SNAKE GAME

CENG 3006: Introduction to Embedded Systems

Spring 2020 - 2021

Ad Astra

Gizem Pesen - 170709050
Fatma Karadağ - 170709061
Peteksu Karpuz - 170709063
Ayşe Ceren Çiçek - 170707009

***Abstract*—The main goal of this project is to design and implement a snake game with an Arduino Uno. The game can be played on a dot matrix by using a joystick. We used Arduino IDE to implement codes with C++ programming language.**

Keywords—Arduino, snake game, dot matrix, joystick, MAX7219

I. INTRODUCTION

Snake game is a very popular video game that a player maneuvers a line called a snake and it grows in length when it reaches (eats) a dot (apple). It is originated in 1976 and become very popular with Nokia mobile phones in 1998.

The aim of the game is simple. The player moves the snake and it tries to eat apples as much as possible. The length of the snake grows with each eaten apple. When the snake becomes longer, it becomes hard to control it. If the head of the snake hits any part of its body, the game ends.

II. PROJECT DETAILS

The subject of this project is to simulate this snake game on a led matrix with Arduino and let the player control the snake with a joystick. The implemented project will be as follows.

Primarily, the screen displays a scrolling text which is "Press Start". A player presses on the push button of the joystick to start the game. It starts with displaying an apple which is randomly set on a place and a snake moving on the led matrix, with a fixed starting length. The player can control the snake by moving the joystick through the x and y-axis. Each eaten apple grows the length of the snake. When the snake becomes larger the death possibility of it increases. The player tries not to collide the head of the snake with its body. If it collides, the game ends, and "Game Over" text is displayed on the led matrix. Subsequently, the game restarts the snake and turns to beginning with displaying the "Press Start" text again. The player can play the game again by pressing the joystick's push button.

III. IMPLEMENTATION

1. Libraries

The implementation of the game is started by including the necessary libraries. The first library "LedControl.h"[1] is an Arduino library for the MAX7219 LED display driver. The second [2] and third libraries are added for tracking joystick movements. The third library which is

```
#include "LedControl.h"
#include <Joystick.h>
#include <AxisJoystick.h>
```

"AxisJoystick.h" [3] has helped us a lot to define the direction of the movement like up, down, right, and left.

2. Declarations

We defined the input pins of the joystick. We connected axis pins VRx and VRy to the A1 and A0 pins and SW(switch) to A2 on the Arduino Uno and we declare the joystick object. We also defined "START_LENGTH" as 3, which corresponds to the starting length of the snake.

```
#define SW_PIN A2   // Joystick select button
#define VRX_PIN A1  // Joystick x-axis
#define VRY_PIN A0  // Joystick y-axis

// Create joystick
Joystick* joystick;

// The starting length of the snake
#define START_LENGTH 3
```

3. Initialization of Variables

```
const int numDevices = 1;     // Number of MAX7219s used
const long scrollDelay = 80;  // Scrolling speed of text
unsigned long bufferLong [14] = {0};

// LedControl(DATA, CLK, LOAD, number of matrices)
LedControl lc = LedControl(11, 13, 10, numDevices);

// Scrolled texts
const unsigned char scrollText[] PROGMEM = {"  G A M E  O V E R  "};
const unsigned char scrollTextStart[] PROGMEM = {"  P R E S S  S T A R T  "};

// Start velocities by direction
int x_direction = 1;
int y_direction = 0;

// Snake array with shape (64,2), 64 for 8x8 dots, 2 for x and y values
int snake[64][2];
int snake_length = 1;

// Apple array for x and y values
int apple[2];

// Boolean to check if game is ended
bool game_over = false;
```

“numDevices" variable defines the number of MAX7219 LED display drivers. To display scrolling text on the Matrix, the speed of the text is defined as 20 which is "scrollDelay". "scrollText" and "scrollTextStart" character arrays hold the texts which will be shown in the beginning and at the end. The Matrix will display "PRESS START" text until the player presses on the start button and the "GAME OVER" text will be shown for one time when the snake collides with itself.

The led control is defined as "lc" with the connected digital pins. DATA pin of the MAX7219 is connected with the Arduino Uno's digital pin 11. CLK pin with digital pin 13 and LOAD pin with digital pin 10.

“x_direction” and “y_direction” correspond to the speed of the snake in a given direction. The starting velocity is 1 in the x-direction.

The snake is defined as a 2-dimensional array. 64 corresponds to the total number of dots on the matrix (where the snake can be) and 2 corresponds to the x and y value of that dot.

The apple is declared as an array that will contain 2 values, x and y coordinate of the apple. Another variable is "game_over" which's type is boolean. It is defined to check if the game is ended or not.

4. Setup Function

```
void setup() {

  lc.shutdown(0, false);    // Open matrix
  lc.setIntensity(0, 15);   // Set brightness of the matrix
  lc.clearDisplay(0);       // Clear the display of the matrix

  joystick = new AxisJoystick(SW_PIN, VRX_PIN, VRY_PIN);

  // Initialize the pseudo-random number generator
  randomSeed(analogRead(0));
  snake[0][0] = random(0, 8);
  snake[0][1] = random(0, 8);

  // Extend the length of the snake to starting length
  for (int i = snake_length; i < START_LENGTH; i++) {
    extendSnake();
  }

  createApple();  // Create a new apple

  for (int x = 0; x < numDevices; x++) {
    lc.shutdown(x, false);
    lc.setIntensity(x, 8);
    lc.clearDisplay(x);
  }
}
```

The setup() function is called when the sketch starts. It is used to initialize variables, pin modes, etc. First of all, we should wake up the led display and do some setup (like setting brightness, clearing the display). Then we created an AxisJoystick object with defined pins.

randomSeed() initializes the pseudo-random number generator. It is used to define the starting location of the snake randomly. "snake[0][0]" corresponds to the x value and "snake[0][1]" is the y value. After the first dot of the snake is found randomly, the length of the snake is extended from 1 to starting length which was defined before as 3. With the createApple() function, the x and y value of the apple is generated randomly. In for loop, for each MAX7219 device, matrixes are waking up and doing setup but in this case, it is only one.

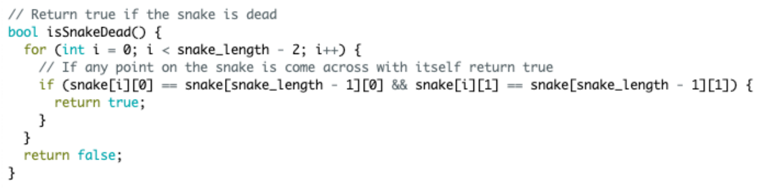
5. Loop Function

```
void loop() {
  bool game_start = false; // Boolean to check if game is started

  // Display "Press Play" text until player presses on joystick's button
  scrollMessage(scrollTextStart);

  // The player pressed on the start button
  game_start = true;


  while (game_start) {

    if (isSnakeDead()) {
      // The snake is dead, "Game Over" text is displayed
      lc.clearDisplay(0);
      scrollMessage(scrollText);
      game_start = false;

      // Turn the snake values to default
      snake_length = 1;
      snake[0][0] = random(0, 8);
      snake[0][1] = random(0, 8);

      for (int i = snake_length; i < START_LENGTH; i++) {
        extendSnake();
      }

      createApple();
      lc.clearDisplay(0);

      for (int x = 0; x < numDevices; x++)
      {
        lc.shutdown(x, false);      //The MAX72XX is in power-saving mod
        lc.setIntensity(x, 8);      // Set the brightness to default val
        lc.clearDisplay(x);         // and clear the display
      }
      return;
    }

    // The game is started
    checkDirection(joystick->singleRead());
    moveSnake();
    checkApple();
    draw();
    delay(150);
  }
}
```

In the loop function, we defined a boolean "game_start" to check if the game is started by the player. If there is no response in the joystick’s button, the "PRESS START" text will be scrolled for one time. When the player presses on the start button, "game_start" variable turns to true, and the game keeps running until the snake is dead. In the while loop, we check if the game has ended or not with the isSnakeDead() function. If the snake is dead, we cleared the led matrix’s screen and "GAME OVER" text starts displaying on the matrix, and all values of the snake will turn to the initial state. If the snake is not dead, we check joystick inputs with the checkDirection function and move the snake with received input from the user. The apple will be checked if it is eaten or not and then the moved snake and apple will be drawn to the screen again.

5. isSnakeDead Function

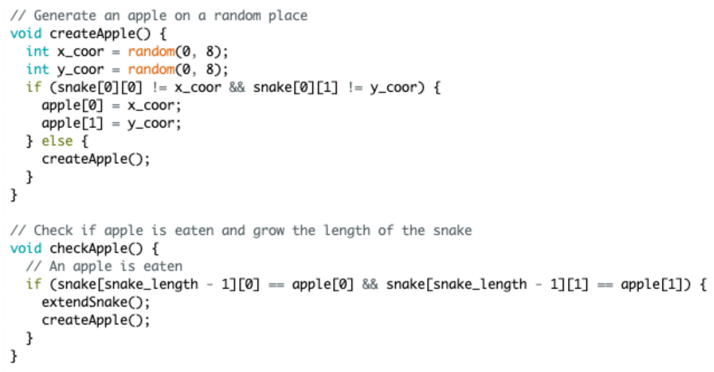
This function checks whether the snake is dead or not. For each point on the snake check if the x and y values are the same as the snake's last point. If it is that means the snake collides with itself and return true.



```
// Return true if the snake is dead
bool isSnakeDead() {
  for (int i = 0; i < snake_length - 2; i++) {
    // If any point on the snake is come across with itself return true
    if (snake[i][0] == snake[snake_length - 1][0] && snake[i][1] == snake[snake_length - 1][1]) {
      return true;
    }
  }
  return false;
}
```

6. checkApple and createApple Functions

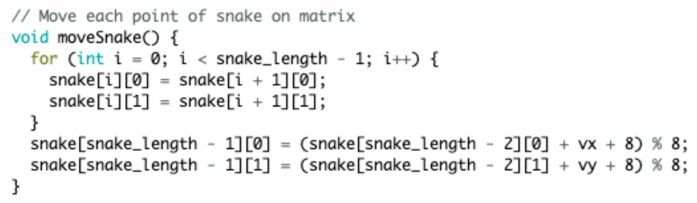
checkApple() function checks if the apple is eaten or not. If the x and y values of the apple are equals to the x and y values of the last point of the snake, the length of the snake extends and we create a new apple with createApple() function.



```
// Generate an apple on a random place
void createApple() {
  int x_coor = random(0, 8);
  int y_coor = random(0, 8);
  if (snake[0][0] != x_coor && snake[0][1] != y_coor) {
    apple[0] = x_coor;
    apple[1] = y_coor;
  } else {
    createApple();
  }
}

// Check if apple is eaten and grow the length of the snake
void checkApple() {
  // An apple is eaten
  if (snake[snake_length - 1][0] == apple[0] && snake[snake_length - 1][1] == apple[1]) {
    extendSnake();
    createApple();
  }
}
```

7. moveSnake Function

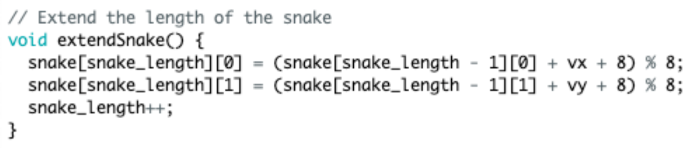
moveSnake() function replaces each point on the snake with the next one. For the last point, we add velocity x and y to the previous one and add 8 to prevent minus values. Then we apply the modulo operator to get a value that is not greater than 8.



```
// Move each point of snake on matrix
void moveSnake() {
  for (int i = 0; i < snake_length - 1; i++) {
    snake[i][0] = snake[i + 1][0];
    snake[i][1] = snake[i + 1][1];
  }
  snake[snake_length - 1][0] = (snake[snake_length - 2][0] + vx + 8) % 8;
  snake[snake_length - 1][1] = (snake[snake_length - 2][1] + vy + 8) % 8;
}
```

8. extendSnake Function

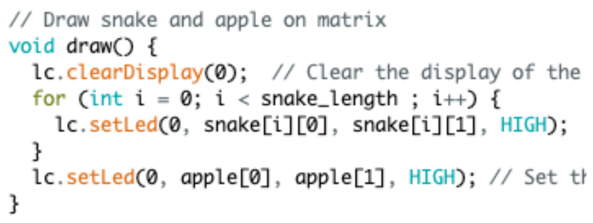
This function extends the length of the snake. It is called in setup function and every time when the snake eats an apple.



```
// Extend the length of the snake
void extendSnake() {
  snake[snake_length][0] = (snake[snake_length - 1][0] + vx + 8) % 8;
  snake[snake_length][1] = (snake[snake_length - 1][1] + vy + 8) % 8;
  snake_length++;
}
```

9. draw Function

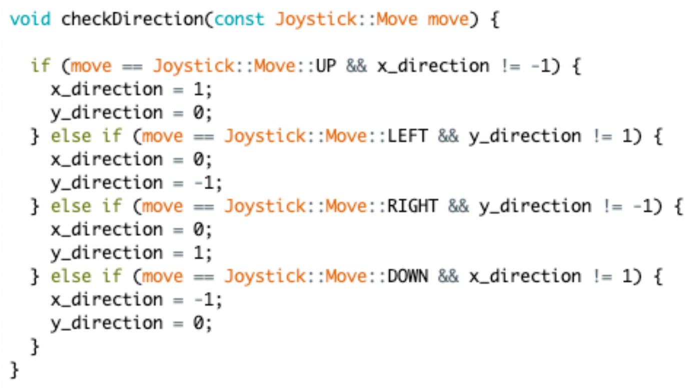
The draw function clears the display every time, sets the apple and each point of the snake on the matrix.



```
// Draw snake and apple on matrix
void draw() {
  lc.clearDisplay(0);  // Clear the display of the
  for (int i = 0; i < snake_length ; i++) {
    lc.setLed(0, snake[i][0], snake[i][1], HIGH);
  }
  lc.setLed(0, apple[0], apple[1], HIGH); // Set t
}
```

10. checkDirection Function

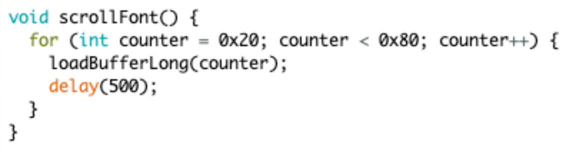
checkDirection function gets the input from the joystick and changes the direction of the snake according to inputs.



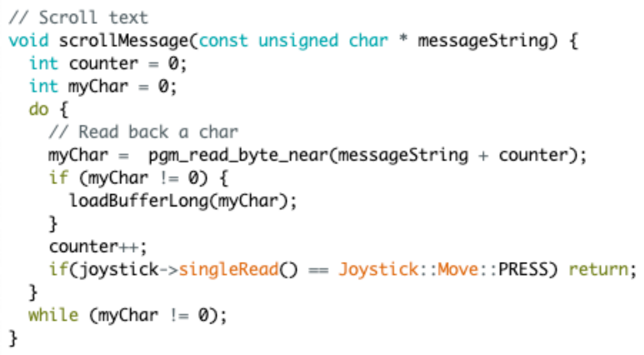
```
void checkDirection(const Joystick::Move move) {

  if (move == Joystick::Move::UP && x_direction != -1) {
    x_direction = 1;
    y_direction = 0;
  } else if (move == Joystick::Move::LEFT && y_direction != 1) {
    x_direction = 0;
    y_direction = -1;
  } else if (move == Joystick::Move::RIGHT && y_direction != -1) {
    x_direction = 0;
    y_direction = 1;
  } else if (move == Joystick::Move::DOWN && x_direction != 1) {
    x_direction = -1;
    y_direction = 0;
  }
}
```

11. scrollFont and scrollMessage Functions

These two functions are for scrolling text on the LED matrix. Each char on text is read and call “loadBufferLong“ function to load the character into scroll buffer. Then the character is moved every time by "rotateBufferLong" function and displayed again on the LED matrix with "printBufferLong" function. This and following 4 functions are taken from a tutorial to print text on LED matrix [4].

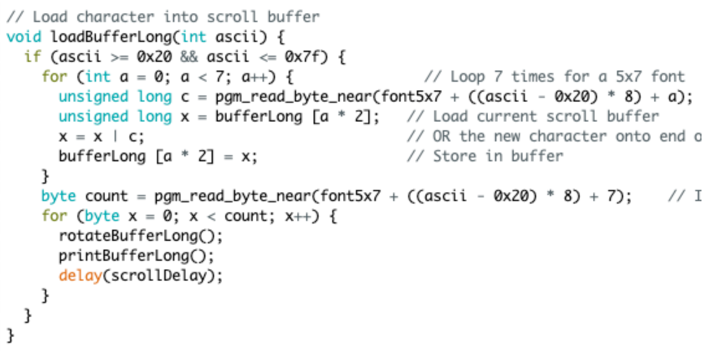


```
void scrollFont() {
  for (int counter = 0x20; counter < 0x80; counter++) {
    loadBufferLong(counter);
    delay(500);
  }
}
```



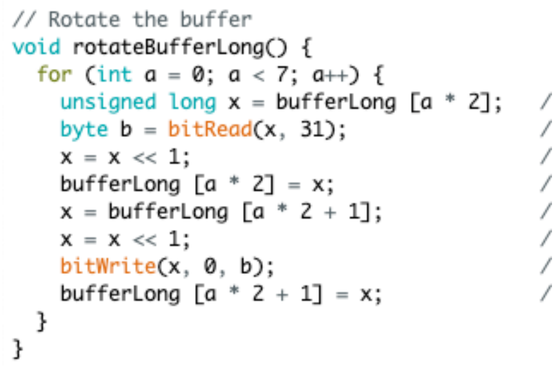
```
// Scroll text
void scrollMessage(const unsigned char * messageString) {
  int counter = 0;
  int myChar = 0;
  do {
    // Read back a char
    myChar =  pgm_read_byte_near(messageString + counter);
    if (myChar != 0) {
      loadBufferLong(myChar);
    }
    counter++;
    if(joystick->singleRead() == Joystick::Move::PRESS) return;
  }
  while (myChar != 0);
}
```

12. loadBufferLong Function

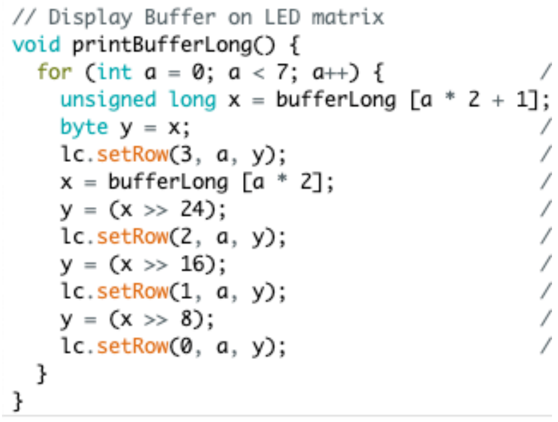


```
// Load character into scroll buffer
void loadBufferLong(int ascii) {
  if (ascii >= 0x20 && ascii <= 0x7f) {
    for (int a = 0; a < 7; a++) {                    // Loop 7 times for a 5x7 font
      unsigned long c = pgm_read_byte_near(font5x7 + ((ascii - 0x20) * 8) + a);
      unsigned long x = bufferLong [a * 2];     // Load current scroll buffer
      x = x | c;                                // OR the new character onto end o
      bufferLong [a * 2] = x;                   // Store in buffer
    }
    byte count = pgm_read_byte_near(font5x7 + ((ascii - 0x20) * 8) + 7);     // I
    for (byte x = 0; x < count; x++) {
      rotateBufferLong();
      printBufferLong();
      delay(scrollDelay);
    }
  }
}
```

13. rotateBufferLong and printBufferLong Functions



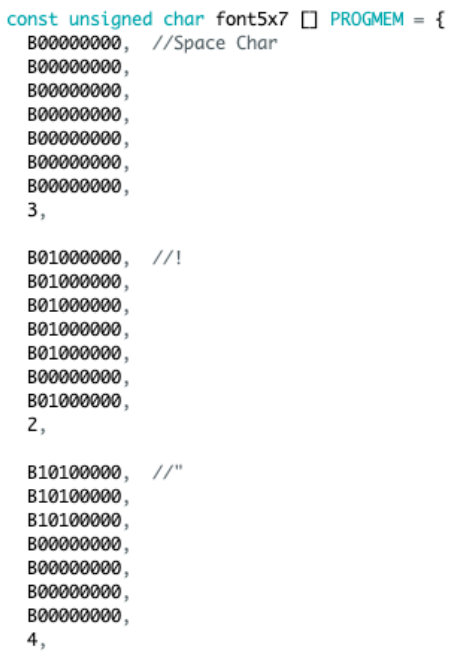
```
// Rotate the buffer
void rotateBufferLong() {
  for (int a = 0; a < 7; a++) {
    unsigned long x = bufferLong [a * 2];     /
    byte b = bitRead(x, 31);                  /
    x = x << 1;                               /
    bufferLong [a * 2] = x;                   /
    x = bufferLong [a * 2 + 1];               /
    x = x << 1;                               /
    bitWrite(x, 0, b);                        /
    bufferLong [a * 2 + 1] = x;               /
  }
}
```



```
// Display Buffer on LED matrix
void printBufferLong() {
  for (int a = 0; a < 7; a++) {               /
    unsigned long x = bufferLong [a * 2 + 1];
    byte y = x;                               /
    lc.setRow(3, a, y);                       /
    x = bufferLong [a * 2];                   /
    y = (x >> 24);                            /
    lc.setRow(2, a, y);                       /
    y = (x >> 16);                            /
    lc.setRow(1, a, y);                       /
    y = (x >> 8);                             /
    lc.setRow(0, a, y);                       /
  }
}
```

14. Numeric Font Matrix

Numeric font matrix is for holding each character.



```
const unsigned char font5x7 [] PROGMEM = {
  B00000000,  //Space Char
  B00000000,
  B00000000,
  B00000000,
  B00000000,
  B00000000,
  B00000000,
  3,

  B01000000,  //!
  B01000000,
  B01000000,
  B01000000,
  B01000000,
  B00000000,
  B01000000,
  2,

  B10100000,  //"
  B10100000,
  B10100000,
  B00000000,
  B00000000,
  B00000000,
  B00000000,
  4,
```

The project components are an Arduino Uno, a breadboard an 8x8 LED matrix, a MAX7219 dot matrix module, jumper wires, and a thumb joystick.

The first component is Arduino Uno which is a microcontroller board based on the ATmega328P (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator (CSTCE16M0V53-R0), a USB connection, a power jack, an ICSP header, and a reset button.

The second and third components are the 8x8 LED matrix and MAX7219 dot matrix module. The MAX7219 LED driver can be used to control 7-segment displays up to 8 digits, bar-graph displays, or 64 individual LEDs. The driver communicates with the Arduino through SPI so we only need three wires to control the display.

The fourth component is the joystick. The Analog Joystick is similar to two potentiometers connected together, one for the vertical movement (Y-axis) and the other for the horizontal movement (X-axis). We check the direction of the snake with movement. The joystick also comes with a Select switch. With the select switch, we check if the player started the game.

First of all, we connected our joystick with female-to-male cables to Arduino Uno and breadboard. The connections of pins are shown in the table down below.

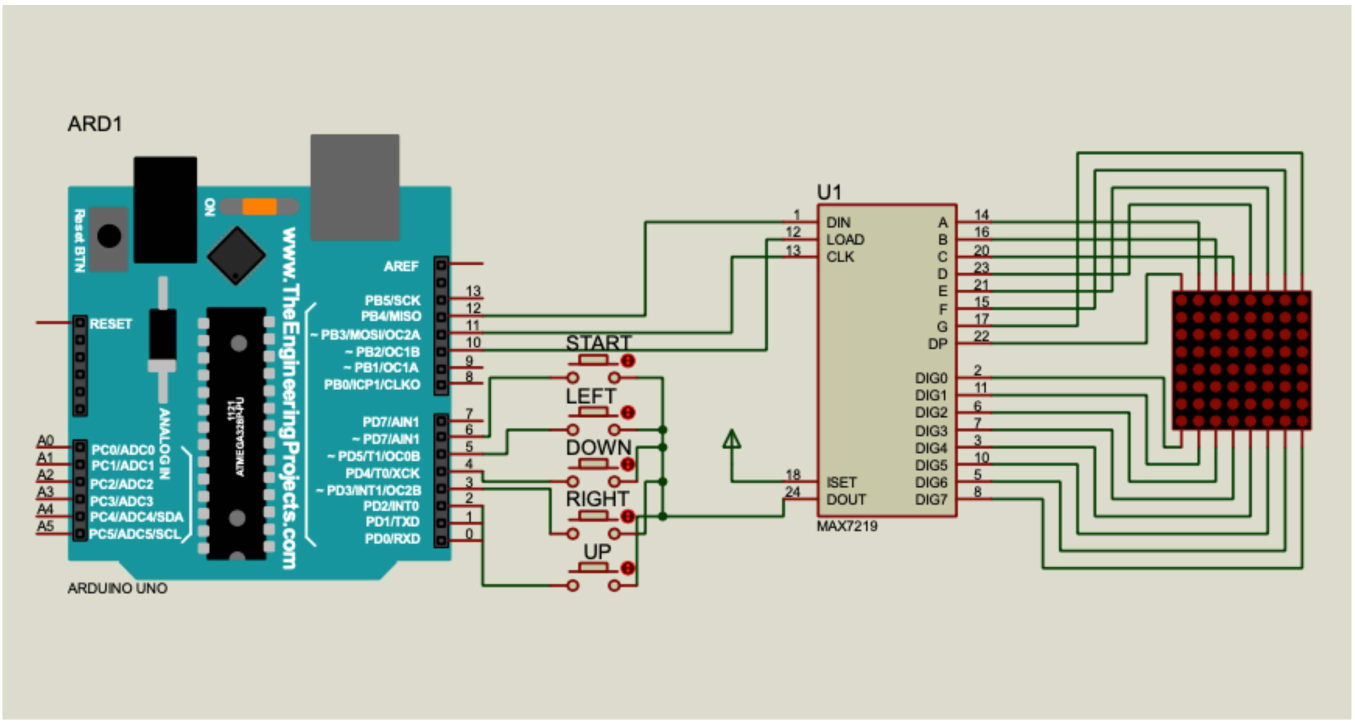
Joystick Pin	Arduino Uno Pin
GND	GND
+5V	5V
VRX	A1
VRY	A0
SW	A2

Secondly, we connected Led Dot Matrix with the same cables to Arduino Uno and breadboard. We also showed the connections of pins on the second table, which is down below.

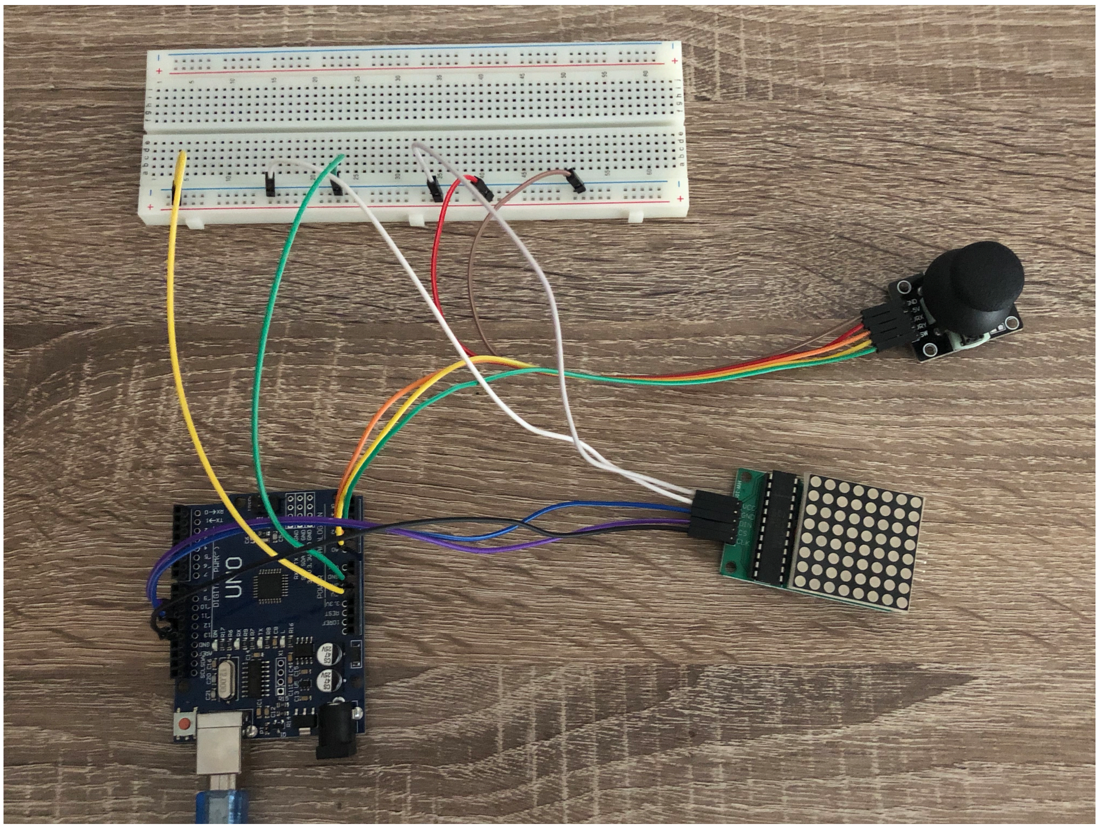
Led Matrix Pin	Arduino Uno Pin
VCC	5V
GND	GN
DIN	11
CS	10
CLK	13

Then we connected the Arduino Uno to our computer with a USB cable. To connect it, we choose the Arduino type and the port in Arduino IDE from the tools menu. In this IDE, we verified our code and upload it to the board and everything worked well as expected.

1. Simulation



Circuit



REFERENCES

1. https://github.com/wayoda/LedControl
2. https://github.com/MHeironimus/ArduinoJoystickLibrary
3. https://github.com/YuriiSalimov/AxisJoystick
4. https://electronoobs.com/eng_arduino_tut56.php