2016

Automated basketball socring syste,

Znanatej Panga

Automated Basketball Scoring System

Znanatej Panga

Problem Report submitted to the

College of Engineering and Mineral Resources

At

West Virginia University

In partial fulfillment of the requirements for the degree of

Master of Science

In

Computer Science

Roy S. Nutter, Jr., Ph.D., Chair

Yanfang (Fanny) Ye, Ph.D.

Hany H. Ammar, Ph.D.

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia

2016

Keywords: [Conductive Stretch Sensor, Arduino, LED, Automatic Basketball Score]

ABSTRACT

Automated Basketball Scoring System

Znanatej Panga

An automated basketball scoring system is configured to detect when a shot is made. In addition, this system can be modified for recreation or training purposes. A Conductive Rubber Cord Stretch Sensor is used to detect if a shot has been successfully made. This Conductive Rubber cord is laced around the net and it stretches when basketball passes through the net. Careful comparison of differences in the analog voltage values helps in identifying when a valid shot is made. In this scoring system, the basketball hoop is lit up using LEDs which change colors in Red-Green-Blue order at regular intervals. The point value of each basket is decided based on the color of the LEDs when the shot has been made unlike in conventional basketball game where the point value depends on from where the shot had been made. This system is also provided with options to manually increase and decrease the score along with start, pause, and restart the game options.

ACKNOWLEDGEMENTS

This work would not have been possible without the guidance and direction of Dr. Roy S. Nutter Jr. I would like to extend my sincere gratitude to Dr. Yanfang (Fanny) Ye and Dr. Hany H. Ammar for their time and consideration. I would also like to thank Mr. Ruston Seaman of New Vision Renewable Energy for the opportunity to work on this project and providing equipment required for this project.

It gives me immense pleasure to thank my family without whose support, this would not be possible. I would like to thank the LCSEE department for their support and opportunity.

Table of Contents

List of Figures

1 Introduction

There are many places in the world without electricity which still live in dark when the sun goes down. A recreational area in such places can be lighted using solar charged battery powered LEDs. Instead of lighting the whole place, lighting the sports equipment with the LEDs should be inexpensive and affordable.

This project is to create a low-cost automated basketball scoring system which can be used for both recreational and training purposes. Section 3 talks about the background of the problem. The problem is further categorized into two parts. First is to automatically detecting when a basketball passes through the net. There are a few existing technologies that automatically detect a shot. Section 4 describes some of these existing. Second is to control the LED's color which also decides the point value of the basket. Section 5 describes the design of the automated basketball scoring system while Section 6 concerns the evaluation of the implemented automated basketball scoring system. Section 7 and 8 are conclusion and future work.

2 Problem Statement

The project is to design a low-cost, battery powered, automated basketball scoring system.

3 Background

Third world countries lack exposure to new-age technology. Absence of electricity is the main reason for technology not being a part of their lifestyle. There are more than a billion people still living in the dark when sun goes down, and many still use fire or kerosene or candles at night.

It is estimated that almost 40% of the childhood deaths in Kenya are caused due to respiratory diseases which are mostly caused because of the smoke generated from fire used to cook food and light their homes. EPA (US Environmental Protection Agency) researchers estimate that the lifetime cancer risk from wood smoke is 12 times greater than a similar amount of cigarette smoke. Apart from these, there are many other disadvantages of not having clean

energy options to cook food like people need to walk long distances to gather wood, sometimes into dangerous places where there are threats to their lives. The generation of electricity mostly depends on non-renewable resources, and includes big investments. Alternatively, renewable energy can be used by tapping smaller amounts of energy when needed. [1]

Ruston Seaman, President and CEO of New Vision Renewable Energy, based in Philippi, West Virginia dedicated most of his life developing under resourced communities in U.S. One of many projects that he has initiated was using renewable energy in his community. The Seaman's are the first family in their community to operate their house totally on alternative energy sources. Using solar powered LEDs, New Vision Renewable Energy has helped to light up many houses in about 37 countries all over the world.

Figure 1 School Girls in Kenya with New Vision's solar power pack and light. Picture Provided by [2]

The Director of Vision Trust International, a non-profit organization visited Philippi to observe the community's response to Mr. Seaman's projects. In that visit, she asked New Vision's help

in lighting up a school in the Dominican Republic. In November 2014, Mr. Seaman along with his family visited the school located at a small village, Coutri, in Dominican Republic. At that time, the school has about 400 children, a master and covered recreational space with no electricity. New Vision Renewable Energy helped light the recreational space using several portable solar power packs which contain bright LED lights powered by a compact 12V/9Ah lithium polymer battery charged using a small 10-watt solar panel. [1]

During Mr. Seaman's stay at Coutri, he noticed that the entire village has no access to a source of electricity. The only place where there is light at night is a bar. Every night, the bar owner fires up his gas generator, cranks up the music, turns on the lights, and sells alcohol. Mr. Seaman observed that many children were drawn towards the bar as that is the only place with lights at night.

After returning back to U.S from Dominican Republic, Mr. Seaman thought that the best way to prevent children from going to the bar is to have an accessible recreational space with lights at night. His passion for Basketball, which is an activity more popular and less expensive than many others, made him think that it could be a good means of recreation in such areas.

His experience of installing lights at the Dominican Republic school taught him that most of the light is lost when installed in higher locations and made the LEDs look like low intensity street lights. Upon further research, he designed a basketball arena such that all the light is focused towards the intended places.

Figure 2 Basketball Arena in daylight at Philippi, WV Which is built with solar powered LEDs and reflective sheets. Picture Provided by [1]

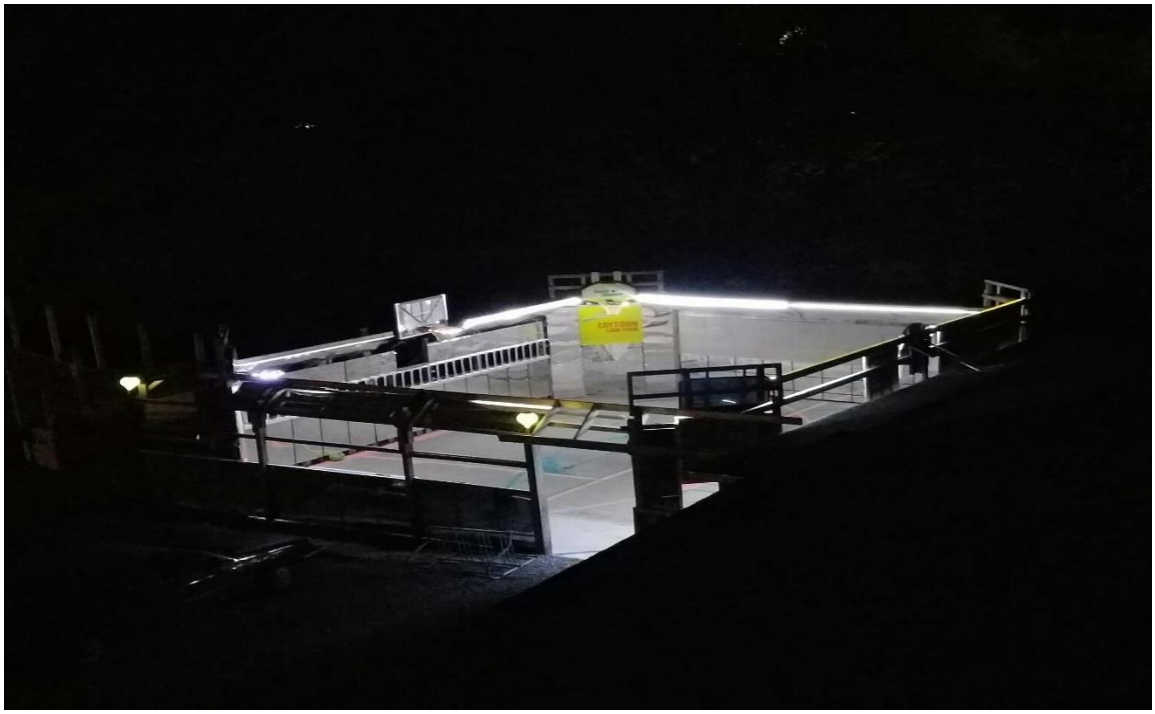

Figure 3 Basketball Arena in night at Philippi, WV. Picture Provided by [2]

In this process, he came up with the idea of using LEDs to light the basketball board itself. After several attempts, he concluded that it is optimal to strategically place LEDs on the hoop itself and use a reflective sheet on the backboard. He tested this concept in the basketball arena he built in Philippi, WV and thinks that this mode is ideal and could be implemented in those places where there is no access to electricity. He also thinks that this mode could be used for basketball training.

With a vision to implement more such creative ideas, Mr. Seaman established Glogames LLC, a start-up venture to create sport products that use LED lighting technology to provide new types of recreational and instructional experiences for people of all age groups.

Figure 4 Basketball hoops with LEDs. Picture Provided by [2]

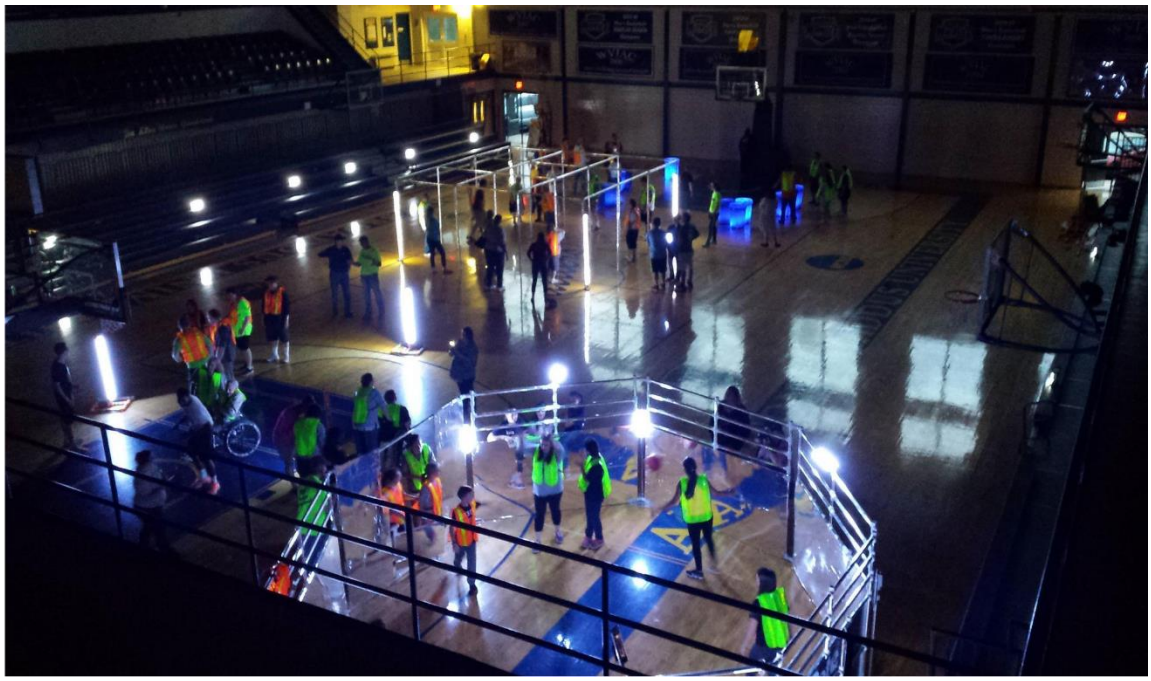

Figure 5 Arena with different recreational games developed using LEDs. Picture Provided by [2]

A discussion with Mr. Seaman on exploring the different possibilities of the use of LEDs on the basketball hoop led to the idea of a new type of recreational and fun basketball game. In this version of basketball, the system could automatically detect when a shot is made, change the LED's color in regular intervals, and the point value of each basket is decided by the color of the LEDs at that moment. To the best of my knowledge, as a whole, there is no pre-existing system with this functionality. There are a few technologies available which automatically detect when a shot is made. The next section describes some of these technologies.

4 Existing Automated Scoring Technologies

There are a few basketball shot detecting devices developed using different techniques. These are complicated and expensive as they use laser beam techniques that require a multitude of laser beams in order to acquire the location of a basketball during play. These might result in false statistics sometimes due to interference of objects other than basketball. Another technique is to use vibration sensors or accelerometers.

In [3], piezo-electric or piezo-resistive accelerometer sensors are used to detect the vibration of the hoop, motion of the basketball net, and pressure from a passing basketball. Three sensors are strategically placed on the net to detect the above properties. One of the sensor attached at upper rear side of the net is connected to the hoop using a rigid material like a hook to transfer high frequency vibrations from the hoop to the sensor. It can also sense the net motion in to detect if the ball passes directly through the net without touching the hoop. The other two sensors namely lower front and rear sensors are placed opposite to each other below the first sensor. These two lower sensors are used to sense combination of net motion and pressure from a passing ball.

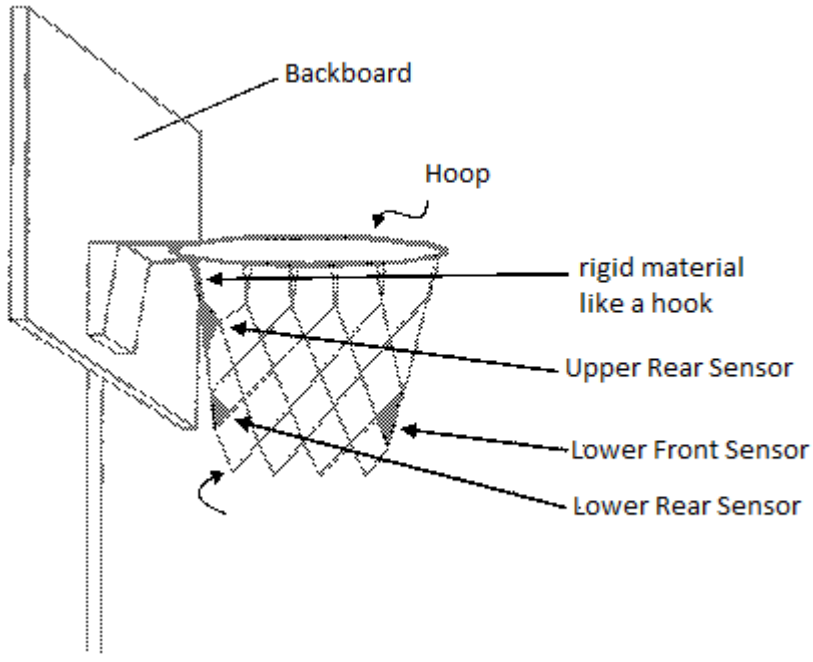

Figure 6 Shot detection system using accelerometers. Picture taken from [3]

The information received from all the three sensors is then sent to a microprocessor embedded on one or more sensor PCBs. After combining and analyzing the information received, the microprocessor sends the data to a console which process the data and displays the results. These results include but not limited to how many shots are attempted and how many shots are successful.

There is a time delay for the upper rear sensor to avoid false counting due to multiple vibrations on the hoop resulted from same shot. As it is not possible to predict how much of bounce a ball can have or how many times a ball can bounce on the hoop, it is not ideal to come up with a perfect time delay. So, there is still possibility of generating false signals due to ball bouncing of the rim. In a situation where, a player hand went from bottom of the net and the ball bounced off the hoop, this system identifies the scenario as a shot made successfully, because, the sensors cannot identify the difference between a ball and other object.

In [4], ultrasonic sensors are used in the system to detect a successful shot. A ultrasonic sensor which typically consists of a transmitter and receiver is placed at downward angle to a work place below the hoop. Placing the sensor in downward angle avoids detection of any object above the hoop. When a ball passes through the hoop, it interferes with the ultrasonic waves generated by the sensor. The distance to the ball is calculated using the speed of the ultrasonic wave and time taken by the receiver to receive the bounced ultrasonic wave. If this distance falls into a valid distance range, then the system detects it as a successful shot.

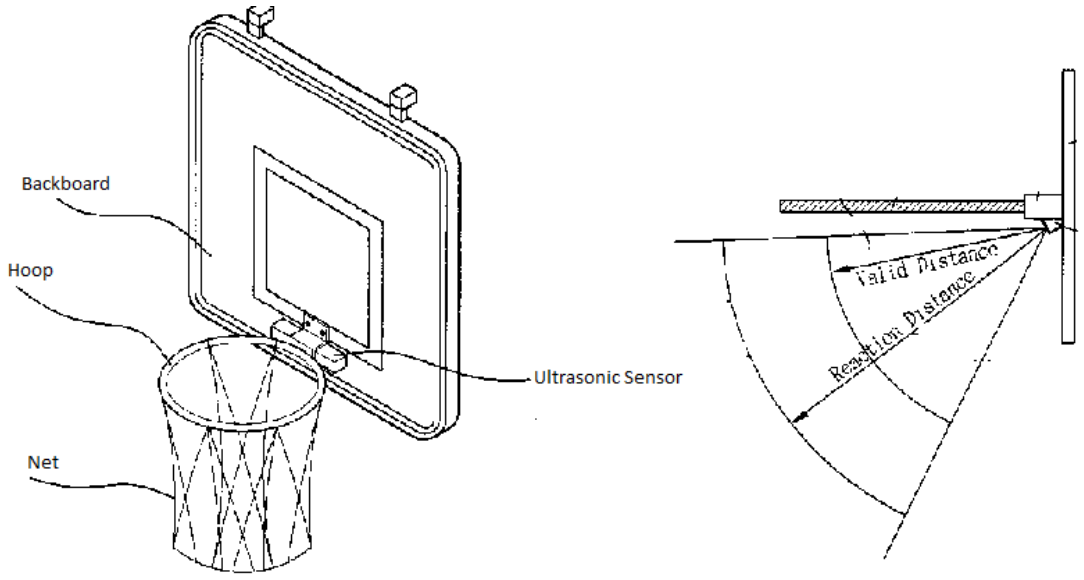

Figure 7 Basketball shot detection using Ultrasonic sensor. Picture taken from [4]

This system might provide false results due to the movement of the net, which can occur due to several other reasons apart from the ball going through it. The ultrasonic sensors cannot distinguish between a ball and any other object. So, it will give false results in scenarios where the player's hand comes in the path of the ultrasonic waves at valid distance.

In [5], one or more sensors are placed within the basketball and the system determines whether a shot is made based on the signals received from the sensors. Different sensors like accelerometers and gyrometers are used to read one or more attributes of a shot such as linear acceleration, spin axis, spin rate, launch velocity, launch direction, launch angle, launch coordinates, backboard vibration, and rim vibration. The raw sensed data received from the sensors within the basketball is compared to predetermined signature characteristics of a made shot or a missed shot to determine if a shot is made. The paper does not talk about what sensors can be used or how to acquire the predetermined data of a made shot or missed shot. Overall, the idea might work to detect a shot, but it is expensive as all the sensors needs to be placed inside the basketball which can happen only at a manufacturing stage of the ball.

In [6], photoelectric sensors are used to detect if a shot is made. A photoelectric sensor is placed on the front side of the bracket that attaches the hoop to backboard. The system determines if a shot is made depending on the time for which the ball breaks the light beam

emitted by the sensor. For more accurate results, a set of photo electronic transmitter and receiver are placed on opposite sides of the hoop.

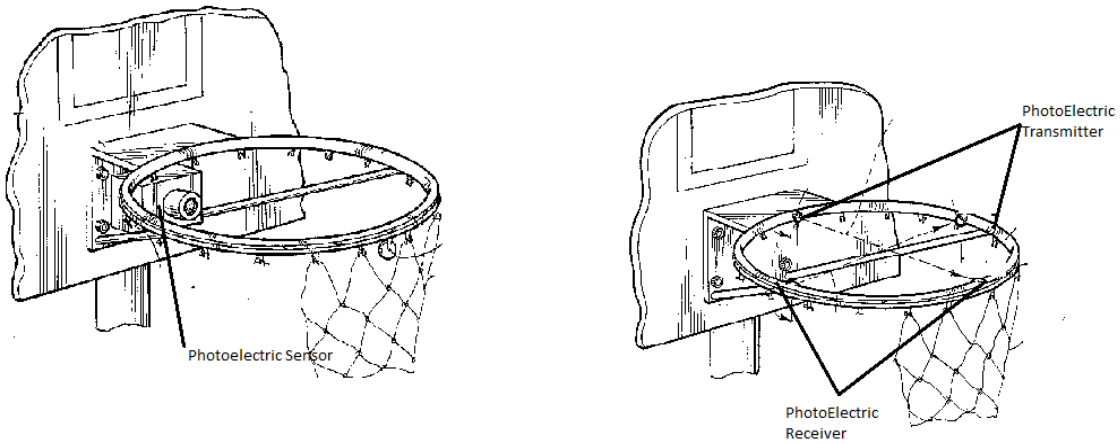

Figure 8 Basketball shot detection system using one or more Photoelectric Sensors. picture taken and modified from [6]

This system also gives false results as it cannot distinguish between a ball and other object. When a player's hand interferes with the light beam of the photoelectric sensor, it might consider it as a shot made. In some scenarios where the ball bounces back from the rim after breaking the light beam, the system might think it as a shot made. If a player dunks to make a shot, he might accidentally break the light beam after the ball passes through the net and this might make the system resulting in multiple shots. These sensors should be encased to protect from the hard hitting by the ball. So, putting them on the hoop might restrict the movement of the ball when a shot is made.

In [7] [8] [9], a conductive material is used to detect when a shot is made. This conductive material is laced to the net and stretches when a ball passes through the net. A shot made is determined by careful comparison of the voltage before and after the ball pass through the net. The net is attached to the hoop using metal hooks on the hoop. The conductive material is laced over the net through the metal hooks.

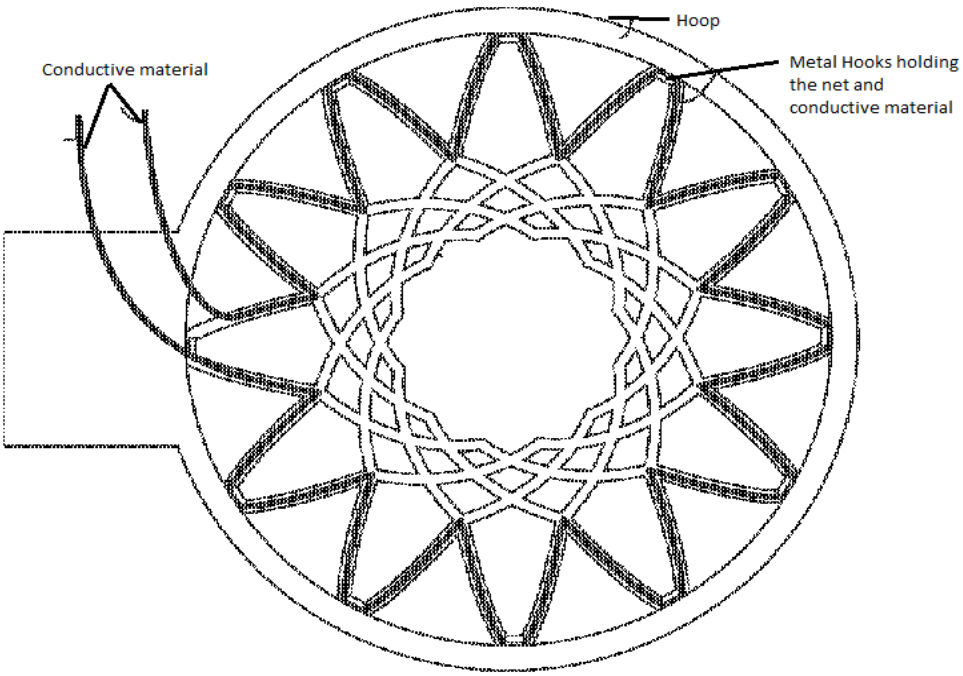

Figure 9 Basketball shot detection using a conductive material. picture taken and modified from [9]

This approach of using a conductive material for detecting a shot made is inexpensive, accurate, and easy to install, maintain and replace than using different sensors like ultrasonic, photoelectric and accelerometers. One of the downside of this design of placing the conductive material to the hoop hooks is that the system might give false results in scenarios such as a player touching the net or holding the hoop while dunking, which might cause the conductive material to stretch.

This project uses the above idea of using a conductive material to detect a shot with some modifications in the placement of the material. There are many elastic conductive materials like carbon nanotubes, silver nanoscale wires, and conductive stretch fabrics made with different polymers. In this project, a rubber cord made of carbon-black impregnated rubber which is inexpensive and readily available, is used as a conductive material. Instead of placing the conductive material through the hooks of the hoop, a single rubber cord of about a meter is laced through the net at approximately 5 inches from the bottom of the net. This design avoids the scenario of interfering with the rubber cord when a player dunks. The next section describes the automated basketball scoring system design in detail.

5 System Design

The automated basketball scoring system is designed based on the below five sub-systems:

- Microcontroller Board: which is used to interact between hardware and software.
- Sensor: which is used to detect if a shot is made successfully
- RIM RGB LED lights: placed on the basketball rim which changes the color and decides the point value of each basket
- LED Score Display: which is used to display the score, and
- Manual Controllers: push buttons used to start and pause the game, increase and decrease the score

These five sub-systems are discussed in detail below.

5.1 Microcontroller Board

Arduino is a microcontroller board with an open-source electronics platform based on easy to use hardware and software. Arduino is mostly used to read data from the sensors and control a variety of things including lights, motors and other outputs. Arduino is inexpensive platform that can use multiple operating systems like Windows, Mac OSX and versions of Linux. Arduino Software (IDE) is a very simple and clear programming environment. It is open sourced, built based on AVR C programming language, and can be expanded through C++ libraries. Arduino connects to a computer using USB. It can be run in standalone mode or as interface connected to a computer. The hardware plans of the Arduino are published under a Creative Commons license, so anyone can make their own version of the module. There are different types of Arduinos and the most common version that is widely used is the Arduino Uno. [10]

The Arduino Uno R3 board is a microcontroller development board built based on the ATmega328P. Arduino Uno R3 has a 16MHz quartz crystal clock, 32KB of flash memory, 14 digital input/output pins, 6 analog inputs, an in-circuit serial programming (ICSP) header, a USB connection, a power jack, and a reset button. Arduino Uno R3 has a very convenient power management and built-in voltage regulation. It can be powered directly using computer USB port and can regulate the input voltage of up to 12V to both 5V and 3.3V. The

ATmega328P chip used on Arduino Uno R3 can be easily replaced in case of damage as it is very low cost and easy to find. [10]

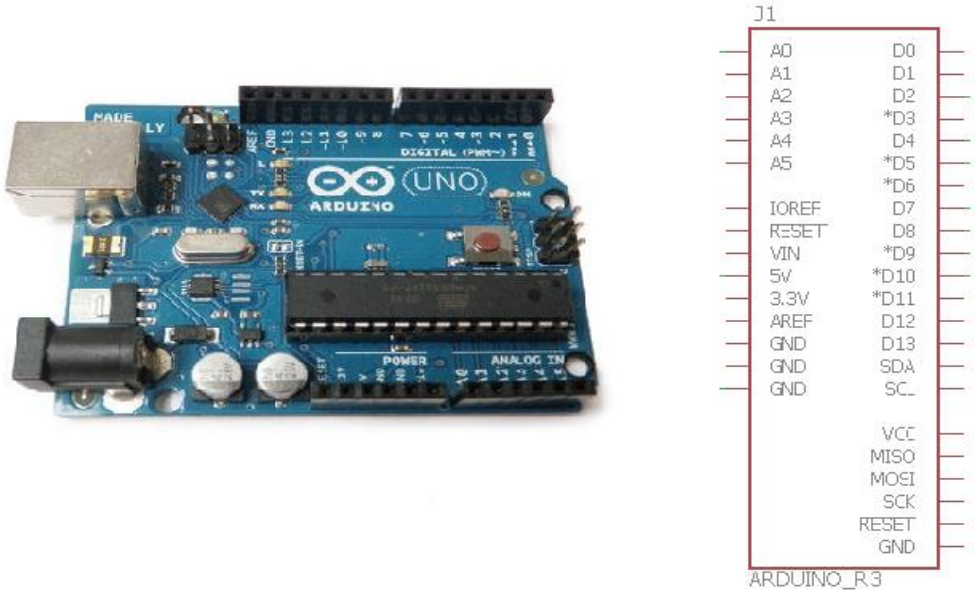

Figure 10 Arduino Uno R3. Picture from [10]

5.2 Sensor

Adafruit Conductive Rubber Cord Stretch Sensor is made of carbon-black impregnated rubber and is used to detect when a shot is made. It is laced around the basketball net approximately 5 inches from the bottom of the net.

A stretch sensor is a component which changes its resistance when stretched and is typically used to measure stretch, displacement and force [11] [12]. They are used in different applications such as robotics, VR gloves and suits, feedback sensor for air muscles, and biometric displacement reading. [11]

The resistance increases when the rubber cord is stretched and it gradually decreases when the rubber shrinks back to its original state once the force is released. It is not ideal to establish a baseline for the resistance values as every conductive rubber cord has a different resistance depending on several factors such as the length of the cord and the material with which it is

made. Also, when the rubber cord is stretched and released, the resistance might increase slightly upon release before reducing to its resting value. [11]

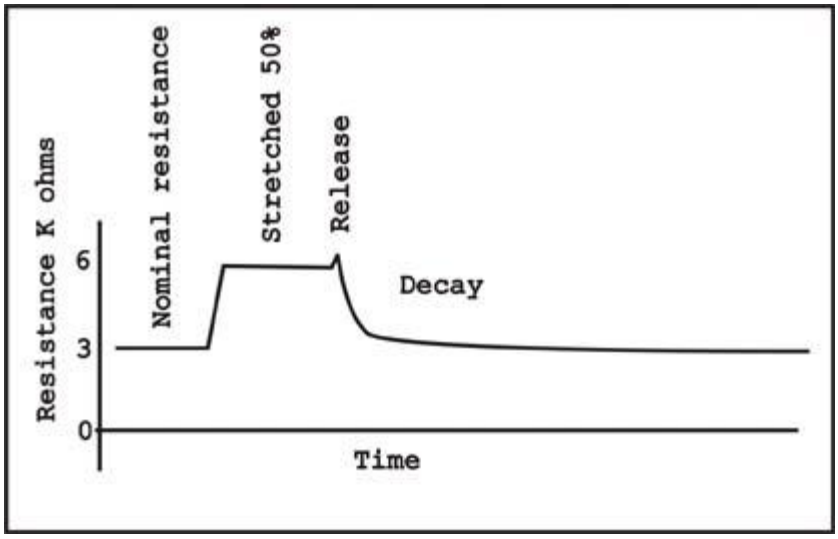

Figure 11 Graph of Stretch Sensor Resistance over time. Picture from [11]

A voltage Divider is created by combining the stretch sensor with a static resistor, which is used to produce a variable voltage that is read by a microcontroller's voltage reader known as analog-to-digital converter.

With resistors R1 and R2 in series, as per voltage divider rule [13], the relation between input voltage Vin and output voltage Vout is:

$$Vout = \left(\frac{R2}{R1 + R2}\right) Vin \qquad (1)$$

The resistance of the Adafruit conductive rubber cord [12] is about 350-400 ohms per inch and 140-160 ohms per centimeter. Let the resistance of the entire conductive rubber cord used in the project is R and, a 10K Ω resistor is connected in series with it to create a resistive voltage divider. The output voltage Vout in the middle changes when the resistance of the rubber cord changes. As per the above equation, the output voltage Vout is,

$$Vout = \left(\frac{R}{R + 10K}\right) Vin \qquad (2)$$

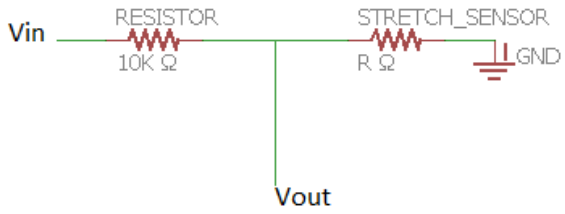

Figure 12 Resistive Voltage divider

Arduino Uno R3 microcontroller reads the output voltage Vout using Analog-to-digital converter or ADC, and converts it into a number between 0 and 1023. The microcontroller's ADC gives a ratiometric value, which means, it returns a value of 1023 when the voltage to analog pin is 5V, a value of 0 when the voltage to analog pin is 0V, and any value between 0-1023 when the voltage to analog pin is a value between 0-5V. [14] [15]

$$\frac{Resolution\ of\ ADC}{System\ Voltage} = \frac{ADC\ Reading}{Analog\ Voltage\ Measured} \tag{3}$$

The above equation can also be written as,

$$ADC\ Reading = \frac{Vout * 1023}{Vin} \tag{4}$$

Substituting the value of Vout from equation (2) in equation (4) gives the unknown resistance value R of the conductive rubber cord stretch sensor.

$$R = \frac{10K}{\left(\frac{1023}{ADC\ Reading} - 1\right)} \tag{5}$$

An increase in the resistance or the voltage read by Arduino ADC means that the conductive rubber cord laced to the basketball net is stretched which in general happens only when a ball passes through the net.

The microcontroller continuously reads the stretch sensor voltage from analog input pin A0. This value will be in the range of 0 and 1023 which is then mapped to a corresponding value in the range of 0 and 255. This value is then converted into resistance using the equation (5). But, instead of resistance, the voltage value is directly used to determine if a shot is made, as the percentage of change in resistance is same as voltage in this case. From Figure 11, the

voltage gradually increases when the rubber cord is stretched and gradually decreases when it is released. The voltage when the rubber cord starts stretching is stored in a variable lowestVoltage, which is used to calculate the reference voltage. The reference voltage is 6% more than the voltage lowestVoltage, the moment ball met the rubber cord when passing through the net. The voltage when the rubber cord is released, moment after the ball passes through the rubber cord attached to the net is stored in a variable highestVoltage. If the highestVoltage is greater than reference voltage, then it means that the change in voltage is due to the ball passing through the net. The score is then updated as per the LED color at that moment. Table 1 shows Arduino code snippet of detecting a basketball successfully and updating the score is accordingly.

```
void detectShot()
{
        sensorValue = analogRead(analogInPin);
        outputValue = map(sensorValue, 0, 1023, 0, 255);
        resistance = fixedResistance/((1023/sensorValue)-1);
        referenceValue = lowestVoltage + (lowestVoltage*6/100);
        if(lowestVoltage == 0 && highestVoltage == 0){
                lowestVoltage = outputValue;
                highestVoltage = outputValue;
        }
        if(outputValue <= previousOutputValue){
                if(trigger = 1){
                        if((highestVoltage > referenceValue) && (previousOutputValue !=
                    0)){
                        score = score + points;
                        }
                        trigger = 0;
                }
                lowestVoltage = outputValue;
                highestVoltage = outputValue;
        }
        if(outputValue > previousOutputValue){
                highestVoltage = outputValue;
                trigger = 1;
        }
        previousOutputValue = outputValue;
}
```

Table 1 Arduino code snippet to detect when a shot is made and increase the score accordingly

5.3 RIM RGB LED Lights

LED strips consists of flexible circuit boards with color LEDs soldered on. There are two different types of LED strips, analog and digital. Digital Strips have a driver chip on the strip that addressed each LED individually. These strips typically take 5V and can be run from a typical microcontroller board. Analog strips are non-addressable, which means all LEDs on the strip act as one, and they come in one color or full color spectrum. As the analog strip LEDs cannot be controlled individually, they fade and blink together. Analog strips are cheaper than digital strips.

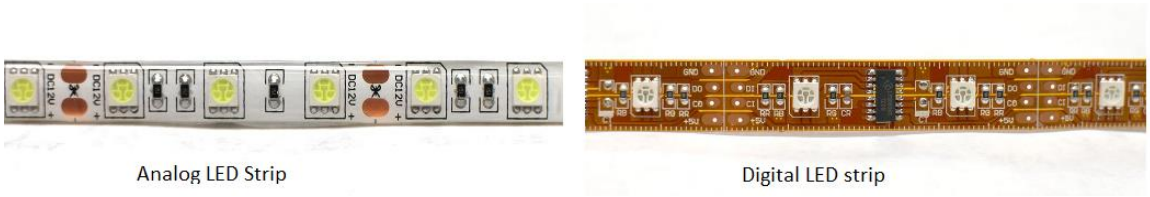

Figure 13 Analog and Digital LED strip

Analog strip of 32 segments which are linked by metal contact pads is attached around the basketball rim. Each segment has 3 LEDs and are wired in series, which means that the operating voltage of each segment is 3 times of a single LED. An average RGB LED requires 3.3V and 20mA, which means that a single segment of 3 RGB LEDs require approximately 9.9V and 60mA. [16] All the segments of same color are wired in parallel so that they all get the same amount of voltage throughout the strip and the current draw for each segment adds up depending upon the length of the strip.

The RGB analog LED strip used in the project has 4 wires, one for power, and one each for red, green, and blue control. The brightness and color of the LEDs can be controlled using microcontroller's Pulse-width modulation (PWM) technique. Depending upon the color and brightness of the LEDs, the strip used in this project requires a voltage in the range of 9-12V and a maximum current draw of 1.920 amps (32*60mA) when all the LEDs are set to ON. The Arduino's output pins can only supply around 40mA each. so, a driver circuit, which is a transistor amplifier, is used to boost the power. Three TIP41C NPN transistors are used in this circuit. TIP41C can handle up to 6 Amps of continuous current, [17] which is enough for about 300 LEDs with full bright white color.

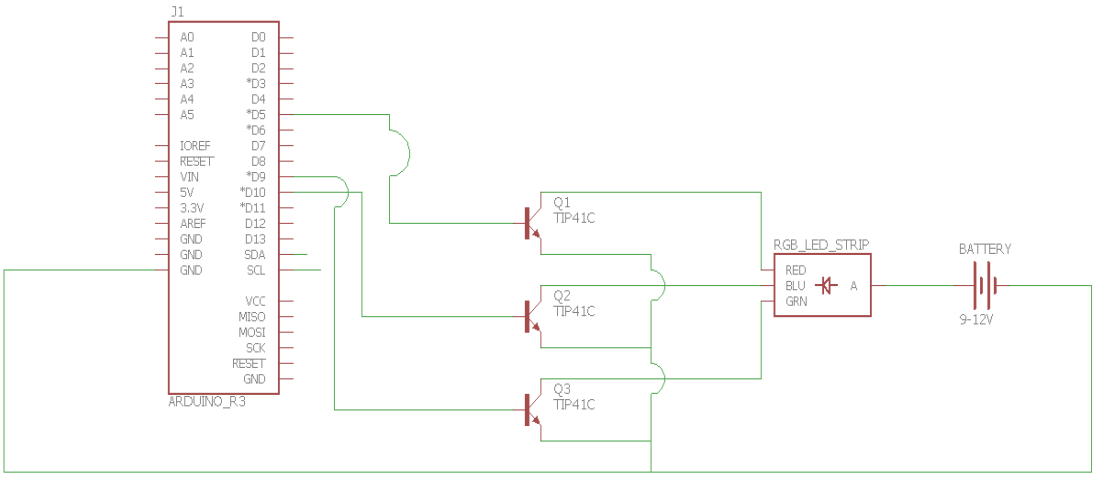

Figure 14 Schematics of RGB LED Strip driver circuit

The base of each transistor is connected to different PWM output ports of the Arduino, collector of each transistor is connected to one of red, blue, and green terminals of the LED strip, and the emitter of each transistor is connected to the ground. Arduino sends a 5V, low-current PWM signal to the transistor's base which switches the transistor, allowing it to conduct higher current at 9-12V across the collector and emitter through the LEDs.

Arduino is programmed such that when the system is ON, the LEDs are set to RED color for the first 30 seconds, changed to Green color for next 20 seconds, and then to Blue color for next 10 seconds. This keeps on rotating until the system is either paused or stopped. Alternatively, the point value of each basket is set depending on the LEDs color state.

LED Color	Duration	Shot Point Value
Red	30 Sec	2
Green	20 Sec	3
Blue	10 Sec	5

The cutoff time for each color is calculated by adding the current time to the duration for which the LEDs need to be set to that color. When the game starts, the LEDs are set to red color and the time until the LEDs should be red is calculated by adding 30000 ms to the current

system time millis(). If the current system time is greater than the red LED cutoff time, then the LEDs are set to green color and the time until the LEDs should be green is calculated by adding 20000 ms to the current system time millis(). If the current system time is greater than the green LED cutoff time, then the LEDs are set to blue color and the time until the LEDs should be blue is calculated by adding 10000 ms to the current system time millis(). If the current system time is greater than the blue LED cutoff time, then the LEDs are set to red color and the time until the LEDs should be red is calculated by adding 30000 ms to the current system time millis(). This process keeps on running in a loop until the system is stopped.

```
void setColor(int red, int green, int blue){
          analogWrite(REDPIN, red);
           analogWrite(GREENPIN, green);
           analogWrite(BLUEPIN, blue);
}
void setLEDColor(){
          if((redLedTime==0) && (greenLedTime==0) && (blueLedTime==0))
          {
                    redLedTime=millis()+30000L;
                    setColor(255,0,0); //Red
                    points = 2;
          }
          if((redLedTime==0) && (blueLedTime!=0) && (millis() > blueLedTime))
          {
                    redLedTime=millis()+30000L;
                    setColor(255,0,0); //Red
                    blueLedTime=0;
                    points = 2;
          }
          if((millis() > redLedTime) && (redLedTime!=0))
          {
                    greenLedTime=millis()+20000L;
                    setColor(0,255,0); //Green
                    redLedTime=0;
                    points = 3;
          }
          if((millis() > greenLedTime) && (greenLedTime!=0))
          {
                    blueLedTime = millis()+10000L;
                    setColor(0,0,255); //Blue
                    greenLedTime=0;
                    points = 5;
          }
}
```

Table 2 Arduino code controlling the LEDs color and point value of a basket

5.4 LED score display

Adafruit 1.2" 4-Digit 7-Segment Display w/I2C Backpack is used to display the score. This 7-segment display along with the backpack is very easy to work with than most other displays. Normally, 16 pins are required to drive most 7-segment displays, whereas, adafruit 7-segment display w/I2C Backpack requires only 2 pins as the backpack uses an I2C constant-current matrix controller on the back of the PCB. [18]

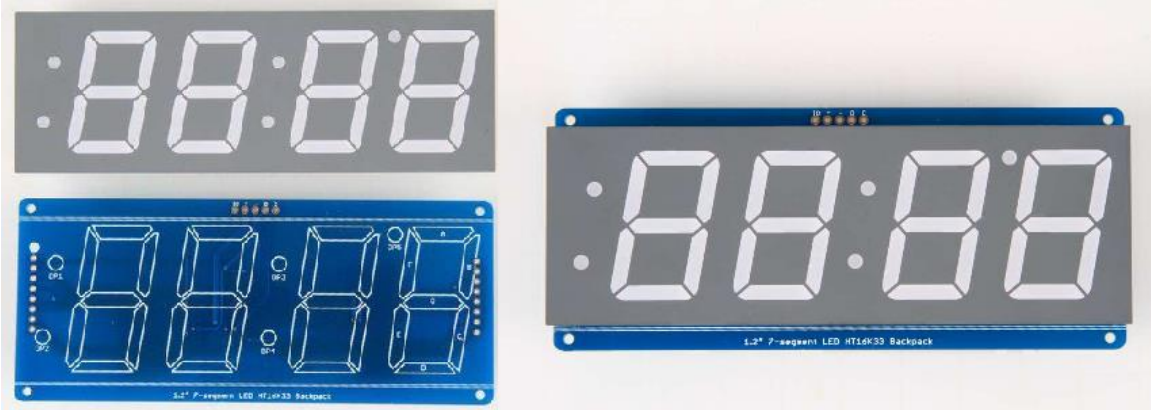

Figure 15 Adafruit 1.2" 4-Digit 7-Segment Display w/12C Backpack

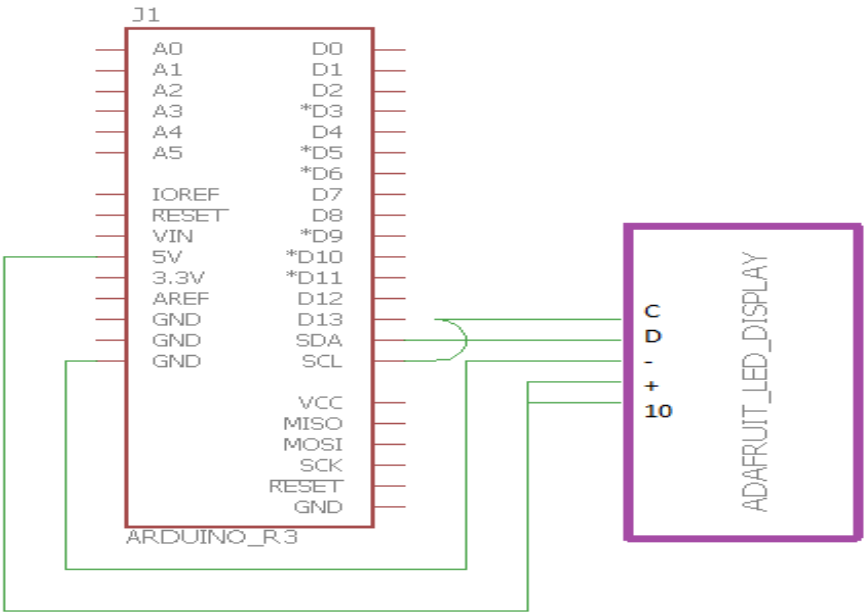

Figure 16 Schematics of LED display connected to Arduino Uno

5.5 Manual Controllers

Four Push buttons are used to start the game, pause the game, and to manually increase or decrease the score. “Push button is a component that connects two points in a circuit when pressed”. [19] One leg of the push button is connected to a 5V supply through a 10K Ohms resistor, which is also connected to a digital I/O pin on Arduino to read the button’s state, and the corresponding leg of the pushbutton is connected to ground.

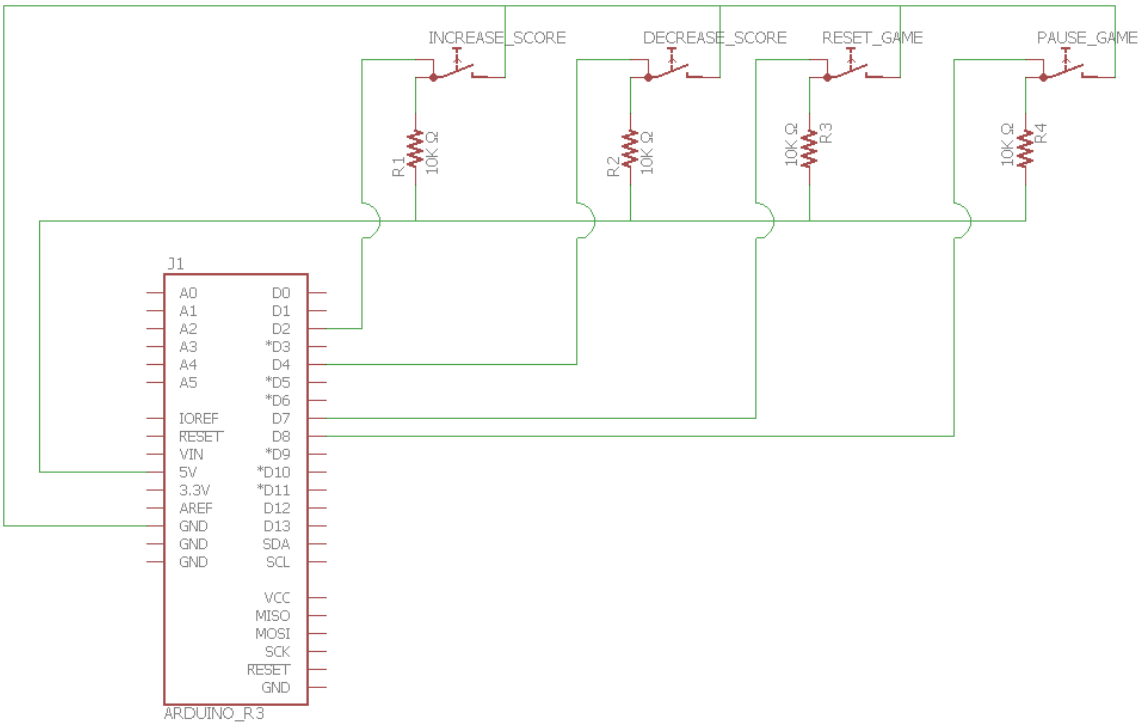

Figure 17 Schematics of manual controlled push buttons

There is no connection between the two legs of the push button when not pressed and is considered to be in open state. When the pushbutton is pressed, it is considered to be in closed state and it makes connection between its two legs. In open state, the Arduino digital I/O pin reads a HIGH as it is connected to 5V through a pull-up resistor and in closed state, it reads a LOW as it is connected to the Ground.

The microcontroller continuously reads the value from the I/O ports connected to the push buttons. All the four push buttons are checked for bouncing if the microcontroller detects a change in the button state. If the score increase button is pressed, the microcontroller reads

a LOW from the I/O port connected to the push button and the score is increased by 1. Similarly, if the score decrease button is pressed, the microcontroller reads a LOW from the I/O port connected to the push button and the score is decreased by 1. If the Reset game button is pressed and the microcontroller reads a LOW from the I/O port connected to it, then the game is restarted by setting required variables to their initial state.

If the Pause game button is pressed and the microcontroller reads a LOW from the I/O port connected to it, then the game is paused. In the process of pausing the game, first the remaining time of the LED color at that moment is calculated by subtracting the current time from the time until which the LEDs are supposed to be in that color. Next, the LEDs are set to Yellow, Purple and aqua with a delay of about 100 ms which makes them look like blinking indicating that the game is paused. This happens until the pause game button is again pressed to resume the game. When the microcontroller reads a LOW from the I/O port connected to pause button, it sets the LED color to the color before the game is paused and resumes the game.

Combining the five sub-systems Microcontroller Board, Sensor, Rim RGB LED strip, LED Score display, and Manual controllers gives a low-cost, and battery-powered, automated basketball scoring system. Arduino Uno sets the color of the LEDs and continuously reads the voltage from the stretch sensor using analog input port A0. The shot made is identified if this voltage value from the stretch sensor is at least greater than 6% of the reference voltage derived using previous voltage values. The score is increased depending upon the color of the LEDs when the shot is made. There are manual push buttons to restart the game, pause the game, and to adjust the score. Figure 18 shows the complete schematic diagram of automated basketball scoring system.

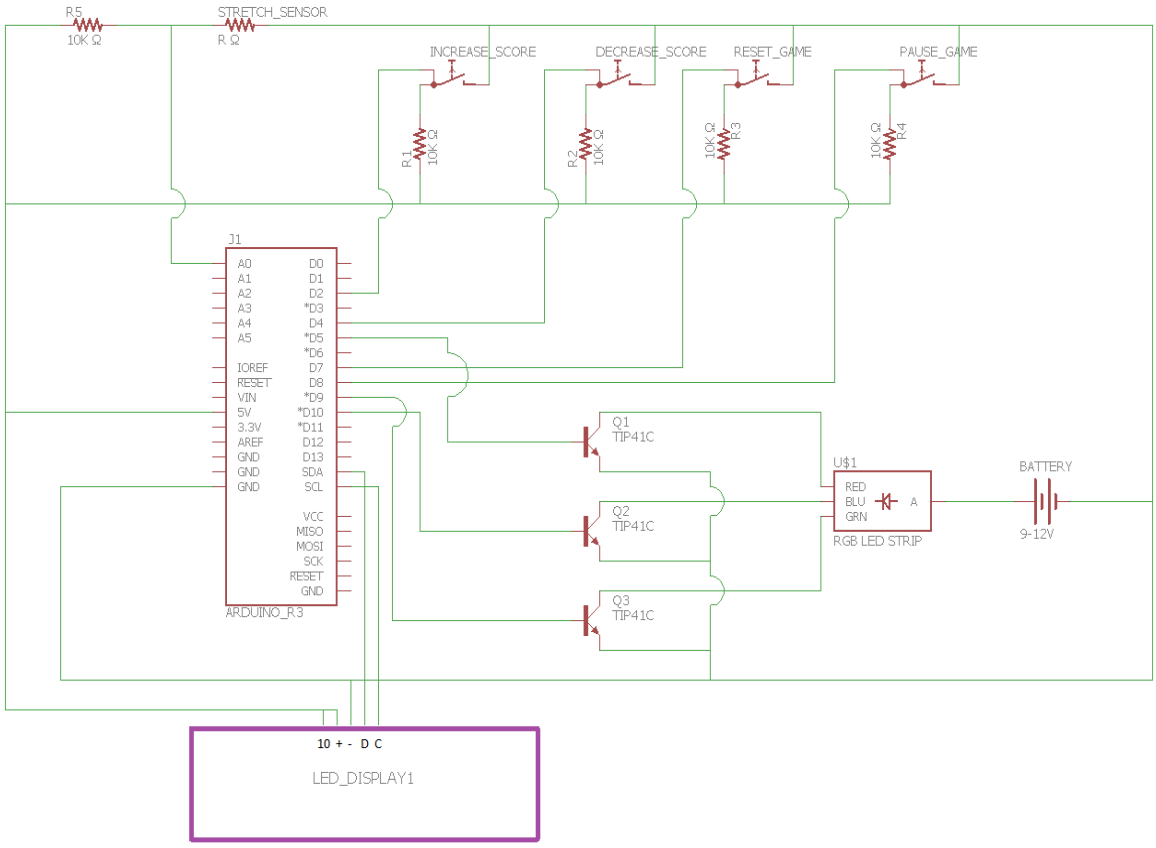

Figure 18 Schematic diagram of Automated Basketball scoring system

The following section describes the evaluation of the designed automated basketball scoring system.

6 System Evaluation

A prototype of the designed automated basketball scoring system is built and tested in the lab by shooting and dropping the ball through the net. The functionality and accuracy of the entire system is tested.

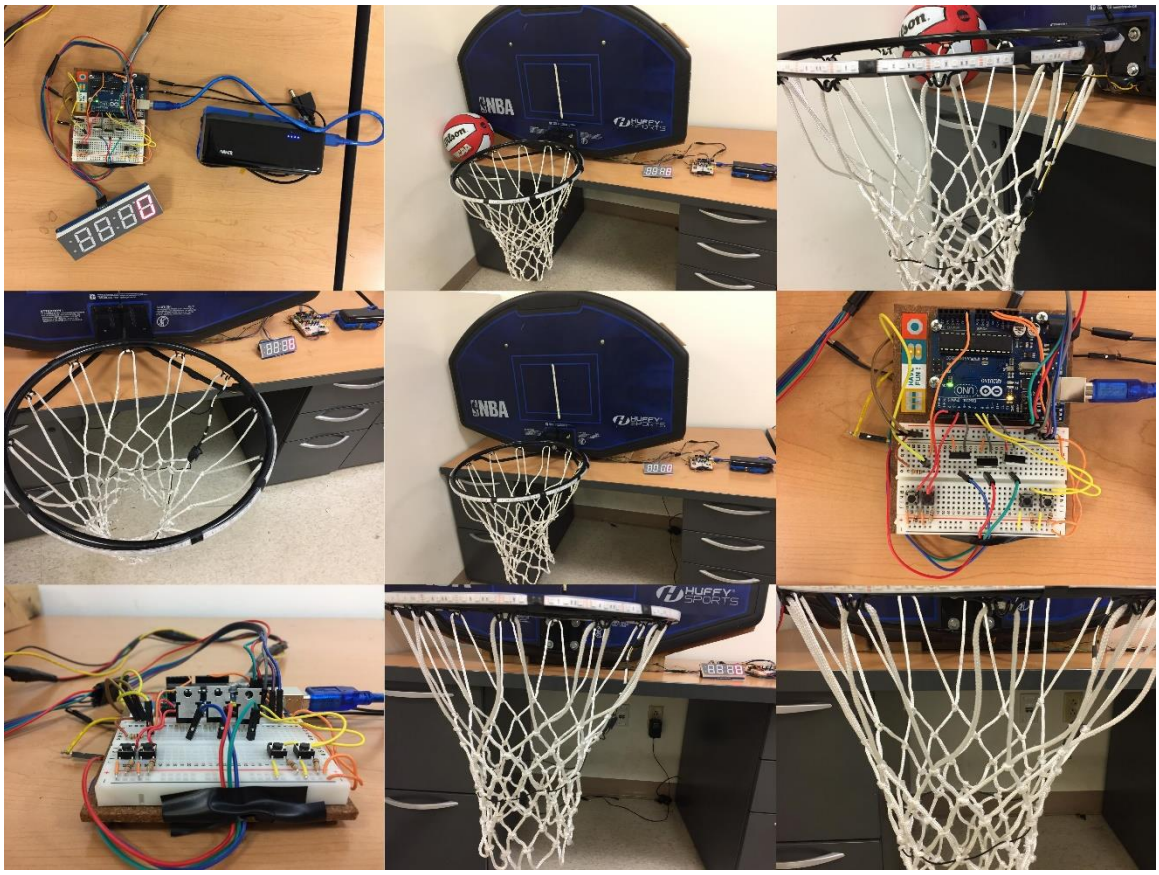

Figure 19 Pictures of the designed automated basketball system prototype

A basketball is dropped through the net for a predefined number of times for each LED color. A minute in which the LEDs change from Red to Green to Blue is considered to be a cycle. The score displayed by the system after each cycle is noted by pausing the game. This system generated score is compared with the theoretical score to measure the accuracy of the system. This process has been repeated 3 times to test the approximate accuracy of the designed automated basketball scoring system.

Time/Cycle	Shots when Red	Shots when Green	Shots when Blue	Expected Score for this cycle	Total expected Score	System Reported score for this cycle	System Reported Total Score
1 min	5	4	2	32	32	32	32
2 min	6	3	1	26	58	26	58
3 min	7	4	2	36	94	36	94
4 min	5	4	2	32	126	32	126

Table 3 Test 1 of automated basketball scoring system

Figure 20 Test 1 picture of scores at the end of each cycle/minute

Time/Cycle	Shots when Red	Shots when Green	Shots when Blue	Expected Score for this cycle	Total expected Score	System Reported score for this cycle	System Reported Total Score
1 min	4	2	1	19	19	22	22
2 min	5	3	2	29	48	31	53
3 min	5	4	1	27	75	29	82

Table 4 Test 2 of automated basketball scoring system

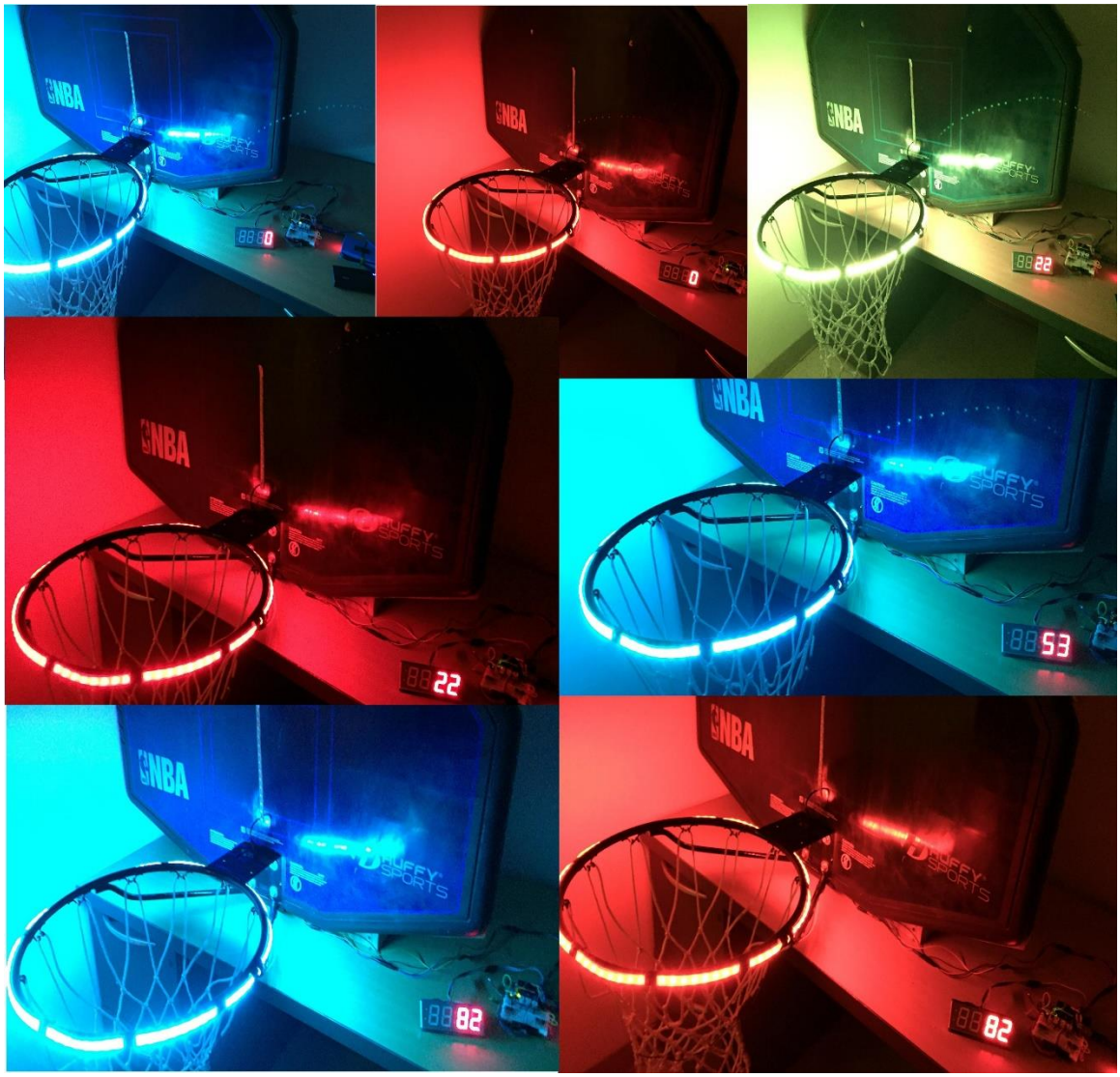

Figure 21 Test 2 Picture of scores at the end of each cycle/minute

Time/Cycle	Shots when Red	Shots when Green	Shots when Blue	Expected Score for this cycle	Total expected Score	System Reported score for this cycle	System Reported Total Score
1 min	5	2	1	21	21	21	21
2 min	6	3	2	31	52	31	52
3 min	5	3	2	29	81	29	81
4 min	4	2	1	19	100	19	100
5 min	5	4	1	27	127	27	127

Table 5 Test 3 of automated basketball scoring system

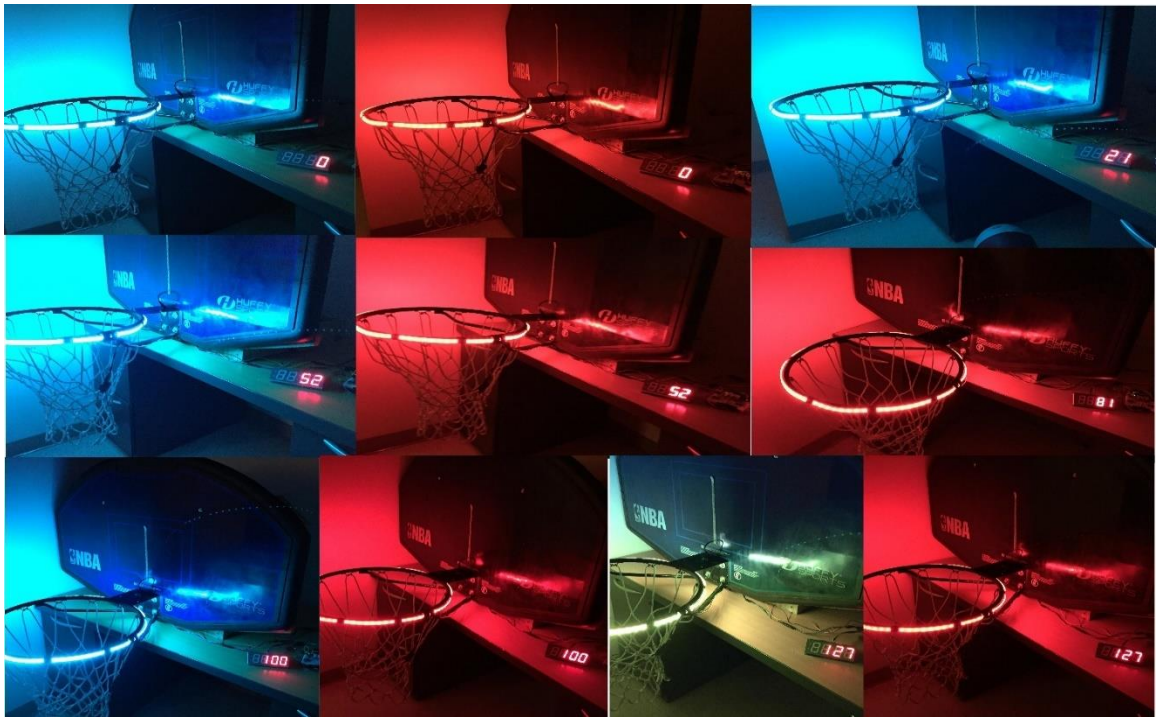

Figure 22 Test 3 Picture of scores at the end of each cycle/minute

Test 1 and Test 3 system results are in line with the expected results. Test 2 system results are slightly deviated from the expected results. This might have been because of the interference of the table on which backboard is placed with the net and ball. Overall, the basketball has been shot or dropped through the net 118 times and the system results are as expected for 115 times which puts the system accuracy at about 97.5% approximately. This accuracy rate might vary when the system is tested in a real time environment with players shooting at the basket.

7 Conclusion

In this project, a low cost, battery powered, automated basketball scoring system has been designed, implemented, and tested to an extent. The designed automated basketball scoring system detects when the ball passes through the net with the point value of each basket depending on the recurrently changing LEDs color placed on the hoop. This system uses an Arduino Uno microcontroller, a conductive rubber cord stretch sensor, an analog RGB LED strip, and other electronic components such as transistors, resistors and pushbuttons which are widely used, readily available, inexpensive, and affordable. Note that the system has not yet been tested under game conditions.

8 Future Work

Further study can be done on finding other inexpensive materials that can be used as a conducting stretch sensor. Different approaches such as creating a certain portion of the net with conductive yarn, or lacing the stretch sensor through the net can also be explored. The manual controls provided to adjust the score, start, and pause the game uses pushbuttons. By using wireless transmitter and receiver, these pushbuttons can be replaced by a wireless controller which is easy to use. The designed system in this project gives false results if the ball passes through the hoop from below the net. This can be fixed by using another sensor which can track the direction of the ball into hoop. Further study can be done in finding an optimal way to resolve this scenario.

9 Bibliography

[1] [Online]. Available: http://www.nvre.org/.

[2] R. Seamen, *President & CEO, New Vision Renewable Energy.*

[3] W. M. Klein, "Real-time wireless sensor scoring". US Patent US7998004 B2, 16 August 2011.

[4] S.-Y. Huang, "Method and device for detecting goal in basketball game device". US Patent US20040224797 A1, 11 November 2004.

[5] K. L. K. D. J. P. A. G. W. Robert T. Thurman, "Basketball sensing apparatus". US Patent US20140200692 A1, 17 July 2014.

[6] J. L. Best, "Score-sensitive basketball hoop". US Patent US4858920 A, 22 August 1989.

[7] D. D. R. C. A. K. D. A. D. Bruce C. Ianni, "Basketball shot-tracking system". US Patent US9254432 B2, 9 February 2016.

[8] D. D. R. C. A. K. T. J. K. H. K. H. J. R. A. G. P. M. H. Bruce C. Ianni, "Basketball net which detects shots that have been made successfully". US, World Patent US20160096067 A1, WO2016057535A1, 7 April 2016.

[9] J. J. M. D. N. S. A. Victor Michael Bove, "Force-sensing net". US Patent US20130172131 A1, US9233287, US20160082331, 4 July 2013.

[10] "Arduino," [Online]. Available: https://www.arduino.cc/en/Guide/Introduction. [Accessed 15 November 2016].

[11] "Images Scientific Instruments," [Online]. Available: http://www.imagesco.com/sensors/stretch-sensor.html. [Accessed 15 November 2016].

[12] "Conductive Rubber Cord Stretch Sensor," Adafruit, [Online]. Available: https://www.adafruit.com/product/519. [Accessed 15 November 2016].

[13] "Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Voltage_divider. [Accessed 15 November 2016].

[14] "Sparkfun ADC," [Online]. Available: https://learn.sparkfun.com/tutorials/analog-to-digital-conversion. [Accessed 15 November 2016].

[15] "Arduino Analog Voltage," [Online]. Available: https://www.arduino.cc/en/Tutorial/ReadAnalogVoltage. [Accessed 15 November 2016].

[16] "Adafruit LED strips," [Online]. Available: https://learn.adafruit.com/rgb-led-strips. [Accessed 16 November 2016].

[17] "STMicroelectronics TIP41C NPN transistor datasheet," [Online]. Available: www.st.com/resource/en/datasheet/tip42c.pdf. [Accessed 15 November 2016].

[18] "Adafruit 7-segment display backpack," [Online]. Available: https://learn.adafruit.com/adafruit-led-backpack/1-2-inch-7-segment-backpack. [Accessed 15 November 2016].

[19] "Arduino push button," [Online]. Available: https://www.arduino.cc/en/tutorial/pushbutton. [Accessed 16 November 2016].

[20] "Portable Solar Power Pack & Light," [Online]. Available: http://www.nvre.org/powerandlight/index.html.

[21] M. Kadlec, Department Of Ecology State of Washington, 1992.

[22] "Adafruit "Using a Thermistor"," [Online]. Available: https://learn.adafruit.com/thermistor/using-a-thermistor. [Accessed 15 November 2015].

[23] "Arduino Uno," [Online]. Available: https://www.arduino.cc/en/Main/ArduinoBoardUno. [Accessed 15 November 2016].

[24] R. R. Hampton, "Basketball goal sensor for detecting shots attempted and made". US Patent US6389368 B1, 14 May 2002.

10 Appendix A

10.1 Arduino Code

```
#include <Wire.h>
#include "Adafruit_LEDBackpack.h"
#include "Adafruit_GFX.h"

#define REDPIN 9
#define GREENPIN 10
#define BLUEPIN 5
#define fixedResistance 10000
Adafruit_7segment matrix = Adafruit_7segment();

const int analogInPin = A0;
const int resetButtonPin = 8;
const int pauseButtonPin = 7;
const int scoreIncreaseButtonPin = 2;
const int scoreDecreaseButtonPin = 4;
int score = 0;
int points = 0;
int trigger = 0;
int outputValue = 0;
int lowestVoltage = 0;
int highestVoltage = 0;
int previousOutputValue = 0;
int resistance;
float sensorValue;
float referenceValue;
unsigned long redLedTime = 0;
unsigned long greenLedTime = 0;
unsigned long blueLedTime = 0;
```

```
unsigned long redLedRemainingTime = 0;
unsigned long greenLedRemainingTime = 0;
unsigned long blueLedRemainingTime = 0;
String color = "";
int resetButtonState;
int pauseButtonState;
int scoreIncreaseButtonState;
int scoreDecreaseButtonState;
int lastResetButtonState = LOW;
int lastPauseButtonState = LOW;
int lastScoreIncreaseButtonState = LOW;
int lastScoreDecreaseButtonState = LOW;
unsigned long lastDebounceTime = 0;
unsigned long debounceDelay = 50;

void setup()
{
        Serial.begin(9600);
        matrix.begin(0x70);
        pinMode(REDPIN, OUTPUT);
        pinMode(BLUEPIN, OUTPUT);
        pinMode(GREENPIN, OUTPUT);
        pinMode(resetButtonPin, INPUT);
        pinMode(pauseButtonPin, INPUT);
        pinMode(scoreIncreaseButtonPin, INPUT);
        pinMode(scoreDecreaseButtonPin, INPUT);
}

void loop()
{
        manualControls();
```

```
        setLEDColor();
        detectShot();
        matrix.print(score, DEC);
        matrix.writeDisplay();
        delay(100);
}

void manualControls()
{
        int readingToResetGame = digitalRead(resetButtonPin);
    if (readingToResetGame != lastResetButtonState) {
        lastDebounceTime = millis();
    }
    if ((millis() - lastDebounceTime) > debounceDelay) {
        if (readingToResetGame != resetButtonState) {
            resetButtonState = readingToResetGame;
        }
        if(readingToResetGame == 0){
                resetGame();
          }
    }
    lastResetButtonState = readingToResetGame;

    int readingToPauseGame = digitalRead(pauseButtonPin);
    if (readingToPauseGame != lastPauseButtonState) {
        lastDebounceTime = millis();
    }
    if ((millis() - lastDebounceTime) > debounceDelay) {
        if (readingToPauseGame != pauseButtonState) {
            pauseButtonState = readingToPauseGame;
        }
```

```
        if(readingToPauseGame == 0){
                pauseGame();
          }
}
lastPauseButtonState = readingToPauseGame;

int readingToIncreaseScore = digitalRead(scoreIncreaseButtonPin);
if (readingToIncreaseScore != lastScoreIncreaseButtonState) {
        lastDebounceTime = millis();
}
if ((millis() - lastDebounceTime) > debounceDelay) {
        if (readingToIncreaseScore != scoreIncreaseButtonState) {
              scoreIncreaseButtonState = readingToIncreaseScore;
        }
        if(readingToIncreaseScore == 0){
                score=score+1;
          }
}
lastScoreIncreaseButtonState = readingToIncreaseScore;

int readingToDecreaseScore = digitalRead(scoreDecreaseButtonPin);
if (readingToDecreaseScore != lastScoreDecreaseButtonState) {
        lastDebounceTime = millis();
}
if ((millis() - lastDebounceTime) > debounceDelay) {
        if (readingToDecreaseScore != scoreDecreaseButtonState) {
              scoreDecreaseButtonState = readingToDecreaseScore;
        }
        if(readingToDecreaseScore == 0 && score > 0){
                score=score-1;
          }
```

```
        }
        lastScoreDecreaseButtonState = readingToDecreaseScore;
}

void detectShot()
{
        sensorValue = analogRead(analogInPin);
        outputValue = map(sensorValue, 0, 1023, 0, 255);
        resistance = fixedResistance/((1023/sensorValue)-1);
        referenceValue = lowestVoltage + (lowestVoltage*6/100);
        if(lowestVoltage == 0 && highestVoltage == 0){
                lowestVoltage = outputValue;
                highestVoltage = outputValue;
        }
        if(outputValue <= previousOutputValue){
                if(trigger = 1){
                        if((highestVoltage > referenceValue) && (previousOutputValue != 0)){
                        score = score + points;
                        }
                        trigger = 0;
                }
                lowestVoltage = outputValue;
                highestVoltage = outputValue;
        }
        if(outputValue > previousOutputValue){
                highestVoltage = outputValue;
                trigger = 1;
        }
        previousOutputValue = outputValue;
}
```

```
void setColor(int red, int green, int blue)
{
        analogWrite(REDPIN, red);
        analogWrite(GREENPIN, green);
        analogWrite(BLUEPIN, blue);
}

void setLEDColor()
{
        if((redLedTime==0) && (greenLedTime==0) && (blueLedTime==0))
        {
                redLedTime=millis()+30000L;
                setColor(255,0,0);
                points = 2;
                color = "Red";
        }
        if((redLedTime==0) && (blueLedTime!=0) && (millis() > blueLedTime))
        {
                redLedTime=millis()+30000L;
                setColor(255,0,0);
                blueLedTime=0;
                points = 2;
                color = "Red";
        }
        if((millis() > redLedTime) && (redLedTime!=0))
        {
                greenLedTime=millis()+20000L;
                setColor(0,255,0);
                redLedTime=0;
                points = 3;
                color = "Green";
```

```
        }
        if((millis() > greenLedTime) && (greenLedTime!=0))
        {
                blueLedTime = millis()+10000L;
                setColor(0,0,255);
                greenLedTime=0;
                points = 5;
                color = "Blue";
        }
}

void resetGame()
{
        previousOutputValue = 0;
        outputValue = 0;
        points = 0;
        score=0;
        redLedTime = 0;
        greenLedTime = 0;
        blueLedTime = 0;
        lowestVoltage = 0;
        highestVoltage = 0;
        color = "";
        trigger = 0;
}

void pauseGame()
{
      if(redLedTime != 0){
            redLedRemainingTime = redLedTime - millis();
            initiatePause();
```

```
            resumeGame(255,0,0,redLedRemainingTime);
        }
    if(greenLedTime != 0){
            greenLedRemainingTime = greenLedTime - millis();
            initiatePause();
            resumeGame(0,255,0,greenLedRemainingTime);
        }
        if(blueLedTime != 0){
            blueLedRemainingTime = blueLedTime - millis();
            initiatePause();
            resumeGame(0,0,255,blueLedRemainingTime);
        }
}

void initiatePause()
{
    delay(250);
    for(int i=0; digitalRead(pauseButtonPin) != 0; i++){
        analogWrite(REDPIN, 255);
            analogWrite(GREENPIN, 255);
            analogWrite(BLUEPIN, 0);
        delay(100);
        analogWrite(REDPIN, 80);
            analogWrite(GREENPIN, 0);
            analogWrite(BLUEPIN, 80);
        delay(100);
        analogWrite(REDPIN, 0);
            analogWrite(GREENPIN, 255);
            analogWrite(BLUEPIN, 255);
        delay(100);
    }
```

```
        delay(100);
}

void resumeGame(int red, int green, int blue, unsigned long remainingTime)
{
        setColor(red,green,blue);
        if(red == 255){redLedTime = millis() + remainingTime;}
        if(green == 255){greenLedTime = millis() + remainingTime;}
        if(blue == 255){blueLedTime = millis() + remainingTime;}
}
```