VIT®

(Estd. u/s 3 of UGC Act 1956)

SNAKE GAME USING ARDUINO

PROJECT FINAL REVIEW

Submitted for CAL in B. Tech Microprocessors and Interfacing
CSE2006

Faculty: - Prof. Sasipriya P
Slot-G2

(SCHOOL OF COMPUTER SCIENCE AND ENGINEERING)

GROUP MEMBERS: -

Piyush Bamel 16BCE1221
Kaustubh Nagar 16BCE1338
Aritra Banerjee 16BCE1339
Deshna Puitandi 16BCE1317

CERTIFICATE

This is to certify that the project "**Snake Game using Arduino**" submitted under course **Microprocessor and Interfacing CSE2006** to **Prof. Sasipriya P** is completed by our group comprising of **Piyush Bamel (16BCE1221), Deshna Puitandi (16BCE1317), Kaustubh Nagar (16BCE1338) and Aritra Banerjee (16BCE1317)**.

It was under the able guidance of our teacher Prof. Sasipriya P.

SIGNATURE OF FACULTY
DATE:

ACKNOWLEDGEMENT

*We would like to express our special thanks of gratitude to our **Microprocessor and Interfacing** faculty, **Prof. Sasipriya P**, who gave us the golden opportunity to do this wonderful project on the topic "***Snake Game using Arduino***", which also helped us in doing a lot of Research and we came to know about so many new things. We are thankful to you. Secondly, we would also like to thank our group members in finalizing this project within the limited time frame.*

INTRODUCTION

The snake game is a very absorbing and interesting game. It is simple to understand but hard to master. The aim of the game is pretty simple. What we basically have to do is make the snake eat the food particle by moving it upwards downwards and sideways. As the snake eats the food its size progressively increases and the difficulty as it gradually becomes an assiduous task to maneuver the snake given that its speed also increases at a brisk pace. To put it succinctly this easily qualifies as one of the simplest, yet challenging game ever made.

As far as the functionality of this project is concerned we can extend it to handheld console or arcade machine. Games like Atari Breakout, Solitaire, Pac-Man etc.

COMPONENTS USED

- **ARDUINO UNO**

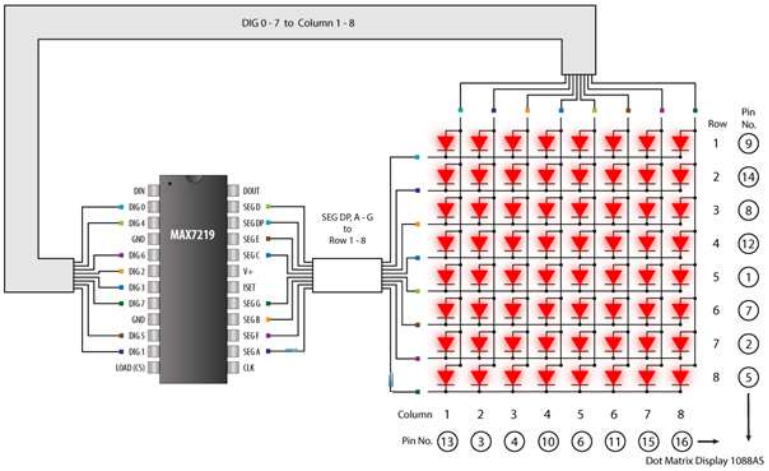
The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. "Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards.

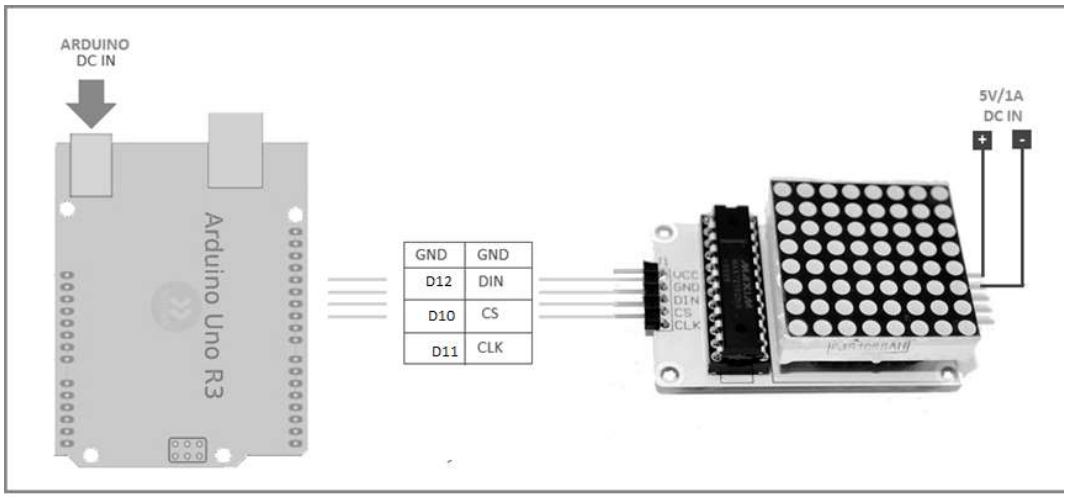
SPECIFICATION

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

- **LED MATRIX MAX7219**

The IC is capable of driving 64 individual LEDs while using only 3 wires for communication with the Arduino, and what's more we can daisy chain multiple drivers and matrixes and still use the same 3 wires.
The 64 LEDs are driven by 16 output pins of the IC. The question now is how is that possible. Well the maximum number of LEDs light up at the same time is actually eight. The LEDs are arranged as 8×8 set of rows and columns. So, the MAX7219 activates each column for a very short period of time and at the same time it also drives each row. So, by rapidly switching through the columns and rows the human eye will only notice a continuous light.





- **JOYSTICK MODULE 2-AXIS**

  This device contains two independent potentiometers (one per axis) for reporting the joystick's position, with wiring options for voltage or resistance outputs. Its modular form-factor allows you to plug the 2-Axis Joystick directly into a breadboard for easy prototyping. Comfortable and convenient to use, this joystick features a comfortable cup-type knob, and springs that auto-return to the center position.

Key Features:

- Easy breadboard connection
- Two independent 10 K potentiometers with common ground
- Spring auto-return to center
- Comfortable cup-type knob

- **JUMPER WIRES**

  A jump wire (also known as jumper, jumper wire, jumper cable, DuPont wire, or DuPont cable – named for one manufacturer of them) is an electrical wire or group of them in a cable with a connector or pin at each end (or sometimes without them – simply "tinned"), which is normally used to interconnect the components of a breadboard or other prototype or test circuit, internally or with other equipment or components, without soldering.

  There are different types of jumper wires. Some have the same type of electrical connector at both ends, while others have different connectors. Some common connectors are:

  Solid tips – are used to connect on/with a breadboard or female header connector. The arrangement of the elements and ease of insertion on a breadboard allows increasing the mounting density of both components and jump wires without fear of short-circuits. The jump wires vary in size and color to distinguish the different working signals.
  Crocodile clips – are used, among other applications, to temporarily bridge sensors, buttons and other elements of prototypes with components or equipment that have arbitrary connectors, wires, screw terminals, etc.
  Banana connectors – are commonly used on test equipment for DC and low-frequency AC signals.
  Registered jack (RJnn) – are commonly used in telephone (RJ11) and computer networking (RJ45).

RCA connectors – are often used for audio, low-resolution composite video signals, or other low-frequency applications requiring a shielded cable.
RF connectors – are used to carry radio frequency signals between circuits, test equipment, and antennas.

- **IC 7809**

7809 is a voltage regulator integrated circuit(IC) which is widely used in electronic circuits. Voltage regulator circuit can be manually built using parts available in the market but it will take a lot of time to assemble those parts on a PCB. Secondly, the cost of those parts is almost equal to the price of 7809 itself so professionals usually prefer to use 7809 IC instead of making a voltage regulator circuit from scratch.
The 7809 regulator line is a positive voltage regulator that is the 7809 voltage regulator ic generate the voltage which is positive with respect to the common ground. In case if both the positive and negative voltage supply is needed in the same circuit. The voltage regulator 7809 is combined with its corresponding 79XX family IC. The voltage regulator 7809 is available most commonly in TO-220 package, more over the IC is also available in TO-3, TO-92 and surface mount Packages.

The 7809 Voltage regulators do operate at their optimal capability if the input voltage is at least 2.5 volt greater than the output voltage and the current is 1 or 1.5 Amperes more. Though the voltage and current different is different for other IC Packages.

- **BREAD BOARD**

A breadboard is a solderless device for temporary prototype with electronics and test circuit designs. Most electronic components in electronic circuits can be interconnected by inserting their leads or terminals into the holes and then making connections through wires where appropriate. The breadboard has strips of metal underneath the board and connect the holes on the top of the board. The metal strips are laid out as shown below. Note that the top and bottom rows of holes are connected horizontally and split in the middle while the remaining holes are connected vertically.

- **POWER SUPPLY**

CODE FOR THE PROJECT IMPLEMENTATION

```
#include "LedControlMS.h"

LedControl lc = LedControl(12, 11, 10, 1);

int had[8][8] = {0};

byte xHad = 4;
byte yHad = 5;
byte xaxis = 0;
byte yaxis = 0;

byte directi = 0;
int step1 = 1;
long time1 = 0;
int speed1 = 500;
byte score1 = 0;

boolean snooze1 = false;

int num[11][7] = {
  {B00000000, B00000000, B00000000, B00000000, B00000001, B01111111,
B00100001},
  {B00000000, B00000000, B00110001, B01001001, B01000101, B00100011},
  {B00000000, B00000000, B01000110, B01101001, B01010001, B01000001,
B01000010},
  {B00000000, B00001000, B11111111, B01001000, B00101000, B00011000,
B00001000},
  {B00000000, B00000000, B00111110, B01000101, B01001001, B01010001,
B00111110},
  {B00000000, B00000000, B00000110, B01001001, B01001001, B00101001,
B00011110},
  {B00000000, B00000000, B01110000, B01001000, B01000111, B01000000,
B01100000},
  {B00000000, B00000000, B00110110, B01001001, B01001001, B01001001,
B00110110},
  {B00000000, B00000000, B00111100, B01001010, B01001001, B01001001,
B00110000},
  {B00000000, B00000000, B00111110, B01010001, B01001001, B01000101,
B00111110},
  {B00000000, B00000000, B00000000, B00000000, B00000000, B00000000,
B00000000},
```

```
};

void setup() {
  Serial.begin(9600);
  Serial.println("Your game starts");
  delay(1000);



  randomSeed(analogRead(A0));
  genseed();
  lc.shutdown(0, false);
  lc.setIntensity(0, 1);
  byte h[7]=
{B10000000,B10000000,B10110000,B11001000,B10001000,B10001000,B10001000};
  lc.setRow(0,0,h[0]);
  lc.setRow(0,1,h[1]);
  lc.setRow(0,2,h[2]);
  lc.setRow(0,3,h[3]);
  lc.setRow(0,4,h[4]);
  lc.setRow(0,5,h[5]);
  lc.setRow(0,6,h[6]);
  delay(500);

byte e[7]=
{B00000000,B00000000,B01110000,B10001000,B11111000,B10000000,B01110000};
  lc.setRow(0,0,e[0]);
  lc.setRow(0,1,e[1]);
  lc.setRow(0,2,e[2]);
  lc.setRow(0,3,e[3]);
  lc.setRow(0,4,e[4]);
  lc.setRow(0,5,e[5]);
  lc.setRow(0,6,e[6]);
  delay(500);


byte
y[7]={B00000000,B00000000,B10001000,B10001000,B01111000,B00001000,B011100
00};
lc.setRow(0,0,e[0]);
  lc.setRow(0,1,y[1]);
  lc.setRow(0,2,y[2]);
  lc.setRow(0,3,y[3]);
```

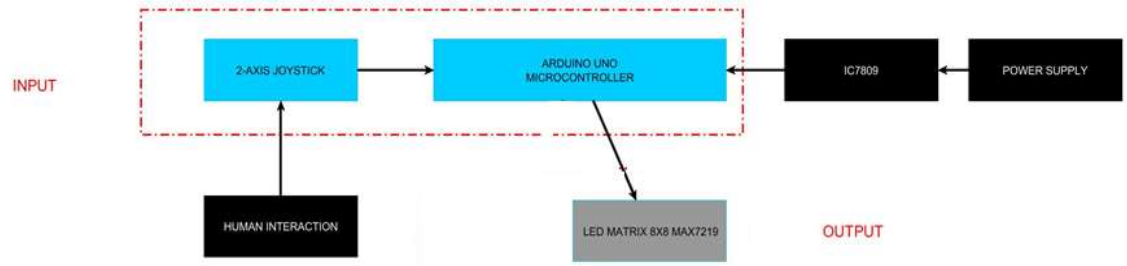
```
  lc.setRow(0,4,y[4]);
  lc.setRow(0,5,y[5]);
  lc.setRow(0,6,y[6]);
  delay(500);

  byte hf[8]=
{B00111100,B01000010,B10100101,B10000001,B10100101,B10011001,B01000010,B
00111100};
  lc.setRow(0,0,hf[0]);
  lc.setRow(0,1,hf[1]);
  lc.setRow(0,2,hf[2]);
  lc.setRow(0,3,hf[3]);
  lc.setRow(0,4,hf[4]);
  lc.setRow(0,5,hf[5]);
  lc.setRow(0,6,hf[6]);
  lc.setRow(0,7,hf[7]);
  delay(500);
  lc.clearDisplay(0);
}

void loop() {

  if ((analogRead(A0) < 200) && (directi != 0) && (directi != 1)) directi = 1;
  if ((analogRead(A0) > 800) && (directi != 1) && (directi != 0)) directi = 0;
  if ((analogRead(A1) < 200) && (directi != 3) && (directi != 2)) directi = 3;
  if ((analogRead(A1) > 800) && (directi != 2) && (directi != 3)) directi = 2;

  if ((millis() - time1) > speed1) {
    time1 = millis();

    delfood(had);
    score();
    mov1();

    if (xHad == xaxis && yHad == yaxis) snooze1 = true;
    if (had[xHad][yHad] != 0) GameOver();

    step1++;
    had[xHad][yHad] = step1;
  }
  draw(had, xaxis, yaxis);
}
```

```
void score() {
  if (snooze1 == true) {
    snooze1 = false;
    genseed();

    score1++;
    speed1 -= 20;

    Serial.print ("Your score is ");
    Serial.println (score1);
  }
}

void draw(int matrix1[8][8], byte x, byte y) {
  for (int i = 0; i < 8; i++) {
    for (int j = 0; j < 8; j++) {
      if ((matrix1[i][j] >= 1) || ((i == x) && (j == y) )) lc.setLed(0, i, j, true);
      else lc.setLed(0, i, j, false);
    }
  }
}

void delfood(int matrix1[8][8]) {
  for (int i = 0; i < 8; i++) {
    for (int j = 0; j < 8; j++) {
      if (matrix1[i][j] == (step1 - score1)) matrix1[i][j] = 0;
    }
  }
}

void mov1() {
  switch (directi) {
    case 0:
      xHad++;
      break;
    case 1:
      xHad--;
      break;
    case 2:
      yHad++;
      break;
    case 3:
      yHad--;
```

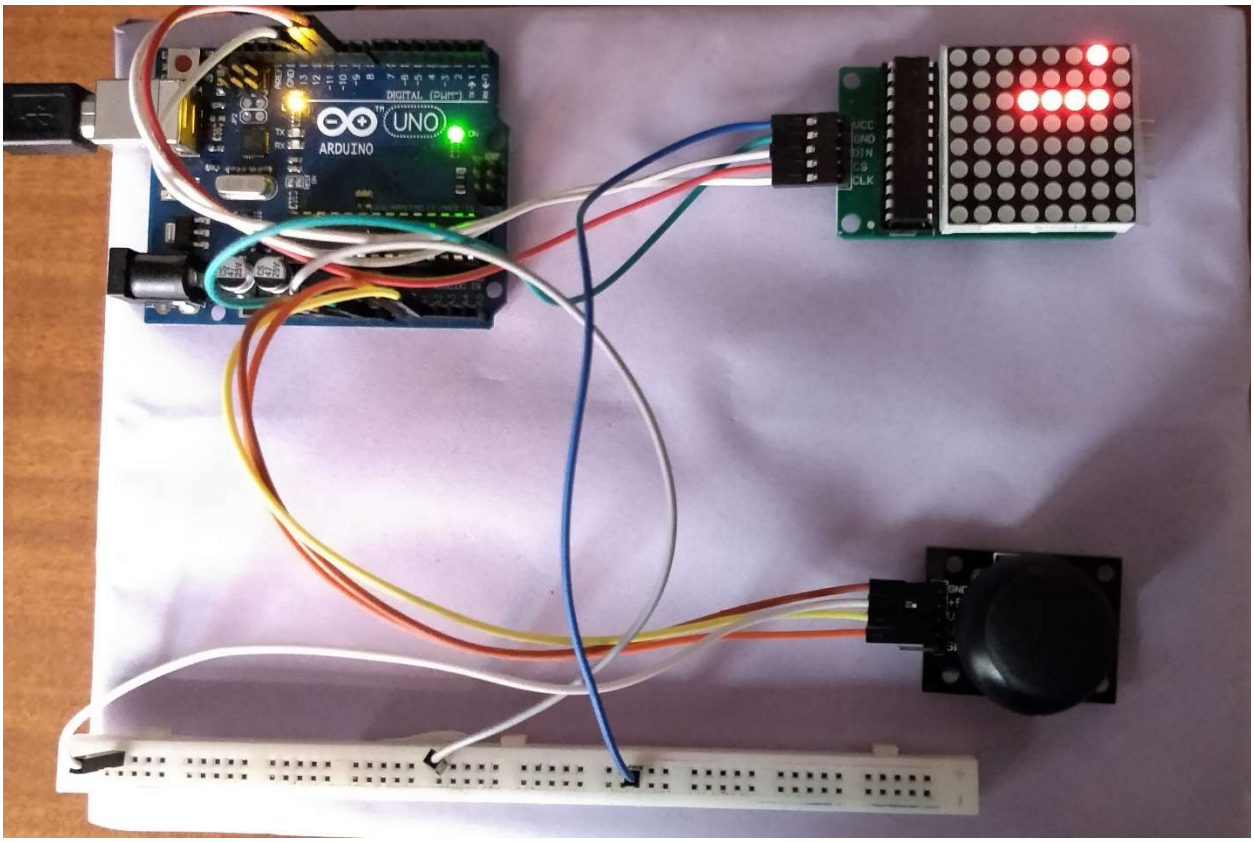
```
      break;
    }
    if (xHad == 8) xHad = 0;
    if (yHad == 8) yHad = 0;
    if (xHad == 255) xHad = 7;
    if (yHad == 255) yHad = 7;
}

void genseed() {
  lc.setLed(0, xaxis, yaxis, false);
  do {
    xaxis = random(0, 8);
    yaxis = random(0, 8);
  } while (had[xaxis][yaxis] != 0 );
}

void draw1(byte num1) {
  if (num1 == 0) num1 = 10;
  for (int col = 0; col < 7; col++) {
    lc.setRow(0, col, num[num1 - 1][col]);
  }
}

void GameOver() {
  Serial.println ("Game over");
  for (int row = 0; row < 8; row++) {
    for (int col = 0; col < 8; col++) {

      lc.setLed(0, col, row, true);
      delay(25);
    }
  }
  for (int row = 0; row < 8; row++) {
    for (int col = 0; col < 8; col++) {
      lc.setLed(0, col, row, false);
      delay(25);
    }
  }
  do {
    draw1((score1 / 10) % 10);
    delay(200);
    draw1(score1 % 10);
    delay(200);
```

```
    draw1(11);
    delay(1000);
  } while (true);
}
```

BLOCK DIAGRAM



IMPLEMENTATION IMAGES



RESULT

We have successfully implemented snake game on Arduino Uno which is controlled by 2-axis joystick and displayed on 8x8 LED matrix.