The Skittle Sorter

By: Maheen Shoaib
Student #: 850420
Mr. Di Iorio
TEJ 4M0

Table of Contents

Situation

This skittle sorting machine is truly unique as it is specifically designed to assist individuals who are colorblind. The use of an RGB sensor and servo motors allows for accurate detection and sorting of skittles by color, making the task of sorting skittles much easier for colorblind individuals. Additionally, the code includes a starting sequence function and a function for controlling the RGB LED, which can be used to indicate the color of the skittle being sorted. This feature provides a visual cue for colorblind individuals and improves the overall usability of the device. The code is also open-source, which means that it can be modified and customized to suit the specific needs of the user. This level of flexibility and adaptability makes this skittle sorting machine even more unique. Furthermore, the device is built with a user-friendly approach and can be used easily by anyone. It's a great example of how technology can be used to make life easier for people with special needs. Furthermore, this skittle sorter also features 3D printed parts which makes it easy to assemble and disassemble. The 3D printed parts also provide a cost-effective solution for building a skittle sorter.

Problems / Possibilities: General

Performance Criteria:

- ➔ Have the ability to hold multiple skittles.
- ➔ Be compact.

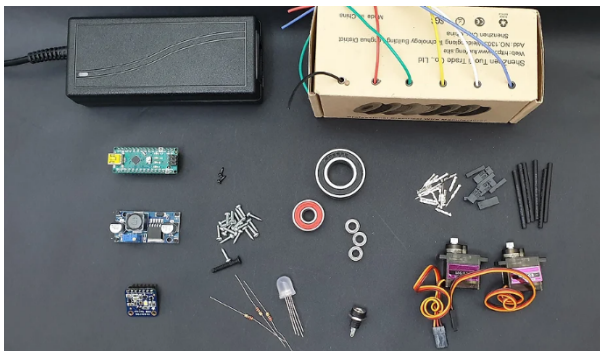
Specifications:

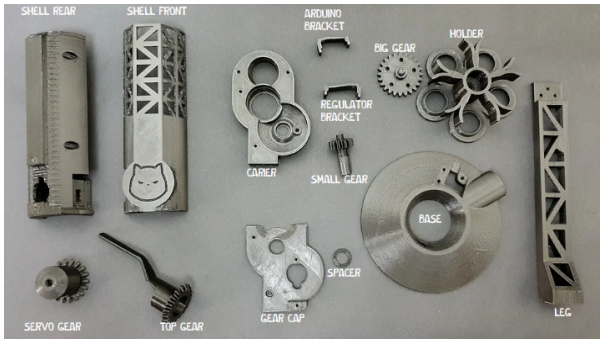
- ➔ Skittles have 5 colors, so the reservoir must use a motor to move the skittle to the needed section.
- ➔ All 3-d printed materials should be screwed together for the machine to be sturdy.

Limitations:

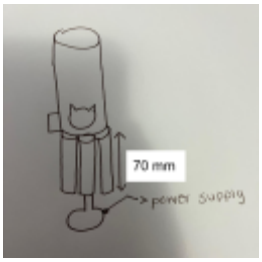
- ➔ One of the main limitations of this project was time. I had a limited amount of time to complete the project, which meant that I had to work efficiently and make quick decisions to stay on schedule. This limitation made it challenging to thoroughly test the circuit, and some components were not properly tested before use.
- ➔ Another limitation of this project was resources/materials. I had a limited budget for the project, which meant that I had to be creative with the materials I used. I had to use 3D printed parts for some of the circuit's components, which turned out to be a weakness of the design as they broke easily.
- ➔ Limited money.
- ➔ Limited tools.
- ➔ Limited time.

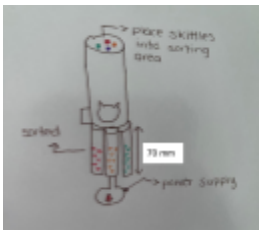
Problems / Possibilities: Parts List

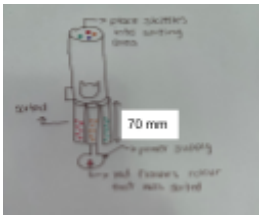
Part(s)	*Picture*
Tools	
Test Tubes	
Skittles	
Hardware: ➔ LM2596 buck converter ➔ arduino nano ➔ 2X MG90S servos ➔ TCS 34725 RGB sensor ➔ 3X ball bearing (5 x 10 x 4 mm) ➔ ball bearing (8 x 22 x 7mm) ➔ ball bearing (17x35x10mm) ➔ countersunk screw	

➔ self tapping screws ➔ RGB led (common Cathode(-)) ➔ 3X 470 ohm resistors ➔ wires ➔ jack plug ➔ dupont connectors ➔ 9V or 12v power supply (1 amp minimum)	
3D Printed Parts	

Investigation: Operating Instructions





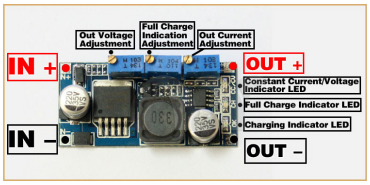
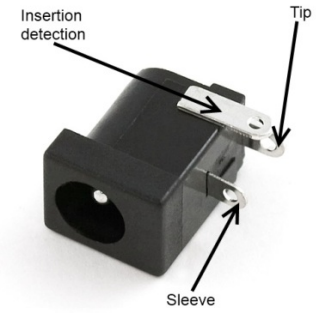


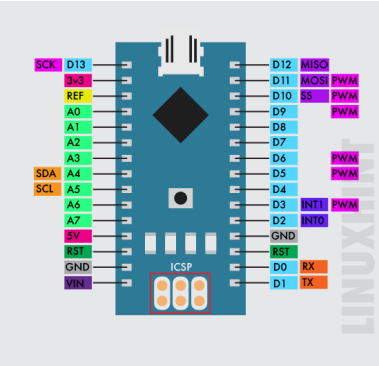
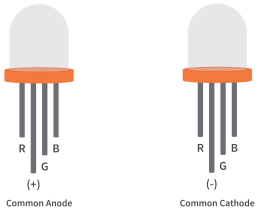
1. Connect the device to a 9V power supply by plugging it into the jack plug. Make sure the power supply is providing enough power and stable voltage.
2. Place skittles into the top of the machine. The skittles should be placed in the designated area for sorting.
3. Wait for the skittles to fall into different test tubes by color while the RGB LED indicates the color of the skittle being sorted. The skittles will fall into the designated test tubes according to their color. The RGB LED will light up with the corresponding color of the skittle being sorted.
4. If there are any issues with the sorting, you can adjust the limit values for each color in the code and re-upload it to the device. The limit values are stored in the arrays, you can adjust the minimum and maximum values for each color, and re-upload the code to the device.

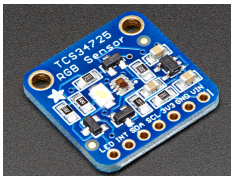
```
uint16_t black_limit_values[] = {0, 200, 0, 200, 0, 200};
uint16_t yellow_limit_values[] = {850, 2200, 800, 2200, 400, 1100};
uint16_t green_limit_values[] = {350, 800, 650, 1400, 250, 700};
uint16_t orange_limit_values[] = {650, 1800, 300, 850, 210, 600};
uint16_t red_limit_values[] = {400, 950, 150, 450, 150, 400};
```

5. To clean the device, you can remove the 3D printed parts and clean them separately. Clean the parts with a mild detergent and a damp cloth. Make sure the parts are dry before reassembling the device.
6. Always make sure that the servos are in the correct positions before turning off the device. The servos should be in the "home" position, which is usually at 0 degrees.
7. Repeat the process as necessary.

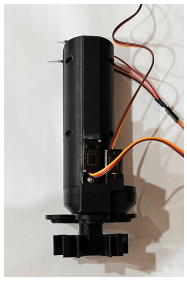
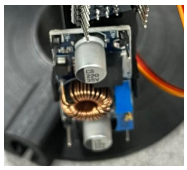
Investigation: Parts with Pin Layout

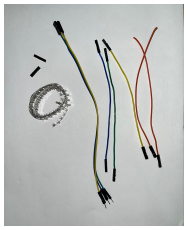
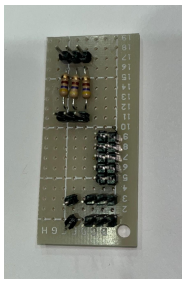
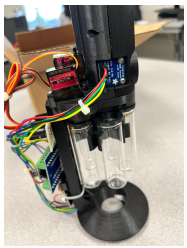
Name	Pin Layout	Findings
Buck Convertor		A buck converter, also known as a step-down converter, is a type of DC-to-DC power converter that reduces the voltage level from a higher value to a lower value while maintaining a constant current. It works by controlling the amount of time that a switch is closed, allowing a small amount of energy to be transferred from the input voltage to the output voltage during each switching cycle. The basic components of a buck converter include an input capacitor, an inductor, a switch, a diode, and an output capacitor. The voltage conversion ratio is determined by the duty cycle of the switch, which is the ratio of the on-time to the total switching period. It is efficient, cost-effective, and widely used in power supplies and battery chargers among other electronic devices.
Jack Plug		To solder a jack plug, strip the insulation from the wires and connect them to the corresponding terminals on the jack plug: tip (positive), ring (negative), and sleeve (ground).

Arduino Nano		Arduino Nano is a small form factor microcontroller board similar to the Arduino Uno, with 14 digital I/O pins, 8 analog inputs, a 16 MHz quartz crystal and USB connection, it is commonly used in hobby projects, education, and small-scale industrial applications.
RGB Led		RGB LEDs typically have a cathode (-) and an anode (+) terminal. The cathode, also known as the common terminal, is the negative terminal of the LED and is connected to the ground. The anode is the positive terminal of the LED and is connected to the power source through a current-limiting resistor. The cathode pin is connected to the microcontroller and the anode pin is connected to the power supply. The cathode is used to control the intensity of the LED, by adjusting the current flowing through it.To code an RGB LED, you can use a color picker tool like the one provided on w3schools. It allows you to select a color by adjusting the values of the red, green, and blue elements, which can then be used in the analogWrite() function to set the intensity of each color element connected to a separate digital pin on the Arduino board.

RGB Sensor		The TCS34725 is an RGB color sensor that has four pins. The pin layout for TCS34725 is as follows: 1. VDD: The power pin which supplies voltage to the sensor typically 2.7V to 3.3V 2. GND: Ground pin which connects the sensor to the ground of the circuit. 3. SDA: Serial data pin, This pin is used to transmit the sensor's output data. 4. SCL: Serial clock pin, This pin is used to synchronize the data transmission between the sensor and the microcontroller.

Construction: Chart

Step	Operation	Tips/Details	Safety	Tools/Materials	Pictures (if applicable)
1	Research	Used websites and YouTube videos to research the Arduino Nano, RGB Led, Buck Convertor and Jack Plug.	N/A	- Computer - Pencil - Paper	
2	Build Base	Used a screwdriver, screws, and parts to build the base.	N/A	- Screwdriver - Screws	
3	Build Main Component	Used the parts, gears and 2 servo motors to build the main component of the machine.	N/A	- Screwdriver - Screws - 2 X Servo Motors	
4	Solder all needed components	Used the soldering tool to make electrical connections on all components via headers.	- Hold pieces with helping hands to ensure no burns - Keep door open for	- Soldering tool - Helping hands - Tin - Flux - Buck Convertor - RGB Sensor - Headers	

			cross ventilation - No loose clothing		
5	Make Female to Female wires	Used a crimping tool to crimp pins onto wires.	- No loose clothing	- Crimping tool - Wires - Pins	 
6	Make the PCB Board in substitute of the bread board (looks better)	Make connections on the PCB board via soldering and male pins.	- Hold pieces with helping hands to ensure no burns - Keep door open for cross ventilation - No loose clothing	- Soldering tool - Helping hands - Tin - Flux	
7	Final Assembly	Use the female to female wires to connect all parts together with reference to the Fritzing Diagram. Also, add on the test tubes.	- Ensure you do not poke yourself with the pins	- Soldered Material - Female to Female wires	

8	Research Code	Making the needed adjustments to the open-source code, make the changes needed.		- Computer - Arduino IDE	
9	Upload Code	Using the cable, connect the Arduino Nano to the computer and upload code.	N/A	- Arduino IDE - Machine - Arduino Nano Cable	
10	Test the machine	After the code is uploaded, test the circuit and program, and make any needed changes.			

Construction: Functioning Code

This code is for a skittle sorting machine. It uses an RGB sensor and servo motors to detect and sort skittles by color.

The first few lines include necessary libraries for the RGB sensor, servo motors, and I2C communication.

There are several variables defined related to the RGB sensor, such as "colour" which keeps track of the current skittle color, and "black_count" which keeps track of empty/unknown colors. There are also several arrays that store the color limit values for each skittle color.

There are also variables defined related to the RGB LED, such as the pin numbers and current color values. There are also arrays that store the RGB values for each skittle color.

There are also variables defined related to the feeder servos, such as "release_delay" and "colour_angles" which determine the timing and angles for releasing skittles.

In the setup function, the serial communication, LED, and servo pins are initialized. The RGB sensor is also initialized and checked for proper connection. If the sensor is not found, the code enters a loop to blink the LED in red.

The code also includes a starting sequence function, which is called during setup, and a set_led function which is used to control the RGB LED.

```
/*
*Skittle Sorter
*This sketch uses an Arduino board and the circuit shown in the diagram above
*to sort skittles by color. The RGB sensor is used to detect the color of each skittle,
*and the servos are used to sort the skittles into designated test tubes.
*Author: Maheen Shoaib
* Date: January 25, 2023
*/

#include <Wire.h> // library for communication with I2C devices
#include "Adafruit_TCS34725.h" // library for the RGB sensor
#include<Servo.h> // library for controlling servo motors

/* Based on example code for the Adafruit TCS34725 breakout library */

//======================Variables related to RGB sensor=========================

Adafruit_TCS34725 tcs = Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_24MS, TCS34725_GAIN_1X);  // setup RGB sensor

int colour = 0;//0-black, 1-yelow, 2-green, 3-orange, 4-red, 5-purple
int previous_colour = colour;
int black_count = 0;  // count of empty/unknown color

int start_measurment_angle = 88; //servo position to align skittle in front of sensor
int measurment_sweep = 2; //number of measures taken . used to average error
int measurment_angle_increment = 4; //angle increment between measurements

//limit values[] = {min_red_values, max_red_values, min_green_values, max_green_values, min_blue_values, max_blue_values}
uint16_t black_limit_values[] = {0, 200, 0, 200, 0, 200};// color limit values for blank (no skittle)
uint16_t yellow_limit_values[] = {850, 2200, 800, 2200, 400, 1100};// color limit values for yellow
uint16_t green_limit_values[] = {350, 800, 650, 1400, 250, 700};// color limit values for green
uint16_t orange_limit_values[] = {650, 1800, 300, 850, 210, 600};// color limit values for orange
uint16_t red_limit_values[] = {400, 950, 150, 450, 150, 400};// color limit values for red
uint16_t purple_limit_values[] = {150, 400, 150, 450, 150, 500};// color limit values for purple

//======================Variables related to RGB led=========================

// pin for the red element of the RGB LED
byte R_pin = 5;
// pin for the green element of the RGB LED
byte G_pin = 6;
// pin for the blue element of the RGB LED
byte B_pin = 11;
// variable to store the current intensity of the red element of the RGB LED
int current_red_value = 255;
// variable to store the current intensity of the green element of the RGB LED
int current_green_value = 255;
```

```
// variable to store the current intensity of the blue element of the RGB LED
int current_blue_value = 255;

// array to store the intensity values for the orange color
int orange[] = {255, 30, 0};
// array to store the intensity values for the red color
int red[] = {255, 0, 0};
// array to store the intensity values for the green color
int green[] = {0, 204, 0};
// array to store the intensity values for the yellow color
int yellow[] = {255, 255, 0};
// array to store the intensity values for the purple color
int purple[] = {102, 0, 204};


//====================== Variables related to feeder servos ========================

// delay before releasing the skittle into the test tube
int release_delay = 220;
// angles of the test tubes
int colour_angles[] = {20, 59, 95, 135, 173};
// constant value used to adjust for backlash when moving the feeder servo in the backward direction
const int backward_anti_backlash = 2;
// constant value used to adjust for backlash when moving the feeder servo in the forward direction
const int forward_anti_backlash = 2;
Servo feeder_servo; //declare feeder servo
Servo holder_servo; //declare tubes holder servo

//================================== SETUP ========================================

void setup(void) {

 Serial.begin(9600); // Start serial communication at 9600 baud rate

 pinMode(R_pin, OUTPUT);   // Set the pin mode for R_pin as OUTPUT
 pinMode(G_pin, OUTPUT);   // Set the pin mode for G_pin as OUTPUT
 pinMode(B_pin, OUTPUT);   // Set the pin mode for B_pin as OUTPUT

 analogWrite(R_pin, 255);  // Turn off the red element of the RGB LED by setting its value to 255
 analogWrite(G_pin, 255);  // Turn off the green element of the RGB LED by setting its value to 255
 analogWrite(B_pin, 255);  // Turn off the blue element of the RGB LED by setting its value to 255

 feeder_servo.attach(3);  // Attach the feeder servo to pin 3
 holder_servo.attach(2);   // Attach the holder servo to pin 2
 feeder_servo.write(0);    // Initial position of the feeder servo is set to 0
 holder_servo.write(colour_angles[0]); // Initial position of the holder servo is set to the first angel in the colour_angles array
```

```
 int white[] = {0, 0, 0}; // Local white values variable
 int black[] = {255, 255, 255}; // Local black values variable
 set_led(1000, white);   // Turn the LED on (white) for 1 second

 if (tcs.begin()) {    // Check if the TCS34725 RGB sensor is found
  Serial.println("Found sensor");
  starting_sequence();
 } else {
  Serial.println("No TCS34725 found ... check your connections");
  while (1)  // Fast red blinking if sensor not found
  {
   set_led(300, red);
   set_led(300, black);
  }
 }
}

//================================== LOOP =======================================

void loop(void) {

 if (black_count < 10) // Check if the black_count is less than 10
 {
  feeder_servo.write(0); // Set the servo position to the top for skittle collection
  delay(450);  // Delay to let the feeder servo enough time to get in position
  previous_colour = colour;
  get_colour(); // Read skittle colour

  if (colour == 0) {
   black_count ++; //If no colour found, increment black count
   shake();// Gives a bit more shake to the reservoir even if no skittle is there
  }
  else if (colour == 6) {
   black_count ++; // If colour unknown, increment black count - no skittle release
  }
  else { // If colour found
   move_holder(); // Move tube holder
   release_skittle(); // Release skittle
   black_count = 0;
  }
 }

 else { // If black_count is greater than or equal to 10
  end_loop(1000); // End the loop and wait for 1 second
 }
}
```

```
//================================== GET COLOUR ======================================

void get_colour() {

 uint16_t r, g, b, c;//, colorTemp, lux;
 uint16_t total_r = 0;
 uint16_t total_g = 0;
 uint16_t total_b = 0;

 feeder_servo.write(start_measurment_angle); //move to start measurement angle
 delay(200);

 Serial.println("----------------");
 for (int i = 0; i <= measurment_sweep; i++) //loop for each measure
 {
  tcs.getRawData(&r, &g, &b, &c);    //get color data

  feeder_servo.write(start_measurment_angle + i * measurment_angle_increment); //increment servo angle for next measure
  total_r += r; //add red value to total red value
  total_g += g; //add green value to total green value
  total_b += b; //add blue value to total blue value
  delay(15);
 }


 total_r /= measurment_sweep;  //average values across all measurements
 total_g /= measurment_sweep;
 total_b /= measurment_sweep;

 Serial.print(total_r); Serial.print(" ");
 Serial.print(total_g); Serial.print(" ");
 Serial.print(total_b); Serial.print(" ");
 Serial.println(" ");

 //compare values to determine the color .
 if ((total_r < black_limit_values[1]) &&    //check for black
    (total_g < black_limit_values[3]) &&
    (total_b < black_limit_values[5]))
 { Serial.println("black");
  colour = 0;
 }
 else if ((total_r >= yellow_limit_values[0]) &&   //check for yellow
        (total_r < yellow_limit_values[1]) &&
        (total_g >= yellow_limit_values[2]) &&
        (total_g < yellow_limit_values[3]) &&
        (total_b >= yellow_limit_values[4]) &&
        (total_b < yellow_limit_values[5]))
```

```
 { Serial.println("yellow");
   colour = 1;
 }
 else if ((total_r >= green_limit_values[0]) &&   //check for green
        (total_r < green_limit_values[1]) &&
        (total_g >= green_limit_values[2]) &&
        (total_g < green_limit_values[3]) &&
        (total_b >= green_limit_values[4]) &&
        (total_b < green_limit_values[5]))
 { Serial.println("green");
   colour = 2;
 }
 else if ((total_r >= orange_limit_values[0]) &&   //check for orange
        (total_r < orange_limit_values[1]) &&
        (total_g >= orange_limit_values[2]) &&
        (total_g < orange_limit_values[3]) &&
        (total_b >= orange_limit_values[4]) &&
        (total_b < orange_limit_values[5]))
 { Serial.println("orange");
   colour = 3;
 }
 else if ((total_r >= red_limit_values[0]) &&   //check for red
        (total_r < red_limit_values[1]) &&
        (total_g >= red_limit_values[2]) &&
        (total_g < red_limit_values[3]) &&
        (total_b >= red_limit_values[4]) &&
        (total_b < red_limit_values[5]))
 { Serial.println("red");
   colour = 4;
 }
 else if ((total_r >= purple_limit_values[0]) &&   //check for purple
        (total_r < purple_limit_values[1]) &&
        (total_g >= purple_limit_values[2]) &&
        (total_g < purple_limit_values[3]) &&
        (total_b >= purple_limit_values[4]) &&
        (total_b < purple_limit_values[5]))
 { Serial.println("purple");
   colour = 5;
 }
 else {
   Serial.println("unknown");    //if color not detected, set colour to 0 to force new check
   colour = 6;
 }

}

//============================== MOVE HOLDER ==============================
```

```
// adjust delay depending how much the holder needs to move
// include an backlash compensation

void move_holder() {

 int new_holder_position = colour_angles[colour - 1]; //gets new position of tube
  int holder_delay = (abs(previous_colour - colour) * 70);// - release_delay; //leaves 70ms for the holder servo to turn 36 degrees

  if (holder_delay < 0) {  //keep delay superior to 0
    holder_delay = 0;
  }
  if (previous_colour > colour)
  {
    //forces the holder servo to go back further than normal position to compensation for gearing backlash
    int anti_backlash_angle = new_holder_position - backward_anti_backlash;
    holder_servo.write(anti_backlash_angle);
    //delay(holder_delay);
  }
  else
  {
    int anti_backlash_angle = new_holder_position + forward_anti_backlash;
    //holder_servo.write(anti_backlash_angle);
    holder_servo.write(new_holder_position); // move holder in position
    //delay(holder_delay);
  }

  if (colour == 1) {
    set_led(holder_delay , yellow);
  }
  else if (colour == 2) {
    set_led(holder_delay , green);
  }
  else if (colour == 3) {
    set_led(holder_delay , orange);
  }
  else if (colour == 4) {
    set_led(holder_delay , red);
  }
  else if (colour == 5) {
    set_led(holder_delay , purple);
  }
  else {}

}

//============================  RELEASE SKITTLE  ============================
```

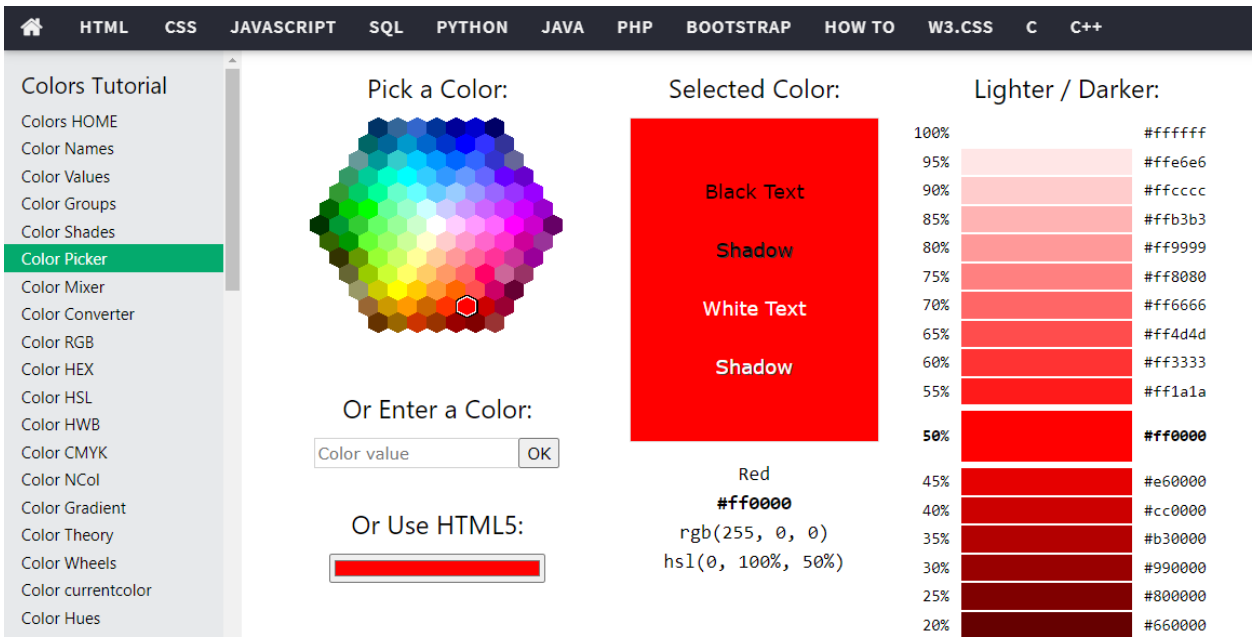
```
void release_skittle() {

  feeder_servo.write(180); // set servo position to the bottom for skittle release
  delay(release_delay);

}

//================================  SHAKE  ================================

void shake() {

  int shake_delay = 80;
  int shake_amount = 5;
  int shake_min_value = 90;
  int shake_max_value = 180;

  feeder_servo.write(180); // set servo position to the bottom for skittle release
  delay(release_delay);
  feeder_servo.write(120); // set servo position to the bottom for skittle release
  delay(80);
  for (int i = 0; i <= shake_amount; i++) //loop for each measure
  {
    feeder_servo.write(shake_min_value); // set servo position to the bottom for skittle release
    delay(shake_delay);
    feeder_servo.write(shake_max_value); // set servo position to the bottom for skittle release
    delay(shake_delay);
  }
}

//============================ STARTING SEQUENCE ============================

void starting_sequence() {

  colour = 1;
  move_holder();
  set_led(300, yellow);
  delay (400);

  colour = 2;
  move_holder();
  set_led(300, green);
  delay (400);

  colour = 3;
  move_holder();
  set_led(300, orange);
```

```
  delay (400);

  colour = 4;
  move_holder();
  set_led(300, red);
  delay (400);

  colour = 5;
  move_holder();
  set_led(300, purple);
  delay (400);

  previous_colour = colour;
  colour = 3;
  move_holder();

  end_loop(200);
}

//================================ END LOOP ================================

void end_loop(int duration) {

  set_led(duration , orange);
  set_led(duration , red);
  set_led(duration , green);
  set_led(duration , yellow);
  set_led(duration , purple);
}

//============================   SET RGB LED COLOUR   =======================

//   This act as a delay() but allows the LED to change color while waiting .
void set_led(int duration, int color[3]) {

  int start_time = millis();     // start time value
  int current_time = start_time; // current time value
  int current_duration = 0;      // total duration

  while (current_duration < duration)
  {
    analogWrite(R_pin, map(current_duration, 0, duration, current_red_value, color[0]));  //ramp up red value for each loop
    analogWrite(G_pin, map(current_duration, 0, duration, current_green_value, color[1])); //ramp up green value for each loop
    analogWrite(B_pin, map(current_duration, 0, duration, current_blue_value, color[2])); //ramp up blue value for each loop

    current_time = millis();   //update current time
    current_duration = current_time - start_time;  //calculate total duration
```

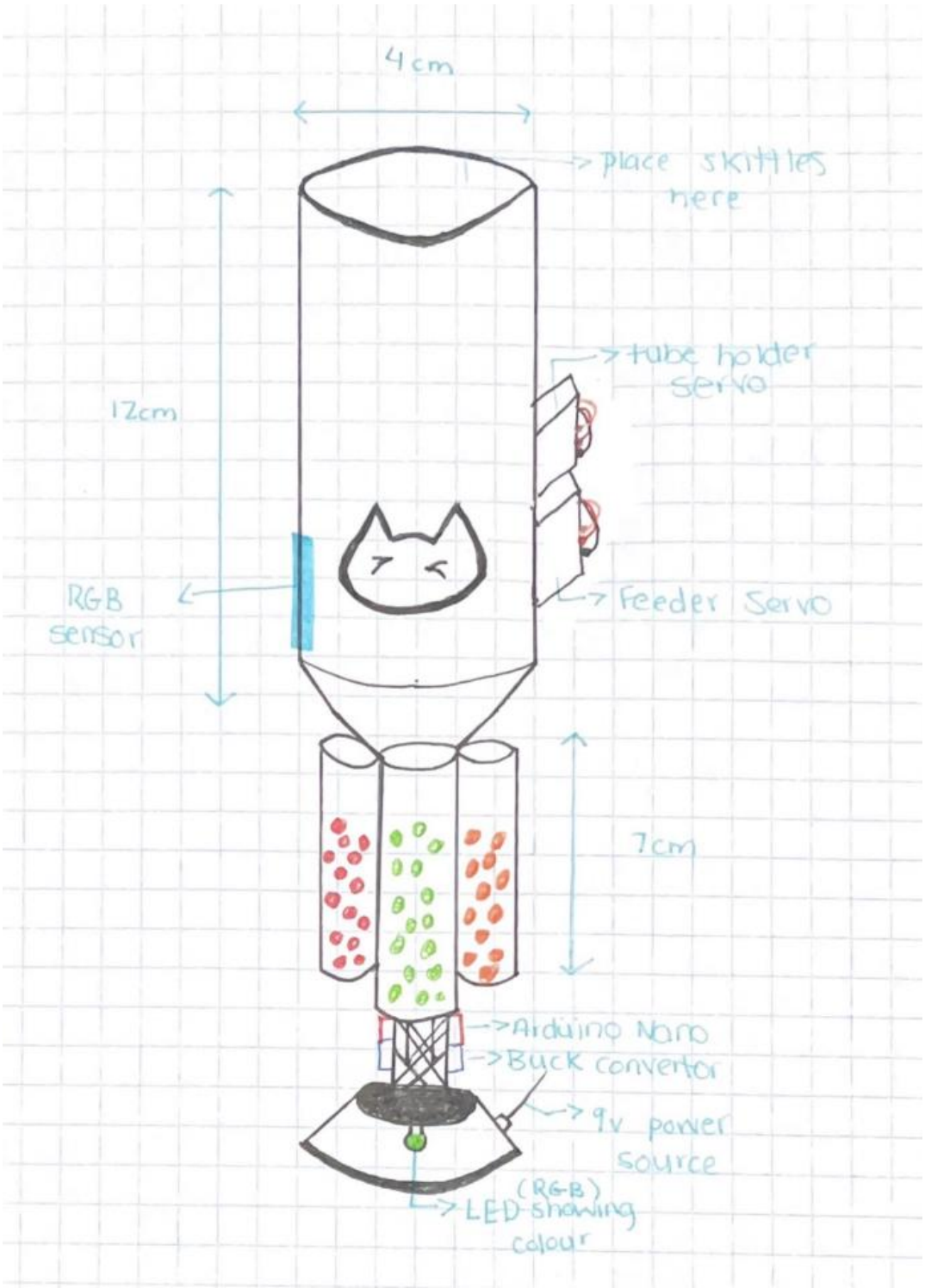
```
 }

 current_red_value = color[0];   // set new red current value
 current_green_value = color[1]; // set new green current value
 current_blue_value = color[2];  // set new blue current value

}
```

Construction: Modification of Code Explanation

This program was referenced from hackday.com. First, to be able to understand the code, I commented on every line. Then, I was able to modify some of the code to my liking. My modification was for the RGB Led, the rotation of the tubes, and the colour range for the skittles. Although the rotation of the tube was done by testing, the RGB and colour range was difficult, and thus, required research. With the help of Mr. Di Iorio, I found https://www.w3schools.com/colors/colors_picker.asp, and used this site to get the RGB values for the specific skittles. Then, I tweaked those with print statements in the Arduino IDE.

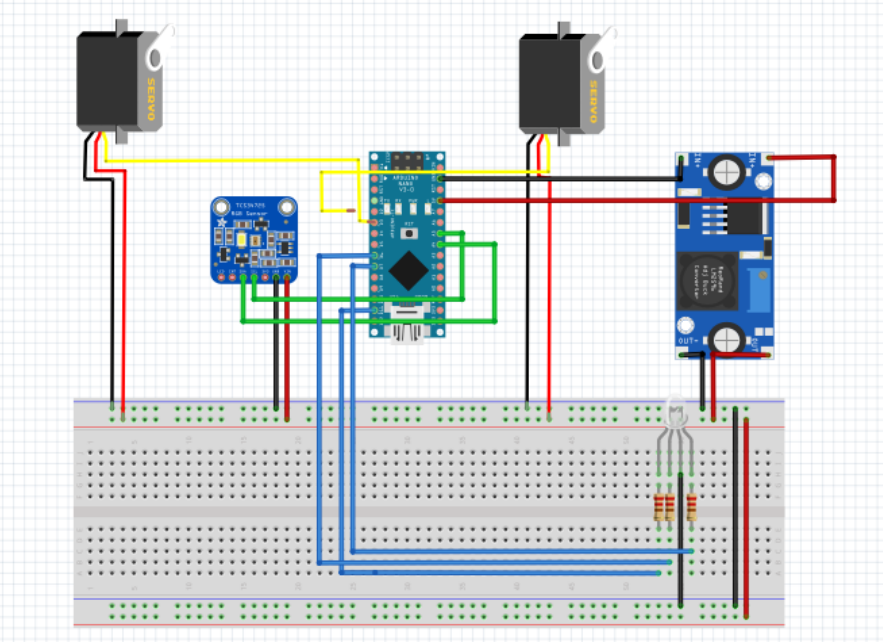


Construction: Final Sketch



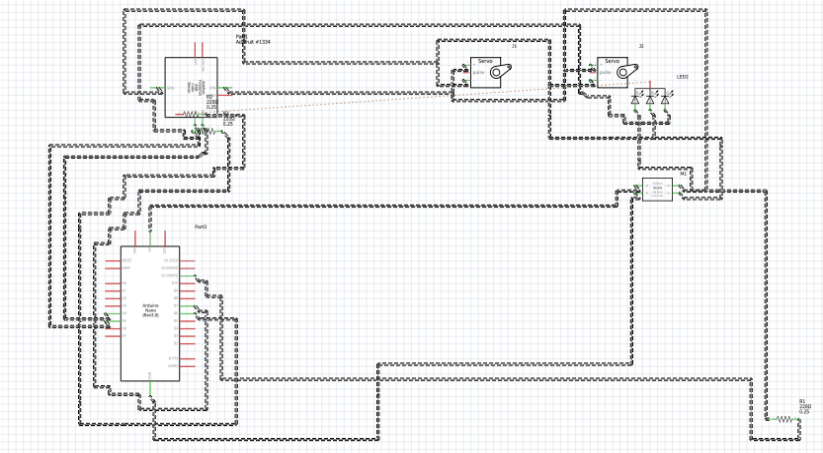
Construction: Fritzing Diagram

Breadboard

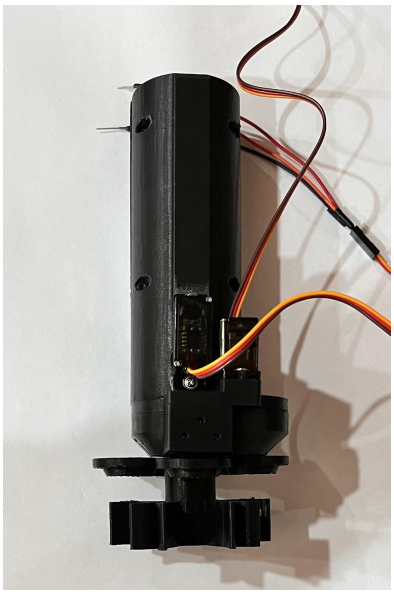


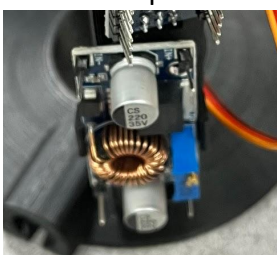
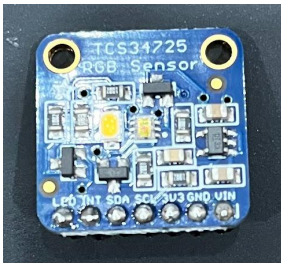
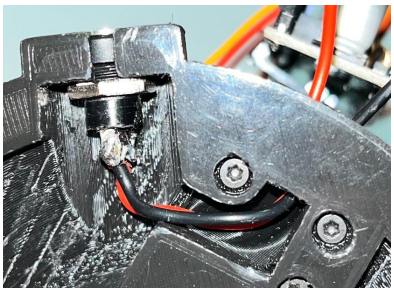
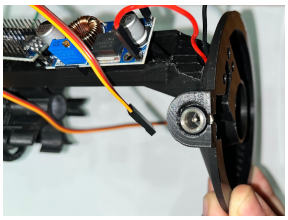
- ➔ Red: Power
- ➔ Black: Ground
- ➔ Blue: RGB Led
- ➔ Motor: Yellow
- ➔ Sensor: Green
- ➔ Ensure Buck Convertor is at 5V

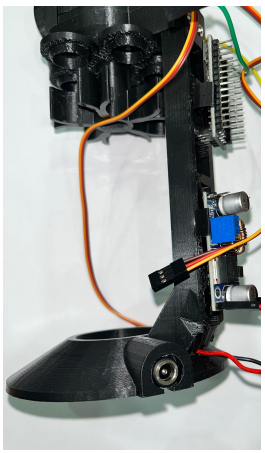
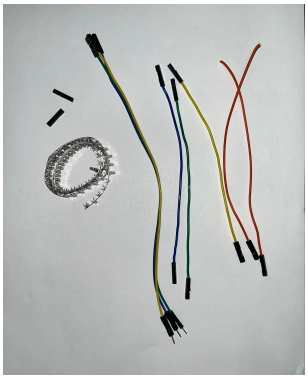
Schematic

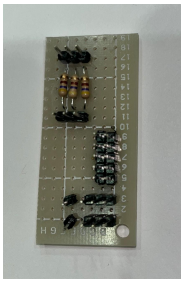
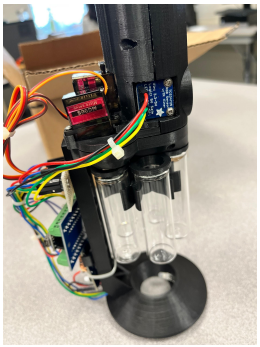


Construction: Activity Log

Day	Activity	Time
Jan 9, 2023	Researching the build and ensuring all components are found	75 minutes
Jan 10, 2023	Make Sketches	75 minutes
Jan 11, 2023	Building Main Components using the 3D printed parts, screws and attached motors.  	50 minutes
Jan 13, 2023	Absent	75 minutes

Jan 16, 2023	Researching Code	75 minutes
Jan 17, 2023	Solder Components    	
Jan 18, 2023	Put soldered parts on the machine via screws. 	75 minutes

	 	
Jan 19, 2023	Create cables  	75 minutes

Jan 20, 2023	Make wiring neat via PCB board (solder) 	75 minutes
Jan 23, 2023	Testing using the sample code and tweaking as needed. (test the machine using a 9v power supply) 	75 minutes
Jan 24, 2023	Complete Fritzing Diagram and complete machine  Then, make modifications to code.	

Evaluation: Overall Experience

The project required me to design and build a circuit, and throughout the process, I encountered several issues that required fixing such as the jack plug had a manufacturing issue, the code would not upload from my laptop, the sensor kept detecting the color of the skittle logo, which gave an error, and the rotation of the inner sensor mechanism was shifting the wrong way.

The manufacturing issue with the jack plug taught me the importance of checking the components before using them in the circuit. I realized that if I had taken the time to properly inspect the jack plug, I could have identified the problem and resolved it before it affected the rest of the circuit. In future projects, I will make sure to thoroughly check all components before using them to avoid similar issues.

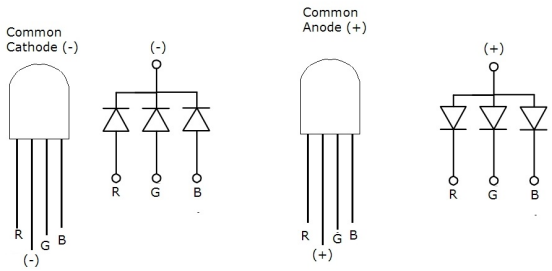
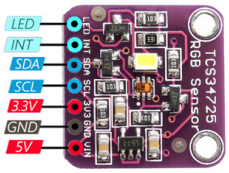
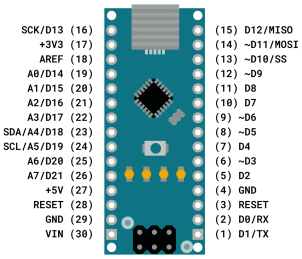
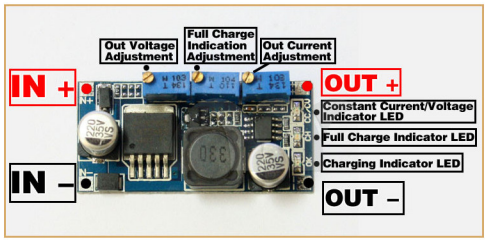
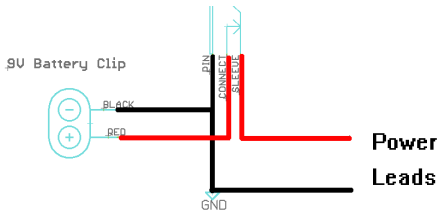
The problem with the code not uploading from my laptop showed me the importance of having a backup plan. I learned that it is always best to have a backup option in case something goes wrong, and in this case, using my sister's laptop was a great solution. This experience has taught me to always have an alternative plan in case of technical difficulties.

The sensor detecting the color of the skittle logo and giving an error, taught me about the importance of thoroughly testing the components before using them in the circuit. I realized that if I had taken the time to properly test the sensor, I could have identified the problem and resolved it before it affected the rest of the circuit. In future projects, I will make sure to thoroughly test all components before using them to avoid similar issues.

The rotation of the inner sensor mechanism shifting the wrong way taught me about the importance of paying attention to small details. I realized that if I had taken the time to properly align the sensor mechanism, I could have avoided the problem altogether. This experience has taught me to always pay attention to small details and to double-check everything before moving on to the next step.

Overall, the project was a good learning experience, and I was able to overcome the challenges faced during the testing phase. Two strengths of my design were its visual appeal and the fact that it worked as intended. Two weaknesses of my design were the use of 3D printed parts which broke easily and the sensor that kept sensing the logo color, which resulted in error. If I were to make it again, I would thoroughly test the sensor before including it in the circuit to avoid the error.

Appendix

Item	Pin Layout
RGB Led	
RGB Sensor	
Arduino Nano	
Buck Convertor	
Jack Plug	

Works Cited

"Arduino Nano." *Arduino Official Store*, https://store.arduino.cc/products/arduino-nano.

//This was used to learn about the Arduino Nano and its pin layout in comparison to an Arduino Uno

HTML Color Picker, https://www.w3schools.com/colors/colors_picker.asp.

//This was used while coding the Led and the rotation of the machine

Industries, Adafruit. "RGB Color Sensor with IR Filter and White LED - TCS34725." *Adafruit Industries Blog RSS*, https://www.adafruit.com/product/1334.

//This informed me about the RGB sensor and its pin layout

Jacques, Villar, et al. "How Do RGB Leds Work?" *Random Nerd Tutorials*, 31 Oct. 2019, https://randomnerdtutorials.com/electronics-basics-how-do-rgb-leds-work/#:~:text=an%20RGB%20LED%20is%20a%20combination%20of%20three%20LEDs%20in,you%20use%20a%20PWM%20signal.

//This was used in the code, and when soldering and wiring the Led.

"Learnabout Electronics." *Buck Converters*, https://learnabout-electronics.org/PSU/psu31.php.

//This was used to learn more about the importance of a buck convertor.

Panos, Kristina. “Sleek, Sophisticated Skittle Sorter.” *Hackaday*, 16 Dec. 2019, https://hackaday.com/2019/12/16/sleek-sophisticated-skittle-sorter/.

// This was the project I used as reference for this final project

“What Is the Difference between a Connector, Jack, Plug, and Port?” *Computer Hope*, 1 Feb. 2021, https://www.computerhope.com/issues/ch001379.htm.

.//This was used to learn about the jack plug and the method that I should use to solder.

Self - Assessment

Comment Requirements

LEVEL	4	3	2	1
Arduino Program Comments (Header and internal documentation) /5	Fully commented code with program header (Author, Date, and Description) organized well at the top. Code can be easily followed and understood by anyone. Every variable is commented, and all sections of code are thoroughly documented	Program contains all required comments and a header at the top but your documentation is somewhat useful in understanding the code.	Program contains only a simple header and comments separating routines. An experienced programmer could understand most of your code but with some difficulty.	An attempt is made to comment your code but many details are left out and do not help anyone understand or troubleshoot your code. Attention to detail is lacking in variable names, functions, etc.

	Level 4	Level 3	Level 2	Level 1
Communication	10 9 8	7	6	5
Title page Table of contents Layout Organization Pictures of progress Required Doc's Activity log Submitted digitally and in a well-organized folder. / 10	Exemplary layout and creative design throughout. No spelling / grammar errors. Easy to understand. Sections of the report are bolded/underlined and there is a clear division of topics / ideas.	Great layout and creative design Minor spelling / grammar errors. Sections of the report are mostly clear and there is a division of topics / ideas.	Good layout and design Some spelling / grammar errors that affect the overall flow of the report. Hard to understand at times. Sections of the report are organized but topics / ideas could be clearer.	Completed sections but design/ organization was not focused on Report is hard to understand and follow throughout. Missing 1-2 required report documents (deliverables) Missing hard copy OR digital copy. Only one is handed in.

Application	**Level 4**	**Level 3**	**Level 2**	**Level 1**
	15 14 13 12	11 10.5	10 9	8 7.5
Construction and Technical skills in design/build. Construction Steps /15	Excellent workmanship and the project build is of excellent quality. Excellent creativity. All steps are listed in full detail and could easily be used by anyone to recreate your project step by step. All safety steps are included.	Project build is of good quality. Good overview of all steps taken but could not be easily be used by anyone to recreate your exact project	project build meets most requirements but is somewhat incomplete and/or not fully functioning. Minor issues with your order of operations – some information is omitted but still makes sense	project build is poor quality, incomplete and/or not fully functioning Major issues with the construction steps – out of order, missing key details that are pertinent to the overall execution of your project
	10 9 8	7	6	5
Programming Requirements met Comments (Header and internal documentation) / 10	Circuit works as per program. No repetitive code. Very efficient programming (eg. use of functions). Additional feature is added. Excellent level of commenting.	Meets requirements discussed in class. No repetitive code. Good level of commenting. Code is efficient.	Some requirements met or exceeded. Some repetitive code. Minimal level of commenting. Code is somewhat efficient.	Few requirements met or exceeded. Code is repetitive. No commenting. Code is not efficiently written.

Thinking	Level 4	Level 3	Level 2	Level 1
	5	4	3	2.5
Wire / Cable Management	Wires are creatively concealed, and the inner components are easily accessed for troubleshooting / repairs.	Wires are concealed and the inner components can be removed if needed.	Wires are not fully concealed throughout your model and the inner components would be removed if needed. Minor issues with wiring: unnecessary long wires, colour code is not used at all times	Wires are messy and hard to follow. Model cannot be easily accessed directly. Minor issues with wiring: unnecessary long wires, colour code is not used at all times.
/5	Wiring follows all colour code conventions, minimal wire is used throughout and is neat throughout.	Minor issues with wiring: unnecessary long wires, colour code is not used.		
	5	4	3	2.5
Sketches Fritzing Diagrams /5	Excellent sketches (clearly labelled and proportional) show clear evidence of planning skills and provide at least one rough draft and one good copy that includes any changes made to the original design. Includes dimensions. Fritzing (Breadboard and Schematic) images are included, neat, components are easily identifiable, and accurate.	Neat sketches and Fritzing diagrams, correct placement of components in drawings but would be somewhat difficult to recreate entirely if given to a new person.	Minor components are incorrect or hard to locate and identify. It could be recreated only if an experienced person were to be given the task of re-building your final exam build.	Components are incorrect or hard to locate and identify. It would be hard for anyone to rebuild or create your project despite all relevant components being present. Lacks attention to detail.

Knowledge	Level 4	Level 3	Level 2	Level 1
	10 9 8	7	6	5
Evaluation Report /10	Good evaluation. Complete description of strengths and weaknesses. Insightful and thoughtful changes are outlined with a complete list of problems faced.	Meets all requirements discussed in class but does not elaborate fully on what could be changed or how it could be improved	Missing 1-2 requirements.	Attempt is made but the connections made lack considerable depth and a detailed explanation is not given for strengths and weaknesses.
	5	4	3	2.5
Self-Evaluation Marked rubric Justification /5	Completed with a justification for each section.	All marks are calculated but there are some minor issues with your justifications	Marks are mostly complete and reasonable. Justifications are lacking in depth but there is effort displayed.	Marks are not reasonable and/or rationale is not justified.
	10 9 8	7	6	5
Circuit Summary Diagram Operating Instructions /10	Circuit explanations, instructions, and block diagram pictorial show an excellent level of understanding. This can be given to anyone and have it understood clearly whether they have a background in this field or not. No information is omitted. Key details are	Circuit explanations, instructions, and block diagram pictorial show a good level of understanding. Information is easy to follow but some of the more technical	1-2 requirements in this area are omitted or vague.	More than 2 key features are omitted which make your project's features not fully understandable. A lack of understanding is evident through the limited circuit explanations.
	included like voltage requirements, pin layouts, full explanations, etc.	information (voltage requirements, I/O sequences) may not be explained in depth.		

Total: 70 /70

Rubric

Comment Requirements

LEVEL	4	3	2	1
Arduino Program Comments (Header and internal documentation) /5	Fully commented code with program header (Author, Date, and Description) organized well at the top. Code can be easily followed and understood by anyone. Every variable is commented, and all sections of code are thoroughly documented	Program contains all required comments and a header at the top but your documentation is somewhat useful in understanding the code.	Program contains only a simple header and comments separating routines. An experienced programmer could understand most of your code but with some difficulty.	An attempt is made to comment your code but many details are left out and do not help anyone understand or troubleshoot your code. Attention to detail is lacking in variable names, functions, etc.

	Level 4	Level 3	Level 2	Level 1
Communication	10 9 8	7	6	5
Title page Table of contents Layout Organization Pictures of progress Required Doc's Activity log Submitted digitally and in a well-organized folder. / 10	Exemplary layout and creative design throughout. No spelling / grammar errors. Easy to understand. Sections of the report are bolded/underlined and there is a clear division of topics / ideas.	Great layout and creative design Minor spelling / grammar errors. Sections of the report are mostly clear and there is a division of topics / ideas.	Good layout and design Some spelling / grammar errors that affect the overall flow of the report. Hard to understand at times. Sections of the report are organized but topics / ideas could be clearer.	Completed sections but design/ organization was not focused on Report is hard to understand and follow throughout. Missing 1-2 required report documents (deliverables) Missing hard copy OR digital copy. Only one is handed in.

Application	**Level 4**	**Level 3**	**Level 2**	**Level 1**
	15 14 13 12	11 10.5	10 9	8 7.5
Construction and Technical skills in design/build. Construction Steps /15	Excellent workmanship and the project build is of excellent quality. Excellent creativity. All steps are listed in full detail and could easily be used by anyone to recreate your project step by step. All safety steps are included.	Project build is of good quality. Good overview of all steps taken but could not be easily be used by anyone to recreate your exact project	project build meets most requirements but is somewhat incomplete and/or not fully functioning. Minor issues with your order of operations – some information is omitted but still makes sense	project build is poor quality, incomplete and/or not fully functioning Major issues with the construction steps – out of order, missing key details that are pertinent to the overall execution of your project
	10 9 8	7	6	5
Programming Requirements met Comments (Header and internal documentation) / 10	Circuit works as per program. No repetitive code. Very efficient programming (eg. use of functions). Additional feature is added. Excellent level of commenting.	Meets requirements discussed in class. No repetitive code. Good level of commenting. Code is efficient.	Some requirements met or exceeded. Some repetitive code. Minimal level of commenting. Code is somewhat efficient.	Few requirements met or exceeded. Code is repetitive. No commenting. Code is not efficiently written.

Thinking	**Level 4**	**Level 3**	**Level 2**	**Level 1**
	5	4	3	2.5
Wire / Cable Management	Wires are creatively concealed, and the inner components are easily accessed for troubleshooting / repairs.	Wires are concealed and the inner components can be removed if needed.	Wires are not fully concealed throughout your model and the inner components	Wires are messy and hard to follow. Model cannot be easily accessed directly.
/5	Wiring follows all colour code conventions, minimal wire is used throughout and is neat throughout.	Minor issues with wiring: unnecessary long wires, colour code is not used.	would be removed if needed. Minor issues with wiring: unnecessary long wires, colour code is not used at all times	Minor issues with wiring: unnecessary long wires, colour code is not used at all times.
	5	4	3	2.5
Sketches Fritzing Diagrams /5	Excellent sketches (clearly labelled and proportional) show clear evidence of planning skills and provide at least one rough draft and one good copy that includes any changes made to the original design. Includes dimensions. Fritzing (Breadboard and Schematic) images are included, neat, components are easily identifiable, and accurate.	Neat sketches and Fritzing diagrams, correct placement of components in drawings but would be somewhat difficult to recreate entirely if given to a new person.	Minor components are incorrect or hard to locate and identify. It could be recreated only if an experienced person were to be given the task of re-building your final exam build.	Components are incorrect or hard to locate and identify. It would be hard for anyone to rebuild or create your project despite all relevant components being present. Lacks attention to detail.

Knowledge	Level 4	Level 3	Level 2	Level 1
	10 9 8	7	6	5
Evaluation Report /10	Good evaluation. Complete description of strengths and weaknesses. Insightful and thoughtful changes are outlined with a complete list of problems faced.	Meets all requirements discussed in class but does not elaborate fully on what could be changed or how it could be improved	Missing 1-2 requirements.	Attempt is made but the connections made lack considerable depth and a detailed explanation is not given for strengths and weaknesses.
	5	4	3	2.5
Self-Evaluation Marked rubric Justification /5	Completed with a justification for each section.	All marks are calculated but there are some minor issues with your justifications	Marks are mostly complete and reasonable. Justifications are lacking in depth but there is effort displayed.	Marks are not reasonable and/or rationale is not justified.
	10 9 8	7	6	5
Circuit Summary Diagram Operating Instructions /10	Circuit explanations, instructions, and block diagram pictorial show an excellent level of understanding. This can be given to anyone and have it understood clearly whether they have a background in this field or not. No information is omitted. Key details are	Circuit explanations, instructions, and block diagram pictorial show a good level of understanding. Information is easy to follow but some of the more technical	1-2 requirements in this area are omitted or vague.	More than 2 key features are omitted which make your project's features not fully understandable. A lack of understanding is evident through the limited circuit explanations.
	included like voltage requirements, pin layouts, full explanations, etc.	information (voltage requirements, I/O sequences) may not be explained in depth.		

Total: /70

Comments Page