Laporan Final *Game PingPong* pada *Dotmatrix Display*

Dosen Pengampu

Eko Pramunanto, S.T. M.T.

Disusun Oleh:

Muhammad Haekal Muhyidin Al-Araby

5024221004

Sistem Tertanam - A

DEPARTEMEN TEKNIK KOMPUTER
FAKULTAS TEKNOLOGI ELEKTRO DAN INFORMATIKA CERDAS
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
2024

1 Gambaran Umum

Pingpong atau tenis meja adalah salah satu olahraga yang dimainkan 2 orang yang saling berlawanan. Untuk mendapat skor pemain harus memasukkan bola ke daerah lawan. Bola digerakkan dengan paddle. Pemain dapat service, smash, maupun menangkis bola.

Project ini adalah implementasi dari pingpong ke dalam dotmatrix display dimana player dapat mengendalikan paddle dengan slide potentiomenter dan smash dengan menekan button. smash akan membuat bola bergerak lurus dan bertambah cepat.

Untuk memenangkan permainan Player harus mendapatkan skor lebih atau sama dengan 11 dan memiliki keunggulan 2 poin dari lawan. Bila keunggulan tersebut tidak dimiliki maka permainan akan berlanjut terus menerus hingga keduanya mendapatkan skor 15 yang akan berakhir seri.

2 Komponen

2.1 Normally Close Switch

Saklar yang yang tertutup pada kondisi normal dan terbuka ketika ditekan. Dengan memanfaatkan pull-up internal pada pin GPIO ESP32 dan menyambungkan ujung lainnya ke ground kita dapat mendapatkan nilai 1 ketika ditekan. Pada project ini switch ini dimanfaatkan untuk player dapat melakukan smash saat ditekan.

2.2 Slide Potentiometer

Potentiometer jenis ini mengubah resistansi pada rangkaian dengan menggunakan mekanisme geser. Karena sifatnya yang dapat menghasilkan data analog biasa digunakan untuk mengatur parameter suatu device dengan akurasi yang lebih tinggi. Pada ESP32 kita dapat membaca nilai dari potentio meter ini dengan menggunakan pin ADC pada ESP32. Pin ADC pada ESP32 dapat membaca nilai tegangan dengan rentang 0 - 3.3v setelah itu maka esp akan menunjukkan nilai maksimum. Resolusi ADC dari ESP32 adalah dari rentang 0 - 12 dimana semakin tinggi resolusi akan semakin akurat namun semakin berat juga pada performa. Pada project ini saya memilih 3 sebagai resolusi karena kita hanya membutuhkan nilai 0 - 8. Karena hanya terdapat 8 pixel pada dotmatrix vertikal.

2.3 Led DotMatrix 8x32

Dotmatrix MAX7219 adalah modul untuk mengendalikan display dotmatrix 8x8 sebanyak 4 buah. Chip yang digunakan mampu mengendalikan seluruh pixel pada dotmatrix. Informasi dikirimkan secara serial. Dan menyebar ke seluruh dotmatrix. Led ini biasa digunakan untuk running text dan game interaktif.

2.4 ESP32

ESP32 adalah Microcontroller yang berfungsi sebagai pengendali utama sistem. ESP32 dapat menerima input analog maupun digital. Lalu dapat mengeluarkan output digital melalui pin GPIO. ESP32 pada project ini diprogram untuk menerima input dari Slide Potentiometer dan Normally Close Switch lalu memberikan keluaran pada 2 device yaitu LED Dotmatrix dan Buzzer Pasif. Keluaran pada LED Dotmatrix berupa data digital yang dikirim secara serial sedangkan pada buzzer berupa PWM.

3 Desain Sistem

3.1 Rangkaian Skematik

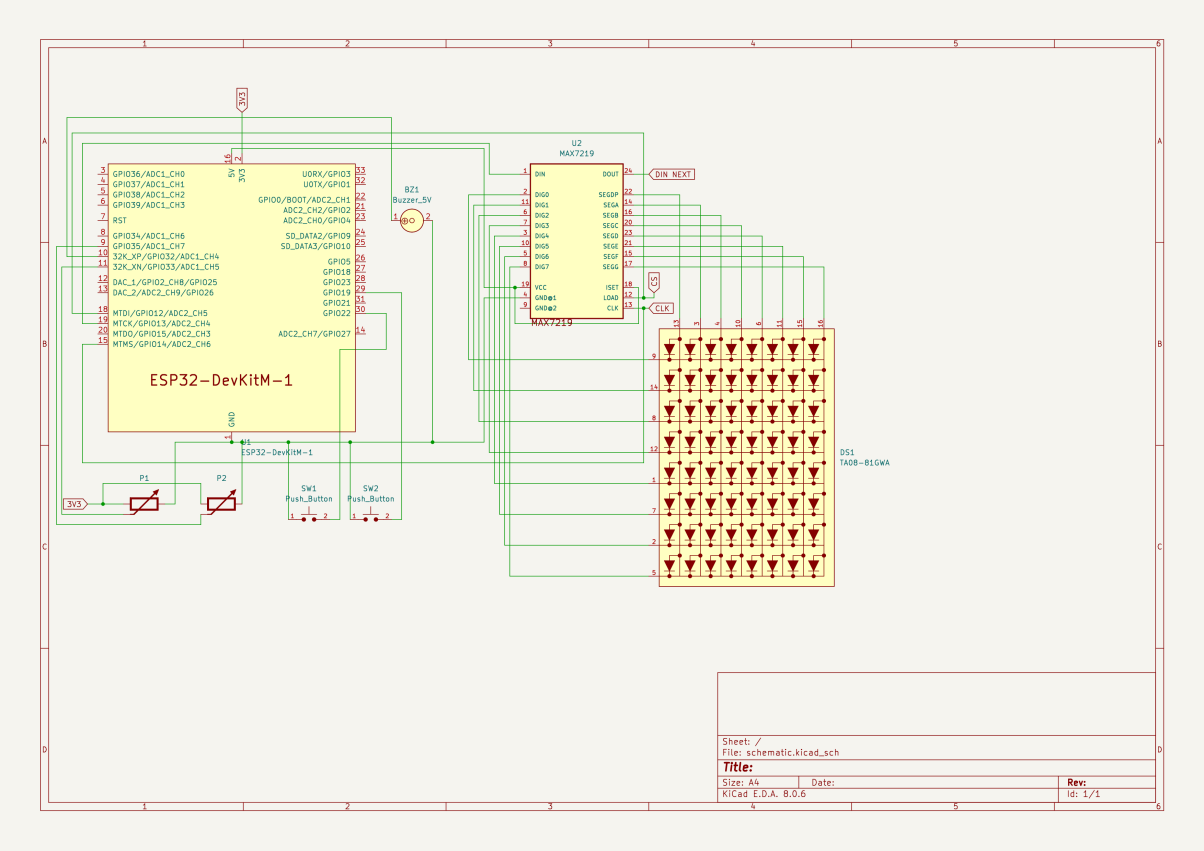

Figure 1: Schematic

3.2 Pemilihan Komponen

1. Potensiometer adalah slide potentiometer dengan resistansi variabel 220k ohm. Pemilihan ini dilakukan karena 220 k ohm adalah nilai yang tepat dimana tidak terlalu rendah sehingga menghasilkan noise maupun terlalu tinggi sehingg tidak kompatibel dengan impedansi ESP32

2. Normally Close Switch digunakan karena input smash adalah salah satu button yang pastinya akan sering ditekan sehingga dengan penggunaan komponen ini kerusakan akibat penggunaan diminimalkan.

3.3 Desain Game

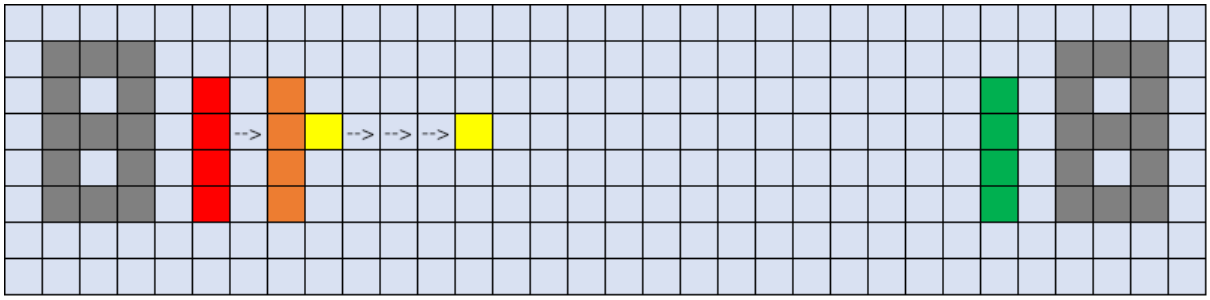

Figure 2: Desain Game

3.3.1 Mekanisme dan Peraturan Game

1. Game memiliki 2 player yang dapat digerakkan dengan slide potentiometer
2. Bola memiliki kecepatan konstan
3. Bola akan memantul saat menabrak pembatas ataupun player
   (a) Ketika bola mengenai pembatas bola akan memantul secara diagonal dengan mempertahankan gerakan horizontal dan membalik gerakan vertikal.
   (b) Ketika bola mengenai player pada bagian tengah bola akan memantul secara lurus dan membalik gerakan horizontallnya
   (c) Ketika bola mengenai player pada bagian atas bola akan memantul ke atas.
   (d) Ketika bola mengenai player pada bagian bawah bola akan memantuk ke bawah.
4. Bila bola melewati daerah player maka player lainnya akan mendapatkan skor 1.
5. Player dapat melakukan smash.
   (a) Smash akan menggerakkan player 2 blok ke depan.
   (b) Bila saat smash player mengenai bola, bola akan bergerak lurus dengan kecepatan tinggi ke arah lawan.
   (c) Durasi smash adalah 0.2 detik
   (d) Smash memiliki durasi cooldown 1 detik dimana setelah smash player tidak dapat melakukan smash lagi hingga cooldown berakhir.
   (e) Ketika smash bola memiliki kemungkinan untuk melewati player dan memberikan skor kepada lawan.

6. Player yang kebobolan akan mendapatkan kesempatan untuk melakukan service.
   (a) Saat service bola akan mengikuti gerakan player.
   (b) Player dapat melakukan service dengan menekan tombol smash.
7. Pemain yang mendapatkan skor di atas 11 dengan selisih 2 dari lawan memenangkan permainan
8. Game akan berakhir imbang ketika kedua pemain mencapai 15 poin

3.3.2 Fitur

1. Permainan diiringi dengan musik yang berasal dari Buzzer.
   (a) terdapat 2 musik yaitu Tetris Theme dan Mario Theme
   (b) Tetris Theme diputar saat menu awal muncul, pemain mencetak skor, dan saat permainan berakhir.
   (c) Mario Theme saat permainan berlangsung. Mario Theme dipilih karena nadanya yang cukup tenang sehingga tidak mengganggu konsentrasi pemain.
2. Tiap event seperti mencetak skor dan kemenangan mempunyai animasi tersendiri
   (a) Saat pemain mencetak skor akan ditampilkan animasi Text dari arah player yang mencetak skor.
   (b) Saat salah satu pemain menang akan tampil juga animasi Text dari player yang menang.
   (c) Ketika imbang animasi akan muncul dari arah bawah.

4 Implementasi

4.1 Hardware

Seperti yang terlihat pada schematic, rangkaian menggunakan ESP32 yang terhubung pada IC MAX7219 dimana CS terhubung pada pin D12, DIN pada D13 dan clock pada D14. Lalu VCC menggunakan pin 5V pada modul terdapat 4 IC MAX7219 yang saling terhubung secara seri.

Untuk input dari push button pada 1 pin dihubungkan pada ground dan 1 nya dihubungkan pada GPIO 19 dan 22 yang masing masing berfungsi untuk input smash button dari player 1 dan 2 berturut-turut. Selanjutnya untuk input gerakan player menggunakan slide potentiometer yang di kedua ujung dihubungkan pad VCC 3.3 V dan Ground. Lalu output dari potentiometer berturut-turut dihubungkan ke GPIO 33 dan 35 untuk mengendalikan gerakan player 1 dan player 2.

Untuk buzzer sendiri digunakan buzzer pasif yang dihubungkan ke GPIO 32 dan GROUND.

Wiring digunakan menggunakan kabel yang disolder pada pcb untuk menjamin konektivitas.

Input dari player 1 dan player 2 dipisahkan dan dihubungkan dengan kabel agar dapat dimainkan dengan jarak yang cukup.

4.2 Software

4.2.1 Alur Program

1. Program dimulai dengan inisialisasi Global Variabel yang terdapat pada global.hpp dan global.cpp, variabel tersebut akan digunakan pada keseluruhan program. Dimana variabel bila memungkinkan akan menggunakan keyword static sehingga akan sama. Sedangkan yang tidak memungkinkan menggunakan keyword static akan menggunakan keyword extern dan didefine pada global.gpp
2. Program lalu menjalankan fungsi Setup pada device untuk melakukan inisiasi dan setup pada device MAX7019 menggunakan library MD_Parola dan MD_MAX72xx
3. Setelah itu program akan menjalankan intro yang berisi animasi dan text nama dan nrp.
4. Lalu program akan menampilkan menu yang bertuliskan "PRESS ANY BUTTON TO START...", dimana program akan menunggu user untuk menekan tombol apapun. Pada saat ini program juga akan memutar musik tetris theme.
5. Sebelum program memasuki loop utama music akan diganti terlebih dahulu ke mario theme dan player 1 diberi kesempatan service.
6. Program memasuki main loop
7. Program akan memeriksa apakah memasuki loop setelah game berahir atau tidak bila iya akan diberi delay 100.
8. Music akan diputar pada tiap loop dengan memanggil fungsi Play pada class MusicPlayer
9. Program akan melakukan update posisi dan memeriksa input pada tiap loop.
10. Setiap 100ms program akan melakukan update display dengan clear display lalu memanggil fungsi Draw pada class Player dan Ball. juga melakukan RenderScore untuk skor Player 1 dan Player 2.
11. Setelah itu program akan memeriksa kondisi kemenangan pada permainan. Dan menampilkan animasi yang sesua dengan kondisi yang terpenuhi.
12. Permainan akan berulang ketika salah satu button ditekan.

4.3 Source Code dan Penjelasan

4.3.1 Main Program

Pada bagian ini program utama akan dijalankan dengan alur yang sudah dijelaskan pada subsection alur code.

```
#include <Arduino.h>
#include <cstdlib>
#include <ctime>

#include "MD_Parola.h"
#include "animation.hpp"
#include "device.hpp"
#include "esp32-hal-adc.h"
#include "esp32-hal-gpio.h"
#include "global.hpp"
#include "music.hpp"
#include "score.hpp"


void setup() {
    Setup();
    Intro();
    analogReadResolution(3);
    display.displayScroll("PRESS ANY BUTTON TO START...", PA_CENTER,
   PA_SCROLL_LEFT, 100);
    mp.Switch(&tetris);
    while (!digitalRead(p1.smashInput) && !digitalRead(p2.smashInput))
   {
        mp.Play();
        if (display.displayAnimate()) {
            display.displayReset();
        }
    }
    delay(300);
    mp.Switch(&mario_underworld);
    p1.service = true;
}

int last = 0;
bool after_win = false;

void loop() {
    if (after_win) {
        delay(100);
        after_win = false;
    }
    mp.Play();
    int current = millis();

    b.Update();
    p1.CheckInput();
    p2.CheckInput();

    if (current - last < 100) {
        return;
    }

    last  = current;

    native_display.clear();
    p1.Draw();
    p2.Draw();
```

```
    b.Draw();

    RightRenderScore(p2.score, 9);
    LeftRenderScore(p1.score, -10);

    if (p1.score >= 11 && p1.score - p2.score >= 2) {
        delay(100);
        mp.Switch(&tetris);
        native_display.clear();
        display.displayText("P1 WIN", PA_CENTER, 50, 1000,
PA_SCROLL_RIGHT, PA_SCROLL_RIGHT);
        while (!digitalRead(p1.smashInput)&& !digitalRead(p2.smashInput
)) {
            mp.Play();
            if (display.displayAnimate()) {
                display.displayReset();
            }
        }
        p1.score = 0;
        p2.score = 0;
        mp.Switch(&mario_underworld);
        after_win = true;
    }
    else if (p2.score >= 11 && p2.score - p1.score >= 2) {
        delay(100);
        mp.Switch(&tetris);
        native_display.clear();
        display.displayText("P2 WIN", PA_CENTER, 50, 1000,
PA_SCROLL_LEFT, PA_SCROLL_LEFT);
        while (!digitalRead(p1.smashInput)&& !digitalRead(p2.smashInput
)) {
            mp.Play();
            if (display.displayAnimate()) {
                display.displayReset();
            }
        }
        p1.score = 0;
        p2.score = 0;
        mp.Switch(&mario_underworld);
        after_win = true;
    } else if (p2.score >= 15 && p1.score >= 15) {
        delay(100);
        mp.Switch(&tetris);
        native_display.clear();
        display.displayText("TIE", PA_CENTER, 50, 1000, PA_SCROLL_UP,
PA_SCROLL_UP);
        while (!digitalRead(p1.smashInput)&& !digitalRead(p2.smashInput
)) {
            mp.Play();
            if (display.displayAnimate()) {
                display.displayReset();
            }
        }
        p1.score = 0;
        p2.score = 0;
        mp.Switch(&mario_underworld);
        after_win = true;
    }
```

```
}
```

Listing 1: main.cpp

4.3.2 Display Functionality

Display dilakukan dengan 2 metode yaitu menggunakan MD_Parola untuk menampilkan animasi text dan MD_MAX72xx untuk menampilkan pixel secara manual. Pada bagian ini program akan menginsialisasi device dotmatrix dengan parameter dan pin yang telah ditentukan.

```
#pragma once

#include <MD_Parola.h>
#include <MD_MAX72xx.h>
#include <SPI.h>
#include <cstdint>

constexpr MD_MAX72XX::moduleType_t HARDWARE_TYPE = MD_MAX72XX::FC16_HW;
constexpr uint8_t MAX_DEVICES = 4;
constexpr uint8_t CS_PIN = 12;
constexpr uint8_t DIN_PIN = 13;
constexpr uint8_t CLK_PIN = 14;
constexpr uint16_t SPEED = 100;

extern MD_Parola display;
extern MD_MAX72XX native_display;

void Setup();
```

Listing 2: device.hpp

```
#include "device.hpp"
#include "HardwareSerial.h"

MD_Parola display = MD_Parola(HARDWARE_TYPE, DIN_PIN, CLK_PIN, CS_PIN,
    MAX_DEVICES);
MD_MAX72XX native_display = MD_MAX72XX(HARDWARE_TYPE, DIN_PIN, CLK_PIN,
     CS_PIN, MAX_DEVICES);

void Setup() {
    native_display.begin();
    native_display.clear();
    native_display.control(MD_MAX72XX::INTENSITY, 0);

    display.begin();
    display.setIntensity(10);
    display.displayClear();

    Serial.begin(9600);
}
```

Listing 3: device.cpp

Class Vector2 digunakan untuk merepresentasikan koordinat kartesian pada koordinat Dotmatrix dan untuk melakukan fungsi Draw dimana fungsi Draw akan memanggil fungsi

setPoint pada MD_MAX72xx untuk menyalakan pixel pada koordinat vector tersebut.

```
#pragma once

class Vector2 {
public:
    int x = 0,y = 0;
    Vector2(int x = 0, int y = 0): x(x), y(y) {}

    Vector2 operator+(const Vector2& other) const {
        return Vector2(x + other.x, y + other.y);
    }
    Vector2 operator-(const Vector2& other) const {
        return Vector2(x - other.x, y - other.y);
    }
    Vector2 operator*(const Vector2& other) const {
        return Vector2(x * other.x, y * other.y);
    }
    Vector2 operator/(const Vector2& other) const {
        return Vector2(x / other.x, y / other.y);
    }
    Vector2 operator+(const float n) const {
        return Vector2(x + n, y + n);
    }
    Vector2 operator-(const float n) const {
        return Vector2(x - n, y - n);
    }
    Vector2 operator*(const float n) const {
        return Vector2(x * n, y * n);
    }
    Vector2 operator/(const float n) const {
        return Vector2(x / n, y / n);
    }
    Vector2 GlobalToDisplay() const;
    void Draw() const;
    void Clear() const;
};
```

Listing 4: vector2.hpp

```
#include "device.hpp"
#include "esp32-hal-gpio.h"
#include <vector2.hpp>

Vector2 Vector2::GlobalToDisplay() const {
    // (0, 0) -> row 3 column 15
    return *this * -1 + Vector2(15, 3) ;
}

void Vector2::Draw() const {
    Vector2 displayPos = this->GlobalToDisplay();
    native_display.setPoint(displayPos.y, displayPos.x, HIGH);
}

void Vector2::Clear() const {
    Vector2 displayPos = this->GlobalToDisplay();
    native_display.setPoint(displayPos.y, displayPos.x, LOW);
}
```

Listing 5: vector2.cpp

Pada bagian ini didefinisikan fungsi intro yang berisi pola-pola untuk ditampilkan pada intro permainan. Dengan memanggil fungsi dari MD_MAX72xx untuk menyalakan pixel dan mematikan pixel secara manual sesuai dengan pola yang sudah ditentukan

```
#pragma once


void Intro();
```

Listing 6: animation.hpp

```
#include "device.hpp"
#include <cstdio>
#include "animation.hpp"
#include "global.hpp"

void Intro() {
    for (int i = 0; i < 8; i++) {
        for (int j = 0; j < 8 * MAX_DEVICES; j++) {
            native_display.setPoint(i, j, HIGH);
            delay(20);
            native_display.setPoint(i, j, LOW);
        }
        delay(20);
    }
    native_display.clear();

    for (int i = 0; i < 8; i++) {
        for (int j = 0; j < 8 * MAX_DEVICES; j++) {
            native_display.setPoint(i, j, HIGH);
            native_display.setPoint(i - 1, j, LOW);
        }
        delay(20);
    }
    native_display.clear();
    for (int i = 8; i >= 0; i--) {
        for (int j = 8 * MAX_DEVICES; j >= 0; j--) {
            native_display.setPoint(i, j, HIGH);
        }
        delay(20);
    }
    native_display.clear();
    for (int j = 0; j < 8 * MAX_DEVICES; j++) {
        for (int i = 0; i < 8; i++) {
            native_display.setPoint(i, j, HIGH);
            native_display.setPoint(i, j - 1, LOW);
        }
        delay(20);
    }
    native_display.clear();
    for (int j = 8 * MAX_DEVICES; j >= 0; j--) {
        for (int i = 8; i >= 0; i--) {
            native_display.setPoint(i, j, HIGH);
        }
```

```
        delay(20);
    }
    native_display.clear();
    display.displayScroll("Muhammad Haekal Muhyidin Al-Araby 5024221030
   ", PA_CENTER, PA_SCROLL_LEFT, 20);
    while(!display.displayAnimate());
    display.displayScroll("PING PONG", PA_CENTER, PA_SCROLL_RIGHT, 20);
    while(!display.displayAnimate());
}
```

Listing 7: animation.cpp

Pada bagian fungsi ini array Vector2 yang merepresentasikan posisi masing masing segment pada display yang digunakan untuk menampilkan score. Terdapat juga array dari binary representation untuk segment yang dinyalakan dan dimatikan pada angkat tersebut. Untuk menampilkan skor program akan mengiterasi masing masing segment dan binary reporesentationnya dan menyalakan pixel sesuai dengan nilainya.

```
#pragma once
#include <cstdint>

void RenderScore(uint8_t score, int x);
void RightRenderScore(uint8_t score, int x);
void LeftRenderScore(uint8_t score, int x);
```

Listing 8: score.hpp

```
#include "score.hpp"
#include "HardwareSerial.h"
#include "vector2.hpp"
#include <cstdint>
#include <random>

Vector2 rightPosition[] = {
    Vector2(0, 0),
    Vector2(1, 0),
    Vector2(2, 0),
    Vector2(0, -1),
    Vector2(2, -1),
    Vector2(0, -2),
    Vector2(1, -2),
    Vector2(2, -2),
    Vector2(0, -3),
    Vector2(2, -3),
    Vector2(0, -4),
    Vector2(1, -4),
    Vector2(2, -4),
};
Vector2 rightPosition90[] = {
    Vector2(0, 0),
    Vector2(0, 1),
    Vector2(0, 2),
    Vector2(1, 0),
    Vector2(1, 2),
    Vector2(2, 0),
    Vector2(2, 1),
    Vector2(2, 2),
    Vector2(3, 0),
```

```
    Vector2(3, 2),
    Vector2(4, 0),
    Vector2(4, 1),
    Vector2(4, 2),
};

Vector2 leftPosition[] = {
    Vector2(0, 0),
    Vector2(1, 0),
    Vector2(2, 0),
    Vector2(0, -1),
    Vector2(2, -1),
    Vector2(0, -2),
    Vector2(1, -2),
    Vector2(2, -2),
    Vector2(0, -3),
    Vector2(2, -3),
    Vector2(0, -4),
    Vector2(1, -4),
    Vector2(2, -4),
};

Vector2 leftPosition90[] = {
    Vector2(0, 0),
    Vector2(0, -1),
    Vector2(0, -2),
    Vector2(-1, 0),
    Vector2(-1, -2),
    Vector2(-2, 0),
    Vector2(-2, -1),
    Vector2(-2, -2),
    Vector2(-3, 0),
    Vector2(-3, -2),
    Vector2(-4, 0),
    Vector2(-4, -1),
    Vector2(-4, -2),
};

uint16_t binaryRep[] = {
    0b1111110111111, // 0
    0b1001010010100, // 1
    0b1110111110111, // 2
    0b1111011110111, // 3
    0b1001011111101, // 4
    0b1111011101111, // 5
    0b1111111101111, // 6
    0b1001010010111, // 7
    0b1111111111111, // 8
    0b1111011111111  // 9
};

void RightRenderScore(uint8_t score, int x) {
    Vector2 position = Vector2(x, 1);

    int firstDigit = 0;
    int secondDigit = 0;

    if (score < 10) {
```

```
        firstDigit = score;
    } else {
        firstDigit = score % 10;
        secondDigit = score / 10;
    }

    int index = 0;
    for (const auto& p : rightPosition90) {
        if ((binaryRep[firstDigit] >> index) & 0x1) {
            (position + p).Draw();
        }
        index++;
    }
    position.y -= 4;
    index = 0;
    for (const auto& p : rightPosition90) {
        if ((binaryRep[secondDigit] >> index) & 0x1) {
            (position + p).Draw();
        }
        index++;
    }
}
void LeftRenderScore(uint8_t score, int x) {
    Vector2 position = Vector2(x, 2);

    int firstDigit = 0;
    int secondDigit = 0;

    if (score < 10) {
        secondDigit = score;
    } else {
        firstDigit = score / 10;
        secondDigit = score % 10;
    }

    int index = 0;
    for (const auto& p : leftPosition90) {
        if ((binaryRep[firstDigit] >> index) & 0x1) {
            (position + p).Draw();
        }
        index++;
    }
    position.y -= 4;
    index = 0;
    for (const auto& p : leftPosition90) {
        if ((binaryRep[secondDigit] >> index) & 0x1) {
            (position + p).Draw();
        }
        index++;
    }
}
```

Listing 9: score.cpp

4.3.3 Player Functionality

Pada class Player terdapat fungsi check input dan draw serta constructor class itu sendiri. Pada constructor program akan setup pin yang ada pada parameter sebagai input pull-up.

Selanjutnya pada fungsi Draw program akan mengaktifkan pixel pada vector2 posisi dia dan 1 diatas dan 1 dibawah.

Lalu pada fungsi CheckInput program akan membaca nilai analog dari pin yang terhubung pada potentiometer, pin ditentukan saat inisialisasi class. Posisi player akan dipindahkan berdasarkan nilai tersebut. Posisi player dibatasi agar tidak melebihi layar.

Pada fungsi ini juga akan dicek apakah player sudah melakukan service atau smash dan apakah sudah 0.2 detik bila iya player akan dikembalikan ke posisi semula.

Pada fungsi ini juga dibaca smash input dan dicek apakah sudah melewati cooldown atau tidak. Bila iya maka player akan smash.

```
#pragma  once

#include "vector2.hpp"
#include <cstdint>

class Player {
public:
    bool after_service = false;
    bool service = false;
    Vector2 position;
    Player(uint8_t dirInput, uint8_t smashInput, Vector2 position =
   Vector2());
    void Draw();
    void CheckInput();
    uint16_t score = 0;
    uint8_t power = 1;
    int last = 0;
    int ready = 0;
    uint8_t dirInput;
    uint8_t smashInput;
};
```

Listing 10: player.hpp

```
#include "global.hpp"
#include "esp32-hal-adc.h"
#include "esp32-hal-gpio.h"
#include "esp32-hal.h"
#include "player.hpp"

Player::Player(uint8_t dirInput, uint8_t smashInput, Vector2 position)
    : dirInput(dirInput), smashInput(smashInput), position(position) {
    pinMode(smashInput, INPUT_PULLUP);
}

void Player::Draw() {
    this->position.Draw();
    (this->position + Vector2(0, 1)).Draw();
    (this->position + Vector2(0, -1)).Draw();
}
```

```
void Player::CheckInput() {
    this->position.y = analogRead(this->dirInput) - 4;
    if (this->position.y > 2) this-> position.y = 2;
    else if (this->position.y < -3) this-> position.y = -3;

    if (this->service) {
        b.position.y = this->position.y;
    }

    int current = millis();
    if (current - last >= 200 && (this-> power != 1 || after_service))
   {
        if (this->position.x < 0) this->position.x -= 2;
        if (this->position.x > 0) this->position.x += 2;
        this->power = 1;
        after_service = false;
    }
    if (digitalRead(this->smashInput) && current >= ready) {
        int delta = this->position.x - b.position.x;
        delta /= abs(delta);
        this->last = current;
        if (this->position.x < 0) this->position.x += 2;
        if (this->position.x > 0) this->position.x -= 2;
        this->power = 8;
        this->ready = current + 1000;
        if (service) {
            this->power = 1;
            service = false;
            b.move = Vector2(delta, 0);
            after_service = true;
        }
    }
// Serial.printf("PlayerPos %d: %d %d\n", (this->dirInput == 35) + 1,
   this->position.x, this->position.y);
}
```

Listing 11: player.cpp

4.3.4 Ball Functionality

Pada class Ball terdapat constructor,update, fungsi hit check, dan score check. pada constructor dimana saat class diinisiasi bola akan berapada pada player 1 dan player 1 dapat melakukan service.

Pada fungsi Update program akan memeriksa apakah sudah melewati periode update belum. Periode update ditentukan dengan membagi 75 / update rate. dimana pada kondisi normal update rate adalah 1 yang berarti bola akan diupdate per 75ms. Semakin cepat update rate maka hal ini akan memengaruhi kecepatan bola. Fungsi ini akan memanggil HitCheck ScoreCheck secara berurutan dan menambahkan move vector pada vector position untuk mengupdate posisi bola.

Pada fungsi HitCheck program akan mengecek apakah bola terkena pembatas atau player. Bila terkena pembatas hanya akan membalikkan nilai y dari move vector. Sedangkan bila terkena Player bola akan memantuk tergantung dari dimana bola tersebut mengenainya.

Pada fungsi ScoreCheck program akan mengecek apakah bola masuk ke daerah player atau tidak dan memberikan skor pada player yang berhasil memasukkan.

```
#pragma once

#include "vector2.hpp"
class Ball {
public:
    Vector2 position;
    Vector2 move;
    Ball(Vector2 position = Vector2());
    void Draw();
    void Update();
    void HitCheck();
    void ScoreCheck();
    int lastUpdate = 0;
    int updateRate = 1;
private:
};
```

Listing 12: ball.hpp

```
#include "ball.hpp"
#include "MD_Parola.h"
#include "global.hpp"
#include "vector2.hpp"
#include "math.h"
#include <cstdlib>
#include "device.hpp"

Ball::Ball(Vector2 position) : position(position) {
    this->position = Vector2(-6, p1.position.y);
    this->move = Vector2(0,0);
    this->updateRate = 1;
}

void Ball::HitCheck() {
    // Serial.printf("P1 X:%d Ball X:%d\n", p1.position.x, this->
   position.x);
    bool ballHitTheWall = this->position.y >= 3 || this->position.y <=
   -4;

    int deltaP1 = this->position.y - p1.position.y;
    int deltaP2 = this->position.y - p2.position.y;

    if (ballHitTheWall) {
        this->move.y *= -1;
    } else if (this->position.x == p1.position.x && abs(deltaP1) < 2 )
   {
        this->move.x *= -1;
        this->move.y = p1.power != 1 ? 0 : deltaP1;
        this->updateRate = p1.power;
    } else if (this->position.x == p2.position.x && abs(deltaP2) < 2) {
        this->move.x *= -1;
        this->move.y = p2.power != 1 ? 0 : deltaP2;
        this->updateRate = p2.power;
    }
}
```

```
void Ball::ScoreCheck() {
    if (this->position.x < -8) {
        display.displayText("SCORED", PA_CENTER, 10, 500,
   PA_SCROLL_LEFT, PA_SCROLL_LEFT);
        mp.Switch(&tetris);
        while(!display.displayAnimate()) {
            mp.Play();
        }
        p2.score++;
        this->position = Vector2(-6, p1.position.y);
        this->move = Vector2(0,0);
        p1.service = true;
        this->updateRate = 1;
        mp.Switch(&mario_underworld);
    } else if (this->position.x > 7) {
        display.displayText("SCORED", PA_CENTER, 10, 500,
   PA_SCROLL_RIGHT, PA_SCROLL_RIGHT);
        mp.Switch(&tetris);
        while(!display.displayAnimate()) {
            mp.Play();
        }
        p1.score++;
        this->position = Vector2(5, p2.position.y);
        this->move = Vector2(0,0);
        p2.service = true;
        this->updateRate = 1;
        mp.Switch(&mario_underworld);
    }

}

void Ball::Update() {
    int current = millis();
    if (current - this->lastUpdate < 75 / updateRate) {
        return;
    }
    this->lastUpdate = current;
    HitCheck();
    ScoreCheck();
    this->position = this->position + this->move;
}

void Ball::Draw() {
    this->position.Draw();
}
```

Listing 13: ball.cpp

4.3.5 Music Functionality

Fungsionalitas utama lagu dikendalikan oleh class MusicPlayer dimana pada MusicPlayer terdapat beberapa fungsi Switch, Start, dan Play lalu constructor dari kelas itu sendiri. Pada saat class diinisiasi akan attach ledc ke pin 32. Hal ini untuk menginsialisasi pin 32 untuk PWM. resolusi yang digunakan adalah 12 bit.

Fungsi Switch digunakan untuk mengganti lagu yang sedang dimainkan dengan meng-

ganti variabel song. Lalu memanggil fungsi start untuk memulai lagu. Pada fungsi start index akan menjadi 0 dan pin akan mengoutputkan notes pertama dari lagu.

Fungsi Play akan memainkan lagu dengan cara mengecek apakah sudah melewati durasi notes atau tidak. durasi tiap notes ditentukan dengan membagi 1000 dengan absolut tempo dari lagu. Bila tempo lagu adalah negatif dimana merepresentasikan tempo titik. Maka durasi akan dikali 1.5. Ketika sudah melewati durasi maka akan memainkan note selanjutnya.

```
#pragma once

#include "HardwareSerial.h"
#include "esp32-hal-ledc.h"
#include "esp32-hal.h"
#include <strings.h>
#include <sys/types.h>
#include "song.hpp"



class MusicPlayer {
public:
    MusicPlayer() {
        ledcAttachPin(32, 0);
        ledcSetup(0, 2048, 12);
    }
    void Switch(Song* song) {
        this->song.tempo = song->tempo;
        this->song.melody = song->melody;
        this->song.size = song->size;
        index = 0;
        this->Start();
    }
    void Start() {
        if (song.size <= 0 || song.melody == nullptr || song.tempo ==
   nullptr) return;
        ledcWriteTone(0, song.melody[0]);
        this->index = 0;
        this->last = millis();
    }
    void Play() {
        if (song.size <= 0 || song.melody == nullptr || song.tempo ==
   nullptr) return;
        int current = millis();
        int duration = 1000 / abs(song.tempo[index]);
        if (tetris_tempo[index] < 0) {
            duration *= 1.5;
        }
        if (current - this->last < duration) return;
        last = current;
        this->index++;
        if (index > song.size - 1) index = 0;
        ledcWriteTone(0, song.melody[this->index]);
    }
private:
    int index = 0;
    int last;
```

```
    Song song = Song(nullptr, nullptr, 0);
};
```

Listing 14: music.hpp

Bagian ini berisi lagu lagu yang dapat dimainkan.

```
#pragma once

#include "note.hpp"

class Song {
public:
    int size;
    Song(int *melody, int *tempo, int size):melody(melody), tempo(tempo
   ), size(size) {}
    int *melody;
    int *tempo;
};

static int mario_melody[] = {
    NOTE_E7, NOTE_E7, 0, NOTE_E7,
    0, NOTE_C7, NOTE_E7, 0,
    NOTE_G7, 0, 0,  0,
    NOTE_G6, 0, 0, 0,

    NOTE_C7, 0, 0, NOTE_G6,
    0, 0, NOTE_E6, 0,
    0, NOTE_A6, 0, NOTE_B6,
    0, NOTE_AS6, NOTE_A6, 0,

    NOTE_G6, NOTE_E7, NOTE_G7,
    NOTE_A7, 0, NOTE_F7, NOTE_G7,
    0, NOTE_E7, 0, NOTE_C7,
    NOTE_D7, NOTE_B6, 0, 0,

    NOTE_C7, 0, 0, NOTE_G6,
    0, 0, NOTE_E6, 0,
    0, NOTE_A6, 0, NOTE_B6,
    0, NOTE_AS6, NOTE_A6, 0,

    NOTE_G6, NOTE_E7, NOTE_G7,
    NOTE_A7, 0, NOTE_F7, NOTE_G7,
    0, NOTE_E7, 0, NOTE_C7,
    NOTE_D7, NOTE_B6, 0, 0
};

static int mario_tempo[] = {
    12, 12, 12, 12,
    12, 12, 12, 12,
    12, 12, 12, 12,
    12, 12, 12, 12,

    12, 12, 12, 12,
    12, 12, 12, 12,
    12, 12, 12, 12,
    12, 12, 12, 12,

    9, 9, 9,
```

```
    12, 12, 12, 12,
    12, 12, 12, 12,
    12, 12, 12, 12,

    12, 12, 12, 12,
    12, 12, 12, 12,
    12, 12, 12, 12,
    12, 12, 12, 12,

    9, 9, 9,
    12, 12, 12, 12,
    12, 12, 12, 12,
    12, 12, 12, 12,
};

static int underworld_melody[] = {
    NOTE_C4, NOTE_C5, NOTE_A3, NOTE_A4,
    NOTE_AS3, NOTE_AS4, 0,
    0,
    NOTE_C4, NOTE_C5, NOTE_A3, NOTE_A4,
    NOTE_AS3, NOTE_AS4, 0,
    0,
    NOTE_F3, NOTE_F4, NOTE_D3, NOTE_D4,
    NOTE_DS3, NOTE_DS4, 0,
    0,
    NOTE_F3, NOTE_F4, NOTE_D3, NOTE_D4,
    NOTE_DS3, NOTE_DS4, 0,
    0, NOTE_DS4, NOTE_CS4, NOTE_D4,
    NOTE_CS4, NOTE_DS4,
    NOTE_DS4, NOTE_GS3,
    NOTE_G3, NOTE_CS4,
    NOTE_C4, NOTE_FS4, NOTE_F4, NOTE_E3, NOTE_AS4, NOTE_A4,
    NOTE_GS4, NOTE_DS4, NOTE_B3,
    NOTE_AS3, NOTE_A3, NOTE_GS3,
    0, 0, 0
};

//Underwolrd tempo
static int underworld_tempo[] = {
    12, 12, 12, 12,
    12, 12, 6,
    3,
    12, 12, 12, 12,
    12, 12, 6,
    3,
    12, 12, 12, 12,
    12, 12, 6,
    3,
    12, 12, 12, 12,
    12, 12, 6,
    6, 18, 18, 18,
    6, 6,
    6, 6,
    6, 6,
    18, 18, 18, 18, 18, 18,
    10, 10, 10,
    10, 10, 10,
    3, 3, 3
```

```
};

static int tetris_melody[] = {
    //Based on the arrangement at https://www.flutetunes.com/tunes.php?
   id=192
    NOTE_E5, NOTE_B4, NOTE_C5, NOTE_D5, NOTE_C5, NOTE_B4,
    NOTE_A4, NOTE_A4, NOTE_C5, NOTE_E5, NOTE_D5, NOTE_C5,
    NOTE_B4,  NOTE_C5, NOTE_D5, NOTE_E5,
    NOTE_C5, NOTE_A4, NOTE_A4, NOTE_A4, NOTE_B4, NOTE_C5,

    NOTE_D5,  NOTE_F5, NOTE_A5, NOTE_G5, NOTE_F5,
    NOTE_E5,  NOTE_C5, NOTE_E5, NOTE_D5, NOTE_C5,
    NOTE_B4, NOTE_B4, NOTE_C5, NOTE_D5, NOTE_E5,
    NOTE_C5, NOTE_A4, NOTE_A4,REST,

    NOTE_E5, NOTE_B4, NOTE_C5, NOTE_D5, NOTE_C5, NOTE_B4,
    NOTE_A4, NOTE_A4, NOTE_C5, NOTE_E5, NOTE_D5, NOTE_C5,
    NOTE_B4,  NOTE_C5, NOTE_D5, NOTE_E5,
    NOTE_C5, NOTE_A4, NOTE_A4, NOTE_A4, NOTE_B4, NOTE_C5,

    NOTE_D5,  NOTE_F5, NOTE_A5, NOTE_G5, NOTE_F5,
    NOTE_E5,  NOTE_C5, NOTE_E5, NOTE_D5, NOTE_C5,
    NOTE_B4, NOTE_B4, NOTE_C5, NOTE_D5, NOTE_E5,
    NOTE_C5, NOTE_A4, NOTE_A4,REST,

    NOTE_E5, NOTE_C5,
    NOTE_D5,  NOTE_B4,
    NOTE_C5,  NOTE_A4,
    NOTE_GS4, NOTE_B4, REST,
    NOTE_E5,  NOTE_C5,
    NOTE_D5,  NOTE_B4,
    NOTE_C5,  NOTE_E5, NOTE_A5,
    NOTE_GS5,
};

static int tetris_tempo[] = {
    //Based on the arrangement at https://www.flutetunes.com/tunes.php?
   id=192
    4,8,8,4,8,8,4,8,8,4,8,8
    , -4,8,4,4,
    4,4,8,4,8,8

    , -4,8,4,8,8,
    -4,8,4,8,8,
    4,8,8,4,4,
    4,4,4, 4,

    4,8,8,4,8,8,
    4,8,8,4,8,8,
    -4,8,4,4,
    4,4,8,4,8,8,

    -4,8,4,8,8,
    -4,8,4,8,8,
    4,8,8,4,4,
    4,4,4, 4,

    2,2,
```

```
    2,2,
    2,2,
    2,4,8,
    2,2,
    2,2,
    4,4,2,
    2,
};
```

Listing 15: song.hpp

Bagian ini berisi nada yang dapat dioutputkan

```
#pragma once

#define NOTE_B0  31
#define NOTE_C1  33
#define NOTE_CS1 35
#define NOTE_D1  37
#define NOTE_DS1 39
#define NOTE_E1  41
#define NOTE_F1  44
#define NOTE_FS1 46
#define NOTE_G1  49
#define NOTE_GS1 52
#define NOTE_A1  55
#define NOTE_AS1 58
#define NOTE_B1  62
#define NOTE_C2  65
#define NOTE_CS2 69
#define NOTE_D2  73
#define NOTE_DS2 78
#define NOTE_E2  82
#define NOTE_F2  87
#define NOTE_FS2 93
#define NOTE_G2  98
#define NOTE_GS2 104
#define NOTE_A2  110
#define NOTE_AS2 117
#define NOTE_B2  123
#define NOTE_C3  131
#define NOTE_CS3 139
#define NOTE_D3  147
#define NOTE_DS3 156
#define NOTE_E3  165
#define NOTE_F3  175
#define NOTE_FS3 185
#define NOTE_G3  196
#define NOTE_GS3 208
#define NOTE_A3  220
#define NOTE_AS3 233
#define NOTE_B3  247
#define NOTE_C4  262
#define NOTE_CS4 277
#define NOTE_D4  294
#define NOTE_DS4 311
#define NOTE_E4  330
#define NOTE_F4  349
#define NOTE_FS4 370
```

```
#define NOTE_G4  392
#define NOTE_GS4 415
#define NOTE_A4  440
#define NOTE_AS4 466
#define NOTE_B4  494
#define NOTE_C5  523
#define NOTE_CS5 554
#define NOTE_D5  587
#define NOTE_DS5 622
#define NOTE_E5  659
#define NOTE_F5  698
#define NOTE_FS5 740
#define NOTE_G5  784
#define NOTE_GS5 831
#define NOTE_A5  880
#define NOTE_AS5 932
#define NOTE_B5  988
#define NOTE_C6  1047
#define NOTE_CS6 1109
#define NOTE_D6  1175
#define NOTE_DS6 1245
#define NOTE_E6  1319
#define NOTE_F6  1397
#define NOTE_FS6 1480
#define NOTE_G6  1568
#define NOTE_GS6 1661
#define NOTE_A6  1760
#define NOTE_AS6 1865
#define NOTE_B6  1976
#define NOTE_C7  2093
#define NOTE_CS7 2217
#define NOTE_D7  2349
#define NOTE_DS7 2489
#define NOTE_E7  2637
#define NOTE_F7  2794
#define NOTE_FS7 2960
#define NOTE_G7  3136
#define NOTE_GS7 3322
#define NOTE_A7  3520
#define NOTE_AS7 3729
#define NOTE_B7  3951
#define NOTE_C8  4186
#define NOTE_CS8 4435
#define NOTE_D8  4699
#define NOTE_DS8 4978
#define REST 0
```

Listing 16: song.hpp

4.3.6 Global Variable

```
#pragma once

#include "ball.hpp"
#include "buzzer.hpp"
#include "player.hpp"
```

```
#include <cstdint>
#include "music.hpp"

extern Ball b;
extern Player p1;
extern Player p2;
static uint8_t score = 0;
extern Buzzer buzzer;
extern MusicPlayer mp;

extern Song mario;
extern Song mario_underworld;
extern Song tetris;
```

Listing 17: global.hpp

```
#include "global.hpp"
#include "music.hpp"

Ball b = Ball(Vector2(0, 0));
Player p1 = Player(33, 19, Vector2(-8, 0));
Player p2 = Player(35, 22, Vector2(7, 0));
MusicPlayer mp = MusicPlayer();

Song mario = Song(mario_melody, mario_tempo, 78);
Song mario_underworld = Song(mario_melody, mario_tempo, 78);
Song tetris = Song(tetris_melody, tetris_tempo, 99);
```

Listing 18: global.cpp

4.4 Hasil

Figure 3: Hasil tanpa package

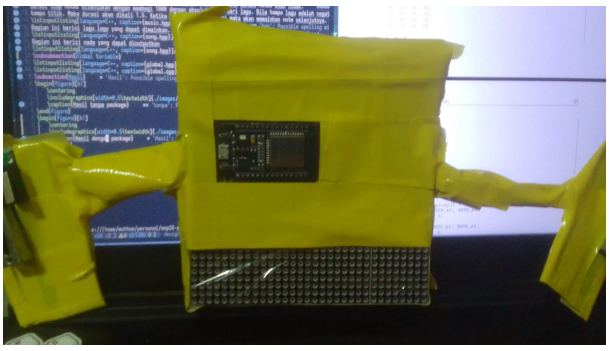

Figure 4: Hasil dengan package