Implementation of Flappy Bird using ESP8266

Sanchit Sharma
Dept. of Electronics and Communication Engineering
Nirma University, Ahmedabad, India
20bec108@nirmauni.ac.in

Siddharth Agarwal
Dept. of Electronics and Communication Engineering
Nirma University, Ahmedabad, India
20bec119@nirmauni.ac.in

***Abstract*—This paper presents the development of a gaming console using the ESP8266 microcontroller. The console is designed to provide a low-cost and accessible option for gamers who want to play retro games on a budget. The ESP8266 microcontroller, which is popular among makers and hobbyists, is utilized as the core component for the console due to its small size, low power consumption, and built-in Wi-Fi connectivity. The console's hardware, software, and user interface are discussed in detail, along with the challenges faced during development and the solutions implemented to overcome them. The final result is a functional gaming console that can run classic games, is easy to assemble and customize, and offers a fun and engaging gaming experience for users.**

I. Introduction

Video games have been a popular form of entertainment for decades, and retro gaming has experienced a resurgence in recent years. While there are many options available for playing classic games, such as emulators and dedicated retro gaming consoles, these can be costly and inaccessible for some gamers. To address this issue, we propose the development of a low-cost, DIY gaming console using the ESP8266 microcontroller. The ESP8266 is a popular microcontroller among makers and hobbyists due to its low cost, small size, low power consumption, and built-in Wi-Fi connectivity. While it was originally designed for Internet of Things (IoT) applications, its versatility and affordability make it a suitable choice for developing a gaming console. By utilizing the ESP8266 as the core component of the console, we can create a budget-friendly and customizable option for retro gaming enthusiasts. In this paper, we present the development of a gaming console using the ESP8266 microcontroller. We discuss the hardware and software components of the console, as well as the user interface and gameplay experience. We also highlight the challenges we faced during development and the solutions we implemented to overcome them. Our goal is to provide a comprehensive guide for hobbyists and makers who are interested in building their own gaming console using the ESP8266 microcontroller.

II. Background

Flappy Bird is a mobile game developed in 2013 by Dong Nguyen, a Vietnamese video game developer, using the Corona SDK game development software. Initially released on May 24, 2013, for iOS, and subsequently for Android devices on January 30, 2014, Flappy Bird's objective is to navigate a bird through a series of pipes without hitting them, by tapping the screen to flap the bird's wings and keep it aloft. The game's design is heavily inspired by the classic 8-bit graphics of the Nintendo Entertainment System (NES) era.

Despite its relatively simple gameplay and graphics, Flappy Bird gained significant popularity, becoming a chart-topping mobile game in several countries. Nevertheless, the game drew criticism for its high level of difficulty and addictiveness. In February 2014, Dong Nguyen announced that he would be removing Flappy Bird from the App Store and Google Play due to the stress and attention it was bringing him.

Flappy Bird's success and subsequent removal from app stores in 2014 has become a notable case study in the mobile gaming industry. Its sudden rise to fame, controversial gameplay mechanics, and the emotional impact on its creator have all contributed to the game's lasting legacy in popular culture.

III. Components used

Name of the Component	Specification	Quantity
ESP 8266	Node MCU	1
Display	OLED I2C	1
Joystick	Dual Axis, Thumb	1
Push Button	Mini	2

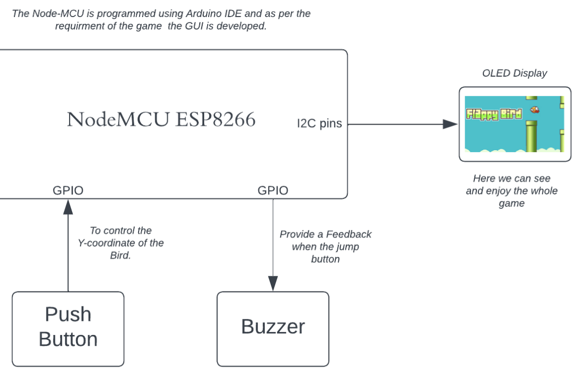
The Node-MCU is programmed using Arduino IDE and as per the requirment of the game the GUI is developed.
NodeMCU ESP8266
I2C pins
OLED Display
Flappy Bird
Here we can see and enjoy the whole game
GPIO
GPIO
To control the Y-coordinate of the Bird.
Provide a Feedback when the jump button
Push Button
Buzzer

Fig. 1. Block Diagram

IV. Components description

The entire circuit is complex,the block diagram makes the circuit easy to understand.The function of each block is discussed below:

A. ESP8266 Node-MCU Board

The ESP8266 Node-MCU is a low-cost, Wi-Fi-enabled microcontroller board that is widely used in Internet of Things (IoT) applications. It features an ESP8266EX microcontroller unit (MCU) with integrated Wi-Fi, a USB-to-serial interface, and a power regulator.

The ESP8266 Node-MCU board is popular among hobbyists and professionals alike due to its ease of use, low cost, and broad compatibility with various programming environments. It can be programmed using the Arduino IDE or Lua programming language, and is compatible with a variety of software development tools and libraries.

One of the primary features of the ESP8266 Node-MCU board is its built-in Wi-Fi capability, which allows it to connect to the internet and communicate with other devices over a wireless network. This feature makes it ideal for IoT projects that require wireless communication, such as home automation systems, sensor networks, and smart devices.

In addition to its Wi-Fi capabilities, the ESP8266 Node-MCU board also features a variety of input and output (I/O) pins that can be used to connect to other electronic components and peripherals. This allows it to interface with a wide range of sensors, actuators, and other devices, making it a versatile platform for building custom electronic systems.

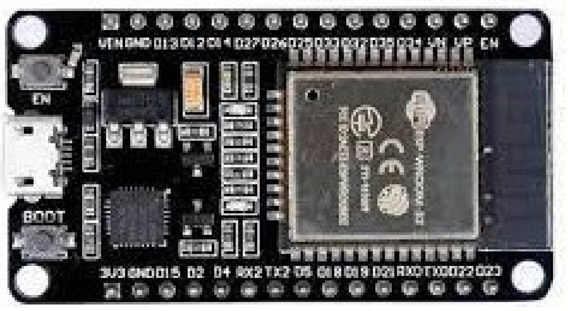

Fig. 2. ESP8266 Node-MCU

B. OLED Display

The 0.96 inch OLED (Organic Light Emitting Diode) display is a compact, low-power display that is commonly used in electronic projects and devices. The display uses organic materials to emit light when an electric current is applied, resulting in high contrast and bright, vivid colors.

One of the key advantages of OLED displays over other types of displays is their low power consumption, which makes them well-suited for battery-powered devices. The 0.96 inch OLED display in particular is popular among hobbyists and professionals alike due to its small size and ease of use.

The display features a resolution of 128 x 64 pixels, which allows it to display text, graphics, and other visual elements with good clarity and detail. It also includes built-in driver circuitry, which simplifies the process of interfacing with other electronic components and microcontrollers.

The 0.96 inch OLED display can be used in a variety of applications, such as wearable devices, smart home systems, and other IoT projects. Its small size and low power consumption make it ideal for projects where space and power are at a premium.

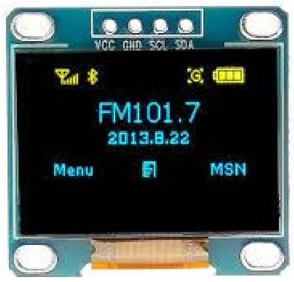

Fig. 3. 0.96 inch OLED Display

C. Buzzer

A buzzer is an electronic component that produces an audible sound when an electric current is passed through it. It typically consists of a small electromechanical device that contains a coil of wire and a metal diaphragm or plate. When an electric current is passed through the coil, it generates a magnetic field that causes the diaphragm or plate to vibrate, producing sound waves that are audible to the human ear.

Buzzer components come in a variety of shapes and sizes, from small piezo buzzers that can fit on a breadboard to larger mechanical buzzers used in industrial applications. Some buzzers are designed to produce a steady tone, while others can be programmed to produce different patterns of sound, such as beeps or alarms.

Buzzer components are commonly used in a variety of electronic devices, such as alarm clocks, smoke detectors, and home security systems. They are also frequently used in hobbyist and DIY electronics projects, such as those involving microcontrollers like the Arduino.

Fig. 4. Buzzer

D. Push Button

A push button is a simple electronic switch that is used to make or break an electrical circuit. It consists of a button or plunger that, when pushed, completes a connection between two electrical contacts. When the button is released, the connection is broken, interrupting the flow of electricity through the circuit.

Push buttons come in a variety of shapes, sizes, and designs, including momentary and latching types. Momentary push buttons are normally open, which means that the circuit is open until the button is pressed, completing the circuit temporarily. Latching push buttons, on the other hand, are normally closed, and when the button is pressed, it remains closed until it is pressed again to open the circuit.

Push buttons are commonly used in electronic devices and circuits for a variety of purposes, such as turning on and off devices, selecting options or modes, and triggering actions. They can be found in devices ranging from simple toys to complex machinery and control systems.

Fig. 5. Push Button

V. Program

```
#include <Wire.h>
#include "SSD1306Wire.h"

#include "images.h"
#include "fonts.h"

SSD1306Wire display(0x3c, 4, 5);

#define DEMO_DURATION 3000
typedef void (*Demo)(void);

float obstacle_idx[4];
int gap_height[4];
int spacing=32;
int width=30;
void setup() {

   Serial.println();
   Serial.println();

   pinMode(2,OUTPUT);
   pinMode(12,OUTPUT);
   pinMode(14,INPUT_PULLUP);
   display.init();

   for(int i=0;i<4;i++)
   {
      obstacle_idx[i]=128+((i+1)*spacing);
      {gap_height[i]=random(8,32);}
   }

   display.flipScreenVertically();
   display.setFont(ArialMT_Plain_10);

   }


   int score=0;
   int stis=0;
   float fx=30.00;
   float fy=22.00;
   int player_moving=0;
   unsigned long moving_time=0;

   int game_mode=0;
   int frame=0;
   int buzzer=0;
   unsigned long buzzer_time=0;


void loop() {
  Serial.begin(300);
  int tst;
  display.clear();

  if(game_mode==0)
  {
   display.setFont(ArialMT_Plain_16);
   display.drawString(0,4,"Flappy ");
  display.drawXbm(0, 0, 128, 64, background);
  display.drawXbm(20, 32, 14, 9, bird);

  display.setFont(ArialMT_Plain_10);
  display.drawString(0,44,"press to start");
   if(digitalRead(14)==0)
   game_mode=1;
   }

  if(game_mode==1)
  {
  display.setFont(ArialMT_Plain_10);
  display.drawString(3,0,String(score));

  if(digitalRead(14)==0)
  {
   if(stis==0)
     {
      moving_time=millis();
      player_moving=1;
      buzzer=1;
      stis=1;
      buzzer_time=millis();

      }

   }else{stis=0;}


   for(int j=0;j<4;j++)
   {
      display.setColor(WHITE);
      display.fillRect(obstacle_idx[j],0,6,64);
      display.setColor(BLACK);
      display.fillRect(obstacle_idx[j],gap_height[j]
      ,6,width);
   }

    display.setColor(WHITE);
    display.drawXbm(fx, fy, 14, 9, bird);

   for(int j=0;j<4;j++)
   {
      obstacle_idx[j]=obstacle_idx[j]-0.01;
      if(obstacle_idx[j]<-7){
      score=score+1;
      digitalWrite(12,1);
```

```
        gap_height[j]=random(8,32);
        obstacle_idx[j]=128;
    }

    }
     if((moving_time+185)<millis())
        player_moving=0;

     if((buzzer_time+40)<millis())
        buzzer=0;

    if(player_moving==0)
     fy=fy+0.01;
    else
     fy=fy-0.03;


     if(buzzer==1)
     digitalWrite(12,1);
     else
     digitalWrite(12,0);

    if(fy>63 || fy<0){
    game_mode=0;
    fy=22;
    score=0;
    digitalWrite(12,1);
    delay(500);
    digitalWrite(12,0);

    for(int i=0;i<4;i++)//generates rectangle
      {
        obstacle_idx[i]=128+((i+1)*spacing);
        {gap_height[i]=random(4,30);}
        }
    }

    for(int m=0;m<4;m++)
    if(obstacle_idx[m]<=fx+7 &&
         fx+7<=obstacle_idx[m]+6)
    {
      if(fy<gap_height[m] ||
           fy+8>gap_height[m]+width){
      game_mode=0;
      fy=22;
      score=0;
      digitalWrite(12,1);
      delay(500);
      digitalWrite(12,0);

      for(int i=0;i<4;i++)
      {
        obstacle_idx[i]=128+((i+1)*spacing);
        {gap_height[i]=random(8,32);}
        }
      }}
       display.drawRect(0,0,128,64);
       }
       display.display();
}
```

VI. Program Description

The program is a simplified version of the popular Flappy Bird game, designed to run on an Arduino device with an OLED display. The game consists of a bird that needs to navigate through a series of obstacles in order to score points.

The program is written in Arduino C language and makes use of several built-in functions and libraries. The images for the bird, buildings, and background are provided in the form of bitmaps, which are stored in header files and loaded into the program using the ”drawXbm” function provided by the SSD1306Wire library.

The obstacles in the game are represented by rectangles, whose size and position are randomly generated using the ”random” function provided by the Arduino IDE. The obstacles are shifted to the left every iteration to create a moving effect, and when they reach the left edge of the display, they are regenerated with a new size and position.

The bird's movement is controlled by a push button, which, when pressed, causes the bird to move up with an offset of 0.01, and when released, causes it to move down with an offset of 0.03. The bird's position is represented by its x and y coordinates, which are updated every iteration based on the current button state.

The program also keeps track of the player's score, which is incremented every time the bird successfully navigates through an obstacle. If the bird collides with an obstacle or goes out of bounds, the game ends, and the score is reset to zero.

Additionally, the program uses an output pin to control a buzzer, which produces a sound effect when the bird collides with an obstacle or when the button is pressed. The buzzer is turned on and off based on a timer, which ensures that the sound effect is played for a short duration.

Overall, this program demonstrates how to use basic Arduino functions and libraries to create a simple game that can be played on an OLED display. While it may not be as advanced as the original Flappy Bird game, it provides a fun and engaging experience for anyone interested in learning about Arduino programming.

VII. Working

Fig. 6 shows the flow of the algoritm. The Flappy Bird game is controlled by an ESP8266 Node-MCU microcontroller circuit, which utilizes a push button to receive user input. Upon pressing the push button, the circuit increases the x-coordinate of the bird by an offset of 0.01, while releasing the push button decreases the bird's x-coordinate by an offset of 0.03.

During gameplay, the circuit constantly checks the x-coordinates of the obstacles. If the obstacle's x-coordinate is less than that of the bird, the circuit updates the player's score. However, if the obstacle's x-coordinate is greater than or equal to that of the bird, the game terminates, and the player is returned to the main screen.

This circuit design allows for intuitive gameplay while incorporating score tracking and obstacle collision detection. By utilizing the ESP8266 Node-MCU microcontroller, the circuit can process user input and gameplay events quickly and efficiently, resulting in a seamless and enjoyable gaming experience.

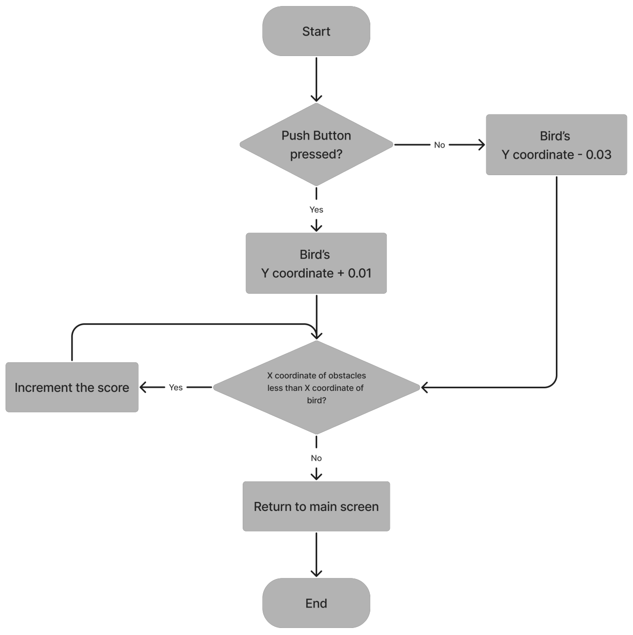
Start
Push Button pressed?
No
Bird's Y coordinate - 0.03
Yes
Bird's Y coordinate + 0.01
X coordinate of obstacles less than X coordinate of bird?
Yes
Increment the score
No
Return to main screen
End

Fig. 6. Flow Chart of the Algorithm

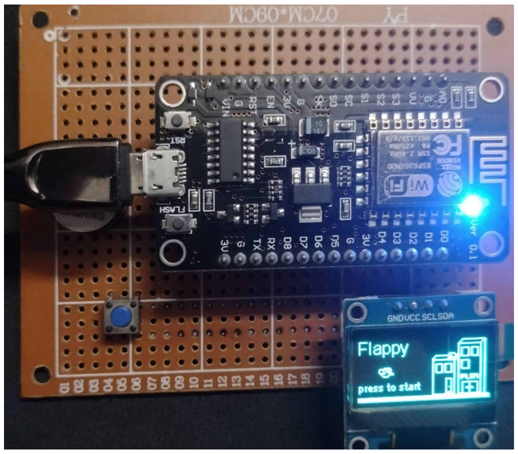

Fig. 7. Circuit Realisation on GPB

VIII. Results

The Flappy Bird game circuit was developed on a General Purpose Breadboard (GPB) using a push button for player input. The ESP8266 Node-MCU microcontroller was utilized to sense the status of the push button and trigger the corresponding actions according to the programmed instructions. The game was then displayed on an OLED display.

This circuit design provides a low-cost and customizable solution for developing interactive games. The use of a push button for player input and an OLED display for game visualization is a simple yet effective way to engage users. By incorporating the ESP8266 Node-MCU microcontroller, the circuit can quickly process player input and display the game output on the OLED display.

Overall, this circuit design demonstrates the flexibility and versatility of using a GPB in combination with a microcontroller to create engaging and interactive games. The Flappy Bird game is a prime example of how a simple circuit can be utilized to create an enjoyable gaming experience.

Fig. 7 shows the circuit realisarion on general purpose board.

IX. Conclusion

In this project, we have discussed the development of a Flappy Bird game using an ESP8266 Node-MCU, a 0.96 inch OLED display, a buzzer, and a push button on a general-purpose board. The flowchart of the game was provided, and the working of the circuit was explained in detail. The ESP8266 Node-MCU was described as a powerful microcontroller that can be programmed using the Arduino IDE, while the 0.96 inch OLED display was described as a compact and easy-to-use display module that can display graphics and text. The buzzer and push button were also explained as essential components of the circuit. Additionally, a general-purpose board was defined as a versatile and flexible platform that can be used for a wide range of electronics projects.

This project highlights the potential for creativity and innovation in the field of electronics and demonstrates the importance of collaboration and support in achieving success.

X. Acknowledgment

We extend our deepest appreciation to Prof Vijay Sawani, our esteemed mentor, for his unwavering guidance and invaluable assistance throughout the duration of this project. His expertise and insights have been instrumental in ensuring the successful completion of this endeavor.

Furthermore, we would like to express our gratitude to all the supporting personnel who provided us with the necessary equipment and resources, which were essential in executing this project efficiently. Without their contributions, it would have been impossible to achieve the desired results.

We acknowledge and value the efforts of all those who played a part in this project, and we are truly grateful for their support and cooperation. Their contributions have enabled us to produce a project of exceptional quality and significance, which we are proud to present as a testament to our collective efforts.

References

[1] https://lastminuteengineers.com/oled-display-esp8266-tutorial/
[2] https://www.instructables.com/ESP8266-with-Multiple-Analog-Sensors/
[3] https://maker.pro/esp8266/projects/joystick-esp8266-mpu6050
[4] https://www.instructables.com/Getting-Started-With-ESP8266-and-AskSensors-IoT-Pl/