Micro-Controller & Micro-processor Sessional (Name : The Coolest Game Ever)

Prepared by
Mohammad Nadimul Abrar Roll:1305107 Phone No:0159301704 &
Atiq Ishraq Arnob Roll:1305103 Phone No:01716925121

YouTube Link:

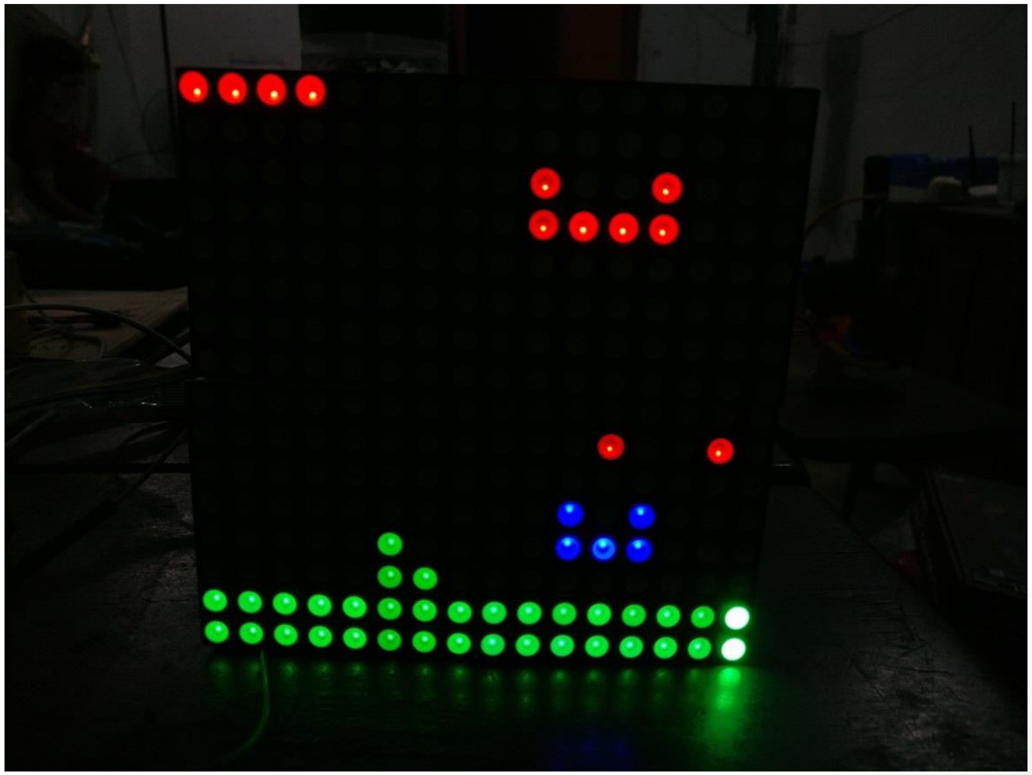
- https://youtu.be/Nf8cRIXtI6o

Implemented Features:

- Using hand gesture to run backwards
- Using scream to jump forward.
- Showing results on 4-colored (Red-Blue-Green&Mixed)LED matrix
- Showing scores in the end.

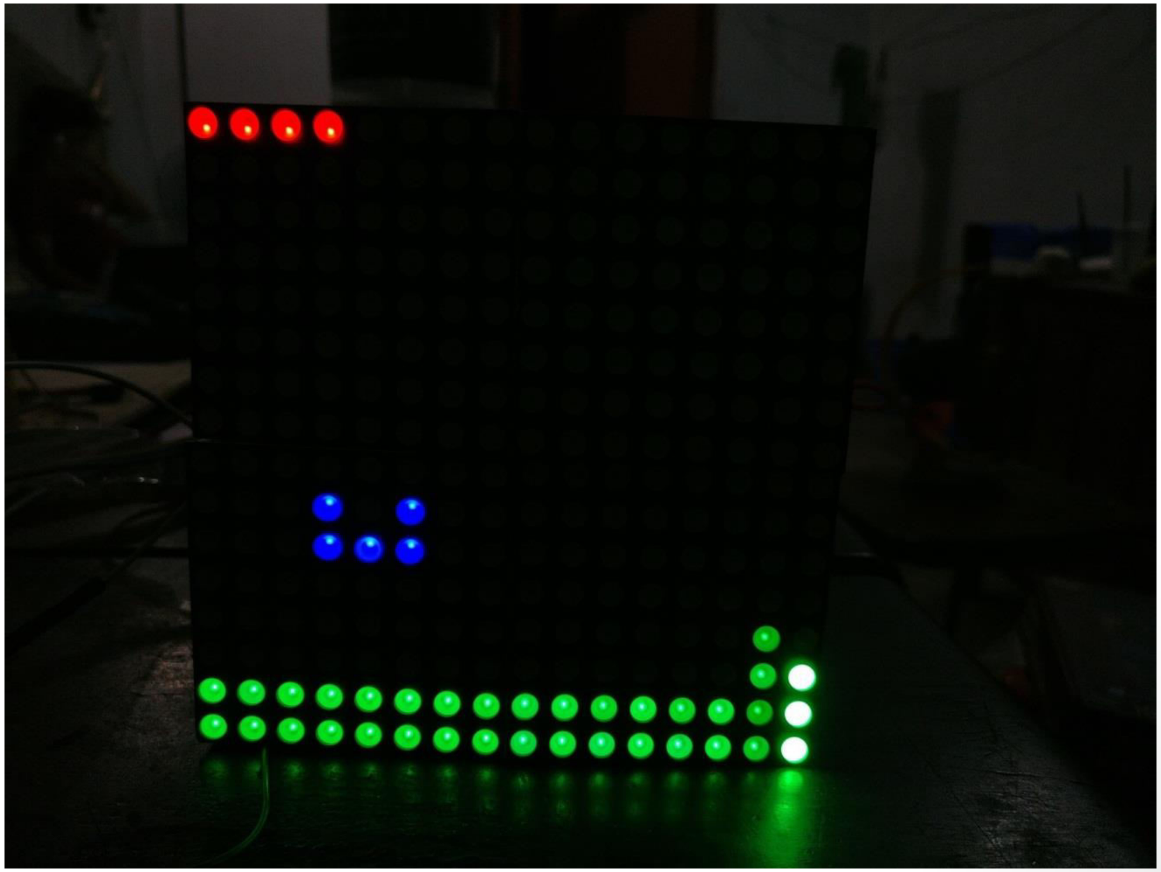
Project Description:

- An 4-colored LED-matrix game of jumping and running.
- In the game, we have a blue hero trying to jump over obstacles while running away from a red evil boss shooting from the sky.
- Figure-1:



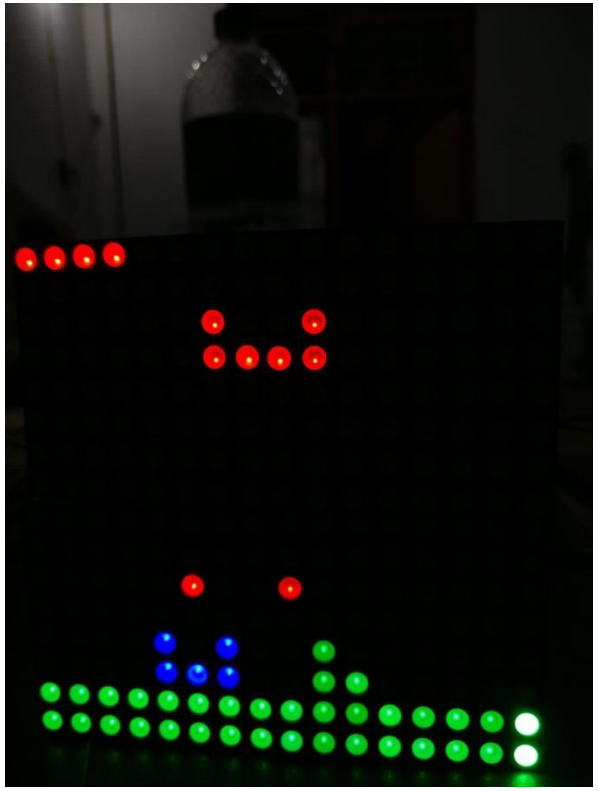
Project Description:

- The green in the bottom represents a field and the green triangle-shaped object represents the obstacle hero jumps over.
- Figure-2:



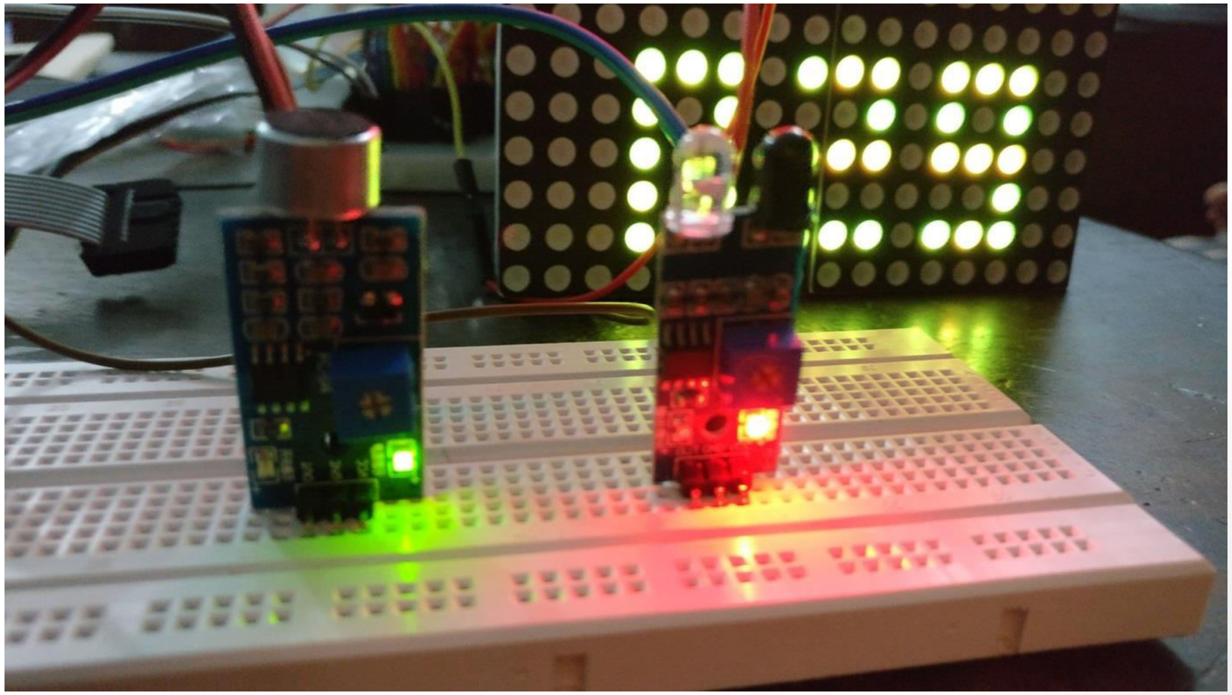
Project Description:

- The 4 red dots in the corner represents 4 lives of the player. Which will decrease gradually if, the player either runs into an obstacle or, get shot by the boss.
- Figure-3:



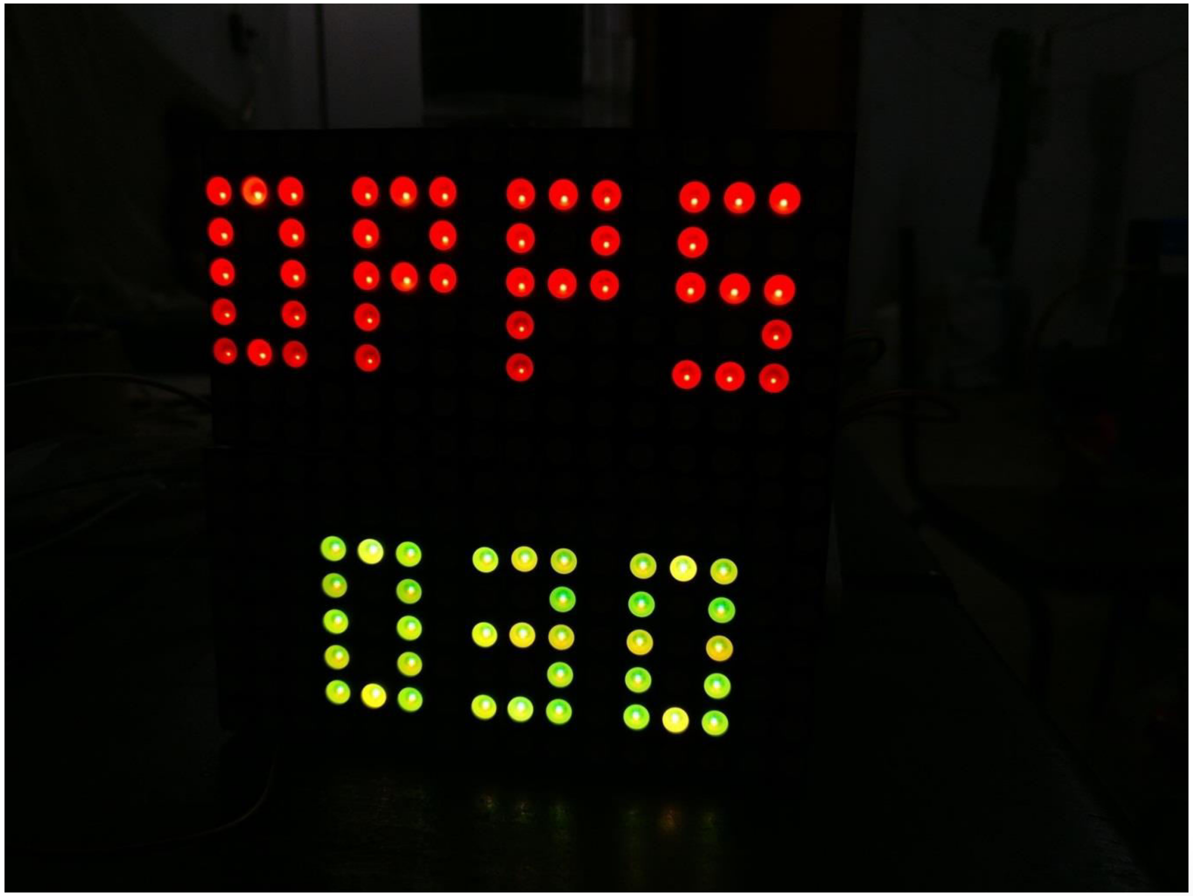
Project Description:

- In order to control the jump movement, we have used a FC-04 sound level detecting sensor. In figure the one in the left.
- Also, running backwards we used the IR Infrared Obstacle Detection Sensor FC-51. It’s the one in the left in the figure.
- Figure-4:



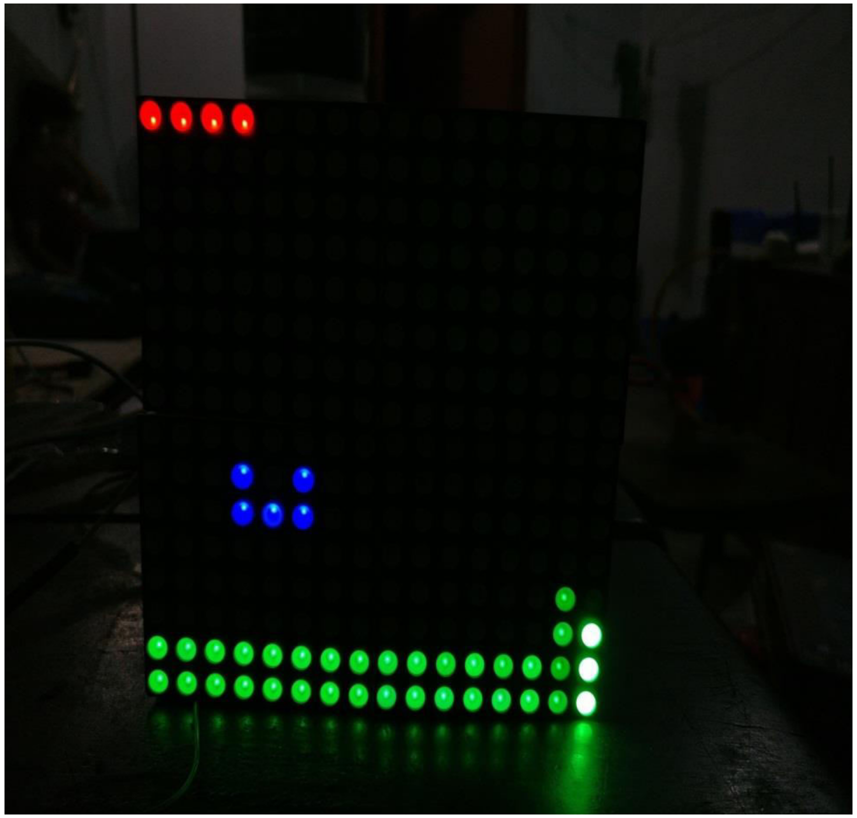
Project Description:

- After the game is over meaning, we have showed the scorecard like this:



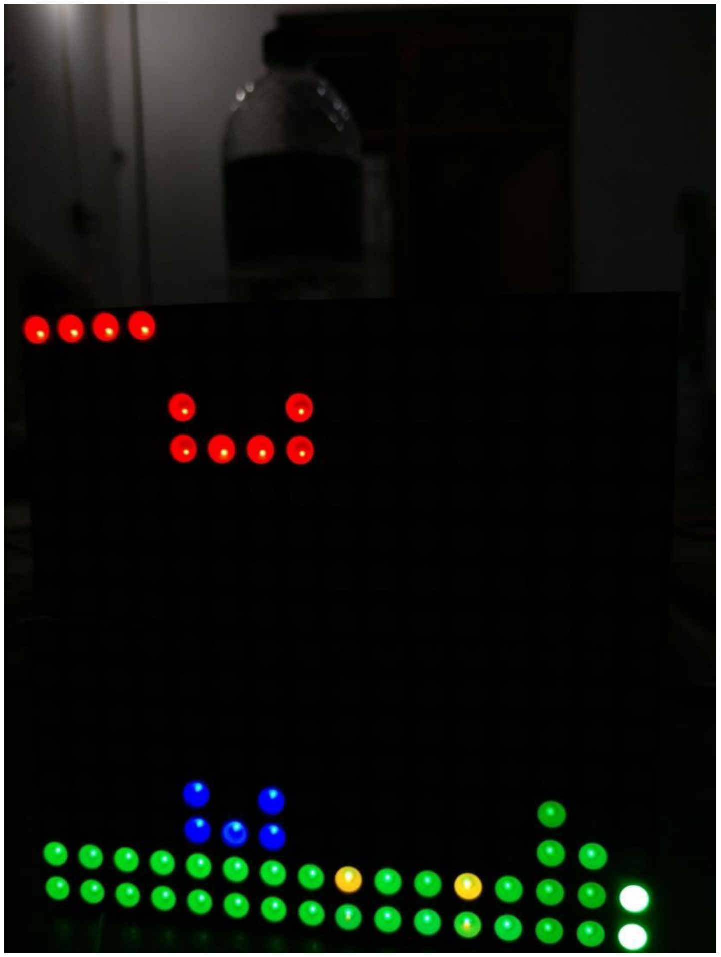
Project Description:

Figure-5: Hero’s Jumping Demonstration.



Project Description:

- Figure-6: Hero's Sliding Demonstration.



Working Prinicpal:

- 1. **Working with single color.**

The first challenge for us was to display a single color figures on our custom made 16x16 LED matrix. We decided to use 74HC154 4x16 decoders to do so. We needed to dominate 64 pins using 4 ports available from ATMega32. At first we shorted all the rows and columns of out four 8x8 LED Matrix. Then it was left for us to dominate only 32 pins. We have used pins PORTB(0-7), PORTA(4-7) and PORTD(4-7) to control 16 pins of the rows. And PORTA(0-4) pins were chosen to be input of our 74HC154 decoder and the 16 output pins were used to dominate remaining 16 pins which actually controls the columns.

- 2. **Working with RGB color.**

The idea actually clicked suddenly. It was like superposition. Firstly we were working with blue color only. But then we made the same circuit with two additional decoder for Green and Red. Our decoders had two enable pin. We realized, we can disable a decoder by setting enable bits to 1. So then we used multiplexing on decoders with PC0, PC1 and PC7 of ATMega32.

Working Principal:

- **3. Working with sensors**

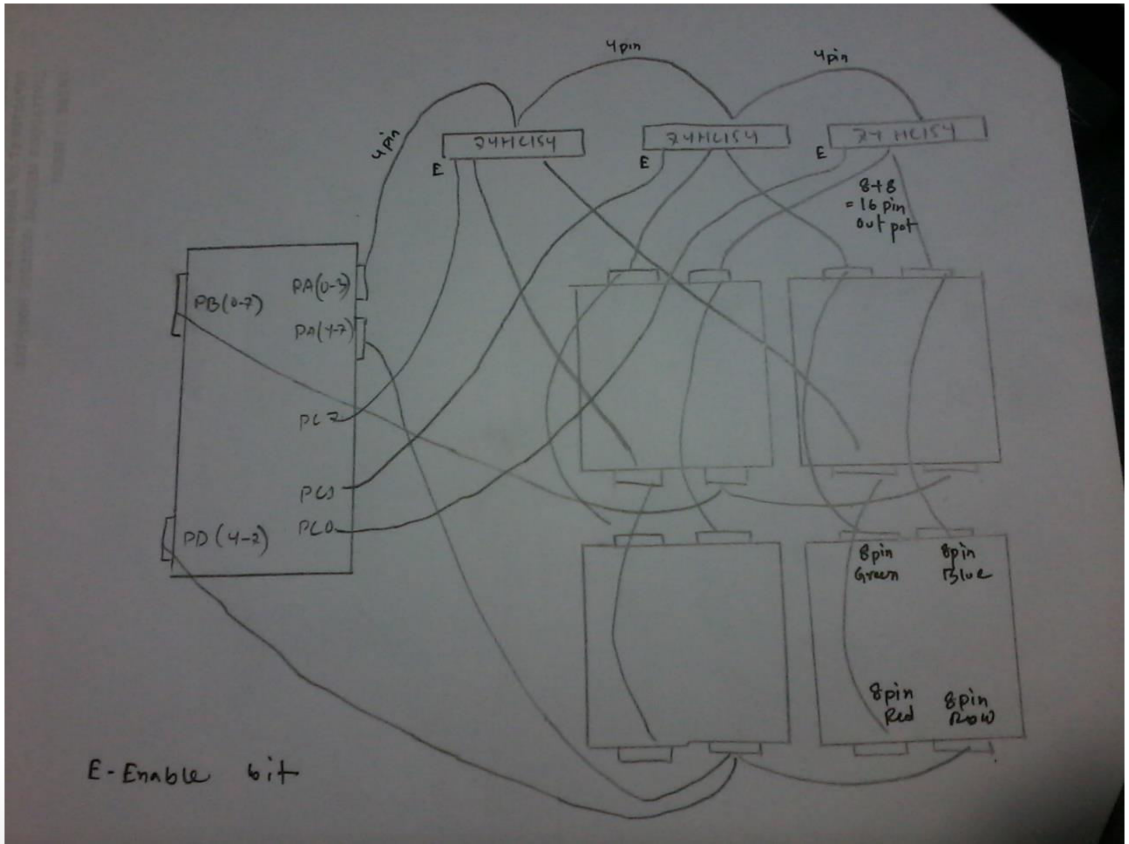
As our sensors returned boolean output, Working with them was pretty easy. But we still had to loose an hour or so to make them work as it was hard to find any datasheet of those sensors to realize that their output was in active low.

- **4. The coding**

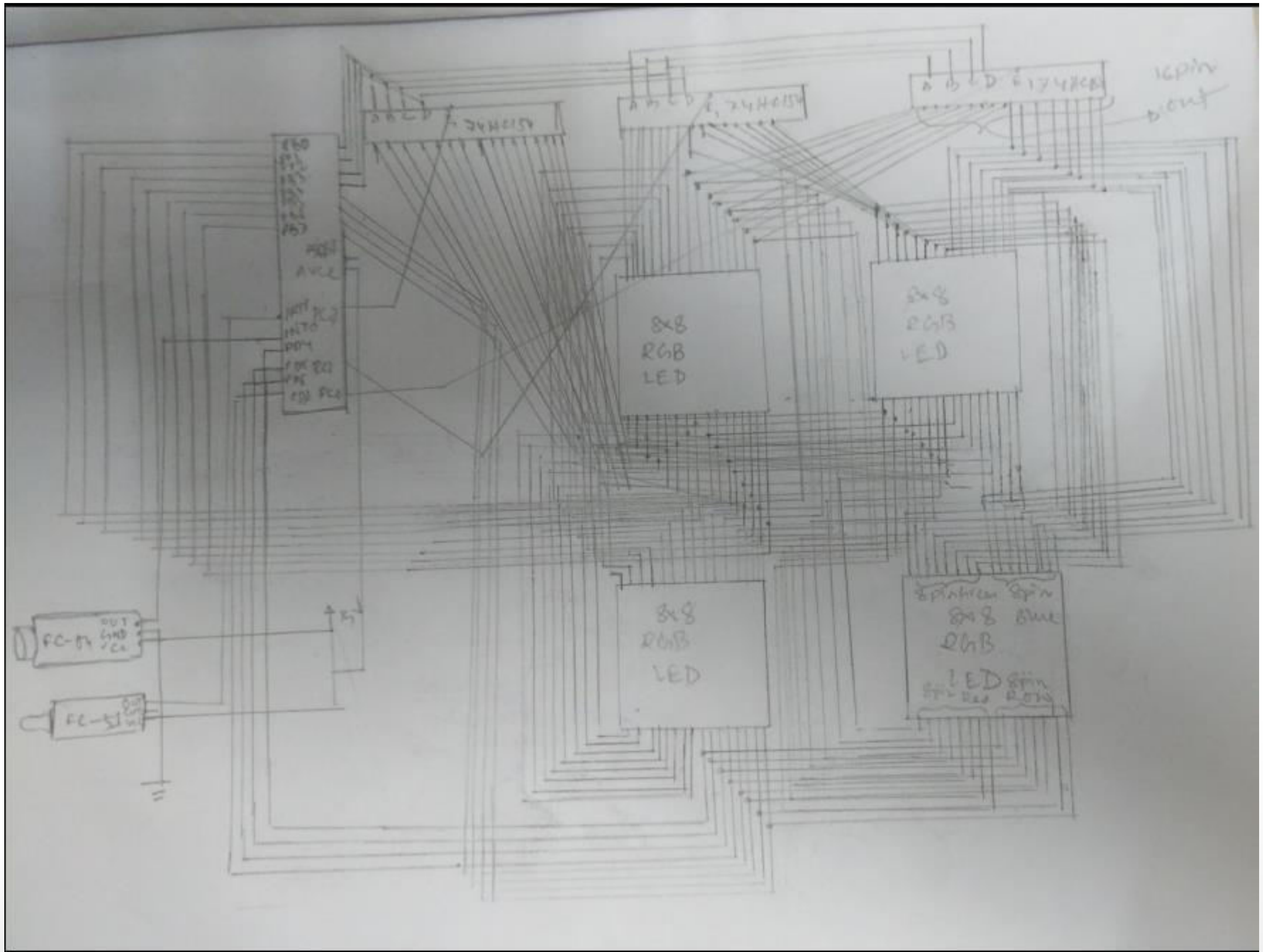
As the game was pretty simple, it was pretty easy to code.

Circuit Diagram:

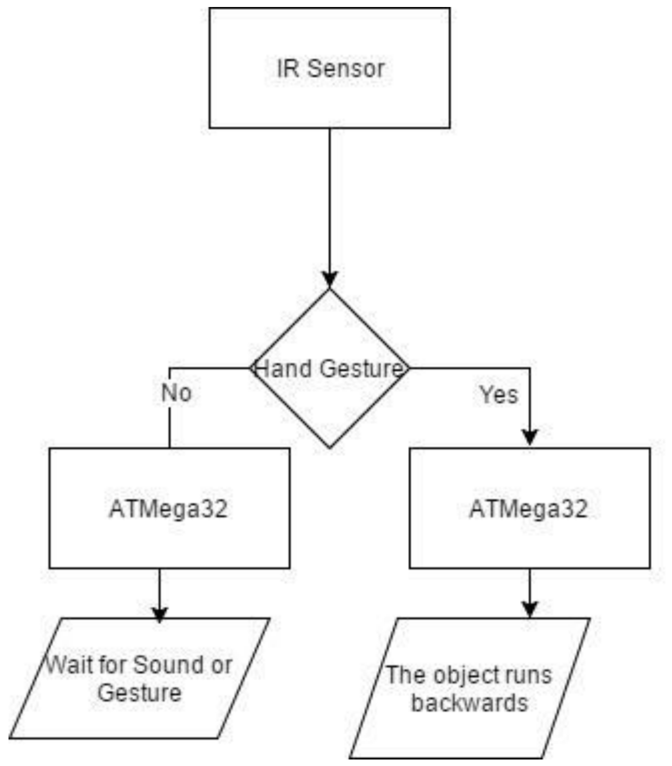
- As there was no 32 pin RGB LED matrix in drawing software we drew the picture by hand.
- **Figure-7: Basic Configuration:**

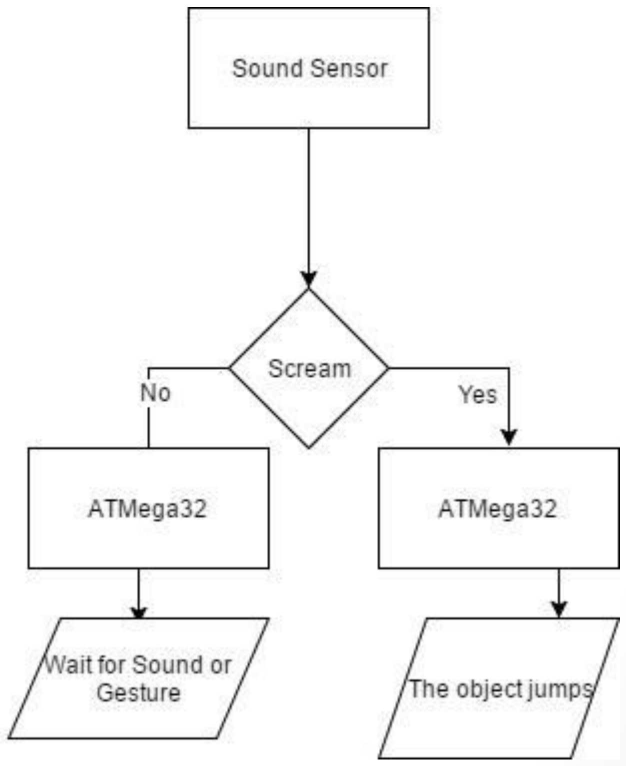


Circuit Configuration (Figure-8):



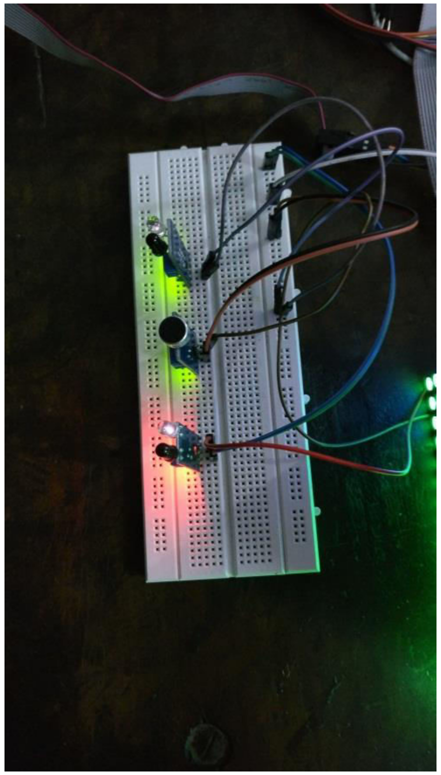
Flow Chart:


IR Sensor
Hand Gesture
No
Yes
ATMega32
ATMega32
Wait for Sound or Gesture
The object runs backwards


Sound Sensor
Scream
No
Yes
ATMega32
ATMega32
Wait for Sound or Gesture
The object jumps

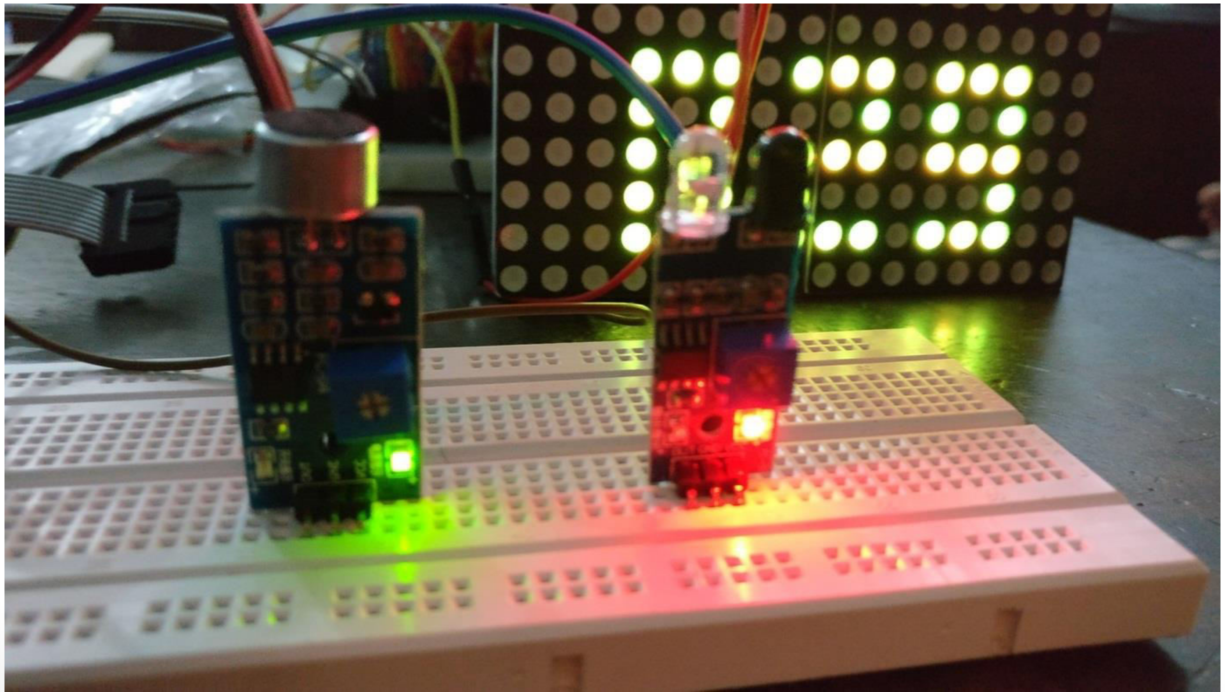
Interfacing with Different Sensors:

- Our IR FC-51 sensor detects any obstacles up to certain point and it changes to logic level low.
- We used ATMega32 interrupts to detect this change of level.
- Figure-9:



Interfacing with Different Sensors:

- Our Sound Sensor FC-04 sensor detects any sound of certain intensity and higher. It also changes to logic level low.
- We again used ATMega32 interrupts to detect this change of level for our sound sensor.
- Figure-10:



Libraries Used:

- We have used basic interrupt library in order to interface with the sound and IR sensor.
- We used both INT0 and INT1 for different sensors.
- Also, we used sei() library function for setting interrupt and ISR() for writing the functions.
- For showing output we used io library.

Features we could not implement:

- We wanted to use two IR sensor to make the object run forward and backwards, but due to time limitations we couldn’t do it.

Obstacles We Faced:

- **Obstacle #1:** We were unable to use interrupt properly at first. While configuring GICR and MCUCR, we got confused. We kept configuring GICR = (1<<INT0) and not getting the result we wanted.
- **Obstacle #2:** Another problem was that we faced was while making circuit for different sensors, we kept getting wrong result as we kept thinking the output was going to be active high.

Overcoming the obstacles:

- **Obstacle #1:** We changed the configuration of GICR to GICR |= (1<<INT0) and problems with interrupt were immediately solved.
- **Obstacle #2:** After we learned that the output was going to be active low, we changed the code accordingly and our problems of detecting output was solved.