GEOMETRY GLOBAL INC.

Microcontroller Project Cse 316

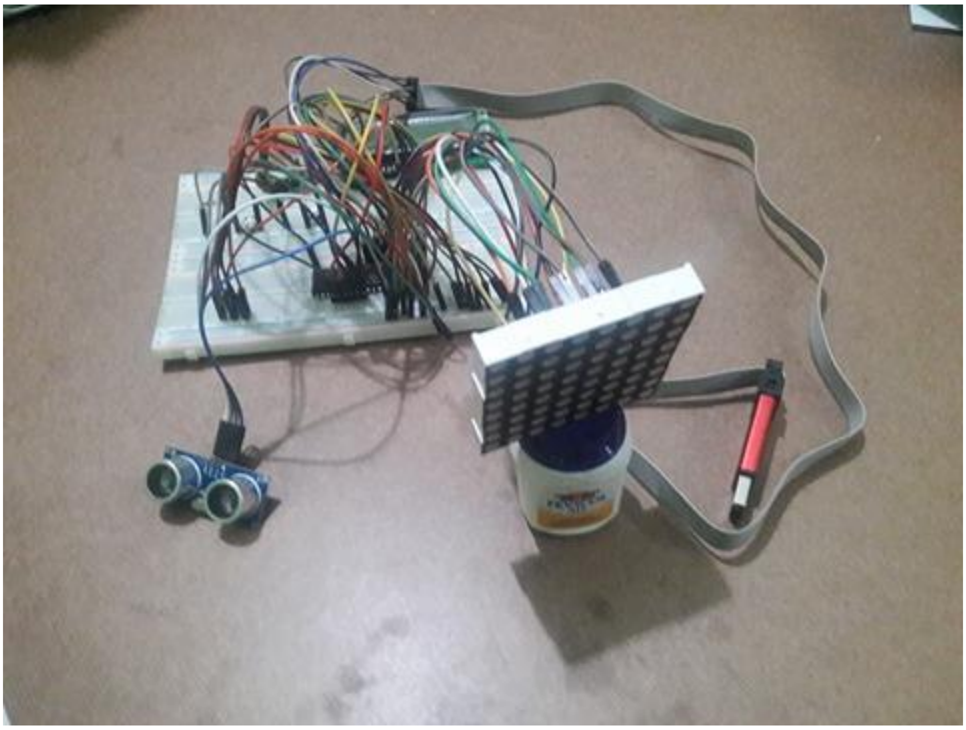
A YEAR OF ACCOMPLISHMENTS

Created by
Anindya Biswas(1405006)
Abhishek Agarwala(1405028)

PROJECT NAME

Our project is snake game

Here is a snapshot of the project:



Our project video link is given below:

https://www.youtube.com/watch?v=CagzMsUUZa8&t=2s

TABLE OF CONTENTS

Introduction

The project is a snake game and as it is a game its only purpose is to provide entertainment. The project starts by displaying a snake using a 8*8 dot matrix. After certain interval of time the dot matrix will update the snake's position in a particular direction. The AtMega32 will also generate a food displayed with a dot in the dot matrix. The user can change the direction of the snake with the help of a sonor sensor HC-SR014. They need to place an object in front of the sonor. If the distance of the object from sensor is greater than its previous position then the snake will move to left or up depending how it is moving and if the distance is smaller then the snake will move right or down depending on its movement. This is how the user can navigate the snake. We have also kept a score display by using an lcd. It would update the score by 5 after everytime the snake eats the food. If the snake goes out of the matrix or catches its body while moving the game will over showing alphabet X. This is our game.

Hardware Requirements

List of hardware used and its cost:

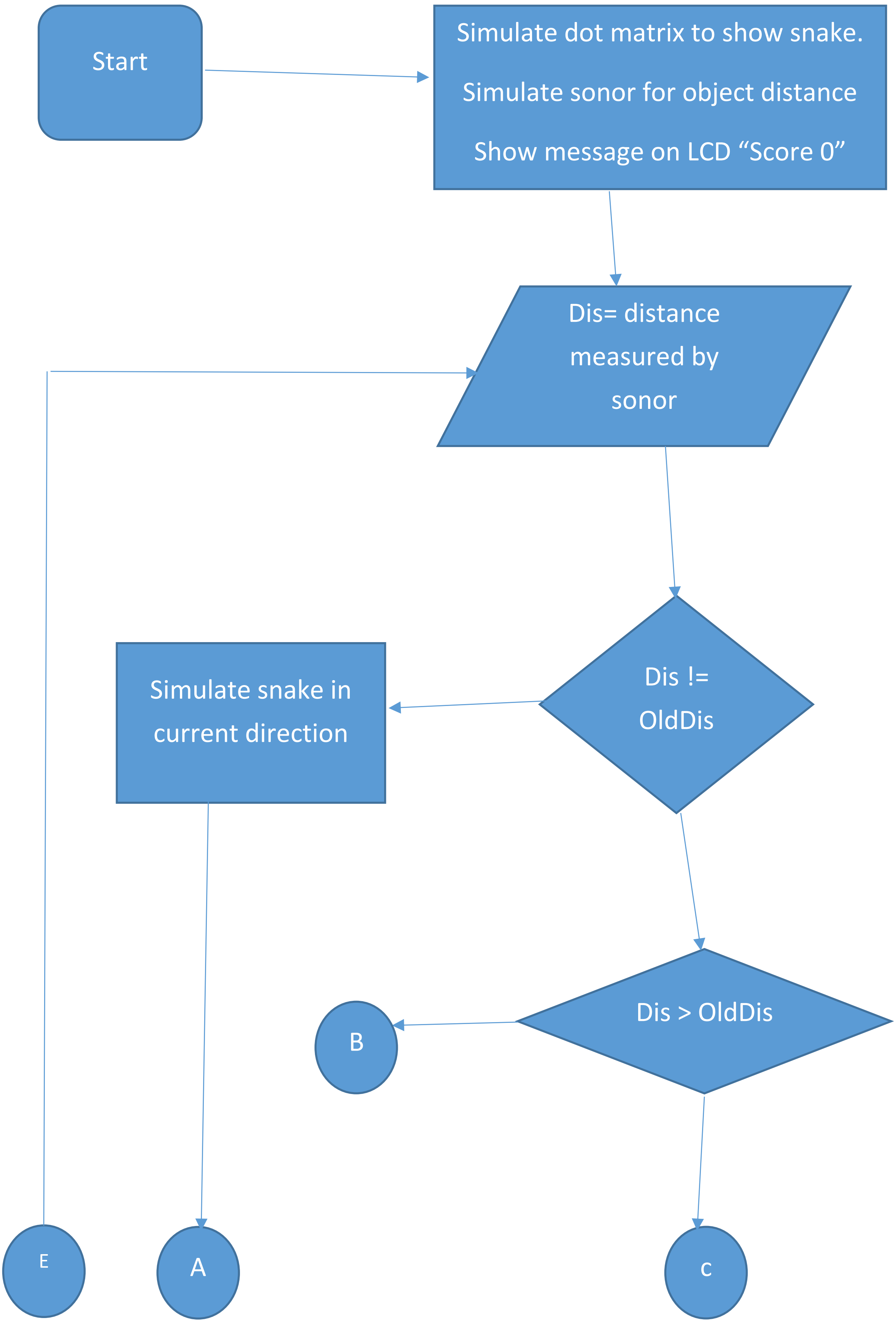
Equipments	Cost(Tk)
Atmega32 microcontroller	150
16x2 LCD Alphanumeric Display	200
1 Breadboard	300
2 Push Buttons	20
Ultrasonic Sensor HC-SR04	70
Many male to male wires	120
Many male to female wires	60

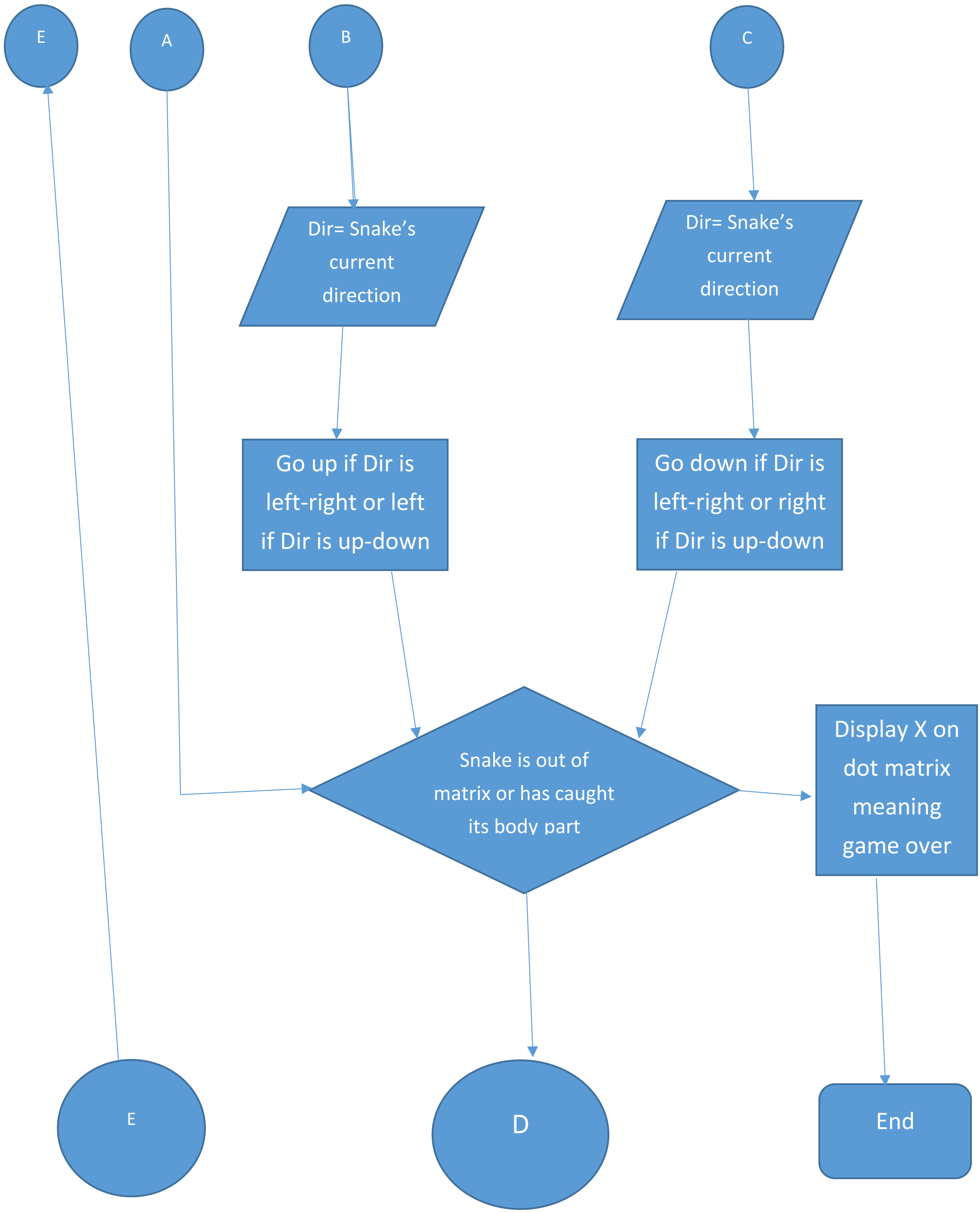
Software Requirements

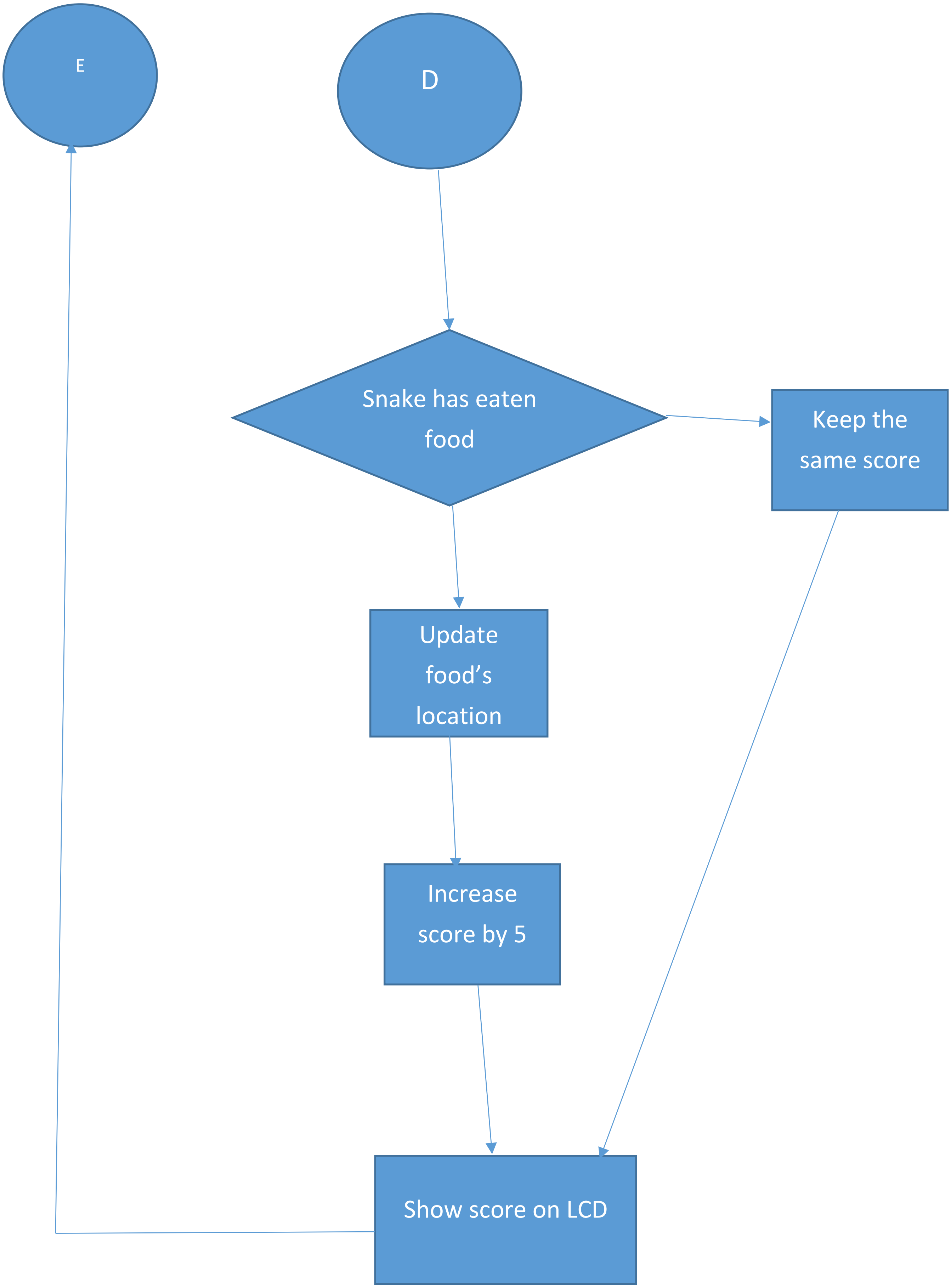
List of Softwares used:

- ATmel Studio 7 (to compile .c code and build .hex and .eep file)
- eXtreme Burner - AVR (to load .hex and .eep file onto ATmega32)
- Proteus 8 Professional (for circuit design)

Flowchart

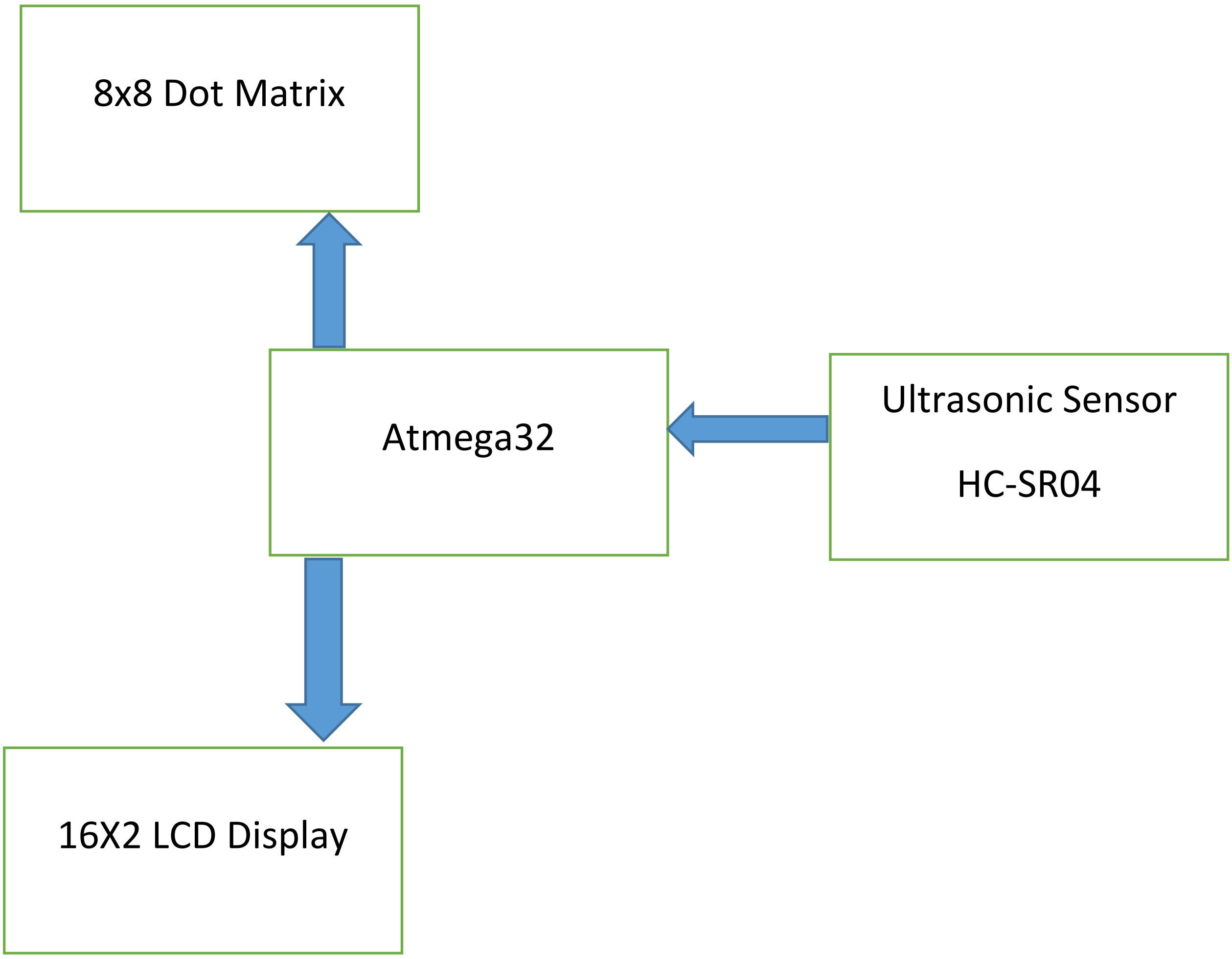

Start
Simulate dot matrix to show snake.
Simulate sonor for object distance
Show message on LCD “Score 0”
Dis= distance measured by sonor
Simulate snake in current direction
Dis != OldDis
B
Dis > OldDis
E
A
C


E
A
B
C
Dir= Snake's current direction
Dir= Snake's current direction
Go up if Dir is left-right or left if Dir is up-down
Go down if Dir is left-right or right if Dir is up-down
Snake is out of matrix or has caught its body part
Display X on dot matrix meaning game over
E
D
End


E
D
Snake has eaten food
Keep the same score
Update food's location
Increase score by 5
Show score on LCD

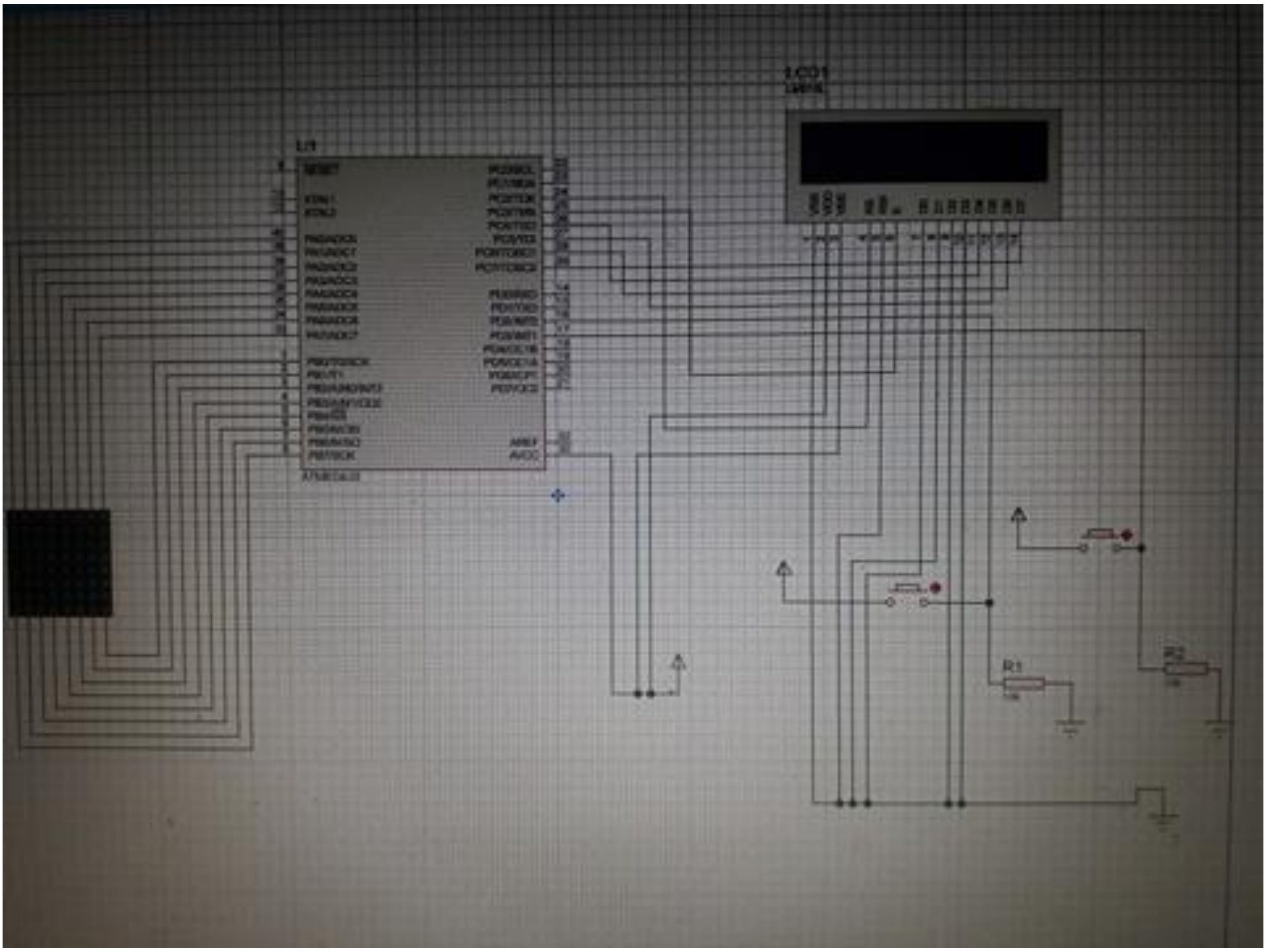
Block Diagram

Showing input and output



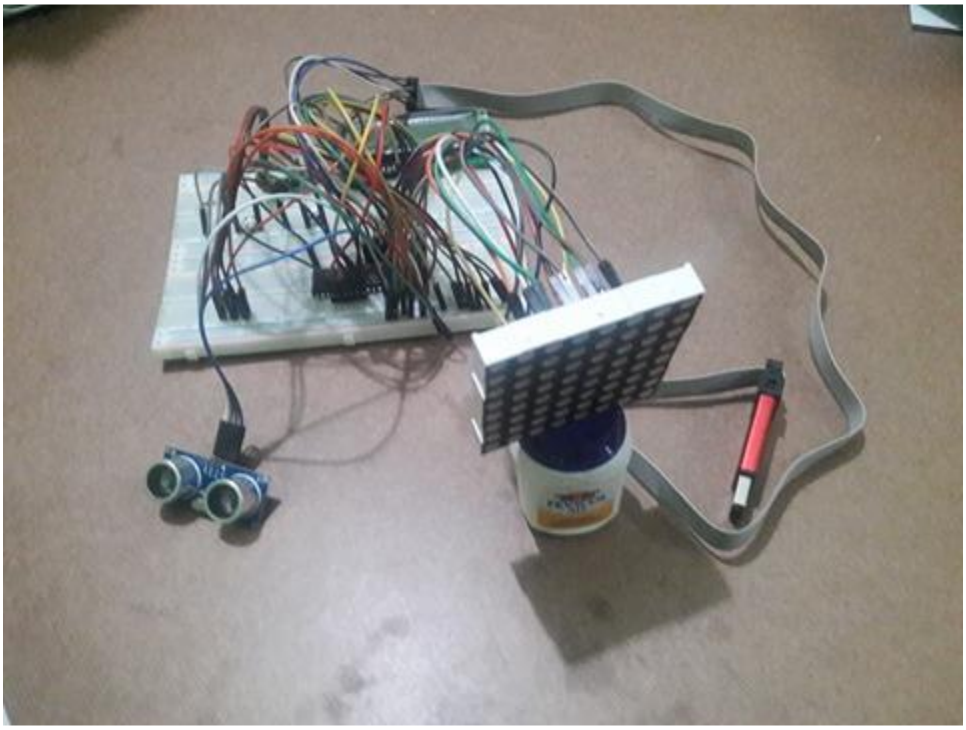
Circuit Diagram

Showing complete connection using buttons:



Actual Circuit

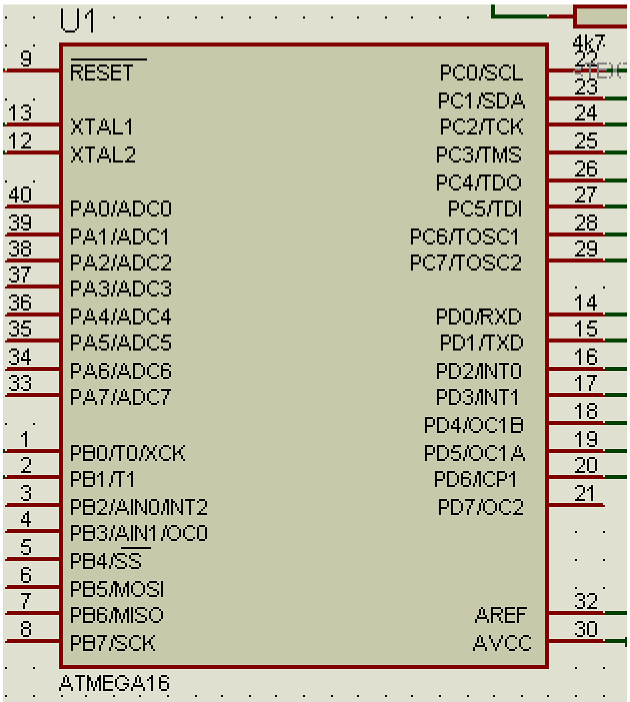
Snapshot of the actual circuit:



Description of modules and used libraries

Describing all the hardware parts and their connections.

Atmega32



ATmega32 is very much similar to ATmega16 microcontroller with certain differences which are discussed below. ATmega32 is an 8-bit high performance microcontroller of Atmel's Mega AVR family. Atmega32 is based on enhanced RISC (Reduced Instruction Set Computing) architecture with 131 powerful instructions. Most of the instructions execute in one machine cycle. Atmega32 can work on a maximum frequency of 16MHz.

Port A (PA7-PA0): Port A serves as the analog inputs to the A/D Converter. Port A also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used. Port pins Port A also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used. Port pins metrical drive characteristics with both high sink and source capability. When pins PA0 to

PA7 are used as inputs and are externally pulled low, they will source current if the internal pull-up.

Port B (PB7-PB0): Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability.

Port C (PC7-PC0): Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The C output buffers have symmetrical drive characteristics with both high sink and source capability.

Port D (PD7.-PD0): Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The D output buffers have symmetrical drive characteristics with both high sink and source capability.

RESET: Reset Input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in Shorter pulses are not guaranteed to generate a reset.

XTAL1: Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

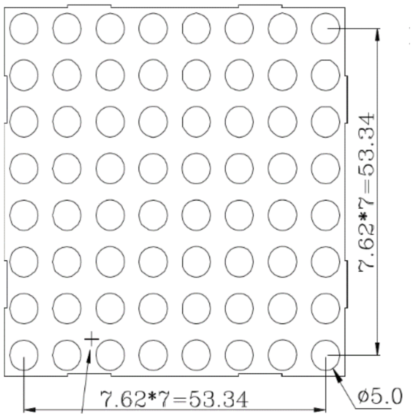
XTAL2: Output from the inverting Oscillator amplifier.

AVCC: AVCC is the supply voltage pin for Port A and the A/D Converter.

AREF: AREF is the analog reference pin for the A/D Converter.

8X8 Dot Matrix

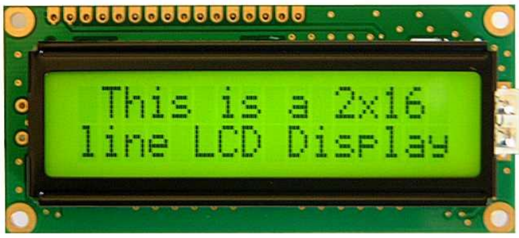
The LED matrices used in the labs are 8x8 and bicolor. Each position (i.e., LED) can glow in green or red.
There are 24 pins in total to select which LED(s) we want to light up. 8 of the pins are used to select the row(s), while the remaining 16 pins are used to select the column(s) (The notion of row and columns are only artificial and depends on the way we consider the orientation of the LED matrix. Rotating the matrix by 90 degrees will alter the orientation. However we will use a fix orientation which is described later in this section). Among these 16 pins, 8 pins are used for lighting the LEDs red while the other 8 pins are used if we want to light the LEDs green. LED matrices can be oriented in two flavors: common (row) anode and common (row) cathode. The difference between these two configurations is how we light up a LED. With common anode orientation positive voltage is attached to rows and ground to columns. With common cathode the connections are reversed. In our experiments we will use the common anode orientation.



The pin diagram of the bicolor LED matrix is given below. It should be held so that the pins are facing u and the leds are at opposite side.

Row4	R4	G4	Row3	R3	G3	Row2	R2	G2	Row1	R1	G1
Row8	R8	G8	Row7	R7	G7	Row6	R6	G6	Row5	R5	G5

16X2 LCD Alphanumeric Display



For LCD we need a header file. The link to the header file is given below:

https://drive.google.com/open?id=1t_WjIpFfDmpzCZMKEHXy3BjRQkXfUaSq

The LCD has 16 connector pins. Connections are:

- LCD 1 (GND) to GND
- LCD 2 (VCC) to 5V
- LCD 3 (contrast) to GND
- LCD 4 (RS) to PD0 (Pin 14 on Atmega32)
- LCD 5 (R/-W) to PD1 (Pin 15 on Atmega32)
- LCD 6 (Clock or Enable) to PD2 (Pin 16 on Atmega32)
- LCD 7, 8, 9, 10 are not connected
- LCD 11 (Data 4) to PC0 (Pin 17 on Atmega32)
- LCD 12 (Data 5) to PC1 (Pin 18 on Atmega32)

● LCD 13 (Data 6) to PC2 (Pin 19 on Atmega32)

● LCD 14 (Data 7) to PC3 (Pin 20 on Atmega32)

● LCD 15 (to VCC) and LCD 16 (to GND) are for background light.

Ultrasonic Sensor HC-SR04:

It has 4 pins.

- VCC is connected to VCC of Atmega32
- Trig is connected to PD0 of Atmega32
- Echo is connected to PD6 of Atmega32
- Gnd is connected to GND of Atmega32

Problems Faced

- The header file of LCD needed two ports. We adapted it to 1 port.
- Atfirst the sonor was not detecting object when we were using interrupt pin. Later we changed it to PD6 and it started working. We also had to modify the code found in the internet for sonor.
- The dot matrix was set up carefully and the pins were chosen and connected precisely
- The code for the snake game was tedious to design and we had to debug quite a while in proteus before getting error free results.
- We also connected a pod to adjust the contrast of LCD.
- Before connecting sonor sensor for direction we used buttons. We configured the buttons in polling method but it gave extremely poor results. Later we switched to ISR or interrupt for better results and the outcome was promising.

Acknowledgements

These are the sites that helped us to set up the project:

- https://electrosome.com/interfacing-lcd-atmega32-microcontroller-atmel-studio/
- https://circuitdigest.com/microcontroller-projects/distance-measurement-using-hc-sr04-avr
- http://www.electronicwings.com/avr-atmega/ultrasonic-module-hc-sr04-interfacing-with-atmega1632
- https://electrosome.com/push-button-switch-atmega32-microcontroller-atmel-studio/
- https://docs.google.com/document/d/1vBv38EOGM-Bvn2SCUHUE8iZCimaciyhjn0eEL9ueWbU/edit

Conclusion

The project helped us get insight about the power of microcontroller and that a small processor such as atmega32 can work wonders if used correctly. The project was fun as well as challenging. Though it was created for entertainment and fun, it had educative values too. We learnt lots. We learnt how atmega32 works and modules such as dot matrix, LCD display and Ultrasonic sensor can be integrated with it. Ultrasonic sensor was truly fun to work with and though frustrating at times we learnt how this sensor works and calculate distance of an object from it. It was truly ingenius.Thanks to our teachers for guiding us when we had problems and giving us the opportunity to create such a wonderful game.