# Object Sorter - R.O.D.G.E.

Clarence Choi , Rajeev Thimmareddy , Kenny Nguyen , Joseph Tai , Joe Wan ,

Melissa Zhang

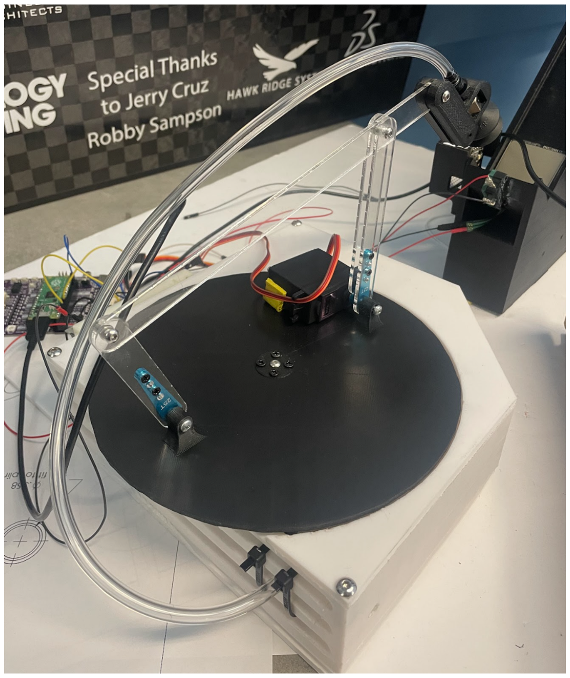
Charles W. Davidson College of Engineering

San Jose State University

Mechanical Engineering 106 Fundamentals of Mechanical Engineering

Dr. Mojtaba Sharifi & Dr. Lin Jiang

May 21, 2022



## Project Summary

We were required to design and build a working prototype encompassing an application process of mechatronics. We proposed a robotic 4-bar sorting arm and apparatus that would take distinguishable objects (painted cubes) from a predestined location. This predestined location acts as a rudimentary conveyor belt, which is similar to industrial standards. This arm collects the objects via suction and sorts each item respectively based on weight or color which is dependent on user input via a digital screen. The 4-bar arm mechanism has two degrees of freedom. Therefore, we required two motors. The first servo is required in order to rotate the entire assembly for sorting. The secondary servo toggles the double rocker mechanism in between two positions. The first position is considered “standby” mode while the forward-facing position is considered “engage” mode. The suction cup is actively picking up or dropping off the object when it is in the engage mode.

The first step we took was designing a four-bar mechanism and deciding on what device would pick up the objects. The following object depicts our graphical synthesis of the four-bar, which was then scaled to an appropriate size based on our constraints.

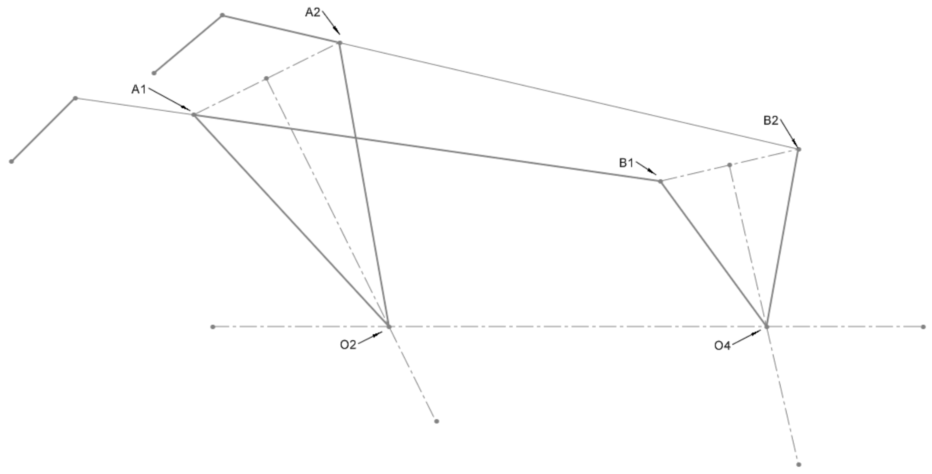


Figure 1. Graphical Synthesis of our Four Bar design

After deciding to use a suction device, cubes were selected as the most appropriate object for our robot to sort because they have flat surfaces that are easy to pick up using suction and are regular on all sides. After finalizing the dimensions for the four-bar, we designed a base and housing for the robot and its components and a ramp for the objects to be delivered to the robot for sorting. We then 3D printed these components, and laser cut the linkages of the four bar mechanism out of acrylic.

Finally, we constructed the system by putting all the components together and securing them on a posterboard at their proper distances. Our final product was a robot that was able to accurately sort by color, however, it was rather inconsistent when sorting by weight. All of our components, including the servos, color sensors, and the suction device worked correctly.

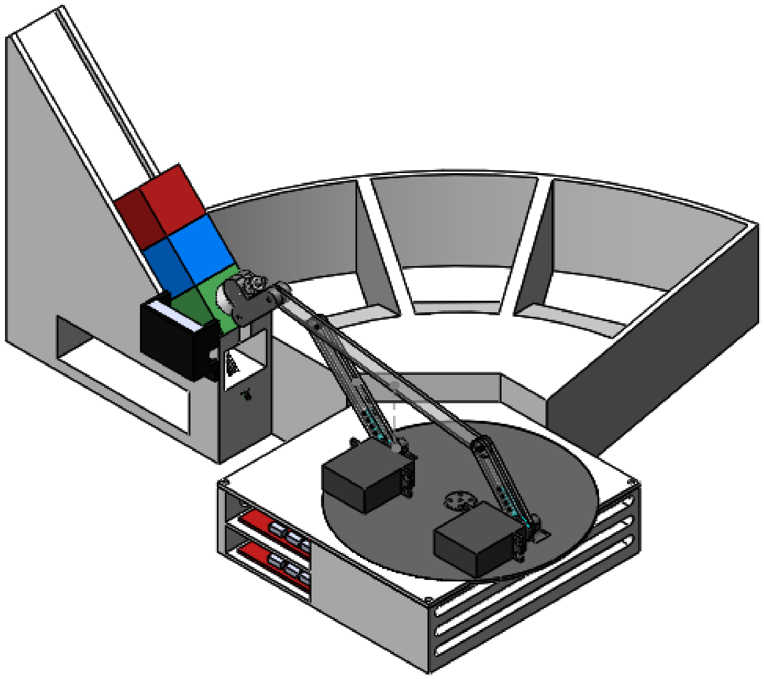


Figure 2. Unsorted ramp, sorted bins, four-arm mechanism, and base.

We learned that our prototype can be improved in a variety of ways. The pressure plate that we sourced was not precise enough for our application, hence why it was inconsistent when sorting by weight. Another consideration is that the suction sometimes needed variable time to release the cube depending on the surface of the cube, which was rougher or smoother depending on how it was printed. This resulted in the cube dropping too late or not at all in some unfortunate instances, which forced us to raise the wait time significantly for the arm to drop the cube. Both of these problems were exacerbated when we used our “unweighted” hollow cubes which were too light to be dropped on time and too light to trigger the pressure plate.

## Introduction

The design and purpose of our project was to build a small-scale object sorter. We knew that in the pharmaceutical or manufacturing industries, there can be small objects like pills or small electronic parts. It is faster and less labor-intensive to have a robot arm with sensors than to have a person sort. We wanted to have a smart object sorter that could differentiate between different objects and sort them correctly, which can be used in a supply line setting.

The main objective of the object sorter was to be able to differentiate between three different colored cubes: red, green, and blue. The secondary objective of the sorter is to measure the mass of the cubes, and then sort them based on that parameter as well. This would require a color sensor and a pressure pad sensor to detect if it was a heavy, light, or hollow cube of three different colors. We later set the hollow cubes to be placed in a bin. Its purpose was only to distinguish each bin for display purposes (via color or weight) as they were unable to be recognized by the pressure plate.

We decided to go with a four-bar mechanism to toggle between a standby position, where the arm would retract and carry the cube as it travels between the unsorted and sorted bins and a grabbing position. In the grabbing position, the air pump would actuate and provide suction to the cube. It would also toggle to the grabbing position when it is dropping the cubes into the sorted bins. In order for the arm to move between the unsorted ramp and the sorted bins, we decided to make a rotating base for the object sorter to be placed on. This would require a strong servo to rotate the base and another servo to actuate the four-bar mechanism between toggle positions.

To help users understand what the object sorter is doing, we used an LCD screen to display what action the robot is currently taking, and what color they are sorting. For instance, if it were in the standby position, the screen would display “Standby”, or “Heavy Cube” when picking up heavy cubes, or “Red/Green/Blue Cube” after the color sensor determines what color it is.

## Design Specifications

The project specifications were to create a four-bar mechanism that can sort varying colored and weighted cubes to simulate an industry object sorting arm. It must sort them all into different bins based on their color or weight depending on how the user wants them to be sorted. It has to be able to actuate between toggle positions and rotate about the base at the correct angles so as to sort into the correct bins without dropping them. Our object sorter must also be able to operate without breaking, as we had a small issue with the acrylic arms breaking.

- 4-bar Linkage
- Double-rocker - motor actuated w/ 2 DOF
- User age >= 12 years
- Lifespan > 3 months
- Easy to manufacture and source materials
- Simple to operate
- Sort objects accurately based on set parameters (color and weight)
- The base must rotate <=180 deg
- The picker must hold on to the object for the duration of sporting action
- The system must detect when sorting has been completed
- The system must be safe
- The system must sort in a timely manner
- The system must be easily portable
- The system must be built in an organized manner for serviceability

## Design in Detail

This design for the object sorter consists of 10 major components: Two microcontrollers, that being a Raspberry Pi Pico, and an Arduino Uno V3, 3 Servos, a Limit switch, a Pressure pad, LCD Screen, a Solenoid, an Air pump, and Color Sensor. The block diagram in Figure 3 depicts how all the components will interact with each other. This section will go in-depth about how all the systems interact with each other to allow for the sorting of an object.

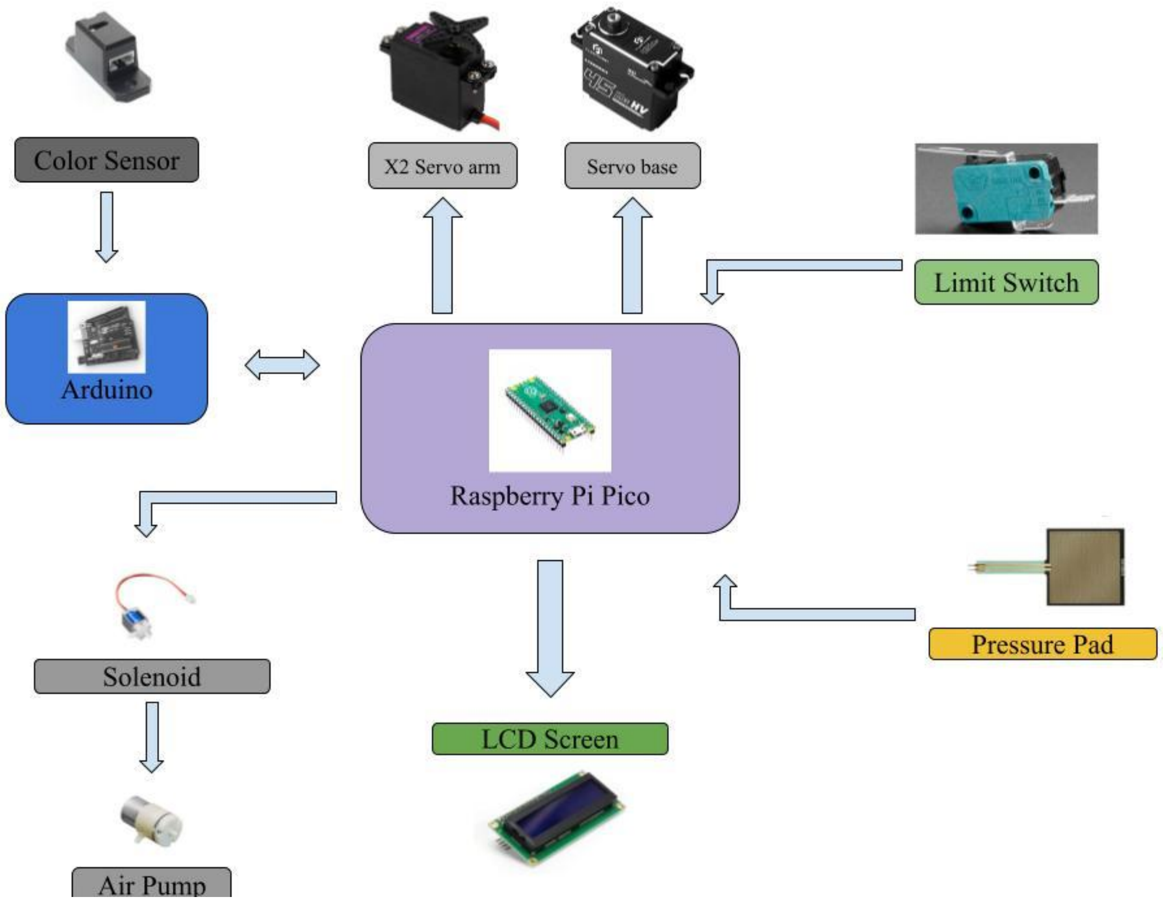


Figure 3. Block diagram of R.O.D.G.E.

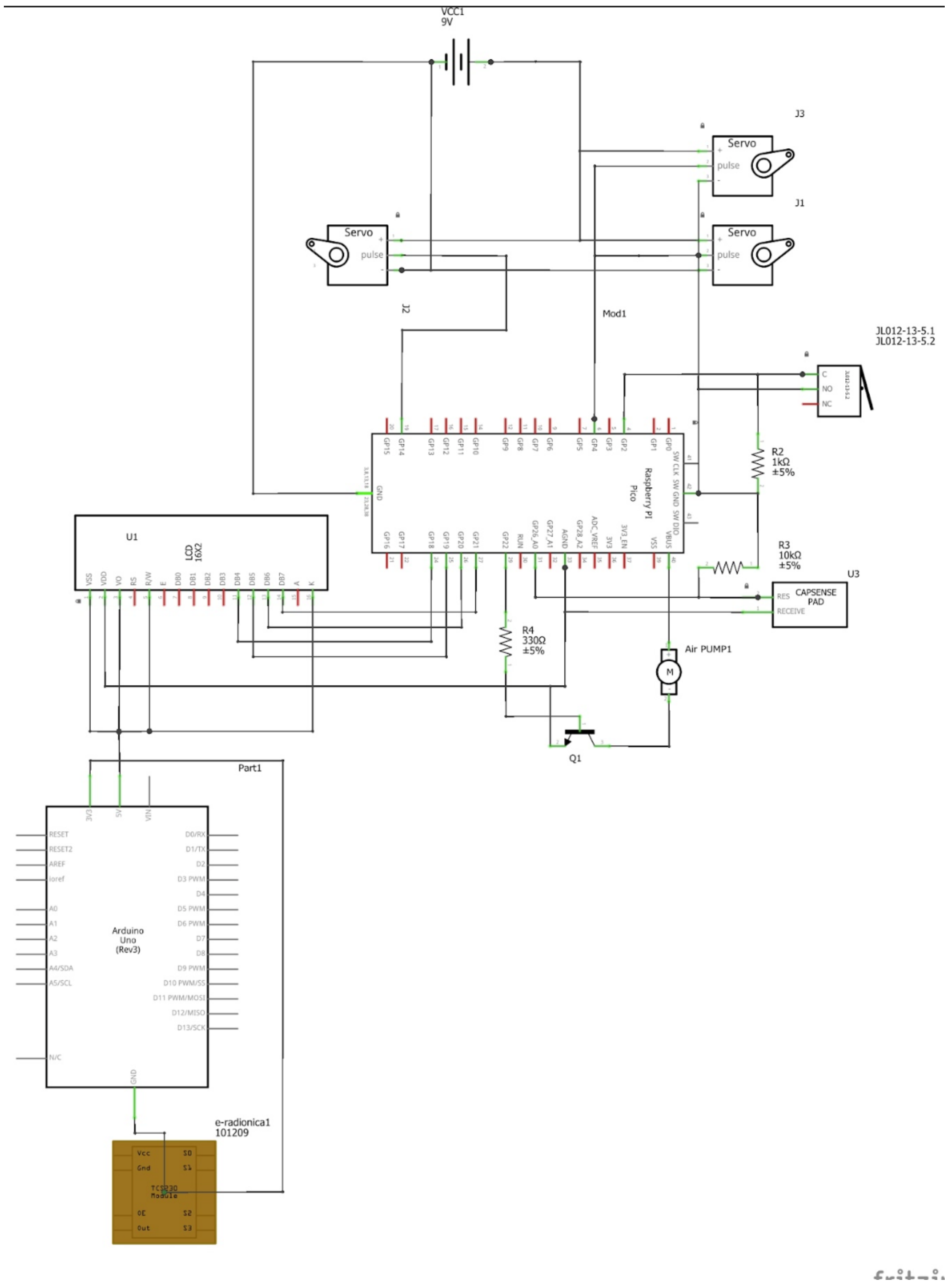


Figure 4. Full Wiring Diagram

The overarching power lines for our project were 9V and 5V in parallel as shown in Figure 4. The 9V was supplied by an external power supply and fed to the servos. All components are connected to the same ground network between multiple GND pins of the Pico, UNO, and external power supply. The PWM signals for each servo were supplied by the Raspberry Pi Pico's GP4 and GP14 pins. Servos J1 and J3 shared the same PWM signal because they were responsible for the same movement of the 4-bar mechanism. Our project ultimately only used one servo because the secondary servo was an unnecessary redundancy. Servo J2 was used to rotate the base of the robot's arm, enabling it to rotate between the pickup ramp and the sorting bins.

The limit switch was used to communicate with the robot whether it should stand by or sort objects. Our ramp was designed to have any objects fall and rest on the limit switch. If the limit switch was not activated, the robot would go to standby mode. If an object was placed on the limit switch, the robot would start to sort the objects. After an object was removed from the ramp by the robot, any object placed above it would fall, activating the limit switch, and telling the robot to sort again. This would continue until there are no more objects to sort. The normally open pin was grounded and the common pin was connected to GP28 of the Pico. The following program was used for the limit switch:

```
from machine import Pin
import time

switch = Pin(28, Pin.IN)

while True:
    print(switch.value())
    time.sleep(0.5)
```

The Arduino UNO has 3 output pins that are directly connected to the Raspberry Pi Pico, but otherwise function as the circuit's sensor and LCD power supply. The Arduino supplies 3.3V to the TCS34725 color sensor and 5V to the LCD display and the pressure pad sensor.

The TCS34725 color sensor and pressure pad sensor were mounted at the base of the ramp, where they would read the values of the next item to be sorted. For the color sensor, the SCL and SDA pins were used to interface with the I2C sensor. The UNO would continuously read the RGB values of the block at the bottom of the ramp and put them in an array. The index of the maximum value would be the index of the output pins of the UNO that outputs high. The Pico would receive all 3 inputs, and whichever input was high, would correspond to the bin index that the block would be sorted into. The pressure pad sensor was easier to integrate because its output was directly connected to the Pico's GP2 pin. Different values would be outputted for objects of different masses, and the robot would sort them in an intelligent way. The first object was sorted such that it would go into the first bin. The second object would be sorted into the same bin if it was within a certain mass tolerance, which can be specified in the code. If it exceeded that tolerance, then it would be sorted into the next bin, and so on.

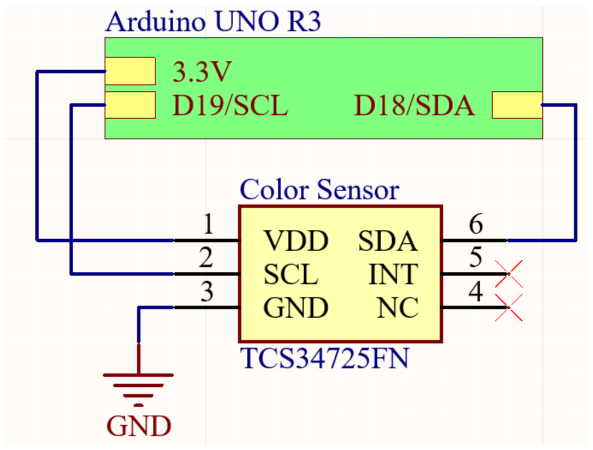


Figure 5. Color Sensor Pinout

Our LCD display was supplied 5V from the Arduino, and was connected to the Raspberry Pi Pico to receive data and display the corresponding message. The 5V pin of the Arduino was connected to Vss, Vd, R/W, and K pins of the LCD The display received data from the Pico by connecting DB4-7 to GP18-21, respectively. Vdd of the LCD was also grounded to the Pico.

The solenoid circuit was responsible for powering our arm's suction to grab onto objects. It was powered by the VBUS of the Raspberry Pi Pico, or 5V, power the air pump. The reason why we powered the air pump separately from the sensors was to limit the current drawn from the Pico. The negative terminal of the air pump was connected to the drain of an N-MOSFET, Q1. The source of the MOSFET was connected to pin AGND of the Pico. The gate of this MOSFET was connected to a GP22, a digital output pin that controls the air pump. For example, while the robot is meant to hold onto an object via the suction cup, GP22 outputs high to Q1, providing the air pump a path to the ground. When the robot arm needs to drop the object, GP22 outputs low, cutting off the air pump's path to the ground and thus turning it off.

In order to run the LCD, we looked online for code support on running our specific screen. We found Github repositories that had drivers for our specific screen, the I2C 1602 LCD. After modifying the code to fit our screen, we could then print whatever statement we needed on the screen. One problem we encountered was the screen turning on partially for a split second, which was later resolved by connecting it to 5V instead of 3.3V, even though it could technically run on a lower voltage.

## Project Outcome

Overall, we believe that the robot functioned as expected. It was able to sort by both parameters to some level, with the color-based sorting being 100% accurate. The weight-based sorting was inconsistent and unfortunately, we didn't have even one entire cycle of correct weight-sorting; a cycle entails the sorting of all 6 cubes.
When it came to the sensors, the RGB sensor worked wonderfully and a big reason for it is its placement and proximity to the cubes. The limit switch worked well too, but we needed to avoid using the hollow cubes as they wouldn't apply enough pressure to engage the limit switch. However, this wasn't the sole reason why the hollow cubes weren't used as they were more susceptible to falling off the ramp when sliding down. Instead, we used two sets of mass-based cubes with different amounts of coins in each of them. The pressure pad worked as expected, but it is the way we designed our ramp around it that led to higher inaccuracies in the sorting; we will address these design shortcomings in the upcoming sections.

We utilized two kinds of actuators- servos and pumps. Both were sourced to run at a common voltage of 6V, and they ran throughout the whole project without any problems. We also sourced a solenoid valve to be paired with the pump but decided to go with an NPN BJT instead, because none of the Pico's GPIOs could supply enough current to run the valve.

Mechanically, the design that came out exceptionally well as the housing's integration of the servos and the four-bar, the suction-based grabbing mechanism, the acrylic links, and servo extensions. However, problems arose with the interference between the housing lid and the four-bar base. We addressed this by grinding the edges of the four-bar base, which seemed to resolve the issue. However, with repeated use, the base kept leaning forward anytime a cube was picked up, so it ran into repeated interference with the housing lid then too; this led to the servo having erratic movements sometimes, as is evident in the demo video. Another issue was the placement of the pressure pad as we didn't tolerate the slot in the ramp within which it sits. We addressed this by cutting some of the edges off of the non-active area, which resolved the problem.

The slot along the ramp for the pressure pad sensor should have been designed to be lower because while objects were falling down the ramp, they would collide with the edge of the sensor, which would peel off bits of the pad's protective layers. Although we are unsure of whether this affected the performance of our project, in the long term, it would damage the sensor and make it unusable or inaccurate. Additionally, the slot for the limit switch at the base of the ramp could be repositioned such that the point of first contact between the switch and the blocks would be near the center of the block. This would better ensure that the switch can more consistently detect objects of even lower masses. The color sensor bracket that is attached to the ramp should also be improved. The current bracket makes wiring the color sensor somewhat difficult and in the long run, the wires would be worn out. However, this design issue is mostly due to last-minute changes in the type of color sensor being used, so there was no time to make a secondary design. Ultimately, it was due to color sensor difficulties, where we tried to use a sensor meant for a specific microcontroller and were unable to get it working properly. If we had started with the ideal sensor, we would have had the time to properly design a mounting bracket that allowed for the wires to experience less strain.

## References (Code and Data Sheets)

Thimmareddy, R. *Rajeev-T/object-sorting-robot: Documentation of a mechatronics project wherein a robot sorts objects based on color and mass.* GitHub. https://github.com/rajeev-t/object-sorting-robot.git

*Object Sorter - SJSU ME106 (Spring 2023)*. YouTube. (2023, May 21). https://youtu.be/CqlPdl_s8rg

# Appendix A -- Source Code for Aurdino

Arduino Uno main.py:

```
#include <Wire.h>
#include "Adafruit_TCS34725.h"
#define redpin 3
#define greenpin 5
#define bluepin 6
#define commonAnode true
#define rgbpins{10, 11, 12}
byte gammatable[256];
Adafruit_TCS34725 tcs =
Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_50MS, TCS34725_GAIN_4X);
void setup() {
  pinMode(3, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  Serial.begin(9600);
  if (tcs.begin()) {
    //Serial.println("Found sensor");
  } else {
    Serial.println("No TCS34725 found ... check your connections");
    while (1);
  }

#if defined(ARDUINO_ARCH_ESP32)
  ledcAttachPin(redpin, 1);
  ledcSetup(1, 12000, 8);
  ledcAttachPin(greenpin, 2);
  ledcSetup(2, 12000, 8);
  ledcAttachPin(bluepin, 3);
  ledcSetup(3, 12000, 8);
#else
  pinMode(redpin, OUTPUT);
  pinMode(greenpin, OUTPUT);
  pinMode(bluepin, OUTPUT);
#endif
  for (int i=0; i<256; i++) {
    float x = i;
    x /= 255;
    x = pow(x, 2.5);
    x *= 255;
    if (commonAnode) {
      gammatable[i] = 255 - x;
    } else {
      gammatable[i] = x;
    }
  }
}
void loop() {
  float red, green, blue;
  digitalWrite(rgbPins[0], LOW); // This & next 2 lines
  digitalWrite(rgbPins[1], LOW); // reset the output pins
  digitalWrite(rgbPins[2], LOW);
  tcs.setInterrupt(false);  // turn on LED
  delay(60);  // takes 50ms to read
  tcs.getRGB(&red, &green, &blue);
  tcs.setInterrupt(true);  // turn off LED
  int color[] = {red, green, blue};
  [maxValue,Index] = max(color);
  digitalWrite(rgbPins[Index], HIGH);
}
```

## Appendix B -- Mock of Schematic

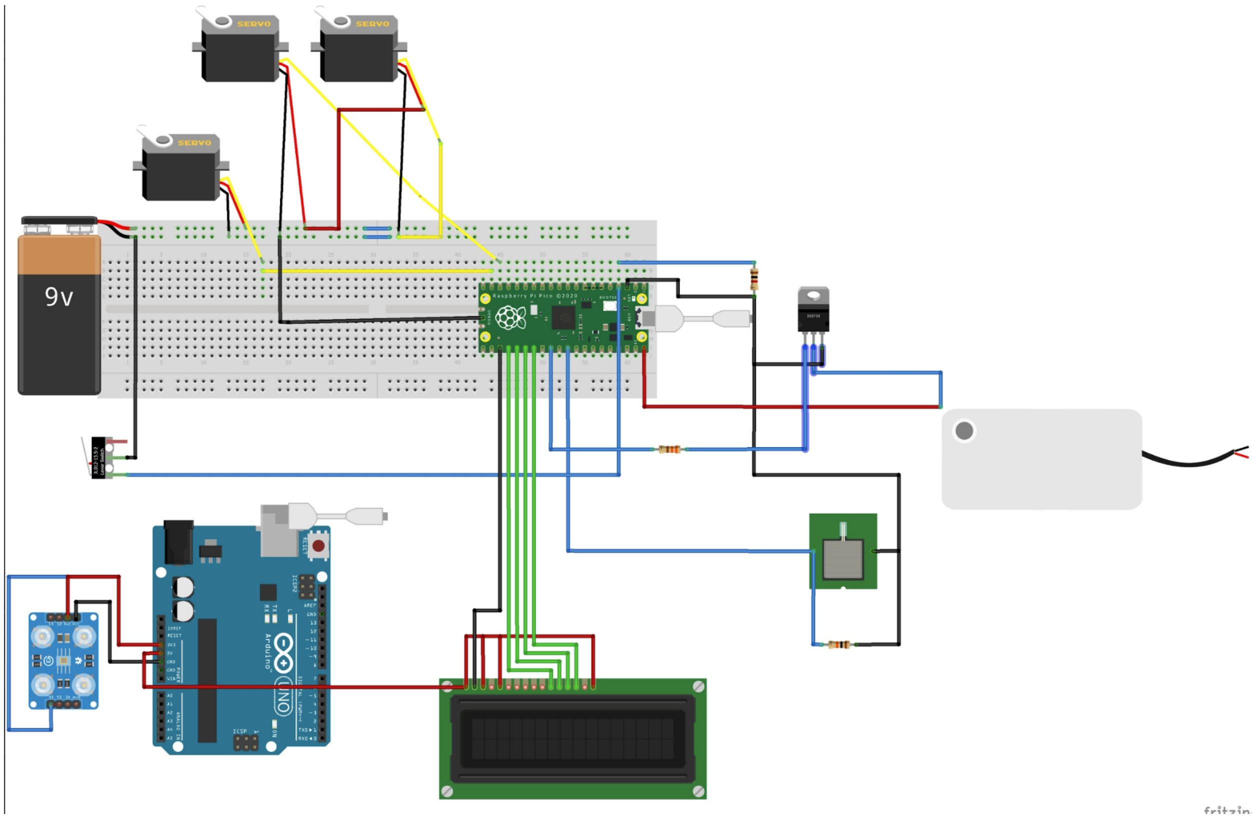


Figure 6. Schematic Diagram of Wired Components

## Appendix C -- Data Sheets

| Item | Data Sheet |
|---|---|
| MG996R 55G Servo | PDF |
| M45CHW Servo | Website |
| Pressure Pad | PDF |
| Color Sensor | PDF |
| Limit Switch | Website |
| Air Pump | HTML |
| LCD Screen | PDF |

## Appendix D -- Build of Materials

| Item | Price |
|---|---|
| 4 Pack - MG996R 55G Servos | 20.99 |
| M45CHW Servo | 41.99 |
| Color Sensor | 19.00 |
| Limit Switch | 1.5 |
| Acrylic Sheet | 17.48 |
| PVC Tubing | 17.77 |
| Vacuum Cup | 12 |
| Valve Fittings | 17.77 |
| Air Pump | 6.95 |
| Air Valve | 5.90 |
| Pressure Pad | 12.40 |
| Arduino Uno Kit | 40.99 |

Links and Further Breakdown