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

3 Desain Sistem

3.1 Rangkaian Skematik

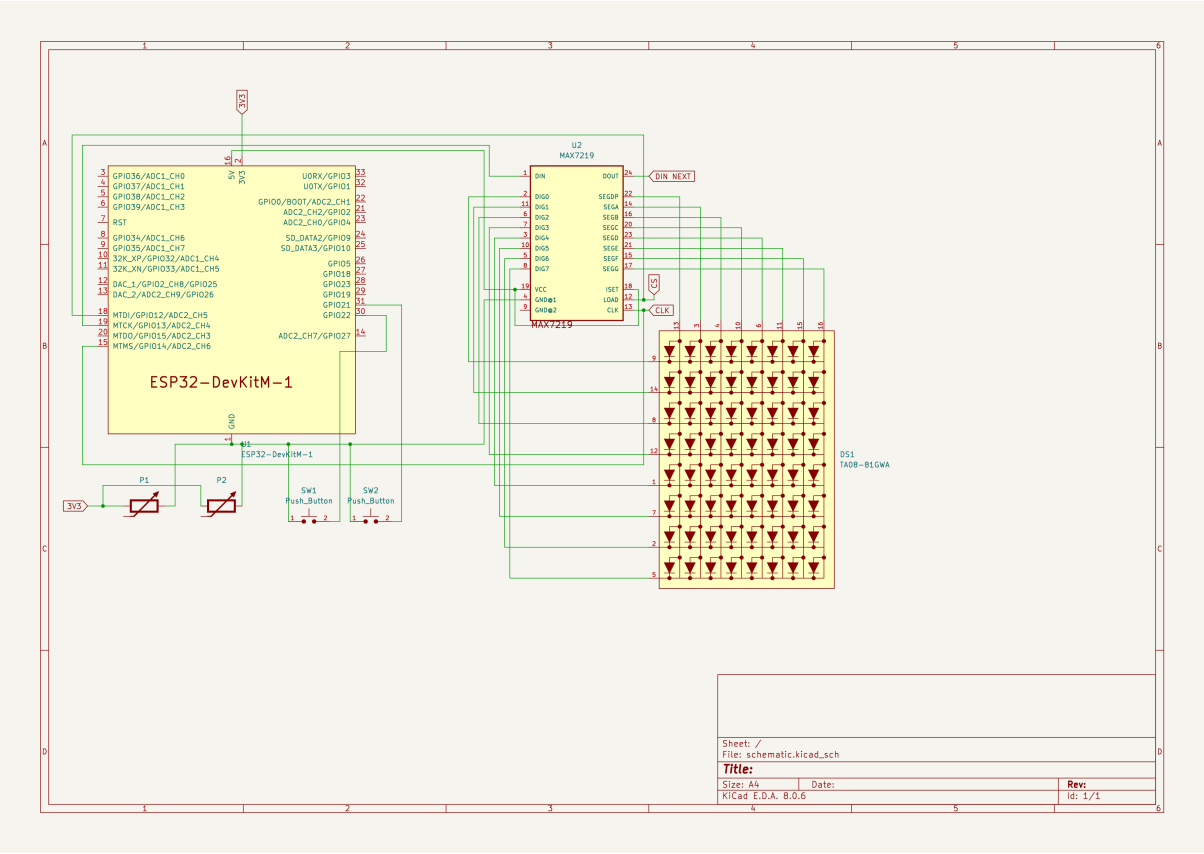

Figure 1: Schematic

Rangkaian menggunakan ESP32 yang terhubung pada IC MAX7219 dimana CS terhubung pada pin D12, DIN pada D13 dan clock pada D14 terdapat 4 IC MAX7219 yang saling terhubung secara seri. Lalu push button dihubungkan dengan ground dan GPIO dan potentiometer dihubungkan ke ground, vcc, dan GPIO.

3.2 Desain Game

1. Game memiliki 2 player yang dapat digerakkan dengan slide potentiometer
2. Bola akan memantul saat menabrak pembatas ataupun player
3. Bola memantul secara diagonal dengan kecepatan konstan
4. Bila player smash dan mengenai bola, bola akan bergerak lurus dan lebih cepat

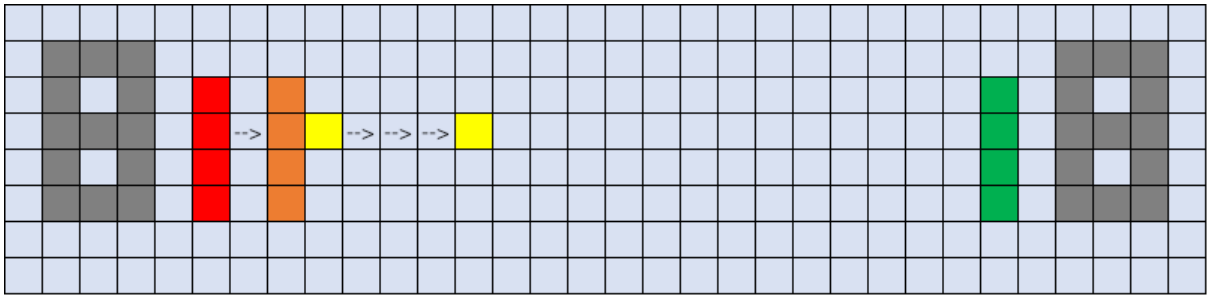

Figure 2: Desain Game

5. Bila bola melewati player maka player lainnya akan mendapatkan skor
6. Player yang kebobolan akan mendapatkan bola dan dapat menggerakkannya dan menekan tombol smash untuk service.
7. Game akan berakhir bila salah satu pemain mencapai skor 11 dan selisih 2 poin.
8. Game akan berakhir imbang ketika kedua pemain mencapai 15 poin

3.3 Program

3.3.1 Menampilkan

Untuk menampilkan ke dotmatrix dapat digunakan 2 library MD_Parola dan MD_MAX72xx. Dimana MD_Parola dapat menampilkan huruf namun kurang dalam kontrol sehingga akan kesulitan untuk menampilkan Game. Sehingga MD_Parola hanya akan digunakan saat menampilkan string. Sedangkan saat ingin menampilkan karakter akan menggunakan MD_MAX72xx yang memberikan raw control pada dotmatrix display. Untuk itu akan dibuat sebuah fungsi yang menerima input berbentuk array 8*32 untuk ditampilkan pada dotmatrix. Berikutnya akan disebut frame. Lalu tiap objek akan diupdate dengan diletakan pada frame tersebut. Sesuai dengan posisi.

3.3.2 Input

Input didapatkan dengan mengambil nilai analogRead pada potentiometer dan digitalRead pada button smash.

3.4 Source Code

Berisi program utama yang mengatur main loop.

```
#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 "score.hpp"


void setup() {
    Setup();
    Intro();
    analogReadResolution(3);
    display.displayScroll("PRESS ANY BUTTON TO START...", PA_CENTER,
   PA_SCROLL_LEFT, 100);
    while (!digitalRead(p1.smashInput) && !digitalRead(p2.smashInput))
   {
        if (display.displayAnimate()) {
            display.displayReset();
        }
    }
}

int last = 0;

void loop() {
    int current = millis();

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

    if (current - last < 50) {
        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);
        native_display.clear();
        display.displayText("P1 WIN", PA_CENTER, 50, 1000,
   PA_SCROLL_RIGHT, PA_SCROLL_RIGHT);
        while (!digitalRead(p1.smashInput)&& !digitalRead(p2.smashInput
   )) {
            if (display.displayAnimate()) {
                display.displayReset();
            }
        }
        p1.score = 0;
        p2.score = 0;
    }
```

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

Listing 1: main.cpp

Program di bawah mengendalikan device dotmatrix menggunakan library Parola dan MDMAX

```
#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

Program di bawah menampilkan untuk menampilkan Intro

```
#pragma once


void Intro();
```

Listing 4: 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 5: animation.cpp

Program di bawah Berisi class vector2 untuk memudahkan posisi game object

```
#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 6: 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 7: vector2.cpp

Program di bawah berfungsi mendapatkan input player dan mengupdate state player

```
#pragma  once

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

class Player {
public:
    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 8: 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) {
        if (this->position.x < 0) this->position.x -= 2;
        if (this->position.x > 0) this->position.x += 2;
        this->power = 1;
    }
    if (digitalRead(this->smashInput) && current >= ready) {
        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) {
            service = false;
            b.move = Vector2(b.position.x - this->position.x, 0);
        }
    }
// Serial.printf("PlayerPos %d: %d %d\n", (this->dirInput == 35) + 1,
   this->position.x, this->position.y);
}
```

Listing 9: player.cpp

Program di bawah untuk mengupdate state bola

```
#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 10: 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) {
    move = Vector2(-1, 0);
}

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) {
        buzzer.Buzz();
        display.displayText("SCORED", PA_CENTER, 10, 500,
   PA_SCROLL_LEFT, PA_SCROLL_LEFT);
        while(!display.displayAnimate());
        p2.score++;
        this->position = Vector2(-5, p1.position.y);
        this->move = Vector2(0,0);
        p1.service = true;
        this->updateRate = 1;
    } else if (this->position.x > 7) {
        buzzer.Buzz();
        display.displayText("SCORED", PA_CENTER, 10, 500,
   PA_SCROLL_RIGHT, PA_SCROLL_RIGHT);
        while(!display.displayAnimate());
        p1.score++;
        this->position = Vector2(4, p2.position.y);
        this->move = Vector2(0,0);
```

```
        p2.service = true;
        this->updateRate = 1;
    }

}

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 11: ball.cpp

Program di bawah untuk menyalakan buzzer

```
#pragma once

#include <cstdint>
class Buzzer {
public:
    Buzzer(uint8_t pin);
    void Buzz();
private:
    uint8_t pin;
};
```

Listing 12: buzzer.hpp

```
#include "buzzer.hpp"
#include "esp32-hal-gpio.h"
#include "esp32-hal.h"
#include <cstdint>

Buzzer::Buzzer(uint8_t pin):pin(pin) {
    pinMode(pin, OUTPUT);
}

void Buzzer::Buzz() {
    digitalWrite(this->pin, HIGH);
    delay(100);
    digitalWrite(this->pin, LOW);
}
```

Listing 13: buzzer.cpp

Program di bawah untuk memunculkan skor

```
#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 14: 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, 0);

    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 15: score.cpp

Program di bawah berisi Global Variabel yang digunakan seluruh program

```
#pragma once

#include "ball.hpp"
#include "buzzer.hpp"
#include "player.hpp"
#include <cstdint>

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

Listing 16: global.hpp

```
#include "global.hpp"
#include "buzzer.hpp"

Ball b = Ball(Vector2(0, 0));
Player p1 = Player(33, 19, Vector2(-8, 0));
Player p2 = Player(35, 22, Vector2(7, 0));
Buzzer buzzer = Buzzer(32);
```

Listing 17: global.cpp

3.5 Hasil

Figure 3: Hasil tanpa package