بسم الله الرحمن الرحيم

AN-NAJAH NATIONAL UNIVERSITY

Computer Engineering Department

Hardware Graduation Project

Snake Hunt

students
Maryam Sholi
Yara Haydariah
marymarie905@gmail.com
yaraalhayd12@gmail.com

Under the supervision of
Dr. Raed Qadi

A report submitted in partial fulfillment of the requirements for bachelor's degree in computer engineering in the Faculty of Engineering Information Technology - Hardware Project

May,2023

Acknowledgements

First and foremost,we thank God for giving us the ability to complete our second graduation project. Efforts have been made on this project. However, without the kind assistance and support of many people, it would not have been possible. We appreciate Dr. Raed Al-Qadi, our project manager, for his guidance and constant supervision,he encouraged us and gave all the necessary data we needed to create and present our project in the best possible way. We really appreciate our families' constant support and encouragement in helping us finish our project, we express our gratitude to the whole faculty in the department of computer engineering for their ongoing help and support.We would also like to thank An-Najah National University for giving us the opportunity to be students with an educational degree in computer engineering. Finally, a big hug to our friends who have always been there for us. Thanks will not be enough for their useful suggestions and additions.

Yours sincerely Maryam Yara.

Disclaimer

This report was written by students at the Computer Engineering Department, Faculty of Engineering, An-Najah National University. It has not been altered or corrected, other than editorial corrections, as a result of assessment and it may contain language as well as content errors. The views expressed in it together with any outcomes and recommendations are solely those of the students. An-Najah National University accepts no responsibility or liability for the consequences of this report being used for a purpose other than the purpose for which it was commissioned.

Contents

List of Figures

Abstract

Recently, technology has found its place in society, as it dominates most of our lives. And the use of technology has become a necessity in every aspect of our lives, from the simplest to the most complex. One of these branches is the entertainment aspect as a great development has recently appeared on games to become more intelligent than before. In this project we are controlling the game of Snake in a smoother way so that we control the movement of Snake without the need for a tangible controller, We will instead use a glove that contains sensors that track the movement of the hand in any direction it will be, and therefore the Snake will be moved in the same direction that was created. As for how to display the game screen, it will be a group of LEDs in the form of matrices. There is a display screen through which the number of points that have been recorded is shown.

1 Introduction

In recent years, we have seen a significant increase in the popularity of gaming across all age groups. One of these games, the snake game, is included in this project in a slightly different method.

Gaming is entertaining and fun. We can relax and decompress after our hard days through it. Many of us like playing and discovering new games in our spare time, while others do the same. Today, games are growing up alongside the rapid technological growth that we have.

This project seeks to combine the simplicity and excitement of the snake game, with a few additional elements. In order to make the classic snake game more exciting and difficult, this project investigates a new dimensions. This game is the perfect choice for a project where we can concentrate on more complicated subjects because of how simple it is.

Snake is a historical game that challenges players to assess their surroundings and choose the shortest or safest route to an objective. This is a great chance to practice their awareness of where they are and think ahead to their next action.

The player controls a long, thin creature that resembles a snake and is moved in a limited area while grabbing food and attempting to avoid bumping into its tail or the borders of the playing area. The snake is moved by a glove equipped with an accelerometer sensor that determines the direction the user wants the snake to travel in. The snake's tail lengthens as it consumes food, making the game more difficult.

The snake and fruits will be displayed on several 8*8 dot matrix linked together and by using gloves the game will work as per our requirements.

1.1 Statement of the problem

The goal of this project is to create a snake game that blatantly fits the following requirements, as the game progresses, the difficulty must grow since a snake‘s size must increase after consuming a unit of food(They will be spawned randomly after each snake eating in different areas around the playing area), the snake may enter a wall from one side and exit the other if and only if the red LED is on, and finally, the moment the snake touches its own body or go out the playing area when the red led is of, the game is ended.

1.2 Problem Depiction

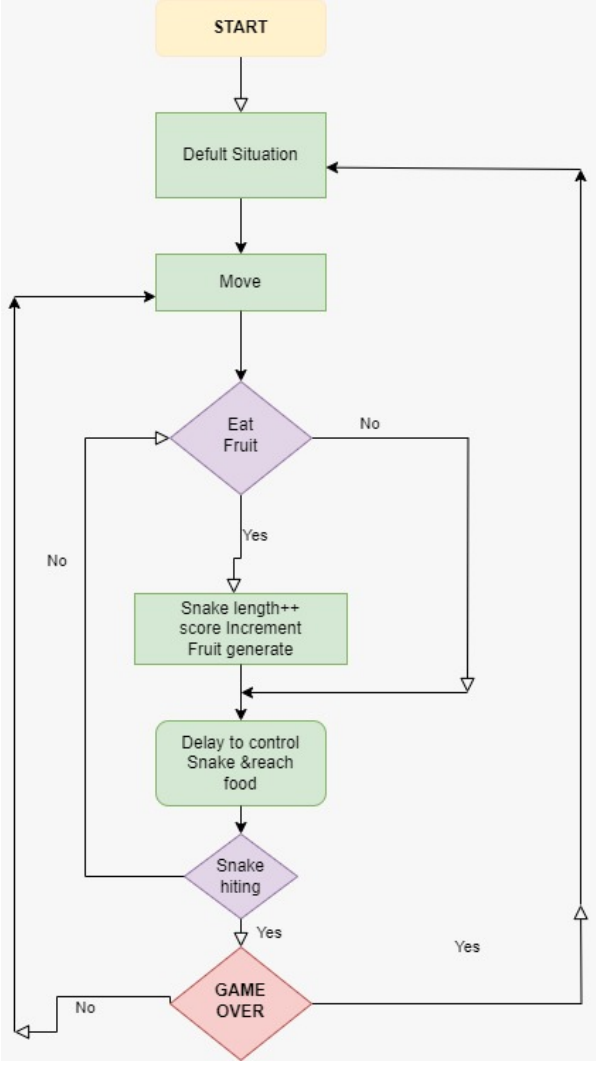

Figure 1.2.1: Chart

1.3 Objectives

The player's main goal in this game is to pick up as many fruits as they can without hitting their snake by itself, once doing this by time so will get increasingly challenging as the snake's length grows.

1.3.1 snake

a. Elements of the snake object (length, tail, and head).
b. Snakes grow longer as they eat more food.
c. The snake appears on the screen at a default Point.
d. The game ends when the snake crosses itself and when it comes out of the frame of the play screen in the event that the red LED is off.

1.3.2 Food

a. Each food unit may be found randomly throughout the Matrix .
b. Each unit of food makes the snake one unit longer.
c. Food cannot appear on any of the leds that the snake is currently occupying.
d. Each unit of food has a value twice the original value if the yellow LED is on.

1.3.3 Screens

a. LED matrices with a dimension of 3*2 (4 matrices), each with 8 rows and 8 columns.
b. OLED screen to display the snake's score had got it at the current game in numerical form, the highest value we got during all the stages we played, and the speed at which the snake moves.

1.3.4 Glove

This is like controlling the movement of the snake. It contains an acceleration sensor in addition to a transmitter. If there is a change in the movement of the hand and its inclination, the direction of this movement is detected using the sensor and according to the difference between the values it reads.. Based on this, the direction that the user wants for the snake to move towards it is detected, and this information is sent to another transmitter located at the end of the LEDs screen.

1.3.5 Boarder

snake can exit this boarder from one side and reenter from the other in the event that the red LED is on and in the event that it is turned off, the snake is not allowed to pass through the borders, and this increases the element of challenge and suspense in the game.

1.4 Scope

Those in need of enjoyment, simplicity, and difficulty are taken care by Snake Hunt.

1.5 Importance

Snake Hunt generates spatial awareness and forces users to think forward to their next step.It also teaches them how to develop long-term strategic plans.

It can be quite upsetting to achieve that height just when you hit your tail, as many parents are aware. The game demands calmness when you lose, and patience to advance, which is inevitable. These are all important life lessons that children need to learn early on since they will help them later in life.

Snake Hunt teaches young people that acquiring new abilities requires practice. It‘s easy to use and includes many features that serve the entertainment and educational requirements of the users.

1.6 Report Organization

The second chapter will discuss the limitations and difficulties we encountered in setting up the system. We will also mention some of the courses that helped us implement and create this application. Chapter three literature review will talk about some of the previous releases in this area and how they differ from this game. Methodology Chapter 4 We will talk about what features our project supports and what technologies we have adopted. The fifth chapter will talk about the results of this project, and the next chapter will talk about recommendations and some future work, and the final chapter will be about references.

2 Constraints and Earlier course work

2.1 Constraints

Because we don't know a lot about mechanical parts, we ran into a lot of difficulties, including finding stores that sell them, how to connect them together, and having to look at most of the circuit separately to discover the malfunctioning component so to solve any problem we faced, it took us a long times.

2.1.1 Data transfer

We had difficulties with the data transfer speed, which caused delays in playing. That's why we used hc-12, because it enabled us to transmit data over long distances with a suitable speed.We also wanted to use the LCD screen for the display, but due to the load on the Arduino, this caused the speed of the snake to decrease..so we chose an OLED screen for the display instead.

2.1.2 bigger screen

We had difficulty using a larger screen with more space to play and converting it from a smaller matrix to a larger one due to the different variables and parameters compared to using only 8 * 8 LED matrix. We also wanted to use many 8 * 8 matrixes in the two dimensions, but the composition of this matrix does not allow two or four of them to be meshed with it and the data is transmitted smoothly.

2.1.3 components on glove

Because the glove is always a moving part, we faced difficulty in installing the pieces on it in a good way, in addition to the vibration in the glove. So we solved this problem by using printed circuit board (PCB) to make the connections internal.

2.2 Earlier Courses

2.2.1 C Language

because our code was written using this language.

2.2.2 Electronic Circuits

The key benefits of this course were learning how to work with electrical circuits and learning the fundamentals of how to work with and connect electronic components.

2.2.3 Arduino course

Arduino course via the Internet, which was very beneficial to us since the Arduino component served as the central axis of our work, and we learned how to use it and employ it to link to other components in order to serve the project in an integrated manner.

2.2.4 CPU lab

This laboratory helped us to identify all the basics necessary to use the hardware components. We also learned how to connect pieces with each other, how to weld them in the correct way, the importance of making sure that they are correct and that they work as required before starting to connect them with others, in addition to how to check them in the event of a specific defect after connecting them to others and discovering the source of this defect.

2.2.5 Wireless

In this course, we learned about the use of wireless devices which helped us during the project to make the game wirelessly controlled.

2.2.6 Microcontroller

One of the computer engineering requirements was the Microcontroller course, which helped us understand how the hardware components work and program. It provided us with all the necessary information, and it also has a lab that shows us through it to deal more concretely with these components, we also dealt with the pins in the microcontroller, in addition to how to download the codes on these components, and how to use Arduino codes.

2.2.7 Microprocessor lab

We learned from this lab how to program the electronic parts, in addition to how to download the codes on these pieces, and this contributed to the smooth implementation and application in such this kind of pieces.

3 Literature Review

Snake is a type of video game that first appeared in arcades in the late 1970s and has since become a classic. Gremlin, a student of computer science, created the first known version of the Snake game in 1976. It was developed as a test to show off a graphical terminal's capabilities at the University of Michigan. The DEC PDP-11, an older computer, was used to run the game. When the Snake game first appeared on arcade machines in the late 1970s and early 1980s, it quickly became widely popular. Gremlin Industries' "Blockade," which was introduced in 1976, was one of the more prominent iterations. In the game Blockade, two snakes had to navigate a small space without bumping with one other, walls, or other objects. The player is in charge of a long, slender snake-like creature that moves around on a bounded plane, gathering food (or other items), while attempting to avoid striking its own tail or the playing area's boundaries. The snake's tail lengthens as it eats food, making the game more difficult with each new piece. The user may move the snake's head down, left, up, or right, and the body will move in the same way.

Cognitive and Psychological Aspects: Scientists look at the psychological and cognitive facets of the Snake game. They investigate the real-time strategy, decision, and response processes of players. It is frequently investigated how the game affects focus, problem-solving abilities, spatial awareness, and attention.

Educational Potential: The educational value of the Snake game is studied by several academics. They look into how it might be applied as a teaching tool in several situations, such as enhancing pattern recognition, decision-making abilities, and hand-eye coordination. The game is an excellent tool for teaching because it's easy to use and accessible.

Here are two of a number of Snake game releases: One of the earliest iterations of the Snake game is **Blockade (1976)** . It showed two snakes slithering around a small space while dodging obstacles and competing with one another for control.

Despite not being a classic Snake game,**Slither.io (2016)** quickly gained popularity as a multiplayer online game. Players took control of a snake, which they tried to develop by ingesting colored pellets while avoiding hitting the snakes of other players.

4 Design and Implementation

4.1 Block description

Using an Arduino (Uno & Nano), we put the snake game into action. In this, the snake is moved left, right, up and down using a glove with a sensor. We will generate the code using Embedded C according to our needs. Snake and fruits will be displayed on a four 8 * 8 dot matrix connected together, and the game will start as we want.

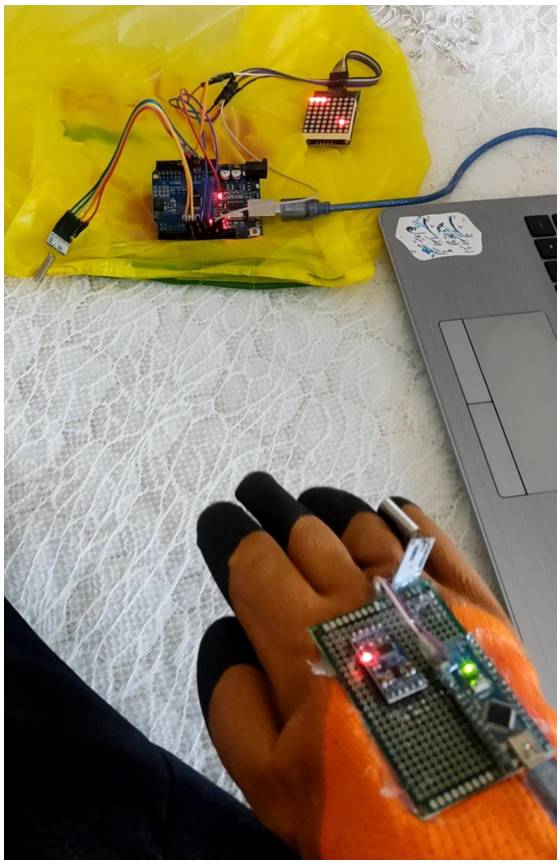

Figure 4.2.1: Component assembly

Figure 4.2.2: Body Game design

4.2 System Design

to design a snake game where players may direct the snake's movement on a screen, get points for consuming food, and avoid colliding with the snake's expanding tail. In this issue, we want to create a game where a snake-like picture slides across the screen. The snake lengthens and we score a point if it comes upon any food. It will die if it touches itself. We will require the following to create this program: 1. A method of expressing the food. 2. A method for our instructions and data to reach the snake 3. how displaying the score 4. A method of detecting our death after colliding with anything.

4.3 Hardware and Software Specifications

4.3.1 Hardware Components

Hardware component :

For this part, we have used an Arduino Uno, Arduino NANO, ADXL345 accelerometer,max7219 led matrix, HC-12, and Display Screen.

• Arduino Uno

The Arduino Uno is an open-source microcontroller board created by Arduino.cc. It is one of the most well-known and extensively used boards in the Arduino ecosystem. It is built around the ATmega328P microcontroller, the board has 14 digital input/output pins, There are 6 analog input pins, clock frequency 16 MHz, a USB connection, Input voltage (limits) 6-20 volts. This piece was used in our project until the hc12 transmitter piece is connected to it, in addition to being connected to the led matrix. It is shown in Figure 4.3.1.

Figure 4.3.1: Arduino Uno

• Arduino Nano

It is a microcontroller with 14 digital I/O pins (as well as 6 PWM output pins). It also contains 8 analog input pins. It has a 5 volt working voltage and a 6-20 volt input voltage. This piece was used in our project until the second hc12 transmitter piece is connected to it, in addition to being connected to the ADXL345 Accelerometer, and it is located on the glove. It is shown in Figure 4.3.2.

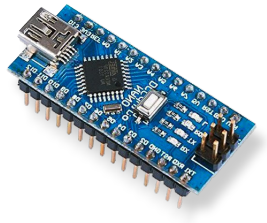

Figure 4.3.2: Arduino Nano

• ADXL345 accelerometer

The ADXL345 is a compact, thin, ultralow power 3-axis accelerometer with high resolution (13-bit) measuring at up to ±16 g. Digital output data is formatted as 16-bit twos complement and is available via an SPI (3- or 4-wire) or I2C digital interface. We used this piece to monitor the movement of the hand in the four directions to move the snake in the appropriate direction. It is shown in Figure 4.3.3.

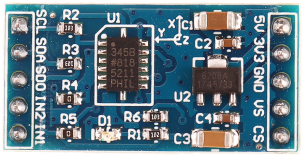

Figure 4.3.3: ADXL345 accelerometer

• max7219 led matrix

The MAX7219 is designed to control LED matrix displays of up to 8x8 configurations. MAX 7219 is a serial and parallel output common-cathode display driver. It connects microprocessors and microcontrollers to 64 individual LEDs. The MAX 7219 is linked to the 8 × 8 LED matrix. The MAX7219 receives data input from the Arduino board. It was used here to display the output coming from the glove movement so that it displays the movements of the snake, we use four of them connected together to scroll the snake throw them. It is shown in Figure 4.3.4.

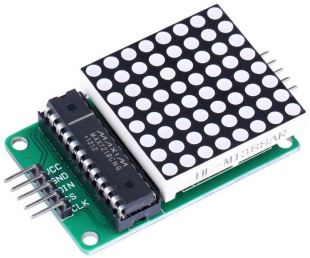

Figure 4.3.4: max7219 led matrix

• HC-12

The HC-12 is a wireless transceiver module with a frequency range of 433 MHz. It is often used for long-distance wireless communication between microcontrollers or other devices. We have used two of this piece so that the first one, which is connected to the glove, was used to send data related to the direction of the glove movement to receive it from the second that monitors the movement to display it on the led matrix. Here are some key features of the HC-12 module:

1. **Long-range communication:** The HC-12 module is capable of achieving communication ranges of up to 1,000 meters in open space, making it suitable for applications that require wireless communication over extended distances.

2. **Simple interface:** The module communicates with Arduino or other microcontrollers using a serial UART interface. It can be connected to the Arduino's RX and TX pins and controlled using standard serial commands.

3. **Multiple operating modes:** The HC-12 module offers different operating modes, including transparent mode and command mode. In transparent mode, it acts as a wireless serial cable, transmitting data between two HC-12 modules without requiring any additional configuration. In command mode, you can send specific commands to the module to change settings or configure features.

4. **Low power consumption:** The HC-12 module is designed to be power-efficient, allowing it to be used in battery-powered applications where power consumption is a concern.

5. **Error correction and addressing:** The HC-12 module supports forward error correction (FEC) to enhance the reliability of wireless communication. It also provides the option to set unique addresses for individual modules, enabling point-to-point or multi-point communication setups.

6. **Integration with Arduino:** The HC-12 module can be easily connected to Arduino boards using software serial or hardware serial interfaces. You can use Arduino libraries specifically designed for the HC-12 module to simplify the communication process. It is shown in Figure 4.3.5.

Figure 4.3.5: HC-12

• OLED Display Screen

An OLED (Organic Light-Emitting Diode) display is a form of display technology that emits light when an electric current is applied using organic components. Its low power consumption, great contrast, and brilliant colors make it a popular choice for Arduino projects. Since each tiny OLED pixel on the display is capable of emitting light independently, clear, detailed images are possible. We have used this screen to display the score the player has reached from controlling the snake and collecting fruits and the degree of difficulty adding to the top score. It is shown in Figure 4.3.6.

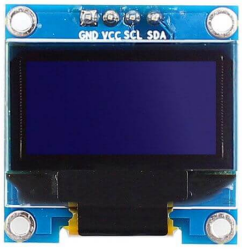

Figure 4.3.6: OLED Screen

• Potentiometer

Potentiometers are a popular electrical component. By rotating a knob or moving a lever, you may adjust the resistance on this analog input device. An analog input pin on the Arduino board can gauge this fluctuation in resistance. We used it to change the speeds of the snake through the different levels.It is shown in Figure 4.3.7.

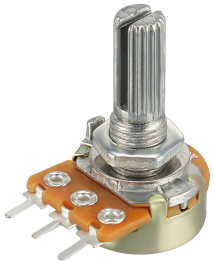

Figure 4.3.7: potentiometer

• Traffic Light

We have used these LEDs to express some things in the game. The green color means that the game is running, while the yellow color lights up randomly to indicate that the value of points during this time will be doubled, and finally the red light when it lights up randomly also allows the player to go beyond the limits of the game. It is shown in Figure 4.3.8.

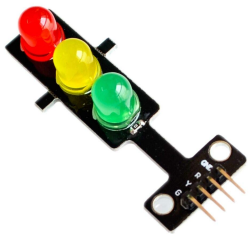

Figure 4.3.8: Traffic Light

• Printed Circuit Board (PCB)

A printed circuit board (PCB) is a flat board made of non-conductive material (typically fiberglass or composite epoxy) that contains electrical connections and components. It provides a platform for assembling and interconnecting electronic components. We used it here to help prevent components on the glove from moving incorrectly, reducing friction with the glove and thus not unraveling the wires. It is shown in Figure 4.3.9.

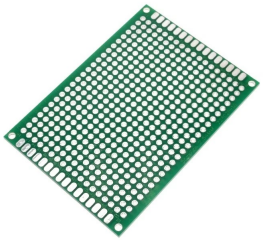

Figure 4.3.9: pcb

• Mini Green Breadboard

A mini green breadboard is a prototyping tool used in electronics projects to create temporary circuits without the need for soldering. They consist of a

plastic board with holes and metal clips that allow you to insert and connect electronic components and wires. We used it to arrange and organize the wires and make it easier to connect them together (especially the GND and VCC). It is shown in Figure 4.3.10.

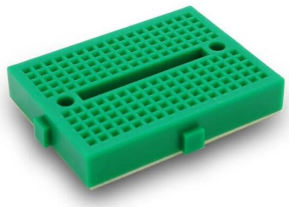

Figure 4.3.10: Breadboard

4.3.2 Software Design

The snake expands if it eats the food. It will move in a certain direction in response to the user's hand motion. Each time the snake moves, the head travels in the new direction, and each of its body pieces advances behind it by assuming the position that the component in front of it once held. The easiest response for feeding the snake is that we assume the snake is eating the meal if its head and the food are in the same location. We therefore need to be aware of the location of the meal. When it is consumed, it vanishes, the snake expands, and food emerges at random in other places. However, we also wanted to display the outcome and results, so we also need a variable to record that. To display the counter value in this instance, we will use a display screen.

functional requirements

- The snake must turn in response to user input.
- The snake must appear to move within the screen.
- The snake will die if it turns around itself .
- The snake will grow longer if it eats food .
- The snake never stops moving .

Non- functional requirements

- Reusability: A single user may utilize the system an unlimited number of times. Additionally, the reusability is stable, flexible, and consistent.
- Effectiveness: The algorithm can either produce the required outcomes or results that are better.
- Reliability: The system is dependable and consistently performs well. It can also be said that the system completes the task without any errors when specific circumstances and time constraints are met.

Arduino IDE

The Arduino IDE (Integrated Development Environment) is a piece of software used to create and develop applications for Arduino microcontroller boards. It is a simple platform for creating, compiling, and uploading code to Arduino boards.

4.4 Arduino Code

Our code was written in two parts, each part for each Arduino (Nano, Uno): Receiver code-Uno:

```
1  //#include <MD_Parola.h>
2  #include <MD_KeySwitch.h>
3  #include <MD_MAX72xx.h>
4  //#define HARDWARE_TYPE MD_MAX72XX::FC16_HW// PAROLA_HW
5  #define MAX_DEVICES 4
6  #define CLK_PIN  13  // or SCK
7  #define DATA_PIN 11  // or MOSI
8  #define CS_PIN  10  // or SS
9  MD_MAX72XX mx = MD_MAX72XX(CS_PIN, MAX_DEVICES);
10
11 #include <SoftwareSerial.h>
12 SoftwareSerial HC12(7, 8); // HC-12 TX Pin, HC-12 RX Pin
13
14
15 #include <Wire.h>
16 #include <ACROBOTIC_SSD1306.h>
17
18
19
20 const int potmeter = 3;
21 int coord[200][2];
22 int leng = 1;
23 int prev;
24 int point[5][2] {{10,5},{3,1},{25,3},{15,7},{11,3}};
25
26 int delay1=3;
27
28 int blue = 12;
```

Figure 4.4.1: Receiver code-1

```
29  int green = 2;
30
31  int screens = 1;
32  int record;
33  int points;
34  int screenpick = 0;
35  int diff = 0;
36  int speedx;
37  int supermode;
38  int doublepoints;
39  long counter;
40
41
42  void slide()
43  {
44    for(int x = 99; x > 0; x--)
45    {
46      coord[x][0] = coord[x-1][0];
47      coord[x][1] = coord[x-1][1];
48    }
49    return;
50  }
51
52
```

Figure 4.4.2: Receiver code-2

```
54  int Joystick()
55  {
56    int valtest = 0 ;
57    while (HC12.available()) {        // If HC-12 has data
58      valtest = (HC12.read());
59      Serial.println(valtest);      // Send the data to Serial monitor
60      Serial.println("---------------");
61     }
62      if(valtest == 100 ){
63     prev = 0;
64     return 0;
65     Serial.println("dddddd");
66    }else if(valtest == 117 ){
67     prev = 1;
68     return 1;
69     Serial.println("uuuuuuuu");
70    }else if(valtest == 108 ){
71     prev = 3;
72     return 3;
73     Serial.println("llll");
74    }else if(valtest == 114 ){
75      prev = 2;
76     return 2;
77     Serial.println("rrrr");
78    }
79    //////////
80    const int VRx = 0;
81    const int VRy = 1;
```

Figure 4.4.3: Receiver code-3

```
  const int SW = 2;
  if(analogRead(SW) == 0)
  {
    return 5;
  }
  return prev;
}


void setup()

{

  Serial.begin(9600);


 Wire.begin();
  oled.init();                      // Initialze SSD1306 OLED display
  oled.clearDisplay();              // Clear screen
  oled.setTextXY(0,0);              // Set cursor position, start of line 0
//  oled.setFont(&FreeSans9pt7b);
  oled.putString("SNAKE GAME");
  delay(2000);

  mx.begin();
  coord[1][0] = coord[0][0] =  5;
  coord[1][1] = coord[0][1] =  5;
```

Figure 4.4.4: Receiver code-4

```
  pinMode(blue, OUTPUT);
  pinMode(green, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(3, HIGH);
  HC12.begin(9600);
 delay(2000);
}



void led(int x, int y)
{
  mx.setPoint(y-1, x-1, 1);
  return;
}


void loop()
{
  while(screens == 1)
  {
    supermode = 0;
    doublepoints = 0;
    digitalWrite(blue,0);
    digitalWrite(green,0);
    mx.clear();
```

Figure 4.4.5: Receiver code-5

```
137       if (points > record)
138       {
139         record = points;
140       }
141       coord[1][0] = coord[0][0] =  5;
142       coord[1][1] = coord[0][1] =  5;
143       while(screens == 1)
144       {
145
146     ////////////
147       int valtest = 0 ;
148      while (HC12.available()) {        // If HC-12 has data
149         valtest = (HC12.read());
150         Serial.println(valtest);      // Send the data to Serial monitor
151         Serial.println("---------------");
152       }
153         diff = analogRead(potmeter) / 100;
154         Serial.println(diff);
155   //      diff = 400 / 100;
156         if (valtest == 114 or valtest == 108)
157         {
158           oled.setTextXY(2,0);             // Set cursor position, line 2 10th character
159           oled.putString("points=     ");
160           screens = 0;
161           speedx = (12-diff)*9;
162           screenpick = 0;
163           leng = 1;
164           points = 0;
```

Figure 4.4.6: Receiver code-6

```
166       HC12.flush();
167
168     }
169     if (valtest == 100)
170     {
171       screenpick++;
172       if(screenpick > 2) screenpick = 0;
173       mx.clear();
174      delay(1500);
175       HC12.flush();
176     }
177     if (valtest == 117)
178     {
179       screenpick--;
180       if(screenpick < 0) screenpick = 2;
181       mx.clear();
182       delay(1500);
183       HC12.flush();
184     }
185
186
187
188     switch(screenpick)
189     {
190       case 0:
191       printNumber(points,0);
192
```

Figure 4.4.7: Receiver code-7

```
229             case 1:
230             printNumber(record,0);
231
232             break;
233
234             case 2:
235             printNumber(diff,0);
236
237             break;
238
239           }
240       }
241
242     }
243     for (int x = 0; x < speedx; x++)
244     {
245       printNumber(points,0);
246     }
247     slide();
248     counter++;
249     if(counter == 40)
250     {
251       supermode = 0;
252       doublepoints = 0;
253       digitalWrite(blue,0);
254       digitalWrite(green,0);
255     }
256     switch(Joystick())
```

Figure 4.4.8: Receiver code-8

```
256   switch(Joystick())
257   {
258     case 0:
259     coord[0][1] = coord[1][1]+1; //y+1
260     break;
261     case 1:
262     coord[0][1] = coord[1][1]-1; //y-1
263     break;
264     case 2:
265     coord[0][0] = coord[1][0]+1; //x+1
266     break;
267     case 3:
268     coord[0][0] = coord[1][0]-1; //x-1
269     break;
270   }
271   for (int x = 0; x < 5; x++)
272   {
273     if (coord[0][0] == point[x][0] && coord[0][1] == point[x][1])
274     {
275       if (random(0,diff+1) == 0)
276       {
277         supermode = 1;
278         counter = 0;
279         digitalWrite(blue,1);
280       }
281       if (random(0,9) == 0)
282       {
283         doublepoints = 1;
```

Figure 4.4.9: Receiver code-9

```
284             counter = 0;
285             digitalWrite(green,1);
286           }
287           leng++;
288           points += 5;
289           if(doublepoints == 1) points += 5;
290           if (speedx > (10-diff)*3 + 10)
291           {
292             speedx--;
293           }
294           point[x][0] = random(1,32);
295           point[x][1] = random(1,8);
296         }
297       }
298        if (supermode == 1)
299        {
300          if(coord[0][0] == 0)
301          {
302            coord[0][0] = 32;
303          }
304          if(coord[0][0] == 33)
305          {
306            coord[0][0] = 1;
307          }
308          if(coord[0][1] == 0)
309          {
310            coord[0][1] = 8;
311          }
```

Figure 4.4.10: Receiver code-10

```
312       if(coord[0][1] == 9)
313       {
314         coord[0][1] = 1;
315       }
316     }
317     mx.clear();
318     for (int x = 0; x < leng; x++)
319     {
320       led(coord[x][0],coord[x][1]);
321     }
322     for (int x = 0; x < 5; x++)
323     {
324       led(point[x][0],point[x][1]);
325     }
326     if (supermode == 0)
327     {
328       if (coord[0][0] == 0 || coord[0][0] == 33 || coord[0][1] == 0 || coord[0][1] == 9)
329       {
330        screens = 1;
331       }
332     }
333
334
335     for(int x = 1; x < leng; x++)
336     {
337       if (coord[0][0] == coord[x][0] && coord[0][1] == coord[x][1])
338       {
339         screens = 1;
340       }
```

Figure 4.4.11: Receiver code-11

```
  }
}
int long coutt = 0;
void printNumber(long number,int place)
{
  int num1 = number / 1000;
  int num2 = number / 100 - num1 * 10;
  int num3 = number / 10  - num1 * 100 - num2 * 10;
  int num4 = number       - num1 * 1000- num2 * 100 - num3 * 10;


if(millis() - coutt > 300){
// oled.clearDisplay();
 oled.setTextXY(2,0);             // Set cursor position, line 2 10th character
 oled.putString("points= ");
  String d_points = String(points);
  const char *mac_points=d_points.c_str();
  oled.putString(mac_points);

   oled.setTextXY(3,0);             // Set cursor position, line 2 10th character
 oled.putString("record= ");
  String d_record = String(record);
  const char *mac_record=d_record.c_str();
  oled.putString(mac_record);

    oled.setTextXY(4,0);             // Set cursor position, line 2 10th character
 oled.putString("diff= ");
```

Figure 4.4.12: Receiver code-12

```
368     String d_diff = String(diff);
369     const char *mac_diff=d_diff.c_str();
370     oled.putString(mac_diff);
371
372     coutt = millis();
373   }
```

Figure 4.4.13: Receiver code-13

Sender code-Nano:

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_ADXL345_U.h>

#include <SoftwareSerial.h>

//SoftwareSerial HC12(8, 7); // HC-12 TX Pin, HC-12 RX Pin

/* Assign a unique ID to this sensor at the same time */
Adafruit_ADXL345_Unified accel = Adafruit_ADXL345_Unified(12345);

void displaySensorDetails(void)
{
  sensor_t sensor;
  accel.getSensor(&sensor);
  Serial.println("------------------------------------");
  Serial.print  ("Sensor:       "); Serial.println(sensor.name);
  Serial.print  ("Driver Ver:   "); Serial.println(sensor.version);
  Serial.print  ("Unique ID:    "); Serial.println(sensor.sensor_id);
  Serial.print  ("Max Value:    "); Serial.print(sensor.max_value); Serial.printl
  Serial.print  ("Min Value:    "); Serial.print(sensor.min_value); Serial.printl
  Serial.print  ("Resolution:   "); Serial.print(sensor.resolution); Serial.print
  Serial.println("------------------------------------");
  Serial.println("");
  delay(500);
}

```

Figure 4.4.14: Sender code-1

```
28  void displayDataRate(void)
29  {
30    Serial.print  ("Data Rate:    ");
31
32    switch(accel.getDataRate())
33    {
34      case ADXL345_DATARATE_3200_HZ:
35        Serial.print  ("3200 ");
36        break;
37      case ADXL345_DATARATE_1600_HZ:
38        Serial.print  ("1600 ");
39        break;
40      case ADXL345_DATARATE_800_HZ:
41        Serial.print  ("800 ");
42        break;
43      case ADXL345_DATARATE_400_HZ:
44        Serial.print  ("400 ");
45        break;
46      case ADXL345_DATARATE_200_HZ:
47        Serial.print  ("200 ");
48        break;
49      case ADXL345_DATARATE_100_HZ:
50        Serial.print  ("100 ");
51        break;
52      case ADXL345_DATARATE_50_HZ:
53        Serial.print  ("50 ");
54        break;
55      case ADXL345_DATARATE_25_HZ:
```

Figure 4.4.15: Sender code-2

```
56          Serial.print  ("25 ");
57          break;
58        case ADXL345_DATARATE_12_5_HZ:
59          Serial.print  ("12.5 ");
60          break;
61        case ADXL345_DATARATE_6_25HZ:
62          Serial.print  ("6.25 ");
63          break;
64        case ADXL345_DATARATE_3_13_HZ:
65          Serial.print  ("3.13 ");
66          break;
67        case ADXL345_DATARATE_1_56_HZ:
68          Serial.print  ("1.56 ");
69          break;
70        case ADXL345_DATARATE_0_78_HZ:
71          Serial.print  ("0.78 ");
72          break;
73        case ADXL345_DATARATE_0_39_HZ:
74          Serial.print  ("0.39 ");
75          break;
76        case ADXL345_DATARATE_0_20_HZ:
77          Serial.print  ("0.20 ");
78          break;
79        case ADXL345_DATARATE_0_10_HZ:
80          Serial.print  ("0.10 ");
81          break;
82        default:
83          Serial.print  ("???? ");
84          break;
```

Figure 4.4.16: Sender code-3

```
85    }
86    Serial.println(" Hz");
87  }
88
89  void displayRange(void)
90  {
91    Serial.print  ("Range:         +/- ");
92
93    switch(accel.getRange())
94    {
95      case ADXL345_RANGE_16_G:
96        Serial.print  ("16 ");
97        break;
98      case ADXL345_RANGE_8_G:
99        Serial.print  ("8 ");
100       break;
101     case ADXL345_RANGE_4_G:
102       Serial.print  ("4 ");
103       break;
104     case ADXL345_RANGE_2_G:
105       Serial.print  ("2 ");
106       break;
107     default:
108       Serial.print  ("?? ");
109       break;
110   }
111   Serial.println(" g");
112 }
```

Figure 4.4.17: Sender code-4

```
114  void setup(void)
115  {
116  #ifndef ESP8266
117    while (!Serial); // for Leonardo/Micro/Zero
118  #endif
119    Serial.begin(9600);
120    Serial.println("Accelerometer Test"); Serial.println("");
121
122    /* Initialise the sensor */
123    if(!accel.begin())
124    {
125      /* There was a problem detecting the ADXL345 ... check your connectior
126      Serial.println("Ooops, no ADXL345 detected ... Check your wiring!");
127      while(1);
128    }
129
130    /* Set the range to whatever is appropriate for your project */
131  //  accel.setRange(ADXL345_RANGE_16_G);
132    // accel.setRange(ADXL345_RANGE_8_G);
133    // accel.setRange(ADXL345_RANGE_4_G);
134     accel.setRange(ADXL345_RANGE_2_G);
135
136    /* Display some basic information on this sensor */
137    displaySensorDetails();
138
139    /* Display additional settings (outside the scope of sensor_t) */
140    displayDataRate();
141    displayRange();
```

Figure 4.4.18: Sender code-5

```
142   Serial.println("");
143  //  HC12.begin(9600);
144  }
145
146  void loop(void)
147  {
148    /* Get a new sensor event */
149    sensors_event_t event;
150    accel.getEvent(&event);
151
152    /* Display the results (acceleration is measured in m/s^2) */
153    float x = event.acceleration.x;
154    float y = event.acceleration.y;
155
156  //  Serial.print("X: "); Serial.print(x); Serial.println("  ");
157  //  Serial.print("Y: "); Serial.print(y); Serial.println("  ");
158  //  Serial.print("Z: "); Serial.print(event.acceleration.z); Serial.print("  ");Serial.pri
159  //
160    if(x>6){
161      Serial.print("l");
162      }else if(x<-6){
163     Serial.print("r");
164    }else if(y<-6){
165     Serial.print("u");
166    }else if(y>6){
167     Serial.print("d");
168    } delay(100);
169  }
```

Figure 4.4.19: Sender code-6

4.5 Circuit Layout

- led matrix:
  clk -> pin10 arduino uno(pb2, ss)
  cs -> pin11 arduino uno(pb3, MOSI)
  DIN -> pin12 arduino uno(pb4, MISO)
  GND -> GND arduino uno
  VCC -> Vin arduino uno

- HC-12 Reciver:
  TXD -> pin7 arduino uno(pd7)
  RXD -> pin8 arduino uno(pb0)
  VCC -> 5-volt arduino uno

- ADXL345 accelerometer:
  SDA -> A4 Arduino NANO(I2C data, SDA)
  SDO -> A5 Arduino NANO(I2C CLK, SCL)
  GND -> GND Arduino NANO

- HC-12 Sender:
  TXD -> D4 Arduino NANO (digital PWM)
  RXD -> D3 Arduino NANO (digital PWM)
  VCC -> vcc accelerometer
  GND -> GND Arduino NANO

- OLED screen:
  SDA -> SDA arduino Uno
  SCL -> SCL arduino uno
  VCC -> 5 volt arduino Uno
  GND -> GND arduino Uno

- Potintiometer:
  output ->D17 arduino Uno
  VCC -> 5 volt arduino Uno

GND -> GND arduino Uno

- Traffic light:
  G ->vin arduino Uno
  Y ->D2 arduino Uno
  R -> D12 arduino Uno
  GND -> GND arduino Uno

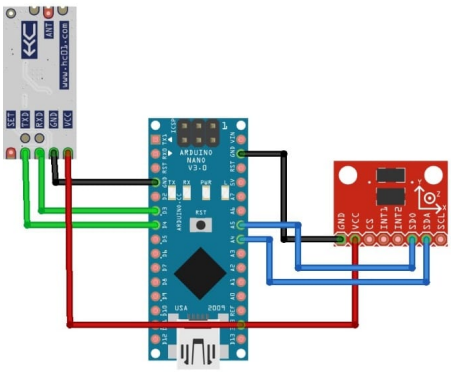

Figure 4.5.1: Sender Circuit

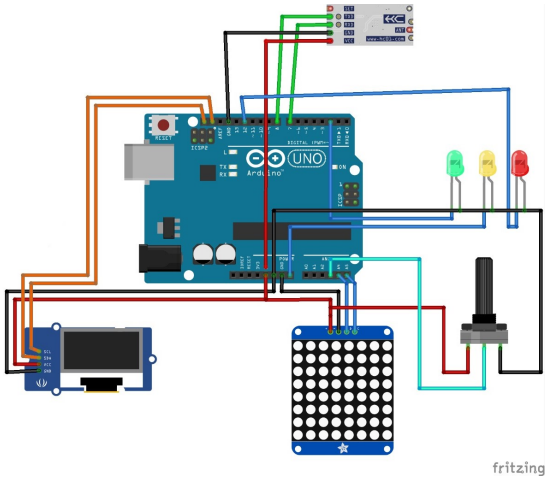

Figure 4.5.2: Receiver Circuit

5 Results and Analysis

Lets start Playing: first the player can select the speed at which he wants to play through the variable resistance, and its value will appear on the OLED screen from 0-8, which expresses the difficulty of the game,at these moments, the play screen does not contain the snake and food, but to start playing and showing these things, the play must tilt the hand to the right or left..and thus the game begins.

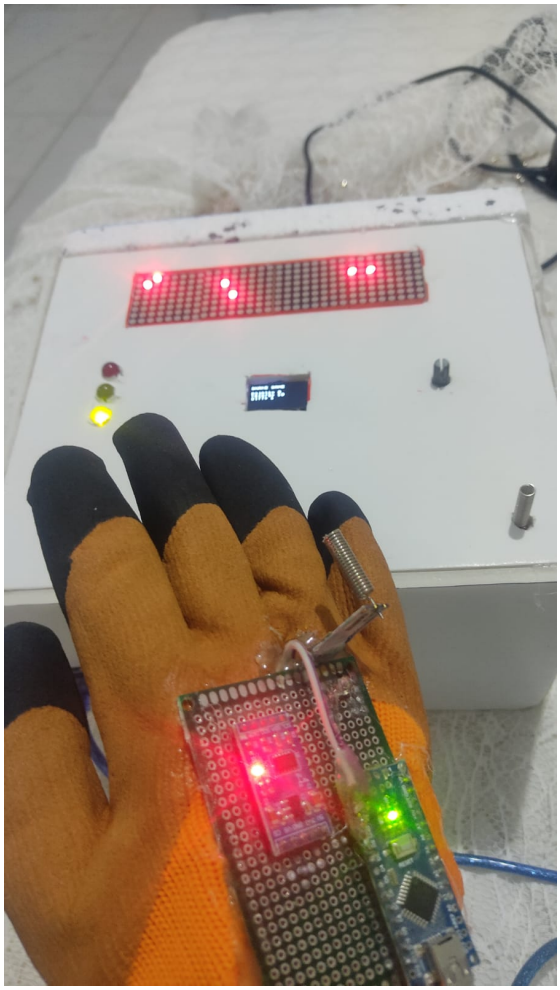

Figure 5.1.1: Start Playing

Then the player tilts the hand to get the direction he wants the snake to

move towards .. right = right, left = left, forward = up, backward = down. This is within the traditional instructions of the snake game, but here there is a difference in the presence of three LEDs (red, green, yellow) and each of them has a specific meaning, green indicates that the game has started, yellow indicates that the currently displayed food points are doubled, and red indicates that we can exceed the border of the screen.

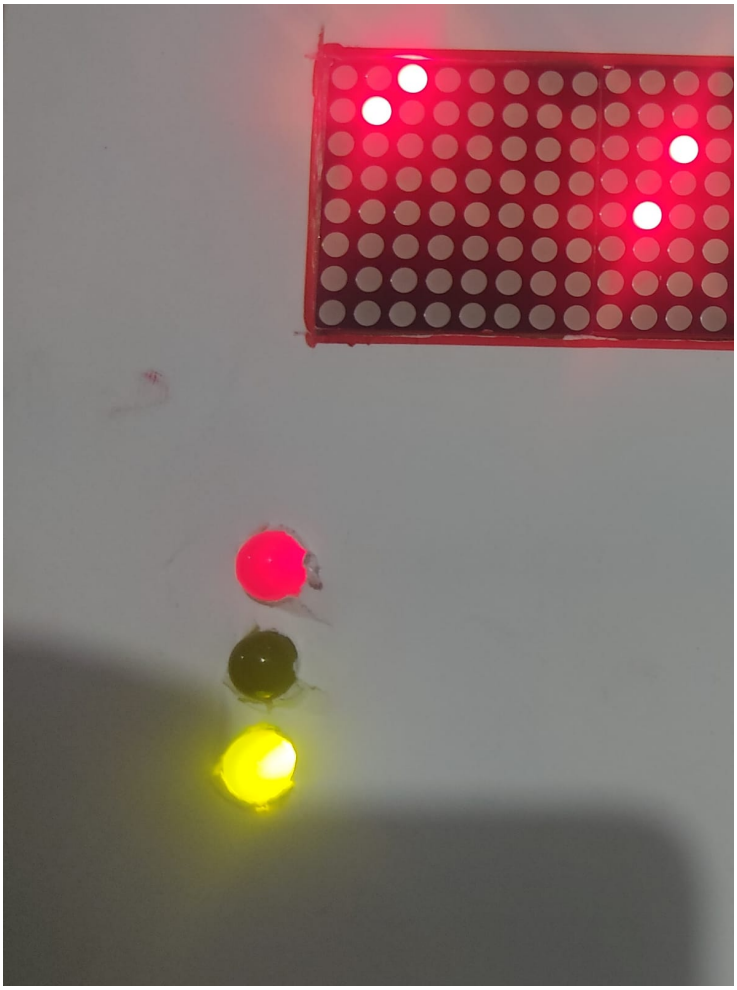

Figure 5.1.2: Leds

Three information will be displayed on the screen, the points collected during the current game, the top score collected by the stages the player is playing, and the difficulty level of the game.

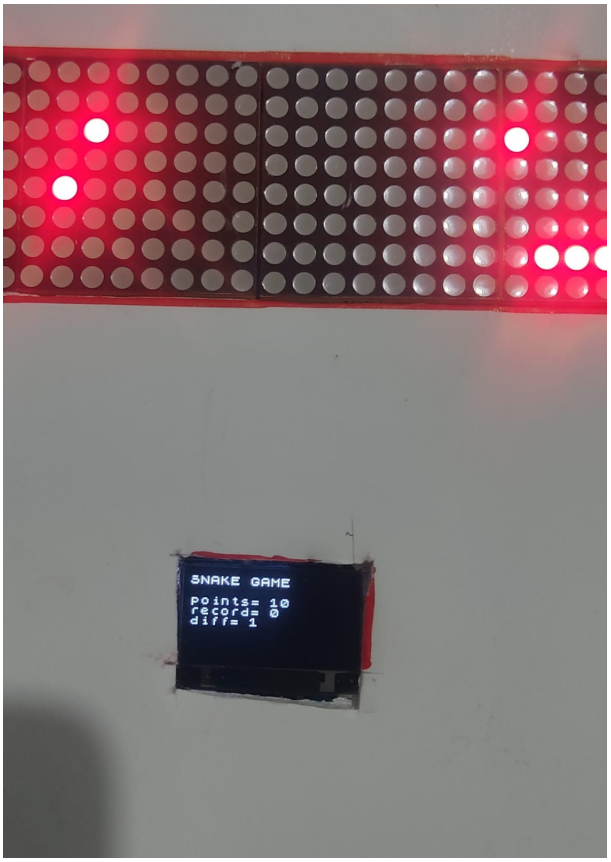

Figure 5.1.3: Oled information

6 Future Work

We will use the RGB led matrix and show fruits of different colors to indicate different values, so that it allows the user additional options in front of him during his movement and test his ability to always get the highest points, and adding another arm so that another player can play and confront so that increased the challenge motive or a snake by the game itself that acts as a competitor to the player and plays against him.

7 Conclusions and Recommendation

In our project, we made a snake game, which aimed to entertain and enjoy, in addition to increasing focus and challenge while controlling it. In the future, we can add improvements to the project by adding another arm so that another player can play and confront.

Bibliography

[1] https://www.youtube.com/playlist?list=PLmotdda810ImH-Nldpl5QOTkHywj3dKE.

[2]$https://www.programmiz.com/python-programming/time.$

[3]$https://www.arageek.com/tech/arduino-boards.$

[4]$https://www.youtube.com/watch?v=Bqgef7v7rj8.$

[5]$https://www.circuitbasics.com/how-to-setup-an-led-matrix-on-the-arduino/.$

[6]$https://howtomechatronics.com/tutorials/arduino/arduino-and-hc-12-long-range-wireless-communication-module/.$

[7]$https://howtomechatronics.com/tutorials/arduino/how-to-track-orientation-with-arduino-and-adxl345-accelerometer/.$