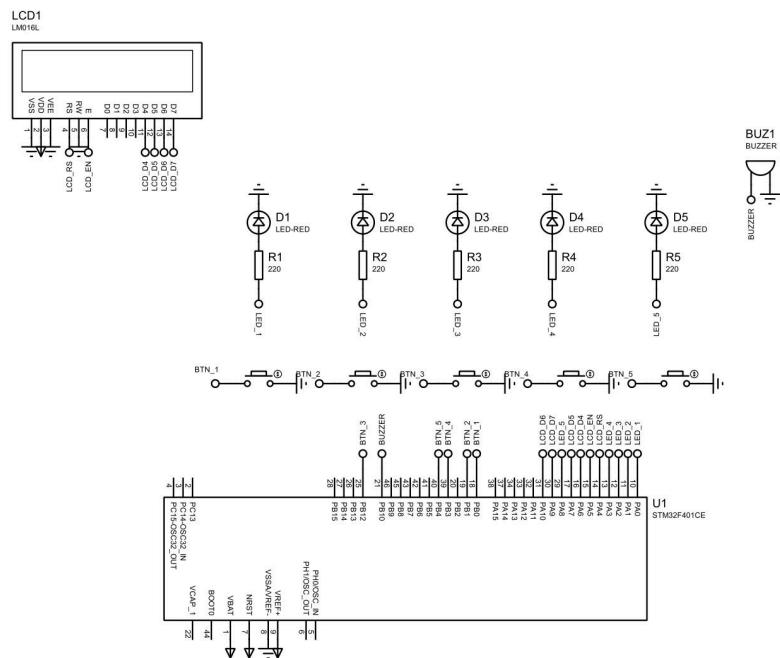
Block Diagram:



Gambar 4.1 Block Diagram

4.2 Desain Hardware

4.2.1 Schematic Diagram



Gambar 4.2 Schematic Diagram

4.2.2 Bill of Materials (BOM)

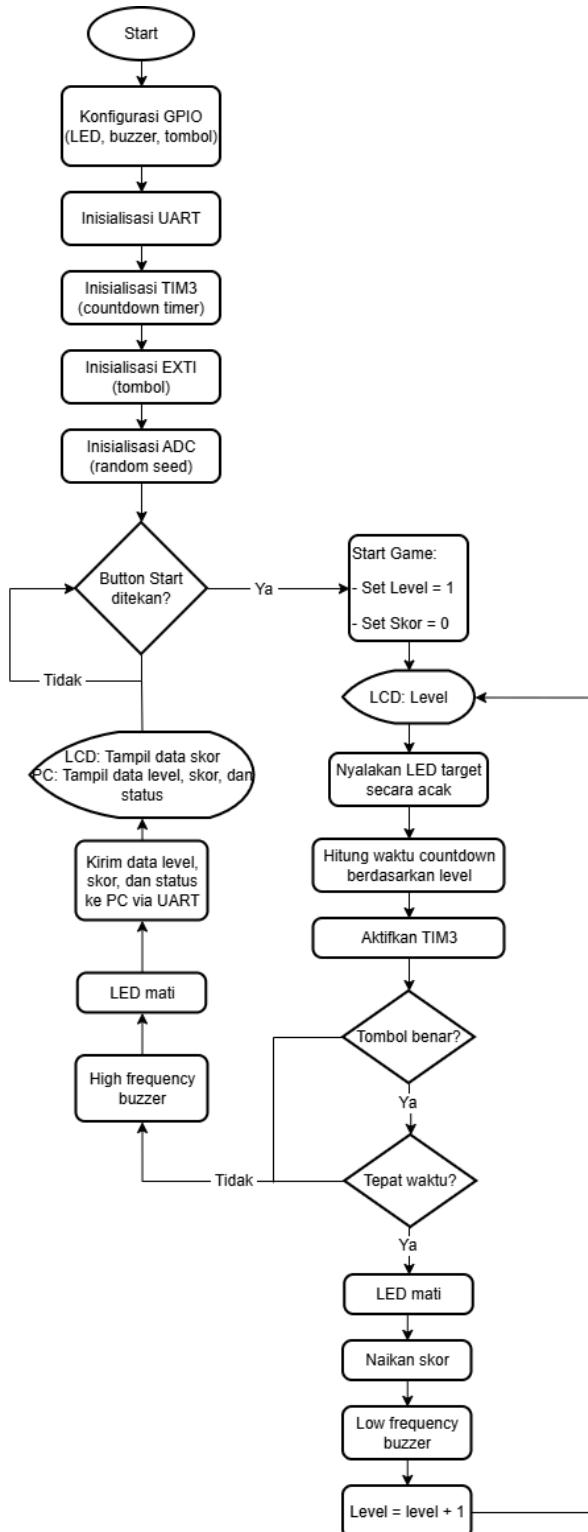
Tabel 4.1 Bill of Materials (BOM)

No	Komponen	Nilai/Tipe	Qty	Harga(Rp)	Total
1	LED	5mm Merah	1	500	500
2	LED	5mm Hijau Doff	1	500	500
3	LED	5mm Kuning	1	500	500
4	LED	5mm Biru Doff	1	500	500
5	LED	5mm Orange	1	500	500
6	Resistor	1/4W	5	200	1.000
7	Kabel Jumper	Male-Female	10	500	5.000
8	Kabel Jumper	Male-Male	10	500	5.000
9	Kabel Jumper	Female-Female	10	500	5.000
10	LCD	Character 2x16 Green + I2C Serial	1	26.000	26.000

No	Komponen	Nilai/Tipe	Qty	Harga(Rp)	Total
11	Buzzer Passive	Kecil	1	3.500	3.500
12	Breadboard	MB102	1	15.000	15.000
13	Breadboard	Sedang	1	12.500	12.500
14	Pushbutton + Knop	B3F Kotak Besar	6	3.500	21.000
15	Blackpill STM32	STM32F411CE U6	1	85.000	85.000
16	Potentiometer	10K Variable	1	5.000	5.000
17	USB to TTL	PL2302	1	10.000	10.000
-	Total Harga		52	196.500	

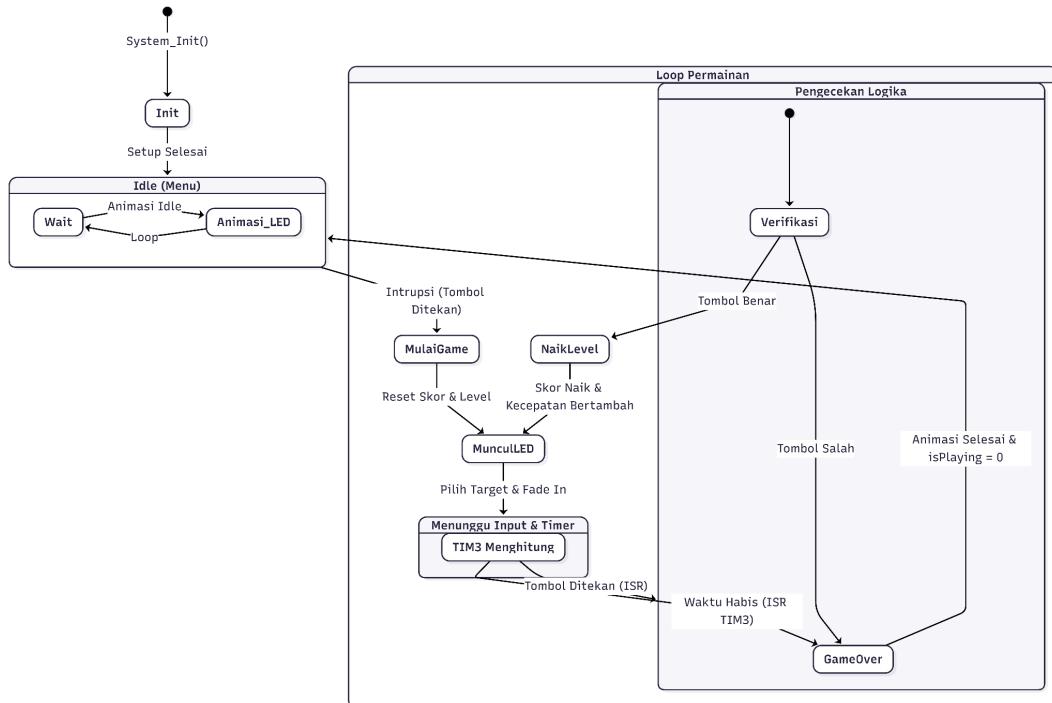
4.3 Desain Software

4.3.1 Flowchart Sistem



Gambar 4.3 Flowchart Sistem

4.3.2 State Machine Diagram



Gambar 4.4 State Machine Diagram

4.4 Desain Subsistem

4.4.1 Subsistem Input (EXTI & GPIO)

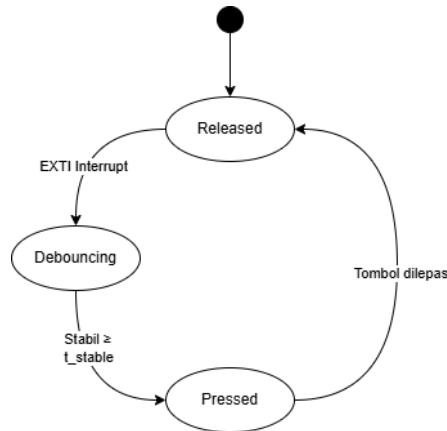
Design Debouncing:

Debouncing tombol diimplementasikan menggunakan metode time-based debounce dengan memanfaatkan timer SysTick beresolusi 1 ms. Setiap interrupt tombol hanya akan diproses apabila selisih waktu dengan penekanan sebelumnya melebihi 200 ms, sehingga efek bouncing dapat diabaikan.

State Machine:

States: [Released, Debouncing, Pressed]

Transitions:



Gambar 4.5 FSM Subsistem Input

4.4.2 Subsistem Timing (Timer Dynamics)

Konfigurasi Timer Counter:

- Clock source: APB1 $\times 1 = 16\text{MHz}$
- Prescaler: PSC = 16 - 1 = 15
- $f_{\text{timer}} = \frac{16\text{MHz}}{6} = 1\text{MHz}$
- Resolution: $1\mu\text{s}$ per count

Auto-Reload Register (ARR):

Untuk interrupt setiap 1ms:

$$\text{ARR} = (1\text{ms} \times 1\text{MHz}) - 1 = 999$$

4.4.3 Subsistem Display (UART Communication)

UART Clock Calculation:

$$\begin{aligned}
 \text{USARTDIV} &= \frac{f_{\text{APB2}}}{16 \times \text{BaudRate}} \\
 &= \frac{50 \times 10^6}{16 \times 9600} \\
 &= 325.52
 \end{aligned}$$

Dengan:

- $f_{APB2} = 50 \text{ MHz}$
- $\text{BaudRate} = 9600$

$$\begin{aligned}
 BRR &= \text{Mantissa} \ll 4 + \text{Fraction} \\
 &= (325 \ll 4) + 8 \\
 &= 0x1458
 \end{aligned}$$

Dengan:

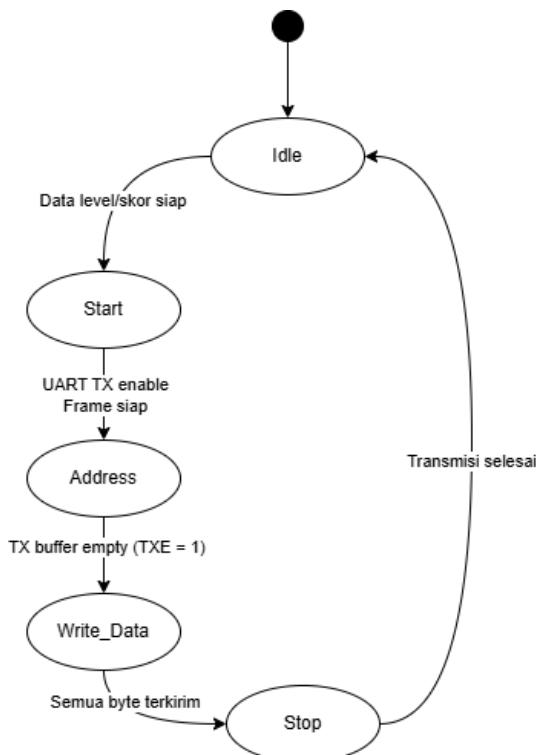
$$\text{Oversampling} = 16 \times$$

- $\text{Mantisa} = 325$
 - $\text{Fraction} = 0.52 \times \text{Oversampling}$
- $$\begin{aligned}
 &= 0.52 \times 16 \\
 &= 8
 \end{aligned}$$

State Machine:

States: [Idle, Start, Address, Write_Data, Stop]

Transitions:



Gambar 4.6 FSM Subsistem Display (UART Communication)

4.4.4 Subsistem Display (I2C Communication)

Kecepatan komunikasi I2C ditentukan oleh frekuensi clock pada bus APB1 dan nilai Clock Control Register (CCR). Berdasarkan spesifikasi sistem, frekuensi APB1 yang digunakan adalah 16 MHz dan frekuensi I2C yang diinginkan adalah 100 kHz (Standard Mode). Perhitungan nilai CCR dilakukan menggunakan persamaan berikut

I2C Clock Calculation:

$$CCR = \frac{f_{APB1}}{2 \times f_{I2C}}$$

Dengan:

$$f_{APB1} = 16\text{MHz} = 16.000.000 \text{ Hz}$$

$$f_{I2C} = 100\text{kHz} = 100.000 \text{ Hz}$$

$$CCR = \frac{16.000.000}{2 \times 100.000}$$

$$CCR = \frac{16.000.000}{2 \times 100.000}$$

$$CCR = 80$$

State Machine:

States: [Idle, Start, Address, Write_Data, Stop]

Transitions:

- Idle → Start: Transisi terjadi ketika fungsi pengiriman dipanggil

- Start → Address: Setelah *Start bit* terkirim (terdeteksi di register status), mikrokontroler mengirimkan alamat I2C LCD

- Address → Write_Data: Jika LCD memberikan sinyal ACK (tanda siap menerima), state berpindah untuk mulai mengirimkan *byte* data
- Write_Data (Loop) → Data dikirim per *byte*. Program akan menunggu hingga register data kosong (TXE) sebelum mengirim *byte* berikutnya.
- Write_Data → Stop: Setelah semua data selesai dikirim, mikrokontroler mengaktifkan bit STOP untuk mengakhiri komunikasi dan mengembalikan bus ke kondisi Idle.

4.4.5 Subsistem ADC

Konversi ADC:

$$\begin{aligned}
 V_{digital} &= \frac{ADC_{value}}{4096} \times V_{ref} \\
 &= \frac{ADC_{value}}{4096} \times 3.3V
 \end{aligned}$$

Dengan:

- $ADC_{value} = 0 - 4095 V$
- $V_{ref} = 3.3 V$

Sampling Time:

$$\begin{aligned}
 t_{sample} &= \frac{(SMP + 12.5)}{f_{ADC}} \\
 &= \frac{(56 + 12.5)}{12.5 \times 10^6} \\
 &= 5.48 \mu s
 \end{aligned}$$

Dengan:

$$SYSCLK = 100 MHz$$

$$APB2 = 50 MHz$$

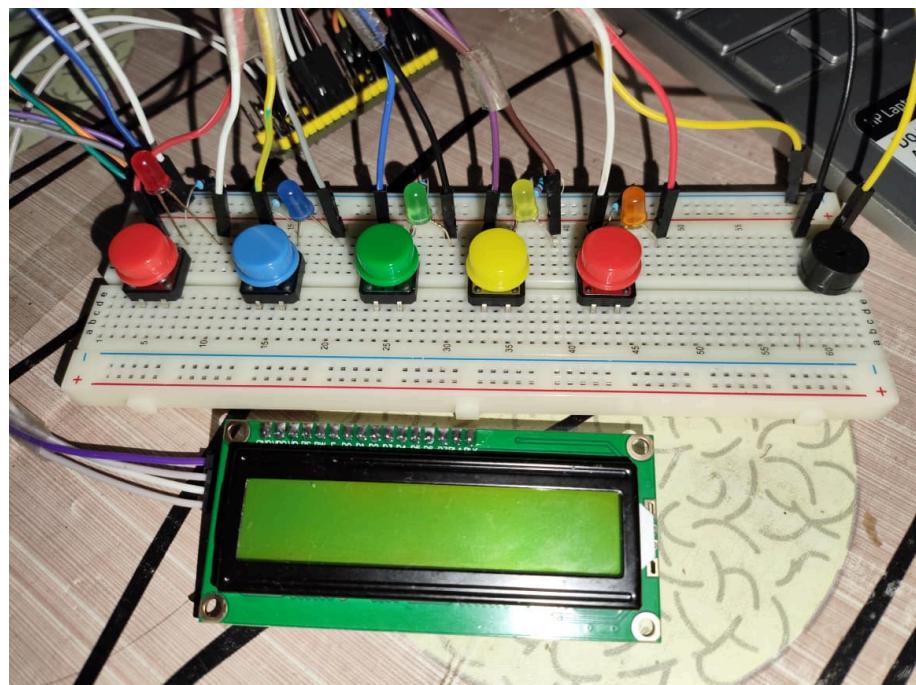
$$ADC prescaler = 4$$

$$\bullet \quad f_{ADC} = \frac{APB2}{ADC prescaler} = \frac{50 MHz}{4} = 12.5 MHz$$

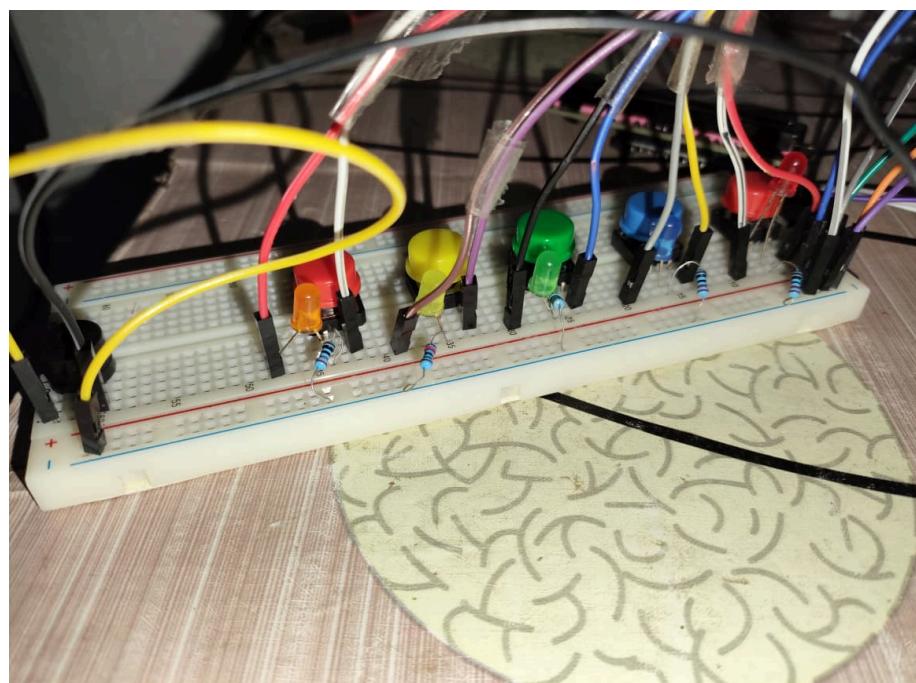
- *SMP = 56 cycles (konfigurasi)*

BAB V: IMPLEMENTASI

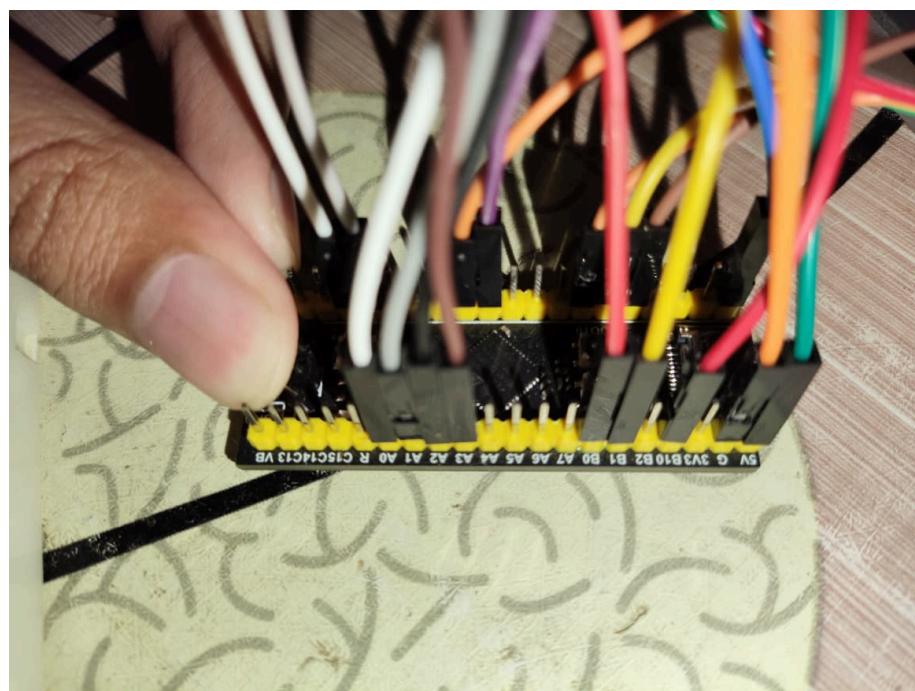
5.1 Implementasi Hardware



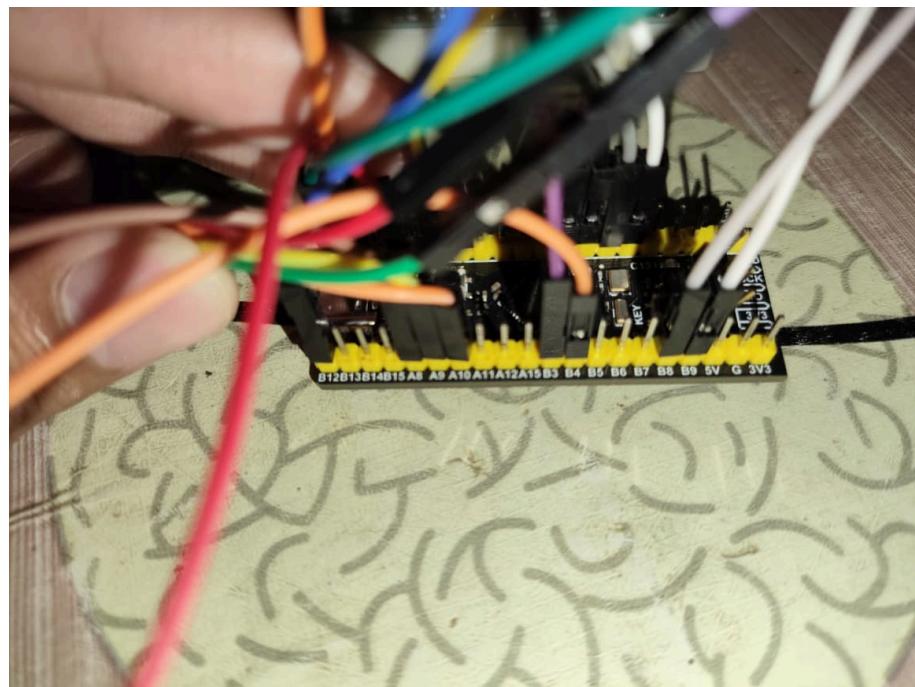
Gambar 5.1 Rakitan Tampak Depan



Gambar 5.2 Rakitan Tampak Belakang



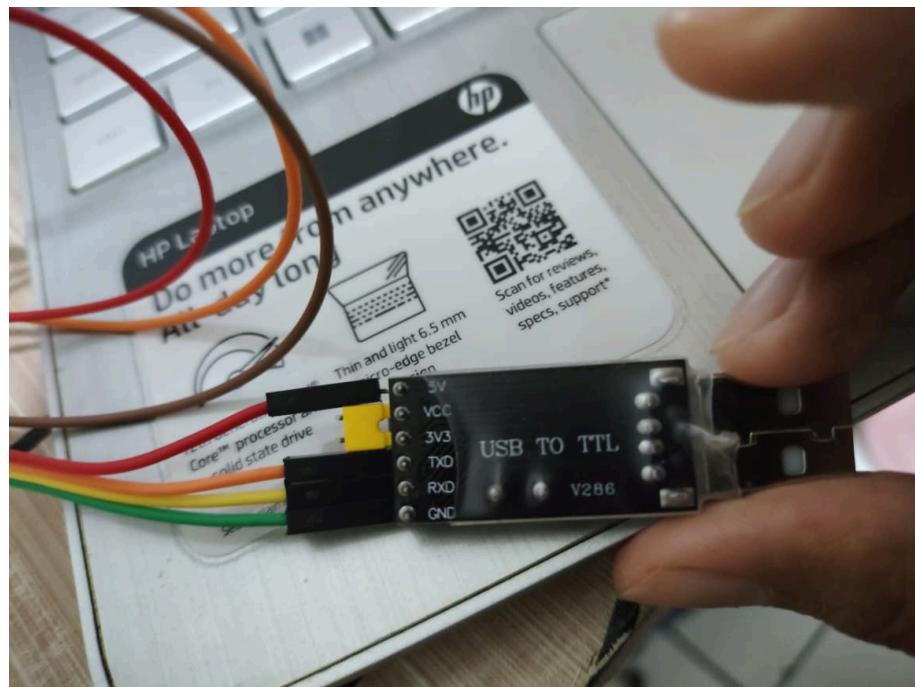
Gambar 5.3 Koneksi Pin Blackpill Tampak Depan



Gambar 5.4 Koneksi Pin Blackpill Tampak Belakang



Gambar 5.5 Koneksi Pin LCD I2C



Gambar 5.6 Koneksi Pin USB TTL

5.2 Implementasi Software (Register-Based)

5.2.1 RCC Configuration

Kode:

```

void System_Init(void) {
    SystemCoreClockUpdate();
    SysTick_Config(SystemCoreClock / 1000); // Konfigurasi SysTick 1ms

    // Enable Clock untuk GPIO A dan B (Bus AHB1)
    RCC->AHB1ENR |= RCC_AHB1ENR_GPIOAEN |
    RCC_AHB1ENR_GPIOBEN;

    // Enable Clock untuk Timer 2 dan 3 (Bus APB1)
    RCC->APB1ENR |= RCC_APB1ENR_TIM2EN | RCC_APB1ENR_TIM3EN;

    // Enable Clock untuk Timer 1, USART1, ADC1, dan SYSCFG (Bus APB2)
    RCC->APB2ENR |= RCC_APB2ENR_TIM1EN |
    RCC_APB2ENR_USART1EN |
    RCC_APB2ENR_ADC1EN | RCC_APB2ENR_SYSCFGEN;
}

}

```

Penjelasan Register:

Tabel 5.1 RCC Configuration

Register	Bit	Value	Fungsi
RCC->AHB1ENR	GPIOAEN	1	Mengaktifkan clock untuk Port A (LED, UART TX).
RCC->APB1ENR	GPIOBEN	1	Mengaktifkan clock untuk Port B (Tombol, LCD, Buzzer).
RCC->APB1ENR	TIM2EN	1	Mengaktifkan <i>clock</i> untuk Timer 2 (PWM LED).
RCC->APB1ENR	TIM3EN	1	Mengaktifkan clock untuk Timer 3 (Game Timer).
RCC->APB2ENR	USART1EN	1	Mengaktifkan clock untuk modul komunikasi Serial USART1

Register	Bit	Value	Fungsi
RCC->APB2ENR	ADC1EN	1	Mengaktifkan <i>clock</i> untuk <i>Analog-to-Digital Converter 1</i> .
RCC->APB2ENR	SYSCFGGEN	1	Mengaktifkan System Configuration Controller untuk keperluan External Interrupt (EXTI)

5.2.2 GPIO Configuration

Kode:

```

void PWM_Init(void) {
    // Reset mode pin PA0-PA3, PA8 lalu set ke Alternate Function (10)
    GPIOA->MODER &= ~((3<<(0*2)) | (3<<(1*2)) | (3<<(2*2)) | (3<<(3*2)) |
(3<<(8*2)));
    GPIOA->MODER |= ((2<<(0*2)) | (2<<(1*2)) | (2<<(2*2)) | (2<<(3*2)) |
(2<<(8*2)));

    // Mapping Pin ke Timer: PA0-PA3 -> TIM2 (AF1), PA8 -> TIM1 (AF1)
    GPIOA->AFR[0] |= (1<<0) | (1<<4) | (1<<8) | (1<<12);
    GPIOA->AFR[1] |= (1<<0);

    // Konfigurasi Timer 2 (PWM Mode)
    TIM2->PSC = 83; TIM2->ARR = 255;
    TIM2->CCMR1 = (6<<4) | (6<<12); // PWM Mode 1 pada CH1 dan CH2
    TIM2->CCMR2 = (6<<4) | (6<<12); // PWM Mode 1 pada CH3 dan CH4
    TIM2->CCER = TIM_CCER_CC1E | TIM_CCER_CC2E | TIM_CCER_CC3E
| TIM_CCER_CC4E;
    TIM2->CR1 |= TIM_CR1_CEN;
}

```

Penjelasan Register:

Tabel 5.2 GPIO Configuration

Register	Bit	Value	Fungsi
GPIOx->MODER	MODERy	10 (Binary)	Mengubah mode pin PA0-PA3 dan PA8 menjadi Alternate Function.
GPIOx->AFR	AFRLy	0001	Menghubungkan pin fisik ke periferal internal TIM1 dan TIM2 (AF1).
TIMx->PSC	-	83	Prescaler. Membagi clock 84MHz menjadi 1MHz.
TIMx->ARR	-	255	Auto-Reload Register. Menentukan resolusi PWM (0-255).
TIMx->CCMRx	OCxM	110 (Bin)	Mengatur mode PWM 1 (Output High saat CNT < CCR).
TIMx->CCER	CCxE	1	Mengaktifkan output sinyal PWM ke pin fisik.

5.2.3 UART Configuration

Kode:

```
void UART_Init(void) {
    // Set PA9 sebagai Alternate Function (TX) -> AF7 (USART1)
    GPIOA->MODER |= (2<<(9*2));
    GPIOA->AFR[1] |= (7<<(1*4));
```

```

// Hitung Baudrate 9600
USART1->BRR = SystemCoreClock / 9600;

// Enable Transmitter (TE) dan UART Enable (UE)
USART1->CR1 |= USART_CR1_TE | USART_CR1_UE;
}

```

Penjelasan Register:

Tabel 5.3 UART Configuration

Register	Bit	Value	Fungsi
GPIOA->AFR[1]	AFRH9	0111	Menghubungkan pin PA9 ke periferal USART1 (AF7).
USART1->BRR	DIV	<i>Calc</i>	Menyimpan nilai pembagi <i>clock</i> untuk menghasilkan 9600 bps.
USART1->CR1	TE	1	<i>Transmitter Enable.</i> Mengizinkan pengiriman data.
USART1->CR1	UE	1	<i>USART Enable.</i> Mengaktifkan modul UART secara keseluruhan.

5.2.4 I2C Configuration

Kode:

```

void I2C_Init_Soft(void) {
    // Set PB8, PB9 sebagai Output
}

```

```

GPIOB->MODER |= ((1<<(8*2)) | (1<<(9*2)));
// Set Output Type ke Open-Drain (Wajib untuk I2C)
GPIOB->OTYPER |= ((1<<8) | (1<<9));
// Set Speed ke High Speed
GPIOB->OSPEEDR |= ((3<<(8*2)) | (3<<(9*2)));
// Set Pull-up Internal
GPIOB->PUPDR |= ((1<<(8*2)) | (1<<(9*2)));
// Set kondisi awal High (Idle)
GPIOB->BSRR = (1<<8) | (1<<9);
}

```

Penjelasan Register:

Tabel 5.4 UART Configuration

Register	Bit	Value	Fungsi
GPIOB->MODER	MODER8/9	01 (Bin)	Mengatur pin PB8 (SCL) dan PB9 (SDA) sebagai output digital.
GPIOB->OTYPER	OT8/9	1	Mengatur tipe output menjadi <i>Open-Drain</i> (Wajib untuk I2C).
GPIOB->PUPDR	PUPDR8/9	01 (Bin)	Mengaktifkan resistor <i>pull-up</i> internal.
GPIOB->BSRR	BS8/9	1	Mengatur pin ke logika 1 (High) secara atomik.

5.2.5 Timer Configuration

Kode:

```

void TIM3_Init(void) {
    // Prescaler agar timer berjalan di 1 kHz (1ms per tick)
    TIM3->PSC = (SystemCoreClock / 1000) - 1;
    TIM3->ARR = 2000; // Nilai awal (akan diubah saat game mulai)
}

```

```

        TIM3->DIER |= TIM_DIER_UIE; // Enable Update Interrupt
        NVIC_EnableIRQ(TIM3_IRQn); // Enable di NVIC
    }
}

```

Penjelasan Register:

Tabel 5.5 Timer Configuration

Register	Bit	Value	Fungsi
TIM3->PSC	-	15999	Membagi frekuensi sistem agar <i>counter</i> bertambah setiap 1 ms.
TIM3->ARR	-	2000	<i>Auto-Reload</i> . Menentukan batas hitungan maksimum sebelum <i>overflow</i> .
TIM3->DIER	UIE	1	<i>Update Interrupt Enable</i> . Mengaktifkan interupsi saat timer <i>overflow</i> .

5.2.6 External Interrupt Configuration

Kode:

```

void EXTI_Init_Custom(void) {
    // Konfigurasi Pull-up untuk tombol PB0, PB1, PB3, PB4, PB12
    GPIOB->PUPDR |= ((1<<(0*2)) | (1<<(1*2)) | (1<<(3*2)) | (1<<(4*2)) |
(1<<(12*2)));
}

// Mapping EXTI Line ke Port B

```

```

SYSCFG->EXTICR[0] |= SYSCFG_EXTICR1_EXTI0_PB |
SYSCFG_EXTICR1_EXTI1_PB | SYSCFG_EXTICR1_EXTI3_PB;
SYSCFG->EXTICR[1] |= SYSCFG_EXTICR2_EXTI4_PB;
SYSCFG->EXTICR[3] |= SYSCFG_EXTICR4_EXTI12_PB;

// Trigger pada Falling Edge dan Unmask Interrupt
EXTI->FTSR |= (1<<0)|(1<<1)|(1<<3)|(1<<4)|(1<<12);
EXTI->IMR |= (1<<0)|(1<<1)|(1<<3)|(1<<4)|(1<<12);

// Enable di NVIC
NVIC_EnableIRQ(EXTI0_IRQn); NVIC_EnableIRQ(EXTI1_IRQn); // dst...
}

```

Penjelasan Register:

Tabel 5.6 EXTI Configuration

Register	Bit	Value	Fungsi
SYSCFG->EXT ICR	EXTIx	0001	Memilih Port B sebagai sumber sinyal untuk jalur EXTI.
EXTI->FTSR	TRx	1	<i>Falling Trigger.</i> Interupsi dipicu saat sinyal turun (tombol ditekan).
EXTI->IMR	MRx	1	<i>Interrupt Mask.</i> Membuka "masker" agar sinyal diteruskan ke CPU.
NVIC->ISER	-	1	Mengaktifkan vektor interupsi

			spesifik pada core Cortex-M4.
--	--	--	-------------------------------

5.2.7 Interrupt Service Routines

Kode:

```
// Handler untuk Tombol 1 (PB0)
void EXTI0_IRQHandler(void) {
    if(EXTI->PR & 1) {      // Cek apakah interrupt berasal dari Line 0
        Button_Handler(0); // Jalankan logika tombol
        EXTI->PR = 1;      // Clear Pending Flag (Wajib)
    }
}

// Handler untuk Timer 3 (Game Timer)
void TIM3_IRQHandler(void) {
    if(TIM3->SR & 1) {      // Cek Update Interrupt Flag
        TIM3->SR = 0;      // Clear Flag
        flag_timeout = 1;   // Set status timeout
        TIM3_Stop();        // Hentikan timer
    }
}
```

Analisis ISR:

- Execution time: Sangat singkat (< 5 μs)
- Priority: Menggunakan prioritas default hardware
- Critical section: Pembersihan Pending Flag (EXTI->PR atau TIM3->SR)

5.3 Pembagian Tugas Kelompok

Tabel 5.7 Pembagian Tugas Kelompok

Anggota	Tugas	Status
Ali Nafis Hamdi	Hardware design & wiring	✓

Anggota	Tugas	Status
Monica Febiana	UART & I2C Implementation	✓
Cheyza Amanda Syafira	Timer & Interrupt	✓
Monica Febiana	Integration & testing	✓
Cheyza Amanda Syafira	Documentation	✓

BAB VI: PENGUJIAN DAN ANALISIS

6.1 Metodologi Pengujian

1. Unit Testing: Tahap ini bertujuan memvalidasi inisialisasi register CMSIS dan fungsi dasar setiap periferal secara terpisah sebelum diintegrasikan ke dalam logika permainan utama.

Tabel 6.1 Tabel Unit Testing

ID Uji	Item Pengujian	Referensi	Parameter Keberhasilan (Success Criteria)
UT-01	Input Tombol & <i>Debouncing</i> (EXTI)	FR-01	Saat tombol ditekan satu kali secara fisik, Serial Monitor hanya menampilkan satu baris log data (tidak terjadi <i>double-trigger</i> atau <i>spamming</i> data).
UT-02	Pembangkitan <i>Seed</i> Acak (ADC)	FR-02	Nilai register ADC yang dicetak ke Serial Monitor menunjukkan angka yang fluktuatif (berubah-ubah/tidak konstan) setiap kali pembacaan dilakukan.
UT-03	Kendali Kecerahan LED (PWM)	FR-02	Intensitas cahaya LED dapat berubah dari redup ke terang secara bertahap (halus) tanpa terlihat adanya kedipan kasar (<i>flickering</i>) oleh mata.

UT-04	Tampilan Karakter LCD (I2C)	FR-06	Teks yang dikirimkan mikrokontroler tampil pada baris dan kolom yang tepat di LCD 16x2 tanpa adanya karakter sampah (<i>glitch</i>).
UT-05	Respons Audio (<i>Buzzer</i>)	FR-03, FR-04	<i>Buzzer</i> dapat menghasilkan dua nada berbeda (nada pendek untuk "Benar" dan nada panjang untuk "Salah") yang terdengar jelas.

2. Integration Testing: Tahap ini menguji interaksi antar-modul untuk memastikan sinkronisasi antara *timer*, logika permainan, dan antarmuka pengguna berjalan sesuai alur desain.

Tabel 6.2 Tabel IntegrationTesting

ID Uji	Item Pengujian	Referensi	Parameter Keberhasilan (Success Criteria)
IT-01	Akurasi <i>Countdown Timer</i>	FR-03, FR-04	Durasi hitung mundur pada LCD sesuai dengan waktu nyata yang diukur menggunakan <i>stopwatch</i> (toleransi keterlambatan visual < 0.5 detik).
IT-02	Logika Perhitungan Skor	FR-03	Skor pada variabel sistem bertambah tepat 10 poin setiap kali tombol yang benar ditekan, diverifikasi melalui tampilan LCD dan log UART.

IT-03	Mekanisme <i>Timeout</i> Otomatis	FR-04	Permainan otomatis berhenti dan masuk ke status "Game Over" tepat saat waktu habis, meskipun tombol tidak ditekan sama sekali.
IT-04	Sinkronisasi Level & Kesulitan	FR-02	Saat skor mencapai ambang batas kenaikan level, sistem otomatis mempercepat durasi waktu atau mengubah kecerahan LED pada ronde berikutnya.

3. System Testing: Pengujian akhir dilakukan dengan menjalankan skenario penggunaan penuh (*End-to-End*) untuk memvalidasi pemenuhan seluruh *Functional Requirements*.

Tabel 6.3 Tabel System Testing

ID Uji	Skenario Pengujian	Referensi	Parameter Keberhasilan (Success Criteria)
ST-01	Inisialisasi Sistem (<i>Startup</i>)	FR-01	Saat sistem dinyalakan atau di-reset, LCD menampilkan pesan pembuka dan menunggu input tombol Start tanpa <i>error</i> .
ST-02	Pola Permainan Acak	FR-02	Dalam 10 kali percobaan berturut-turut, urutan LED yang menyala sebagai target selalu berbeda (tidak terpola statis).

ST-03	Respons Kondisi Menang	FR-03	Ketika tombol yang sesuai dengan LED ditekan: LED mati, skor bertambah, dan bunyi <i>beep</i> pendek terdengar secara instan.
ST-04	Respons Kondisi Kalah/Salah	FR-04	Ketika tombol yang salah ditekan: permainan langsung berhenti, bunyi nada panjang terdengar, dan skor akhir terkunci.
ST-05	Telemetri Data ke PC	FR-05	Serial Monitor pada PC menerima rekap data (Level Akhir, Skor Total, Waktu Reaksi) sesaat setelah permainan berakhir (<i>Game Over</i>).
ST-06	Stabilitas Sistem (<i>Stress Test</i>)	FR-01 s/d FR-06	Sistem tidak mengalami <i>hang</i> atau <i>freeze</i> (macet) saat dimainkan berulang kali atau saat tombol ditekan secara acak dengan cepat.

4. Tools:

- a. Visual Studio Code dengan Ekstensi PlatformIO. Fitur Serial Monitor bawaan PlatformIO digunakan untuk memantau status *upload firmware* dan memastikan tidak ada *error* pada saat inisialisasi sistem.
- b. Modul USB to TTL (UART Bridge) untuk membuktikan FR-05 (Telemetri Data), yaitu mengirimkan data skor, level, dan waktu reaksi dari *hardware* ke PC untuk dianalisis.

- c. Serial Monitor Console (Terminal) untuk memvalidasi FR-01 dan FR-02 dengan cara menampilkan nilai variabel internal (seperti nilai register ADC dan status tombol) secara *real-time* yang tidak bisa dilihat secara visual pada fisik mikrokontroler.
- d. Stopwatch Digital untuk mengukur akurasi *timer* permainan (apakah *game over* tepat di 2 detik) dan membandingkannya dengan logika program.

6.2 Pengujian Sistem Keseluruhan

Skenario Pengujian:

1. Hubungkan mikrokontroler ke PC menggunakan kabel USB dan buka fitur Serial Monitor pada VSCode (PlatformIO).
2. Tekan tombol Reset pada *board*.
3. Pastikan LCD menampilkan teks pembuka "Tel-U Game", tidak ada karakter *glitch*, dan Serial Monitor menerima pesan "==== SYSTEM READY ===".
4. Tekan salah satu tombol input untuk memulai permainan.
5. Mainkan permainan sebanyak 5 putaran (ronde) dengan menekan tombol yang benar sesuai LED yang menyala.
6. Pastikan:
 - LED target berpindah secara acak (tidak berpola statis).
 - Setiap kali tombol yang benar ditekan, *buzzer* harus berbunyi nada pendek (*beep*) dan skor pada LCD bertambah 10 poin.
7. Saat permainan memasuki ronde ke-6, lakukan penekanan tombol secara acak dan cepat (*spamming*) selama 1-2 detik pada tombol yang tidak aktif.
8. Pastikan sistem tetap berjalan stabil, tidak mengalami *freeze* (macet), dan tidak me-reset diri sendiri.
9. Pada ronde berikutnya, biarkan permainan berjalan tanpa menekan tombol apapun hingga batas waktu habis (*timeout*), atau tekan tombol yang salah dengan sengaja.

10. Pastikan sistem harus merespons dengan bunyi *buzzer* nada panjang, mematikan seluruh LED permainan, dan LCD menampilkan teks "GAME OVER" beserta skor akhir.
11. Segera setelah status *Game Over* muncul, alihkan perhatian ke layar PC.
12. Periksa Serial Monitor untuk memastikan data statistik permainan (Skor Akhir, Level, dan Waktu Reaksi) telah diterima dan sesuai dengan yang ditampilkan pada LCD.

Hasil:

Tabel 6.4 Hasil Pengujian Sistem

ID	Skenario / Fokus Uji	Hasil Pengamatan (Observed Result)	Kesimpulan
ST-01	Inisialisasi Sistem (<i>Startup</i>)	Setelah tombol Reset ditekan, LCD menyala menampilkan teks "Tel-U Game" dan "Press Any Btn". Tidak ada karakter aneh pada LCD. Serial Monitor menampilkan log "==== SYSTEM READY ===".	Berhasil
ST-02	Pola Permainan Acak	Dalam 5 ronde permainan awal, LED target berpindah posisi secara dinamis dan tidak membentuk pola berulang yang mudah ditebak.	Berhasil
ST-03	Respons Kondisi Menang	Saat tombol yang benar ditekan, <i>buzzer</i> berbunyi 'bip' pendek ($\pm 100\text{ms}$), skor di LCD bertambah +10, dan LED target langsung mati.	Berhasil

ST-04	Respons Kondisi Kalah	Saat tombol salah ditekan, permainan berhenti instan, <i>buzzer</i> berbunyi nada panjang, dan LCD menampilkan "GAME OVER" beserta skor akhir. Hal sama terjadi saat didiamkan >2 detik (<i>timeout</i>).	Berhasil
ST-05	Telemetri Data ke PC	Serial Monitor menampilkan log data yang akurat pasca permainan. Contoh: '[RESULT] TIMEOUT	Berhasil
ST-06	Stabilitas Sistem (<i>Stress Test</i>)	Tombol ditekan acak dengan cepat tidak menyebabkan mikrokontroler macet. Input tombol saat inisialisasi diabaikan sistem (tidak <i>error</i>) hingga pesan "Ready" muncul.	Berhasil

Dokumentasi:

 VIDEO DEMONSTRASI TUBES_Kelompok 2_TK 47-03.mp4

6.3 Troubleshooting

Tabel 6.4 Troubleshooting

Masalah	Penyebab	Solusi
Button saat diklik responnya lambat	Debounce manual terlalu lama atau variabel <code>last_debounce_time</code> belum tepat	Sesuaikan delay <code>debounce</code> di <code>Button_Handler();</code> di kode sekarang <code>thresholdnya 200 ms</code> , bisa dikurangi untuk respon lebih cepat

Game over terjadi sebelum bermain	Variabel kontrol seperti flag_button_pressed, dan input_allowed belum berada pada state awal	Reset semua flag dan LED saat start: di kode sekarang sudah dilakukan flag_button_pressed=-1; input_allowed=1; sebelum permainan dimulai
Buzzer berbunyi meski belum diaktifkan	Pin PB10 tidak diinisialisasi low secara eksplisit sebelum digunakan, sehingga bisa floating	Pastikan pin GPIOB10 di-reset ke 0 saat init; di kode sekarang sudah ada GPIOB->ODR &= ~(1<<10) sebelum dan sesudah play_tone
LCD paralel tanpa I2C menampilkan karakter aneh	Tegangan MCU (STM32 3.3V) tidak kompatibel dengan LCD 5V; timing paralel manual terlalu sensitif	Tambahkan modul I2C (backpack PCF8574) yang mengurus sinyal paralel; sambungkan SDA/SCL ke STM32
Komunikasi UART tidak terkirim ke PC saat menggunakan USB Type-C	USB Type-C biasa tidak langsung menyediakan level TTL yang sesuai untuk UART, sehingga data serial tidak terbaca	Gunakan USB to TTL converter agar level tegangan UART sesuai dengan PC

BAB VII: KESIMPULAN DAN SARAN

7.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian sistem permainan *Reflex Master* berbasis mikrokontroler Blackpill STM32F411, dapat disimpulkan beberapa hal khususnya pada aspek komunikasi serial, timer dan interrupt, pemrograman tingkat register, serta hasil pengujian sistem.

Pertama, penerapan komunikasi serial pada sistem telah berhasil diimplementasikan menggunakan protokol UART dan I2C. UART digunakan sebagai media komunikasi antara mikrokontroler dan PC untuk mengirimkan informasi skor, level, waktu reaksi, serta status permainan secara real-time melalui serial monitor. Sementara itu, I2C dimanfaatkan untuk mengendalikan LCD 16×2 dalam menampilkan informasi permainan secara lokal. Hasil pengujian menunjukkan bahwa kedua protokol komunikasi tersebut dapat bekerja secara stabil, sinkron, dan sesuai dengan spesifikasi kecepatan yang ditetapkan, sehingga mendukung kebutuhan monitoring dan tampilan data pada sistem permainan.

Kedua, konfigurasi timer dan interrupt berhasil diterapkan untuk mendukung sistem yang bersifat real-time dan responsif. Timer TIM3 digunakan sebagai countdown timer untuk membatasi waktu respons pemain pada setiap level, sedangkan interrupt timer dimanfaatkan untuk mendeteksi kondisi *timeout* secara presisi. Selain itu, penggunaan external interrupt (EXTI) pada tombol input memungkinkan sistem merespons penekanan tombol secara cepat tanpa bergantung pada metode polling. Implementasi ini terbukti meningkatkan ketepatan pengukuran waktu reaksi dan menjaga alur permainan tetap sinkron.

Ketiga, penerapan pemrograman tingkat register (register-level programming) menggunakan CMSIS memberikan pemahaman yang mendalam mengenai cara kerja internal mikrokontroler STM32F411. Seluruh peripheral utama seperti GPIO, USART, I2C, timer, EXTI, ADC, dan NVIC dikonfigurasi secara langsung melalui register, tanpa menggunakan library tingkat tinggi seperti HAL atau CubeMX. Pendekatan ini memungkinkan mahasiswa memahami

hubungan antara konfigurasi register, arsitektur hardware, dan perilaku sistem secara nyata, serta meningkatkan kemampuan analisis dan debugging pada sistem embedded.

Terakhir, hasil pengujian sistem secara keseluruhan menunjukkan bahwa permainan *Reflex Master* dapat berjalan sesuai dengan rancangan dan spesifikasi yang telah ditetapkan. Sistem mampu mendeteksi input tombol dengan cepat, menghasilkan stimulus LED dan buzzer secara tepat waktu, menampilkan informasi permainan pada LCD, serta mengirimkan data hasil permainan ke PC melalui UART tanpa error yang signifikan. Dengan demikian, sistem dinyatakan berfungsi dengan baik, stabil, dan telah memenuhi tujuan pembelajaran serta indikator pencapaian CLO 3 pada mata kuliah Mikroprosesor dan Antarmuka.

7.2 Saran

Pengembangan sistem *Reflex Master* ke depan dapat dilakukan dengan menambahkan fitur permainan yang lebih kompleks, seperti variasi level, pola LED yang lebih dinamis, penyimpanan skor permanen, serta integrasi komunikasi nirkabel untuk meningkatkan fleksibilitas dan pengalaman pengguna. Dari sisi pembelajaran, proyek ini disarankan digunakan sebagai media praktikum lanjutan untuk memperdalam pemahaman mahasiswa terhadap komunikasi serial, timer, interrupt, dan pemrograman tingkat register, serta sebagai bahan perbandingan dengan pendekatan library tingkat tinggi seperti HAL atau RTOS.

Selain itu, analisis datasheet dan reference manual secara lebih mendalam perlu terus dilatih agar mahasiswa mampu merancang sistem embedded secara mandiri. Untuk penelitian lanjutan, sistem ini dapat dikembangkan menjadi alat pengukuran waktu reaksi yang lebih presisi melalui optimasi konfigurasi timer dan interrupt, serta diperluas dengan analisis performa sistem real-time atau penerapan algoritma adaptif untuk menyesuaikan tingkat kesulitan permainan secara otomatis.

AI Editing Statement

Saya menyatakan bahwa saya menggunakan alat kecerdasan buatan (AI) hanya untuk keperluan editing, termasuk perbaikan tata bahasa, penyusunan kalimat, konsistensi istilah, dan peningkatan keterbacaan dokumen ini.

Seluruh ide, analisis, perhitungan, desain, dan konten substantif merupakan hasil pekerjaan saya sendiri. AI tidak digunakan untuk menghasilkan konten inti, menyelesaikan tugas teknis, atau membuat analisis secara otomatis.

Sebagai bentuk transparansi, berikut adalah daftar penggunaan AI beserta nama alat AI yang digunakan:

No	Bagian yang Diedit	Jenis Bantuan	Nama AI yang Digunakan
1	Bab II - Landasan teori	Perbaikan tata bahasa, struktur paragraf, dan konsistensi istilah teknik	ChatGPT
2	Bab VII - Kesimpulan dan Saran	Penyusunan kalimat yang digunakan	ChatGPT
3	Kode Program	Troubleshooting kode yang error	Gemini

Nama Mahasiswa:

- Cheyza Amanda Syafira
- Monica Febyana
- Ali Nafis Hamdi

NIM:

- 101032300081
- 101032330068
- 101032330202

Tanggal: 07/01/2026

DAFTAR PUSTAKA

- [1] STMicroelectronics. (2024). STM32F411xx Reference Manual
- [2] ARM. (2024). Cortex-M3 Technical Reference Manual.
- [3] "How to connect a 4 pin tactile switch?," Arduino Forum, Oct. 17, 2019. [Online]. Available:
<https://forum.arduino.cc/t/how-to-connect-a-4-pin-tactile-switch/616033> [Accessed: Dec. 14, 2025].
- [4] "The Basics of Tactile Switches," DigiKey, Feb. 28, 2022. [Online]. Available:
<https://www.digikey.com/en/articles/the-basics-of-tactile-switches> [Accessed: Dec. 14, 2025].
- [5] "Pull-up Resistors," SparkFun Learn. [Online]. Available:
<https://learn.sparkfun.com/tutorials/pull-up-resistors/all> [Accessed: Dec. 14, 2025].
- [6] "Why isn't internal pull-down resistor working on STM32F411CEU6 Black Pill board?," Stack Overflow, Aug. 10, 2021. [Online]. Available:
<https://stackoverflow.com/questions/68719668/why-isnt-internal-pull-down-resistor-working-on-stm32f411ceu6-black-pill-board> [Accessed: Dec. 14, 2025].
- [7] "STM32 Buzzer | Piezo Buzzer Example + Tone [Active & Passive]," DeepBlue Embedded, Jun. 5, 2024. [Online]. Available:
<https://deepbluemedded.com/stm32-buzzer-piezo-active-passive-buzzer-example-code-tone/> [Accessed: Dec. 14, 2025].
- [8] "Interface Passive Buzzer with STM32," ControllersTech, Sep. 9, 2025. [Online]. Available:
<https://controllerstech.com/interface-passive-buzzer-with-stm32/> [Accessed: Dec. 14, 2025].

- [9] "Using NPN transistor to make piezo beep louder," Arduino Forum, Apr. 28, 2015. [Online]. Available: <https://forum.arduino.cc/t/using-npn-transistor-to-make-piezo-beep-louder/30829> [Accessed: Dec. 14, 2025].
- [10] "Interfacing a Piezo Buzzer with Stm32 Microcontroller," Engineers Garage, Jun. 5, 2019. [Online]. Available: <https://www.engineersgarage.com/interfacing-buzzer-with-stm32-microcontroller/> [Accessed: Dec. 14, 2025].
- [11] "Playing popular songs with Arduino and a buzzer," Arduino Project Hub, Nov. 18, 2022. [Online]. Available: <https://projecthub.arduino.cc/tmekinyan/playing-popular-songs-with-arduino-and-a-buzzer-546f4a> [Accessed: Dec. 14, 2025].
- [12] "Playing Melodies using Arduino Tone() Function," Circuit Digest, Apr. 13, 2022. [Online]. Available: <https://circuitdigest.com/microcontroller-projects/playing-melodies-on-piezo-buzzer-using-arduino-tone-function> [Accessed: Dec. 14, 2025].
- [13] "Arduino Passive Buzzer Tutorial: Alarm, Siren & Tone," ControllersTech, Dec. 12, 2025. [Online]. Available: <https://controllerstech.com/arduino-passive-buzzer-tutorial/> [Accessed: Dec. 14, 2025].
- [14] "Breadboard Organization," Instructables, May 28, 2022. [Online]. Available: <https://www.instructables.com/Breadboard-Organization/> [Accessed: Dec. 14, 2025].
- [15] "Mastering Breadboard Circuits: A Comprehensive Guide for Beginners," AnyPCBA, Jan. 22, 2025. [Online]. Available: <https://www.anypcba.com/blogs/electronic-component-knowledge/mastering-breadboard-circuits-a-comprehensive-guide-for-beginners> [Accessed: Dec. 14, 2025].

- [16] "How Do You Layout a Breadboard?," RayPCB, Feb. 9, 2025. [Online]. Available: <https://www.raypcb.com/breadboard-layout/> [Accessed: Dec. 14, 2025].
- [17] "Breadboard Power Distribution Wires," Instructables, Sep. 28, 2017. [Online]. Available: <https://www.instructables.com/Breadboard-Power-Distribution-Wires/> [Accessed: Dec. 14, 2025].
- [18] "How to use a Breadboard for Beginners? Wiring, Circuit, Arduino," Seeed Studio, Jan. 5, 2020. [Online]. Available: <https://www.seeedstudio.com/blog/2020/01/06/how-to-use-a-breadboard-wiring-circuit-and-arduino-interfacing/> [Accessed: Dec. 14, 2025].
- [19] "How to Use a Breadboard," SparkFun Learn, Aug. 1, 2001. [Online]. Available: <https://learn.sparkfun.com/tutorials/how-to-use-a-breadboard/all> [Accessed: Dec. 14, 2025].
- [20] "Bagaimana membuat LED Hidup, Redup dan Mati secara Perlahan," Electronics Bot, Jan. 2018. [Online]. Available: <http://electronicsbot.blogspot.com/2018/01/bagaimana-membuat-led-hidup-redup-dan.html> [Accessed: Dec. 14, 2025].
- [21] "Cara Mengontrol LED di Arduino (Uno) | SOLVED," Nashrul, Mar. 14, 2022. [Online]. Available: <https://www.nashrul.com/2022/02/cara-mengontrol-led-di-arduino-uno.html> [Accessed: Dec. 14, 2025].
- [22] "Program Membuat LED Mati Secara Perlahan di Arduino," Belajar IT, Jun. 30, 2022. [Online]. Available: <https://www.belajarit.net/2022/07/program-membuat-led-mati-secara.html> [Accessed: Dec. 14, 2025].

- [23] "stm32duino/Arduino_Core_STM32: STM32 core support for Arduino," GitHub, May 9, 2017. [Online]. Available: https://github.com/stm32duino/Arduino_Core_STM32 [Accessed: Dec. 15, 2025].
- [24] "STM32 HAL Integration | stm32duino/Arduino_Core_STM32," DeepWiki, Apr. 23, 2025. [Online]. Available: https://deepwiki.com/stm32duino/Arduino_Core_STM32/3.2-stm32-hal-integration [Accessed: Dec. 15, 2025].

LAMPIRAN

LAMPIRAN A: Full Source Code

Tabel A.1 Full Source Code

```
#include "stm32f4xx.h"
#include <stdlib.h>
#include <stdio.h>

#define LCD_ADDR_FIXED 0x27
#define NOTE_G4 392
#define NOTE_B4 494
#define NOTE_D5 587
#define NOTE_E5 659
#define NOTE_G5 784
#define NOTE_B5 988
#define NOTE_D6 1175
#define NOTE_E6 1319

// --- GLOBAL VARIABLES ---
volatile int flag_button_pressed = -1;
volatile int flag_timeout = 0;
volatile uint32_t msTicks = 0;
volatile uint32_t last_debounce_time = 0;
volatile int input_allowed = 1;

void SysTick_Handler(void) { msTicks++; }
uint32_t get_millis(void) { return msTicks; }

void delay_audio_math(uint32_t us) {
    volatile uint32_t count = us * (SystemCoreClock / 1000000) /
8;
    while(count--);
}

void delay_real_ms(uint32_t ms) {
    uint32_t start = get_millis();
    while ((get_millis() - start) < ms);
```

```

}

void delay_dumb(int count) {
    volatile int i;
    for (i = 0; i < count; i++);
}

void System_Init(void) {
    SystemCoreClockUpdate();
    SysTick_Config(SystemCoreClock / 1000);

    RCC->AHB1ENR |= RCC_AHB1ENR_GPIOAEN | RCC_AHB1ENR_GPIOBEN;
    RCC->APB1ENR |= RCC_APB1ENR_TIM2EN | RCC_APB1ENR_TIM3EN;
    RCC->APB2ENR |= RCC_APB2ENR_TIM1EN | RCC_APB2ENR_USART1EN |
RCC_APB2ENR_ADC1EN | RCC_APB2ENR_SYSCFGEN;
}

void ADC_Init(void) {
    GPIOA->MODER |= (3 << (7 * 2));

    ADC1->CR1 = 0;
    ADC1->CR2 = ADC_CR2_ADON;

    delay_real_ms(1);

    ADC1->SQR3 = 7;
    ADC1->SMPR2 |= ADC_SMPR2_SMP7;
}

// --- I2C & LCD ---
void I2C_Init_Soft(void) {
    GPIOB->MODER &= ~((3<<(8*2)) | (3<<(9*2)));
    GPIOB->MODER |= ((1<<(8*2)) | (1<<(9*2)));
    GPIOB->OTYPER |= ((1<<8) | (1<<9));
    GPIOB->OSPEEDR |= ((3<<(8*2)) | (3<<(9*2)));
    GPIOB->PUPDR &= ~((3<<(8*2)) | (3<<(9*2)));
    GPIOB->PUPDR |= ((1<<(8*2)) | (1<<(9*2)));
    GPIOB->BSRR = (1<<8) | (1<<9);
}

```

```

void SCL_H(void) { GPIOB->BSRR = (1<<8); }
void SCL_L(void) { GPIOB->BSRR = (1<<24); }
void SDA_H(void) { GPIOB->BSRR = (1<<9); }
void SDA_L(void) { GPIOB->BSRR = (1<<25); }

void I2C_Start(void) { SDA_H(); SCL_H(); delay_dumb(50); SDA_L();
delay_dumb(50); SCL_L(); delay_dumb(50); }
void I2C_Stop(void) { SDA_L(); delay_dumb(50); SCL_H();
delay_dumb(50); SDA_H(); delay_dumb(50); }
void I2C_Write(uint8_t data) {
    for (int i = 0; i < 8; i++) {
        if (data & 0x80) SDA_H(); else SDA_L();
        data <= 1; delay_dumb(50); SCL_H(); delay_dumb(50);
SCL_L(); delay_dumb(50);
    }
    SDA_H(); delay_dumb(50); SCL_H(); delay_dumb(50); SCL_L();
delay_dumb(50);
}

uint8_t BL_STATE = 0x08;

void LCD_Nibble(uint8_t nibble, uint8_t rs) {
    uint8_t data = (nibble & 0xF0) | BL_STATE | (rs ? 1 : 0) | 4;
    I2C_Start(); I2C_Write(LCD_ADDR_FIXED << 1); I2C_Write(data);
I2C_Stop(); delay_dumb(500);
    data &= ~4;
    I2C_Start(); I2C_Write(LCD_ADDR_FIXED << 1); I2C_Write(data);
I2C_Stop(); delay_dumb(500);
}
void LCD_Command(uint8_t cmd) { LCD_Nibble(cmd & 0xF0, 0);
LCD_Nibble((cmd << 4) & 0xF0, 0); }
void LCD_Data(uint8_t data) { LCD_Nibble(data & 0xF0, 1);
LCD_Nibble((data << 4) & 0xF0, 1); }
void LCD_String(char *str) { while(*str) LCD_Data(*str++); }
void LCD_SetCursor(uint8_t row, uint8_t col) { uint8_t addr = (row
== 0) ? 0x80 : 0xC0; LCD_Command(addr + col); }
void LCD_Clear(void) { LCD_Command(0x01); delay_real_ms(2); }
void LCD_Init(void) {

```

```

I2C_Init_Soft(); delay_real_ms(50);
LCD_Nibble(0x30, 0); delay_real_ms(5); LCD_Nibble(0x30, 0);
delay_real_ms(2);
LCD_Nibble(0x30, 0); delay_real_ms(2); LCD_Nibble(0x20, 0);
delay_real_ms(2);
LCD_Command(0x28); LCD_Command(0x08); LCD_Command(0x01);
delay_real_ms(5); LCD_Command(0x0C);
}

// --- UART ---
void UART_Init(void) {
    GPIOA->MODER |= (2<<(9*2)); GPIOA->AFR[1] |= (7<<(1*4));
    USART1->BRR = SystemCoreClock / 9600; USART1->CR1 |=
USART_CR1_TE | USART_CR1_UE;
}
void UART_Print(char *str) { while (*str) { while (!(USART1->SR &
USART_SR_TXE)); USART1->DR = *str++; } }
void UART_SendStats(int skor, int level, int reactionTime, int
status) {
    char buf[64];
    UART_Print("-----\r\n");
    if(status == 1) UART_Print("[RESULT] BENAR!\r\n");
    else UART_Print("[RESULT] SALAH/TIMEOUT\r\n");
    sprintf(buf, "Lvl: %d | Skor: %d | Time: %d ms\r\n", level,
skor, reactionTime);
    UART_Print(buf);
}

// --- BUZZER ---
void Buzzer_Init(void) {
    GPIOB->MODER &= ~(3U << (10 * 2)); GPIOB->MODER |= (1U << (10
* 2));
    GPIOB->OTYPER &= ~(1U << 10); GPIOB->OSPEEDR |= (3U << (10 *
2)); GPIOB->PUPDR &= ~(3U << (10 * 2));
}
void play_tone(uint32_t frequency, uint32_t duration_ms) {
    if (frequency == 0) { GPIOB->ODR &= ~(1<<10);
delay_real_ms(duration_ms); return; }
    uint32_t period_us = 1000000 / frequency; uint32_t half_period

```

```

= period_us / 2;
    long cycles = (long)duration_ms * 1000 / period_us;
    for (long i = 0; i < cycles; i++) {
        GPIOB->ODR |= (1<<10); delay_audio_math(half_period);
        GPIOB->ODR &= ~(1<<10); delay_audio_math(half_period);
    }
    GPIOB->ODR &= ~(1<<10);
}
void soundClick() { play_tone(NOTE_D5, 30); }

// --- LED PWM ---
void PWM_Init(void) {
    GPIOA->MODER &= ~((3<<(0*2)) | (3<<(1*2)) | (3<<(2*2)) |
(3<<(3*2)) | (3<<(8*2)));
    GPIOA->MODER |= ((2<<(0*2)) | (2<<(1*2)) | (2<<(2*2)) |
(2<<(3*2)) | (2<<(8*2)));
    GPIOA->AFR[0] |= (1<<0) | (1<<4) | (1<<8) | (1<<12);
    GPIOA->AFR[1] |= (1<<0);
    TIM2->PSC = 83; TIM2->ARR = 255; TIM2->CCMR1 = (6<<4) |
(6<<12); TIM2->CCMR2 = (6<<4) | (6<<12);
    TIM2->CCER = TIM_CCER_CC1E | TIM_CCER_CC2E | TIM_CCER_CC3E |
TIM_CCER_CC4E; TIM2->CR1 |= TIM_CR1_CEN;
    TIM1->PSC = 83; TIM1->ARR = 255; TIM1->CCMR1 = (6<<4);
TIM1->CCER = TIM_CCER_CC1E;
    TIM1->BDTR |= TIM_BDTR_MOE; TIM1->CR1 |= TIM_CR1_CEN;
}
void set_led_pwm(int led_index, int value) {
    if (value > 255) value = 255; if (value < 0) value = 0;
    switch(led_index) {
        case 0: TIM2->CCR1 = value; break; case 1: TIM2->CCR2 =
value; break;
        case 2: TIM2->CCR3 = value; break; case 3: TIM2->CCR4 =
value; break;
        case 4: TIM1->CCR1 = value; break;
    }
}

// --- EXTI ---
void EXTI_Init_Custom(void) {

```

```

    GPIOB->MODER &= ~((3<<(0*2)) | (3<<(1*2)) | (3<<(3*2)) |
(3<<(4*2)) | (3<<(12*2)));
    GPIOB->PUPDR |= ((1<<(0*2)) | (1<<(1*2)) | (1<<(3*2)) |
(1<<(4*2)) | (1<<(12*2)));
    SYSCFG->EXTICR[0] |= SYSCFG_EXTICR1_EXTI0_PB |
SYSCFG_EXTICR1_EXTI1_PB | SYSCFG_EXTICR1_EXTI3_PB;
    SYSCFG->EXTICR[1] |= SYSCFG_EXTICR2_EXTI4_PB;
SYSCFG->EXTICR[3] |= SYSCFG_EXTICR4_EXTI12_PB;
    EXTI->FTSR |= (1<<0)|(1<<1)|(1<<3)|(1<<4)|(1<<12);
    EXTI->IMR |= (1<<0)|(1<<1)|(1<<3)|(1<<4)|(1<<12);
    NVIC_EnableIRQ(EXTI0_IRQn); NVIC_EnableIRQ(EXTI1_IRQn);
NVIC_EnableIRQ(EXTI3_IRQn);
    NVIC_EnableIRQ(EXTI4_IRQn); NVIC_EnableIRQ(EXTI15_10_IRQn);
}

void Button_Handler(int idx) {
    if (input_allowed == 0) return;

    uint32_t now = get_millis();
    if (now - last_debounce_time > 200) {
        flag_button_pressed = idx;
        last_debounce_time = now;
    }
}

void EXTI0_IRQHandler(void) { if(EXTI->PR & 1) {
Button_Handler(0); EXTI->PR = 1; } }

void EXTI1_IRQHandler(void) { if(EXTI->PR & 2) {
Button_Handler(1); EXTI->PR = 2; } }

void EXTI3_IRQHandler(void) { if(EXTI->PR & 8) {
Button_Handler(3); EXTI->PR = 8; } }

void EXTI4_IRQHandler(void) { if(EXTI->PR & 16) {
Button_Handler(4); EXTI->PR = 16; } }

void EXTI15_10_IRQHandler(void) { if(EXTI->PR & (1<<12)) {
Button_Handler(2); EXTI->PR = (1<<12); } }

// --- TIMER 3 ---
void TIM3_Init(void) {
    RCC->APB1ENR |= RCC_APB1ENR_TIM3EN;
}

```

```

TIM3->PSC = (SystemCoreClock / 1000) - 1;
TIM3->ARR = 2000;

TIM3->EGR |= TIM_EGR_UG;

TIM3->SR = 0;

TIM3->DIER |= TIM_DIER_UIE;
NVIC_EnableIRQ(TIM3_IRQn);
}

void TIM3_Start(int ms) {
    flag_timeout = 0;

    TIM3->CR1 = 0;
    TIM3->ARR = ms;
    TIM3->CNT = ms;

    TIM3->CR1 |= TIM_CR1_DIR;

    TIM3->SR = 0;
    NVIC_ClearPendingIRQ(TIM3_IRQn);

    TIM3->CR1 |= TIM_CR1_CEN;
}

void TIM3_Stop(void) { TIM3->CR1 = 0; }
void TIM3_IRQHandler(void) { if(TIM3->SR & 1) { TIM3->SR = 0;
flag_timeout = 1; TIM3_Stop(); } }

// --- LOGIKA GAME ---
int score = 0;
int level = 1;
int timeLimit = 2000;
int isPlaying = 0;
int currentLed = 0;
int fadeValue = 0;
int fadingIn = 1;
uint32_t lastFadeTime = 0;

```

```

const int fadeSpeed = 10;
char lcdBuf[17];

void melodyStart() {
    int melody[] = {NOTE_G4, NOTE_B4, NOTE_D5, NOTE_G5};
    int durations[] = {100, 100, 100, 200};
    for (int i = 0; i < 4; i++) play_tone(melody[i], durations[i]);
}

void melodyCorrect() {
    play_tone(NOTE_E6, 100); delay_real_ms(20); play_tone(NOTE_G5, 100);
    delay_real_ms(20); play_tone(NOTE_D6, 200); delay_real_ms(50);
}

void melodyGameOver() {
    for(int i=0; i<5; i++) set_led_pwm(i, 255); play_tone(NOTE_D6, 150);
    for(int i=0; i<5; i++) set_led_pwm(i, 0);
delay_real_ms(150);
    for(int i=0; i<5; i++) set_led_pwm(i, 255); play_tone(NOTE_B5, 150);
    for(int i=0; i<5; i++) set_led_pwm(i, 0);
delay_real_ms(150);
    for(int i=0; i<5; i++) set_led_pwm(i, 255); play_tone(NOTE_G5, 400);
    for(int i=0; i<5; i++) set_led_pwm(i, 0);
delay_real_ms(300);
}

void gameOverAnim() {
    LCD_Clear();
    LCD_SetCursor(0, 0); LCD_String("GAME OVER!");
    LCD_SetCursor(1, 0); sprintf(lcdBuf, "Skor: %d", score);
    LCD_String(lcdBuf);
    melodyGameOver();
    for(int i=0; i<5; i++) set_led_pwm(i, 0);
}

```

```

// --- MAIN FUNCTION ---
int main(void) {
    System_Init();
    ADC_Init();
    PWM_Init();
    Buzzer_Init();
    EXTI_Init_Custom();
    UART_Init();
    TIM3_Init();
    LCD_Init();

    LCD_SetCursor(0, 0); LCD_String("Tel-U Game");
    LCD_SetCursor(1, 0); LCD_String("Press any Btn..."); 
    UART_Print("==== SYSTEM READY ===\r\n");
    delay_real_ms(1000);

    flag_button_pressed = -1;
    input_allowed = 1;

    while (1) {
        if (!isPlaying) {
            if (get_millis() - lastFadeTime >= fadeSpeed) {
                lastFadeTime = get_millis();
                for(int i=0; i<5; i++) if (i != currentLed)
                    set_led_pwm(i, 0);
                if (fadingIn) {
                    fadeValue += 8; if (fadeValue >= 255) {
                        fadeValue = 255; fadingIn = 0; }
                } else {
                    fadeValue -= 8;
                    if (fadeValue <= 0) {
                        fadeValue = 0; fadingIn = 1;
                        currentLed++; if (currentLed > 4)
                            currentLed = 0; }
                }
            }
            set_led_pwm(currentLed, fadeValue);
        }
    }
}

```

```

if (flag_button_pressed != -1) {
    input_allowed = 0;
    flag_button_pressed = -1;

    soundClick();
    LCD_Clear();
    LCD_SetCursor(0, 0); LCD_String("Siap...");
    LCD_SetCursor(1, 0); LCD_String("Mulai!");
    melodyStart();
    delay_real_ms(800);

    score = 0; level = 1; timeLimit = 2000; isPlaying
= 1;
    UART_Print("--- GAME START! ---\r\n");
    LCD_Clear();
    LCD_SetCursor(0, 0); LCD_String("Cari LED!");
    LCD_SetCursor(1, 0); LCD_String("Skor: 0");
}
}

else {
    input_allowed = 0;

    for(int i=0; i<5; i++) set_led_pwm(i, 0);
    ADC1->CR2 |= ADC_CR2_SWSTART;
    while(!(ADC1->SR & ADC_SR_EOC));
    uint32_t adc_noise = ADC1->DR;

    uint32_t entropy = adc_noise ^ TIM3->CNT ^ msTicks;

    int target = entropy % 5;

    int levelBrightness = 100 + (level * 20);
    if(levelBrightness > 255) levelBrightness = 255;

    for (int b = 0; b <= levelBrightness; b += 20) {
        set_led_pwm(target, b); delay_real_ms(30);
    }
}

```

```

delay_real_ms(50);

EXTI->PR = 0xFFFFFFFF;
flag_button_pressed = -1;
flag_timeout = 0;

input_allowed = 1;
TIM3_Start(timeLimit);

int last_rem_display = -1;
while (flag_timeout == 0 && flag_button_pressed == -1)
{
    int remaining = TIM3->CNT;
    if (remaining < 0) remaining = 0;

    int display_val = remaining / 100;

    if (display_val != last_rem_display) {
        last_rem_display = display_val;

        int sec = remaining / 1000;
        int ms_part = (remaining % 1000) / 100;

        LCD_SetCursor(0, 11);
        sprintf(lcdBuf, "%d.%ds ", sec, ms_part);
        LCD_String(lcdBuf);
    }
}

TIM3_Stop();
input_allowed = 0;

int reactionTime = timeLimit - TIM3->CNT;
set_led_pwm(target, 0);

if (flag_button_pressed != -1) {
    if (flag_button_pressed == target) {
        // BENAR
        score += 10;
        if (score % 50 == 0) level++;
    }
}

```

```

        UART_SendStats(score, level, reactionTime, 1);
        LCD_SetCursor(1, 0); sprintf(lcdBuf, "Skor: %d
Lvl:%d", score, level); LCD_String(lcdBuf);

        for (int k=0; k<2; k++) {
            set_led_pwm(target, 255);
delay_real_ms(50);
            set_led_pwm(target, 0); delay_real_ms(50);
        }
        melodyCorrect();
        if (timeLimit > 300) timeLimit -= 50;
        delay_real_ms(300);
    } else {
        // SALAH
        UART_SendStats(score, level, reactionTime, 0);
        gameOverAnim();
        isPlaying = 0;
    }
} else {
    // TIMEOUT
    UART_SendStats(score, level, timeLimit, 0);
    gameOverAnim();
    isPlaying = 0;
}

flag_button_pressed = -1;

if(!isPlaying) {
    delay_real_ms(1000);
    LCD_Clear();
    LCD_SetCursor(0, 0); LCD_String("Tel-U Game");
    LCD_SetCursor(1, 0); LCD_String("Play Again?");
}
input_allowed = 1;
}
}
}

```

LAMPIRAN B: Datasheet Excerpts

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw										

Gambar B.1 GPIOx_MODER

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFRL7[3:0]				AFRL6[3:0]				AFRL5[3:0]				AFRL4[3:0]			
rw	rw	rw	rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFRL3[3:0]				AFRL2[3:0]				AFRL1[3:0]				AFRL0[3:0]			
rw	rw	rw	rw												

Gambar B.2 GPIOx_AFRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFRH15[3:0]				AFRH14[3:0]				AFRH13[3:0]				AFRH12[3:0]			
rw	rw	rw	rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFRH11[3:0]				AFRH10[3:0]				AFRH9[3:0]				AFRH8[3:0]			
rw	rw	rw	rw												

Gambar B.3 GPIOx_AFRH

Position	Priority	Type of priority	Acronym	Description	Address
-	-	-	-	Reserved	0x0000 0000
-	-3	fixed	Reset	Reset	0x0000 0004
-	-2	fixed	NMI	Non maskable interrupt, Clock Security System	0x0000 0008
-	-1	fixed	HardFault	All class of fault	0x0000 000C
-	0	settable	MemManage	Memory management	0x0000 0010
-	1	settable	BusFault	Pre-fetch fault, memory access fault	0x0000 0014
-	2	settable	UsageFault	Undefined instruction or illegal state	0x0000 0018
-	-	-	-	Reserved	0x0000 001C - 0x0000 002B
-	3	settable	SVCall	System Service call via SWI instruction	0x0000 002C
-	4	settable	Debug Monitor	Debug Monitor	0x0000 0030
-	-	-	-	Reserved	0x0000 0034
-	5	settable	PendSV	Pendable request for system service	0x0000 0038
-	6	settable	Systick	System tick timer	0x0000 003C
0	7	settable	WWDG	Window Watchdog interrupt	0x0000 0040
1	8	settable	EXTI16 / PVD	EXTI Line 16 interrupt / PVD through EXTI line detection interrupt	0x0000 0044
2	9	settable	EXTI21 / TAMP_STAMP	EXTI Line 21 interrupt / Tamper andTimeStamp interrupts through the EXTI line	0x0000 0048
3	10	settable	EXTI22 / RTC_WKUP	EXTI Line 22 interrupt / RTC Wake-up interrupt through the EXTI line	0x0000 004C
4	11	settable	FLASH	Flash global interrupt	0x0000 0050
5	12	settable	RCC	RCC global interrupt	0x0000 0054
6	13	settable	EXTI0	EXTI Line0 interrupt	0x0000 0058
7	14	settable	EXTI1	EXTI Line1 interrupt	0x0000 005C
8	15	settable	EXTI2	EXTI Line2 interrupt	0x0000 0060
9	16	settable	EXTI3	EXTI Line3 interrupt	0x0000 0064
10	17	settable	EXTI4	EXTI Line4 interrupt	0x0000 0068
11	18	settable	DMA1_Stream0	DMA1 Stream0 global interrupt	0x0000 006C

Position	Priority	Type of priority	Acronym	Description	Address
12	19	settable	DMA1_Stream1	DMA1 Stream1 global interrupt	0x0000 0070
13	20	settable	DMA1_Stream2	DMA1 Stream2 global interrupt	0x0000 0074
14	21	settable	DMA1_Stream3	DMA1 Stream3 global interrupt	0x0000 0078
15	22	settable	DMA1_Stream4	DMA1 Stream4 global interrupt	0x0000 007C
16	23	settable	DMA1_Stream5	DMA1 Stream5 global interrupt	0x0000 0080
17	24	settable	DMA1_Stream6	DMA1 Stream6 global interrupt	0x0000 0084
18	25	settable	ADC	ADC1 global interrupts	0x0000 0088
23	30	settable	EXTI9_5	EXTI Line[9:5] interrupts	0x0000 009C
24	31	settable	TIM1_BRK_TIM9	TIM1 Break interrupt and TIM9 global interrupt	0x0000 00A0
25	32	settable	TIM1_UP_TIM10	TIM1 Update interrupt and TIM10 global interrupt	0x0000 00A4
26	33	settable	TIM1_TRG_COM_TIM11	TIM1 Trigger and Commutation interrupts and TIM11 global interrupt	0x0000 00A8
27	34	settable	TIM1_CC	TIM1 Capture Compare interrupt	0x0000 00AC
28	35	settable	TIM2	TIM2 global interrupt	0x0000 00B0
29	36	settable	TIM3	TIM3 global interrupt	0x0000 00B4
30	37	settable	TIM4	TIM4 global interrupt	0x0000 00B8
31	38	settable	I2C1_EV	I ² C1 event interrupt	0x0000 00BC
32	39	settable	I2C1_ER	I ² C1 error interrupt	0x0000 00C0
33	40	settable	I2C2_EV	I ² C2 event interrupt	0x0000 00C4
34	41	settable	I2C2_ER	I ² C2 error interrupt	0x0000 00C8
35	42	settable	SPI1	SPI1 global interrupt	0x0000 00CC
36	43	settable	SPI2	SPI2 global interrupt	0x0000 00D0
37	44	settable	USART1	USART1 global interrupt	0x0000 00D4
38	45	settable	USART2	USART2 global interrupt	0x0000 00D8
40	47	settable	EXTI15_10	EXTI Line[15:10] interrupts	0x0000 00E0
41	48	settable	EXTI17 / RTC_Alarm	EXTI Line 17 interrupt / RTC Alarms (A and B) through EXTI line interrupt	0x0000 00E4
42	49	settable	EXTI18 / OTG_FS_WKUP	EXTI Line 18 interrupt / USB On-The-Go FS Wake-up through EXTI line interrupt	0x0000 00E8
47	54	settable	DMA1_Stream7	DMA1 Stream7 global interrupt	0x0000 00FC

Position	Priority	Type of priority	Acronym	Description	Address
49	56	settable	SDIO	SDIO global interrupt	0x0000 0104
50	57	settable	TIM5	TIM5 global interrupt	0x0000 0108
51	58	settable	SPI3	SPI3 global interrupt	0x0000 010C
56	63	settable	DMA2_Stream0	DMA2 Stream0 global interrupt	0x0000 0120
57	64	settable	DMA2_Stream1	DMA2 Stream1 global interrupt	0x0000 0124
58	65	settable	DMA2_Stream2	DMA2 Stream2 global interrupt	0x0000 0128
59	66	settable	DMA2_Stream3	DMA2 Stream3 global interrupt	0x0000 012C
60	67	settable	DMA2_Stream4	DMA2 Stream4 global interrupt	0x0000 0130
67	74	settable	OTG_FS	USB On The Go FS global interrupt	0x0000 014C
68	75	settable	DMA2_Stream5	DMA2 Stream5 global interrupt	0x0000 0150
69	76	settable	DMA2_Stream6	DMA2 Stream6 global interrupt	0x0000 0154
70	77	settable	DMA2_Stream7	DMA2 Stream7 global interrupt	0x0000 0158
71	78	settable	USART6	USART6 global interrupt	0x0000 015C
72	79	settable	I2C3_EV	I ² C3 event interrupt	0x0000 0160
73	80	settable	I2C3_ER	I ² C3 error interrupt	0x0000 0164
81	88	Settable	FPU	FPU global interrupt	0x0000 0184
84	91	settable	SPI4	SPI 4 global interrupt	0x0000 0190
85	92	settable	SPI5	SPI 5 global interrupt	0x0000 0194

Gambar B.4 Interrupt Vector Table

Port	AF00	AF01	AF02	AF03	AF04	AF05	AF06	AF07	AF08	AF09	AF10	AF11	AF12	AF13	AF14	AF15
	SYS_AF	TIM1/TIM2	TIM3/ TIM4/ TIM5	TIM9/ TIM10/ TIM11	I2C1/I2C2/ I2C3	SPI1/SPI2/ I2S2/SPI3/ I2S3/SPI4	SPI2/I2S2/ SPI3/I2S3	SPI3/I2S3/ USART1/ USART2	USART6	I2C2/ I2C3	OTG1_FS	SDIO				
Port A	PA0	-	TIM2_CH1/ TIM2_ETR	TIM5_CH1	-	-	-	USART2_ CTS	-	-	-	-	-	-	-	EVENT OUT
	PA1	-	TIM2_CH2	TIM5_CH2	-	-	-	USART2_ RTS	-	-	-	-	-	-	-	EVENT OUT
	PA2	-	TIM2_CH3	TIM5_CH3	TIM9_CH1	-	-	USART2_ TX	-	-	-	-	-	-	-	EVENT OUT
	PA3	-	TIM2_CH4	TIM5_CH4	TIM9_CH2	-	-	USART2_ RX	-	-	-	-	-	-	-	EVENT OUT
	PA4	-	-	-	-	SPI1_NSS	SPI3_NSS/ I2S3_WS	USART2_ CK	-	-	-	-	-	-	-	EVENT OUT
	PA5	-	TIM2_CH1/ TIM2_ETR	-	-	SPI1_SCK	-	-	-	-	-	-	-	-	-	EVENT OUT
	PA6	-	TIM1_BKIN	TIM3_CH1	-	-	SPI1_MISO	-	-	-	-	-	-	-	-	EVENT OUT
	PA7	-	TIM1_CH1N	TIM3_CH2	-	-	SPI1_MOSI	-	-	-	-	-	-	-	-	EVENT OUT
	PA8	MCO_1	TIM1_CH1	-	-	I2C3_SCL	-	-	USART1_ CK	-	-	OTG_FS_ SOF	-	-	-	EVENT OUT
	PA9	-	TIM1_CH2	-	-	I2C3_SMB	-	-	USART1_ TX	-	-	OTG_FS_ VBUS	-	-	--	EVENT OUT
	PA10	-	TIM1_CH3	-	-	-	-	-	USART1_ RX	-	-	OTG_FS_I D	-	-	-	EVENT OUT
	PA11	-	TIM1_CH4	-	-	-	-	-	USART1_ CTS	USART6_ TX	-	OTG_FS_DM	-	-	-	EVENT OUT
	PA12	-	TIM1_ETR	-	-	-	-	-	USART1_ RTS	USART6_ RX	-	OTG_FS_DP	-	-	-	EVENT OUT
	PA13	JTMS_SWDIO	-	-	-	-	-	-	-	-	-	-	-	-	-	EVENT OUT
	PA14	JTCK_SWCLK	-	-	-	-	-	-	-	-	-	-	-	-	-	EVENT OUT
	PA15	JTDI	TIM2_CH1/ TIM2_ETR	-	-	SPI1_NSS	SPI3_NSS/ I2S3_WS	-	-	-	-	-	-	-	-	EVENT OUT

Port	AF00	AF01	AF02	AF03	AF04	AF05	AF06	AF07	AF08	AF09	AF10	AF11	AF12	AF13	AF14	AF15
	SYS_AF	TIM1/TIM2	TIM3/ TIM4/ TIM5	TIM9/ TIM10/ TIM11	I2C1/I2C2/ I2C3	SPI1/SPI2/ I2S2/SPI3/ I2S3/SPI4	SPI2/I2S2/ SPI3/I2S3	SPI3/I2S3/ USART1/ USART2	USART6	I2C2/ I2C3	OTG1_FS	SDIO				
Port B	PB0	-	TIM1_CH2N	TIM3_CH3	-	-	-	-	-	-	-	-	-	-	-	EVENT OUT
	PB1	-	TIM1_CH3N	TIM3_CH4	-	-	-	-	-	-	-	-	-	-	-	EVENT OUT
	PB2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	EVENT OUT
	PB3	JTDO_SWO	TIM2_CH2	-	-	SPI1_SCK	SPI3_SCK/ I2S3_CK	-	-	I2C2_SDA	-	-	-	-	-	EVENT OUT
	PB4	JTRST	-	TIM3_CH1	-	-	SPI1_MISO	SPI3_MISO	I2Sext_SD	-	I2C3_SDA	-	-	-	-	EVENT OUT
	PB5	-	-	TIM3_CH2	-	I2C1_SMB	SPI1_MOSI	SPI3_MOSI/ I2S3_SD	-	-	-	-	-	-	-	EVENT OUT
	PB6	-	-	TIM4_CH1	-	I2C1_SCL	-	-	USART1_ TX	-	-	-	-	-	-	EVENT OUT
	PB7	-	-	TIM4_CH2	-	I2C1_SDA	-	-	USART1_ RX	-	-	-	-	-	-	EVENT OUT
	PB8	-	-	TIM4_CH3	TIM10_CH1	I2C1_SCL	-	-	-	-	-	-	SDIO_D4	-	-	EVENT OUT
	PB9	-	-	TIM4_CH4	TIM11_CH1	I2C1_SDA	SPI2 NSS/ I2S2_WS	-	-	-	-	-	SDIO_D5	-	-	EVENT OUT
	PB10	-	TIM2_CH3	-	-	I2C2_SCL	SPI2_SCK/ I2S2_CK	-	-	-	-	-	-	-	-	EVENT OUT
	PB11	-	TIM2_CH4	-	-	I2C2_SDA	-	-	-	-	-	-	-	-	-	EVENT OUT
	PB12	-	TIM1_BKIN	-	-	I2C2_SMB	SPI2 NSS/ I2S2_WS	-	-	-	-	-	-	-	-	EVENT OUT
	PB13	-	TIM1_CH1N	-	-	-	SPI2_SCK/ I2S2_CK	-	-	-	-	-	-	-	-	EVENT OUT
	PB14	-	TIM1_CH2N	-	-	-	SPI2_MISO	I2S2ext_SD	-	-	-	-	-	-	-	EVENT OUT
	PB15	RTC_REFN	TIM1_CH3N	-	-	-	SPI2_MOSI/ I2S2_SD	-	-	-	-	-	-	-	-	EVENT OUT

Gambar B.5 Alternate Function Mapping

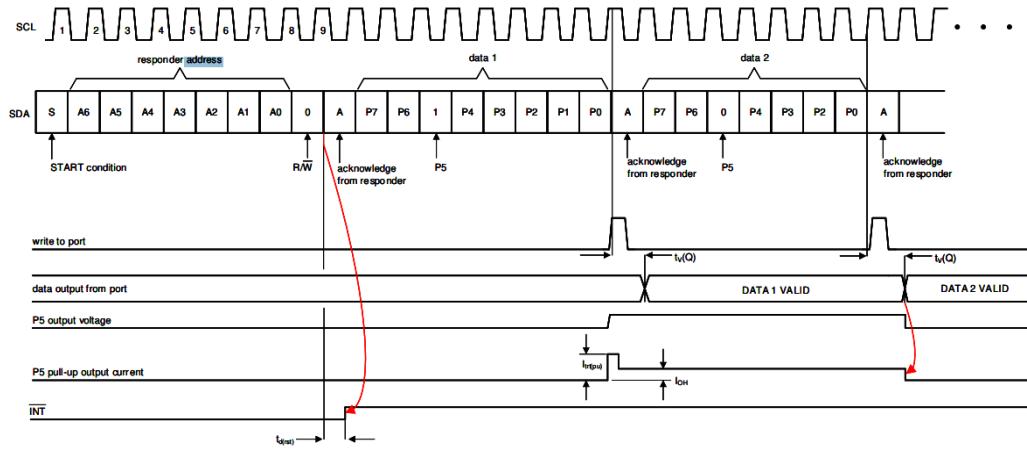
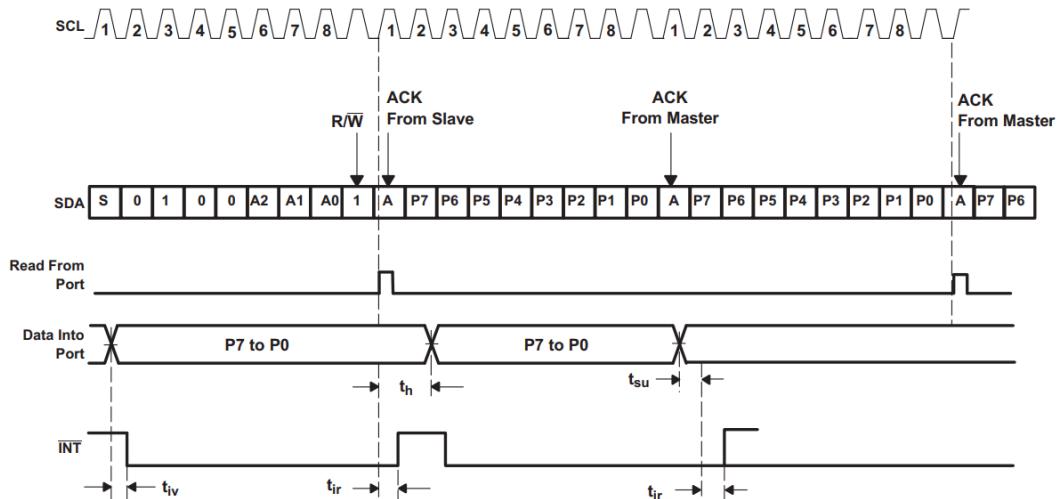


Figure 7-3. Write Mode (Output)



- A. A low-to-high transition of SDA while SCL is high is defined as the stop condition (P). The transfer of data can be stopped at any moment by a stop condition. When this occurs, data present at the latest ACK phase is valid (output mode). Input data is lost.

Figure 7-4. Read Mode (Input)

Gambar B.6 Device Address

LAMPIRAN C: Testing Results

Tabel C.1 Testing Results

ID	Skenario / Fokus Uji	Hasil Pengamatan (Observed Result)	Kesimpulan

ST-01	Inisialisasi Sistem (<i>Startup</i>)	Setelah tombol Reset ditekan, LCD menyala menampilkan teks "Tel-U Game" dan "Press Any Btn". Tidak ada karakter aneh pada LCD. Serial Monitor menampilkan log "==== SYSTEM READY ===".	Berhasil
ST-02	Pola Permainan Acak	Dalam 5 ronde permainan awal, LED target berpindah posisi secara dinamis dan tidak membentuk pola berulang yang mudah ditebak.	Berhasil
ST-03	Respons Kondisi Menang	Saat tombol yang benar ditekan, <i>buzzer</i> berbunyi 'bip' pendek ($\pm 100\text{ms}$), skor di LCD bertambah +10, dan LED target langsung mati.	Berhasil
ST-04	Respons Kondisi Kalah	Saat tombol salah ditekan, permainan berhenti instan, <i>buzzer</i> berbunyi nada panjang, dan LCD menampilkan "GAME OVER" beserta skor akhir. Hal sama terjadi saat didiamkan >2 detik (<i>timeout</i>).	Berhasil

ST-05	Telemetri Data ke PC	Serial Monitor menampilkan log data yang akurat pasca permainan. Contoh: '[RESULT] TIMEOUT	Berhasil
ST-06	Stabilitas Sistem (<i>Stress Test</i>)	Tombol ditekan acak dengan cepat tidak menyebabkan mikrokontroler macet. Input tombol saat inisialisasi diabaikan sistem (tidak <i>error</i>) hingga pesan "Ready" muncul.	Berhasil

LAMPIRAN D: Video Demonstration

Link: [VIDEO DEMONSTRASI TUBES_Kelompok 2_TK 47-03.mp4](#)

QR Code:



Gambar D.1 QR Code Video Demonstrasi

CHECKLIST KELENGKAPAN LAPORAN

Content: 10%

- [x] Abstrak
- [x] BAB I-VII lengkap
- [x] Minimum 20 halaman (tidak termasuk lampiran)
- [x] Semua gambar, table diberi caption dan nomor

Technical: 40%

- [x] Register configuration explained dengan tabel
- [x] Schematic diagram, jelas dan lengkap
- [x] Flowchart dan state machine included
- [x] Photo dokumentasi hardware
- [x] Video demonstration

CLO3 Requirements: 40 %

- [x] Penerapan Serial communication (I2C/UART/SPI)
- [x] Penerapan Timer configuration explained
- [x] Interrupt system explained
- [x] Register-level code documented
- [x] Integration testing performed

Formatting: 10%

- [x] Font: Times New Roman 12pt
- [x] Line spacing: 1.5
- [x] Margin: 4-3-3-3 cm
- [x] Page numbers