Garage

ENGR102 Intro to Electronics

Shoreline Community College

November 2017

Description of the problem. Why are you building this circuit?

Having finished the 16 experiments in the SIK Guide I wanted to mix all of the gained knowledge and even more, so I obtained the Elegoo 37 Kit Sensors which also came with a guide that was very easy to follow. Garage features a laser and two photoresistors that will keep calibrating the system while listening for abrupt changes in the absolute value of the difference between the average difference between the reading of the two photoresistors over the last 200 samples and the current difference being read, meaning a car (or an object) is in the way of the laser requesting entrance. The user who wants to open the door will have to enter the code into the keypad and then the servo will open the door when the code is correct (Garage also features a buzzer and a LED for wrong-correct animations).

What things can you do with this circuit?

You can simulate how a modern garage door (or simply a toll) will work by using a laser that detects obstructions (cars) requesting entrance and a keypad for authentication. This system is also a security system provided with a secret key that only the owner knows (it is #102 resembling ENGR102 for demonstration purposes).

Knowledge from which prior experiments is required?

Knowledge from the following circuits in the SIK Guide (The codes for these circuits can be found at: https://www.sparkfun.com/sikcode):

- Circuit #3: RGB LED; needed for animation and user interfacing/experience purposes.
- Circuit #6: Photo Resistor; needed for the core of Garage.
- Circuit #8: A Single Servo; needed for opening the door.
- Circuit #11: Piezo Element; needed for animation and user interfacing/experience purposes.

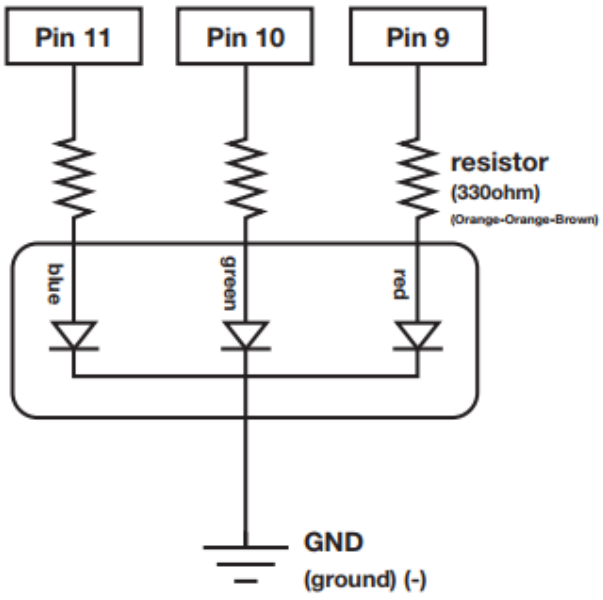

Figure 1. Circuit #3 Schematic (RGB LED).

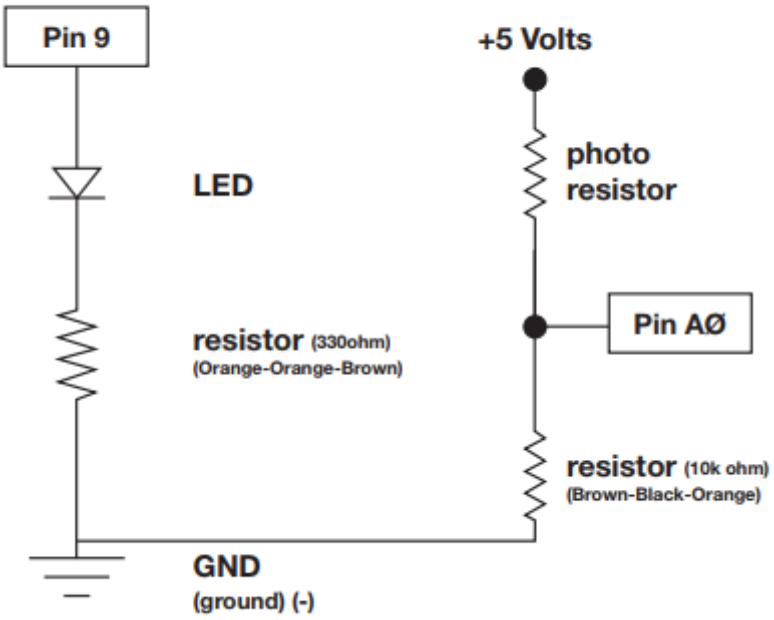

Figure 2. Circuit #6 Schematic (Photo Resistor).

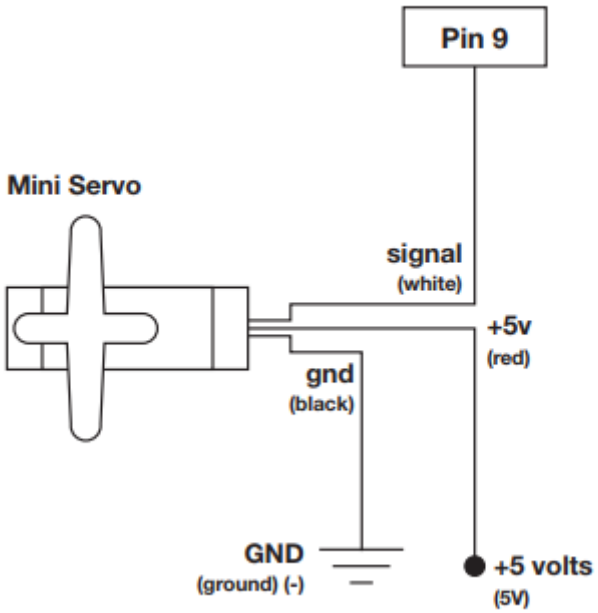

Figure 3. Circuit #8 Schematic (Servo).

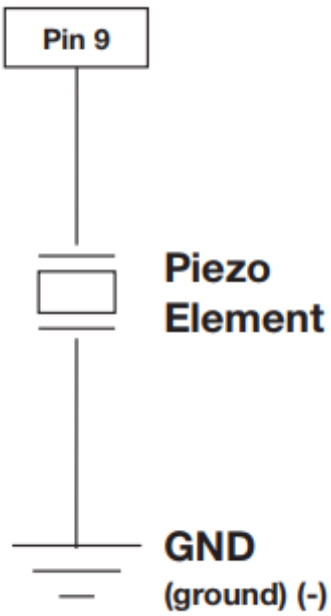

Figure 4. Circuit #11 Schematic (Piezo Buzzer).

And from the following lessons in the Elegoo 37 Sensors Kit Guide (The codes come with the bundled CD):

- Lesson 8: Laser Module; needed for the core of Garage.
- Lesson 12: Photoresistor Module; needed for the core of Garage.
- Lesson 31: Keypad Module; needed for authentication purposes.

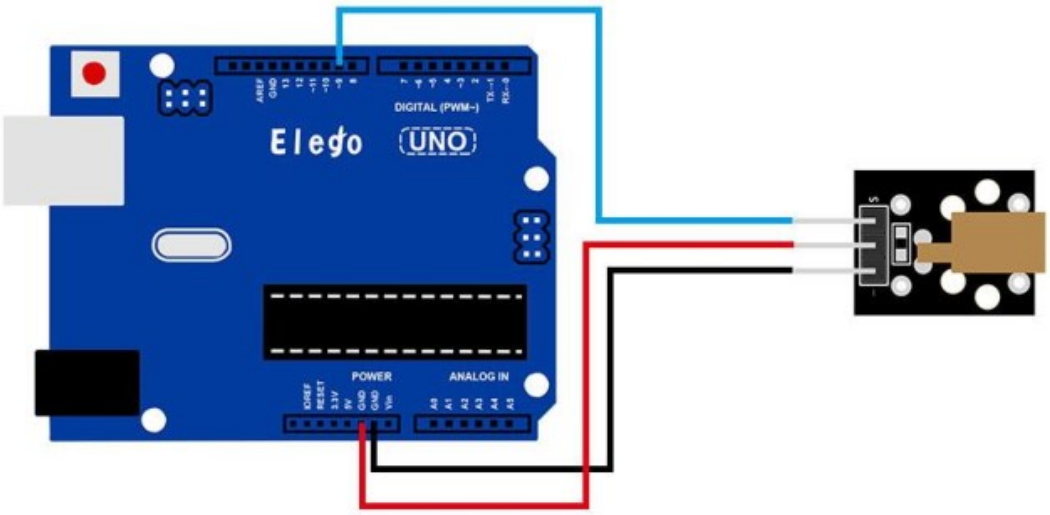

Figure 5. Lesson 8 Wiring Diagram (Laser).

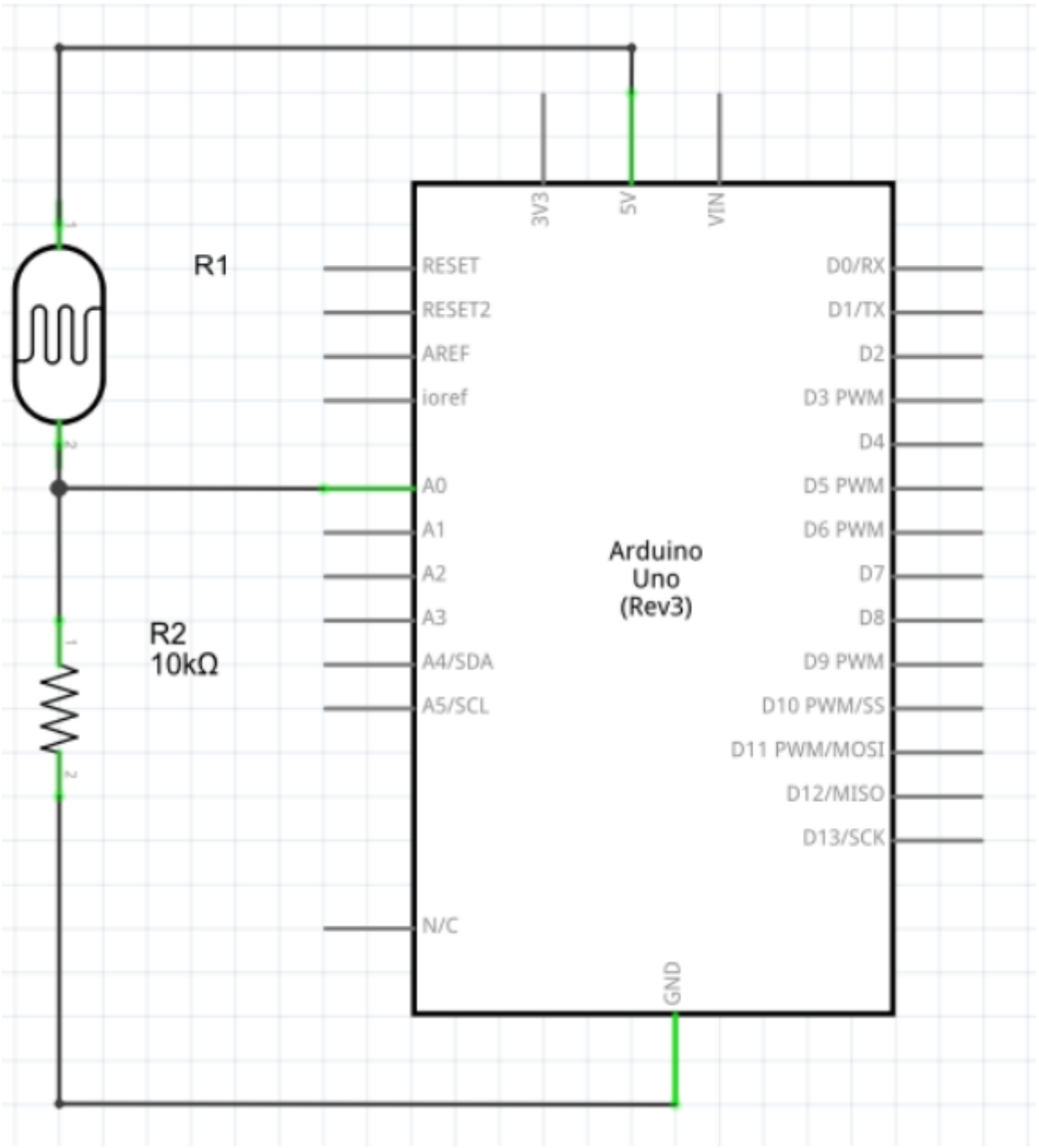

Figure 6. Lesson 12 Schematic (Photo Resistor).

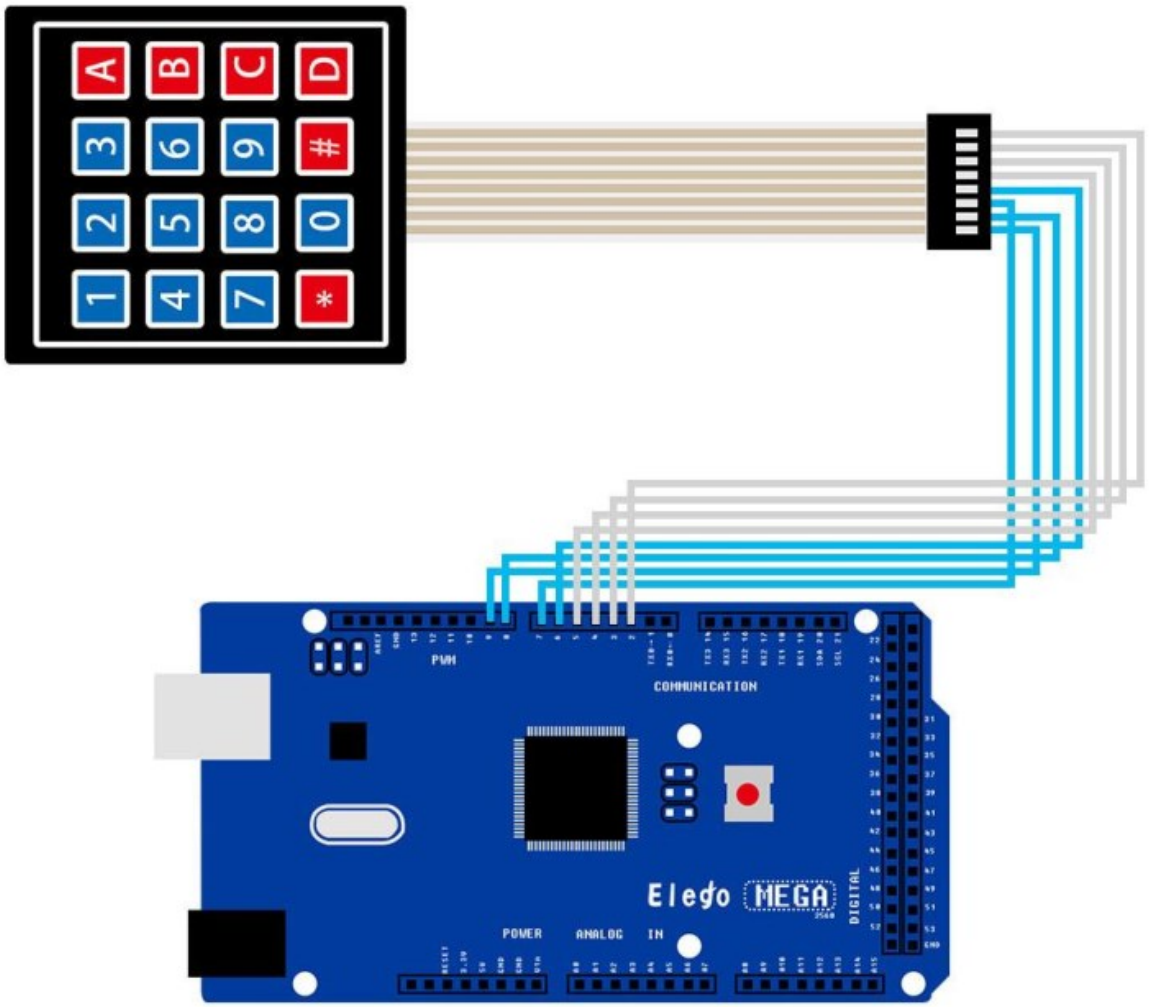

Figure 7. Lesson 31 Wiring Diagram (Keypad).

Parts Required

- One (1) Arduino Board from the Sparkfun kit.
- One (1) Piezo Buzzer
- One (1) RGB LED.
- One (1) Laser Module.
- One (1) Servo.
- One (1) 4x4 Keypad of eight (8) Digital Pins.
- Tape as needed.
- Two (2) mini breadboards.
- Two (2) 10kΩ Resistor.
- Two (2) 330Ω Resistors.
- Two (2) Photo Resistors.
- Some paper/plastic (for making a shield for the targeted by the laser photoresistor).
- Jumpers/Wire as needed.

Problem-solving approach

I followed the following steps:

1) Learn how to control each element with the Arduino Board:
   1.1) Learn how to turn green and red colors with the RGB LED.
   1.2) Learn how to play some melodies (correct/incorrect melodies) with the piezo buzzer.
   1.3) Learn how to use the photoresistors.
   1.4) Learn how to calibrate the photoresistors.
   1.5) Learn how to use the laser module.
   1.6) Learn how to use the servo to move a gate.
2) Put everything together.
3) Do the coding.
4) Debug.
5) Done.

Problems

- Selecting melodies for the correct/incorrect case was not so easy. Basically I borrowed some code from Discoteca and began testing many melodies until I found one that I liked for each case.
- Calibrating the photoresistors was not so easy. This is what I ended up doing (I really loved this approach):
  - Two photo-resistors, one for reference of ambient light (which we'll call Ref) and the target of the laser (which we'll call laser).
  - Read both photoresistors (one is being pointed with the laser).
  - The difference is abs(reading at Ref - Reading at target).
  - This is stored in an array of 200 other most recent samples (the system is calibrating itself continuously, because light conditions may change during the day).
  - The average of those differences is computed.
  - Take the difference between the current difference and the average difference, this is abs(current difference – Average Difference from the 200 most current samples). If this value is higher than some fraction of the Average, there must be something in the way. This fraction is taken by testing under the worse light conditions our system will have to handle which is the case that lights are too bright and the laser is barely distinguishable (the difference will be too small and thus harder to notice the obstacle. This actually cannot be resolved 100% (as we go to infinity levels of brightness), but in real life scenario cases it was found 40% is the best number.
  - Another solution (the issue was solved in the last few bullet points, but this helps even more) is making a hut for the target photoresistor.

Schematic

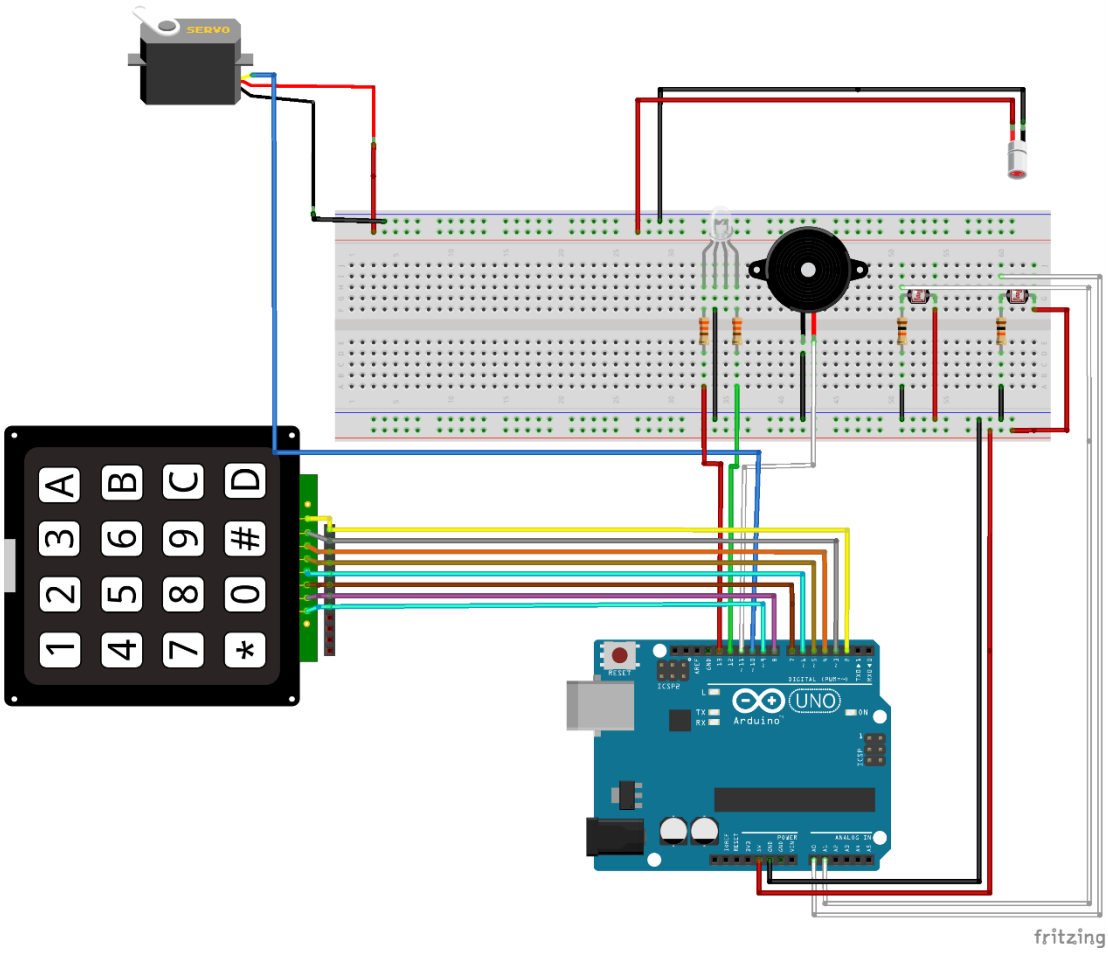

Figure 8. Breadboard View

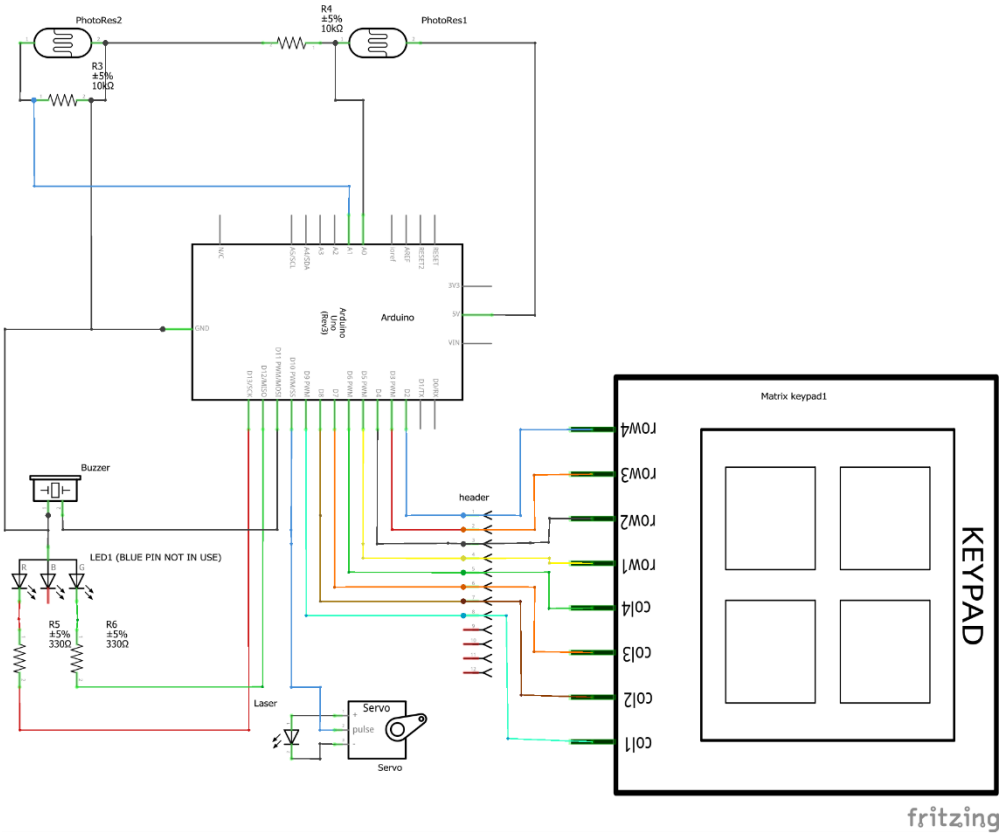

Figure 9. Schematic View

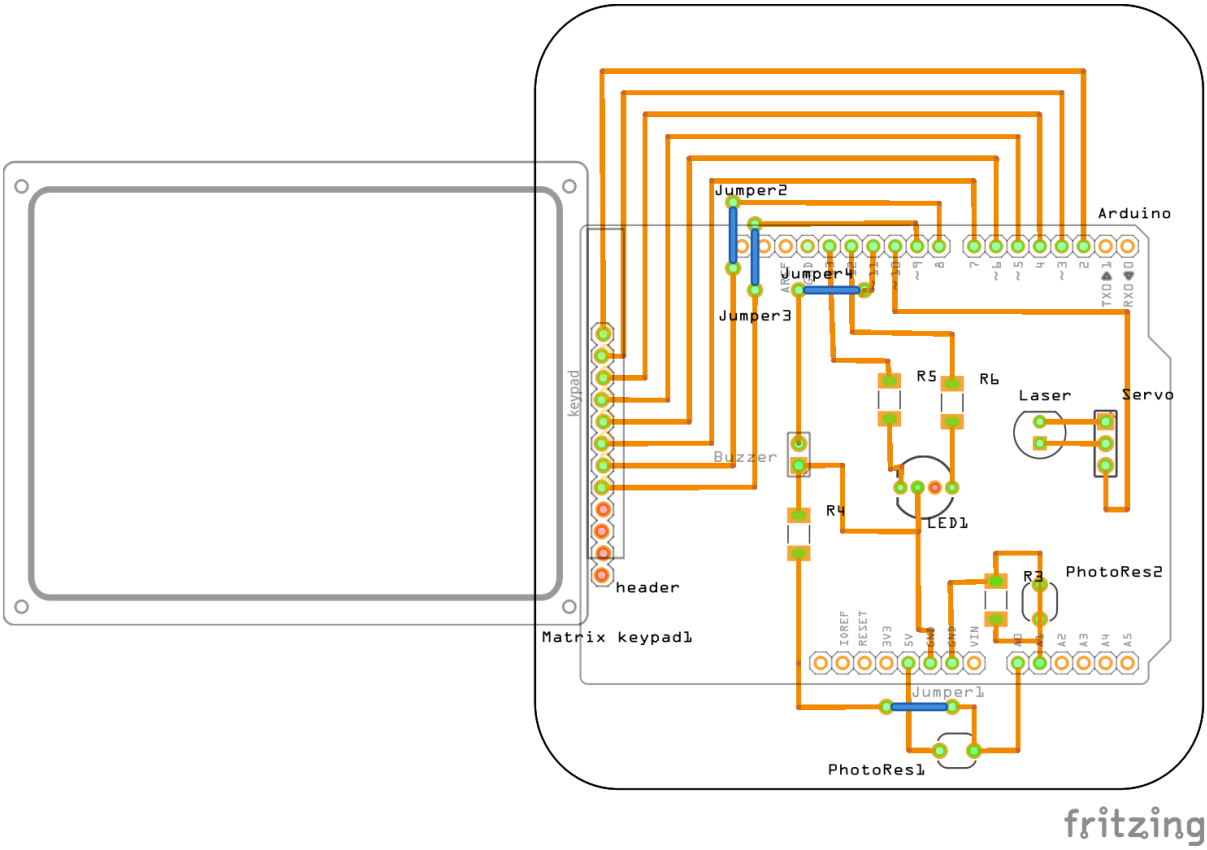

Figure 10. PCB View.

Code

```
/*
 * Name: Rafael Antonio Laya Alfonzo
 * November 2017
 * Class: ENGR 102 intro to electronics (Fall 2017)
 * Software Version: Arduino 1.8.2
 * Source code for Discoteca 2.0 more information on the report
 * The reader can use this code as they please but I won't be liable of anything
*/


#include <Keypad.h>
#include <Servo.h>

Servo servo1;

// constants and global variables
const int lightTargetPin = A0;
const int lightRefPin = A1;
int lightTargetVal;
int lightRefVal;

// this keeps track of the callibration samples
int const numOfSamples = 200;
int diffBetweenSensors[numOfSamples];
int counter = 0;
```

```
String defaultPassword = "#102";
String password;

const int servoClosedAngle = 150;
const int servoOpenAngle = 0;

const byte ROWS = 4; //four rows
const byte COLS = 4; //four columns
//define the cymbols on the buttons of the keypads
char hexaKeys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};

const int maxPasswordLength = 4;

byte rowPins[ROWS] = {9, 8, 7, 6}; //connect to the row pinouts of the keypad
byte colPins[COLS] = {5, 4, 3, 2}; //connect to the column pinouts of the keypad

//initialize an instance of class NewKeypad
Keypad customKeypad = Keypad( makeKeymap(hexaKeys), rowPins, colPins,
ROWS, COLS);

const int redLed = 13;
const int greenLed = 12;
const int buzzerPin = 11;
const int tempo = 113;
const int beats = 4;
const int numNotes = 8;
const int duration = beats * tempo;
const int NUM_NOTES = 8;


int GREEN_RG[2] = {0, 1};
int RED_RG[2] = {1, 0};
int RG_PINS[2] = {redLed, greenLed};
int OFF_RG[2] = {0, 0};

void turnGreenLights();

void setup()
{
  pinMode(redLed, OUTPUT);
  pinMode(greenLed, OUTPUT);
  setPasswordTo(defaultPassword);

  servo1.attach(10, 900, 2100);  //Connect the servo to pin 9
              //with a minimum pulse width of
              //900 and a maximum pulse width of
              //2100.
```

```
}

void loop()
{
  // make sure the doors are closed and keep listening for cars
  closeDoors();
  listenForCars();
}

void setPasswordTo(String newPassword)
{
  password = newPassword;
}

void listenForPassword()
{
  String attempt = readKeypadForAttempt(maxPasswordLength);

  if(attempt == password)
  {
    playCorrectPassword();
    turnGreenLights();
    openDoors();
    turnLightsOff();
  }
  else
  {
    playIncorrectPassword();
    turnRedLights();
    delay(500);
    turnLightsOff();
  }
  return;
}

void closeDoors()
{
  servo1.write(servoClosedAngle);
}

void openDoors()
{
  int currentDiff;
  do
  {
    servo1.write(servoOpenAngle);
    lightTargetVal = analogRead(lightTargetPin);
    lightRefVal = analogRead(lightRefPin);
    currentDiff = abs(lightTargetVal - lightRefVal);
    delay(5000);
    closeDoors();
  } while(laserIsCovered(currentDiff));
```

```
  return;

}

void listenForCars()
{
  lightTargetVal = analogRead(lightTargetPin);
  lightRefVal = analogRead(lightRefPin);

  int currentDiff = abs(lightTargetVal - lightRefVal);


  if(laserIsCovered(currentDiff) and (counter > numOfSamples - 1))
  {
    listenForPassword();
    return;
  }
  else
  {
    diffBetweenSensors[counter % numOfSamples] = currentDiff;
    counter++;
  }
}

boolean laserIsCovered(int currentDiff)
{
  return ((counter > 99) and (abs(currentDiff - avgDiffBetweenSensors()) >
((40.0/100.0) * float(avgDiffBetweenSensors()))));
}

int avgDiffBetweenSensors()
{
  unsigned int sum = 0;
  for(int i = 0; i < numOfSamples; i++)
  {
    sum += diffBetweenSensors[i];
  }

  if (counter > numOfSamples * 2)
  {
    counter = numOfSamples;
  }
  return sum / numOfSamples;
}

String readKeypadForAttempt(int maxLength)
{
  return readKeypadForString(maxLength);
}

String readKeypadForString(int maxLength)
```

```
{
  String string;
  while (string.length() < maxLength)
  {
    char customKey = customKeypad.getKey();
    if (customKey)
    {
      string += customKey;
    }
  }
  return string;
}

void playCorrectPassword()
{
  char notes[] = {'c'};
  int durations[] = {duration};
  soundMelody(buzzerPin, notes, 1, durations, 1, 8);
  delay(duration);
  notes[0] = {'g'};
  soundMelody(buzzerPin, notes, 1, durations, 1, 8);
  delay(duration);
  notes[0] = {'f'};
  soundMelody(buzzerPin, notes, 1, durations, 1, 8);
}

void turnLightsOff()
{
  turnRGLed(RG_PINS, OFF_RG);
}

void turnRedLights()
{
  turnRGLed(RG_PINS, RED_RG);
}

void turnGreenLights()
{
  turnRGLed(RG_PINS, GREEN_RG);
}

void playIncorrectPassword()
{
  char notes[] = {'C'};
  int durations[] = {duration};
  soundMelody(buzzerPin, notes, 1, durations, 1, 8);
  delay(duration);
  notes[0] = {'c'};
  soundMelody(buzzerPin, notes, 1, durations, 1, 8);
  delay(duration);
  notes[0] = {'C'};
  soundMelody(buzzerPin, notes, 1, durations, 1, 8);
```

```
}

void turnRGLed(int rgPins[],int colors[])
{
  for(int i = 0; i < 2; i++)
  {
    digitalWrite(rgPins[i], colors[i]);
  }
}

void soundMelody(int buzzer, char melodyNotes[], int numNotes, int durations[], int
numDurations, int numSounds)
{
  for(int i = 0; i < numSounds; i++)
  {
    tone(buzzer, noteToFreq(melodyNotes[i % numNotes]), durations[i %
numDurations]);
  }
}

int noteToFreq(char note)
{
  int i;

  // these are the notes
  char notesList[numNotes] = {
    'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C'};

  // frequency[i] matches names[i]
  int frequencies[NUM_NOTES] = {
    262, 294, 330, 349, 392, 440, 494, 523};

  // Now we'll search through the letters in the array, and if
  // we find it, we'll return the frequency for that note.
  for (i = 0; i < numNotes; i++)  // Step through the notes
  {
    if (notesList[i] == note)         // Is this the one?
    {
      return (frequencies[i]);    // Yes! Return the frequency and exit function.
    }
  }
  return 0;  // We looked through everything and didn't find it,
  // but we still need to return a value, so return 0.
}
```

The code acts as follows:

1) Keeps making sure doors are closed and listens for cars (obstacles) to put themselves between the laser and the target photoresistor.
2) If there is an obstacle, then listen for input over the keypad
3) If the key is correct, play some sound and the green light, then open the door.

4) Wait 5 seconds and see if the obstacle is still there. Keep repeating until the obstacle is out of the way
5) Wait 5 more seconds and then close the door
6) Keep repeating from step 1

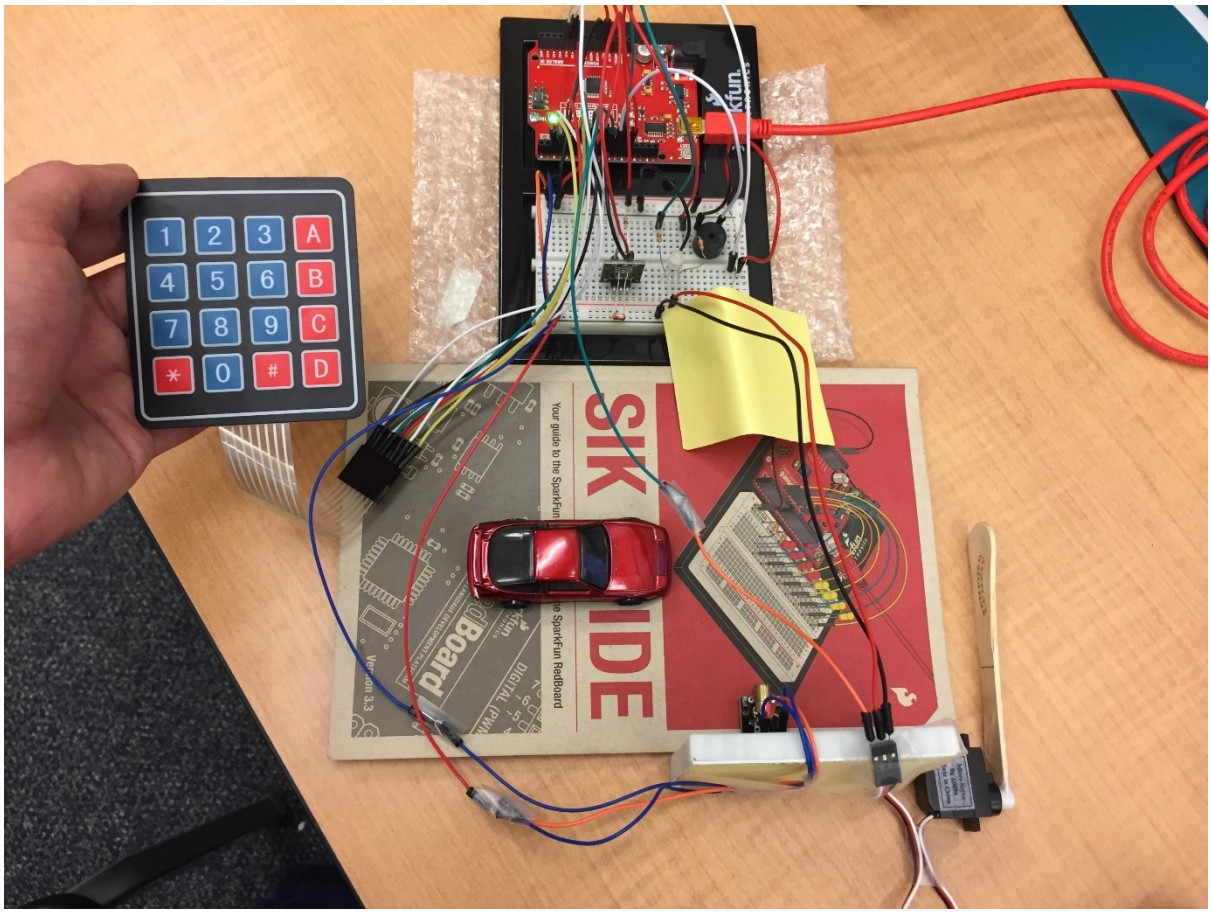

Figure 11. Photo of the circuit

Resources

https://www.sparkfun,com/sikcode

https://www.sparkfun,com/sikguide

http://fritzing.org/home/

https://www.elegoo.com/product/elegoo-37-in-1-sensor-module-kit/