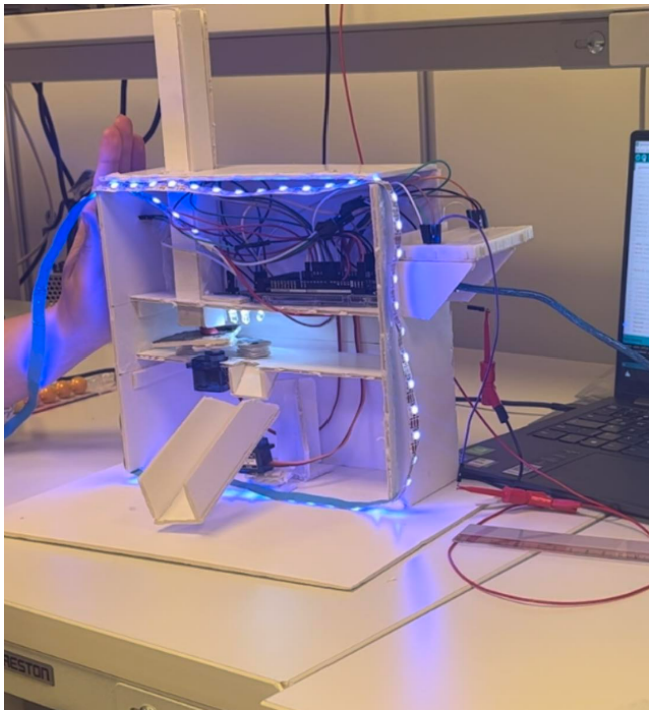


SEMESTER PROJECT

Touchless candy dispenser

Author: Justyna Kluska

Date of submission: 1.06.2023

Characters with spaces: 28871

Abstract

This report focuses on the development and evaluation of my individual project where the main objective is to deliver a self-built and programmed product that performs a specific activity. I chose to build a candy dispenser that offers a unique way of getting your favorite color of candy.

In a logical manner, I was able to implement a VL53L1X Time-of-Flight Distance Sensor, TCS3200 Color Sensor, and two servo micro motors facilitated by C++ coding and an Arduino microcontroller. Additionally, a WS2811 LED strip was chosen as a visual appeal. The dispenser successfully manages to perform its specific activity of detecting and sorting candy by color and finally, displays the right color on LEDs.

The dispenser works in a way that after dropping candy into a tube one can activate the distance sensor by putting their hand no further than 10 cm away from the sensor. This will then activate the first servo to move the candy underneath the color sensor. Once the right color is detected the second servo is triggered and moves a different tube into the desired direction. Almost simultaneously, the first servo keeps rotating to maneuver the candy into the now rightly positioned tube where it will drop down into the user's hand. By this time the LED strip is lit up in the detected color.

The report focuses on each individual module necessary within the sorting process. Different aspects such as design, implementation, and testing will be presented and discussed within the paper for each module. Through an in-depth investigation of the original and self-developed C++ code an analysis of how all of these different modules work both individually as well as in logical order will be possible.

Despite its conclusion of successful implementation and activation of the dispenser, the paper reflects on different shortcomings in terms of design, mechanisms, and durability due to deficits in material quality, its installations of moving parts, and arrangements of devices. In the last part, different future perspectives are introduced such as the idea of including a wider range of colors and an implementation of an additional distance sensor for customized candy selection.

Table of Contents

1. Introduction

Introducing an innovative touchless candy dispenser, a unique and funny device that offers numerous advantages. Do you have a favorite flavor of M&M's candies? Thanks to the sorting function, you can now easily select your preferred candies first. Furthermore, the candy dispenser is touchless, prioritizing hygiene, especially important in a post-COVID world. Whether you're sharing your candies with your family or a group of friends, it's crucial to remember cleanliness. Whether you're hosting a party, searching for a unique surprise for your child, or aiming to stand out on Halloween, the Touchless Candy Dispenser is the must-have accessory that brings joy, convenience, and hygiene to any occasion.

2. Requirements

The requirements of Project 4 are very broad in terms of the choice of the final product and its components. However, the main objective is to deliver a self-built and programmed product that performs a specific activity. It is crucial to apply the skills acquired during the 3rd semester, demonstrating practical know-how in the design, implementation, and testing phases of the project.

3. Methods and Tools

3.1 Methods

Each device will in the first step be tested individually for general functionality. Next, I will implement C++ coding for each device to perform the desired motion or action. Furthermore, I will install all the devices into the specifically designed container and combine the different tasks performed by each individual part by creating one large, fully self-developed C++ code. Once all the devices are in place, the dispenser should be able to perform the color-sorting process of a candy.

3.2 Tools

The used devices are the following:

- Arduino MEGA 2560 microcontroller
- VL53L1X Time-of-Flight Distance Sensor
- Servo Micro Motor x2
- TCS3200 Color Sensor
- WS2811 LED strips
- Breadboard

Additionally, Arduino IDE is used as a software to write and upload C++ codes onto the Arduino MEGA microcontroller.

4. Hardware

In order for the different modules to fit into one container and function properly in the aspired manner I had to plan in advance how everything will look like. I adjusted the measurements to make sure to have enough space for everything to work properly and for the cables to have enough space as well.

The container of the candy dispenser is made out of cardboard and glued together with hot glue.
The quadratic shape of the container has the height and length of a standard DIN A for a page in order to have sufficient space and simplify the cutting and preparing process. In terms of depth, I adjusted the box to comfortably support an Arduino Mega microcontroller. Thus I choose the box to be roughly nine centimeters in depth. What follows is a 2-dimensional sketch of the setup with numbers to describe each part.

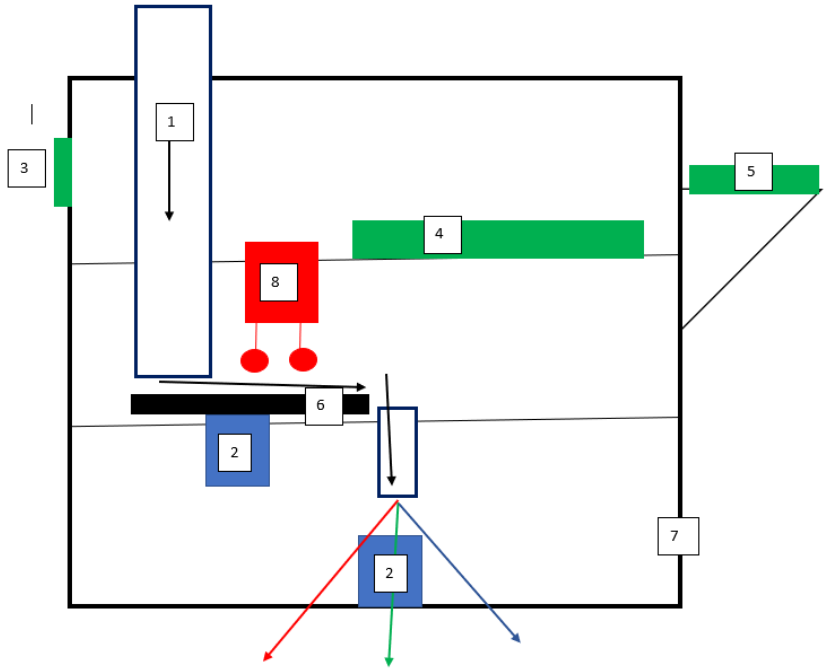


Figure 1 - sketch of the hardware setup

1. This tube lets the candy fall one by one and stops the catching mechanism from taking more than one candy at a time.
2. Number two are the top and bottom servos. The top servo is placed on the first level and has a quarter of a circle with a hole in it horizontally attached to it. This hole is where the candy will go into after dropping down the tube. The top servo will then start to rotate in the direction of the arrow and stops as soon as the candy is underneath the color sensor (8). Once the color is detected the servo keeps rotating in the direction of the arrow and brings the candy to the next tube. This tube is directly attached to the bottom servo and the candy will be directed in the right way according to the detected color.
3. The touch sensor is installed on the exterior of the container in order to have maximum accessibility.
4. The Arduino is placed on the top shelf in order to have enough space above it while still being fairly close to all the other devices. Furthermore, the Arduino is close to one of the sides to have both the USB port as well as the and power input easily accessible.

5. The Breadboard is placed on the outside of the container because this is where the power supply for all the devices will go. In this way, it is easily accessible for adjustments and yet remains close to the Arduino.
6. Number six depicts a simple semi-circle-shaped installment which is to be found underneath the extended turning part of the top servo. This installment ensures that the candy can be moved smoothly from the dropping point to the color sensor and further to the second dropping point. One should note that the top of this installment as well as the top of the extended turning part of the top servo were colored in black in order to increase the color sensor's accuracy.
7. The outside of the container is equipped with LED strips to indicate the color one will receive.
8. The color sensor will find its place on the top shelf next to the Arduino and has its LEDs downwards. This way it is close to the Arduino. The second shelf's height is properly adjusted for the light sensor to be close enough to the candy to have precise measurements.

The block diagram of the candy dispenser is presented below:

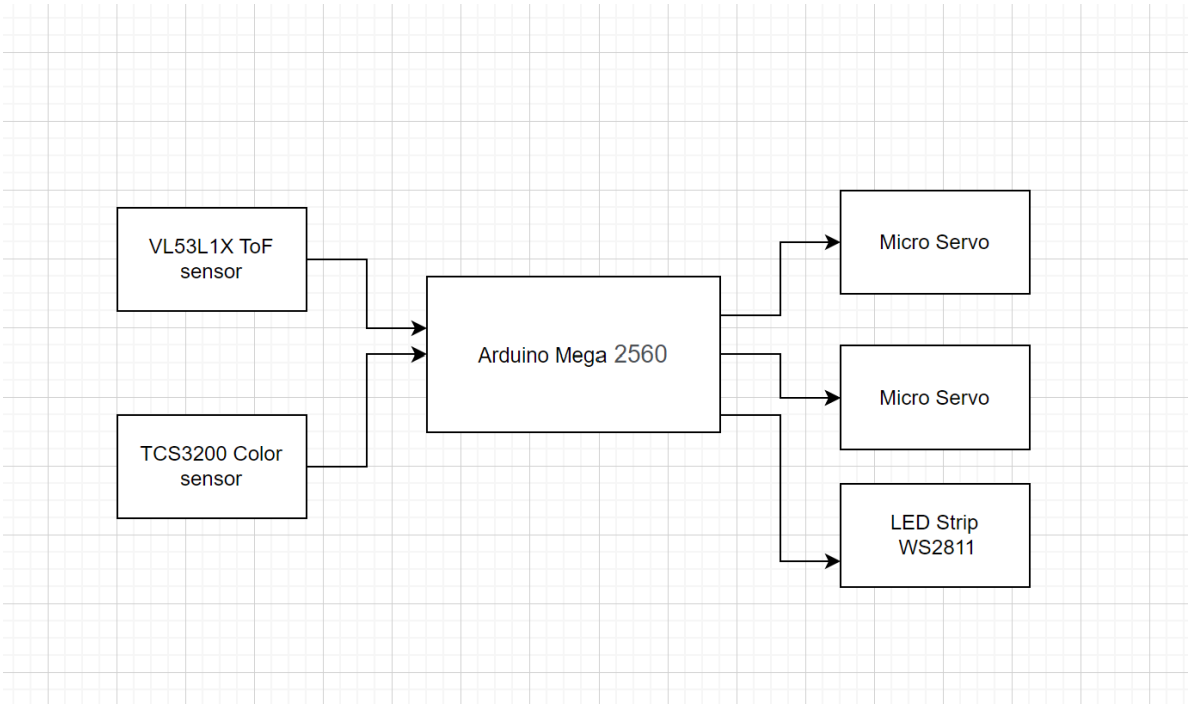


Figure 2 - Block Diagram

5. Modules

I have organized the description of my solution into different modules that are logical, testable, and manageable. Each module consists of the Design, Implementation, and Test phases.

5.1 Module 1 – Arduino MEGA 2560

5.1.1 Module 1 Design

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 chip. It contains 54 digital input/output pins (of which 15 can be used as PWM outputs), and 16 analog inputs. “It was chosen among other microcontrollers, because of the many available pins which are necessary to connect all the external input devices to the board. The operating voltage of this microcontroller is either 3.3V or 5V. This aligns with the servo’s 4.0 - 7.2 V operating voltage, the TCS3200 color sensor’ 2.7V t- 5.5V operating voltage, and the VL53L1X Time-of-Flight Distance Sensor’s 2.6 V - 5.5 V operating voltage. Thus, its voltage pins can be used to power the external equipment. Despite being larger than other microcontrollers, the Arduino Mega board is still suitable for this project, as the container has 21 cm in width and height as well as 9 cm in depth.

5.1.2 Module 1 Implementation

The implementation of an Arduino board is very simple. To check whether the board is functional it is necessary to connect the board to the computer using an USB cable. In the next step, one opens the ide environment and selects the appropriate Arduino model and the corresponding port. One can then upload code onto the board to test its functionality. I did so by using a Serial.print() function which allowed me to print a short message on the serial monitor.

5.1.3 Module 1 Test

To test if the Arduino Mega board is functional, I wrote a simple code and displayed the message on the Serial Monitor.

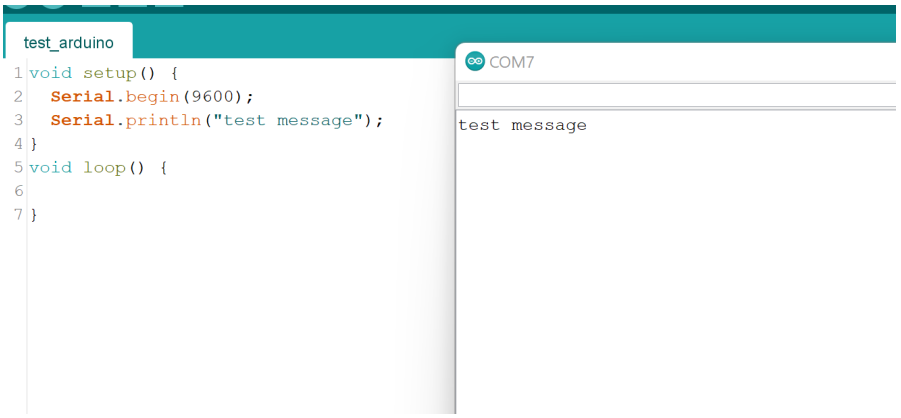


Figure 3 - Arduino basic functionality test

This quick test proves the basic functionality of the microcontroller and it is therefore ready to have further components implemented.

5.2 Module 2 – TCS3200 Color Sensor

5.2.1 Module 2 Design

The TCS3200 color sensor was chosen for this project due to its low cost, its programmable nature, and because it is ability to communicate directly with a Microcontroller via digital input and output pins. Furthermore, its size is perfect to keep the candy dispenser small and handy.

The color sensor comes with four LEDs to lighten up the object. A picture of it can be found below.

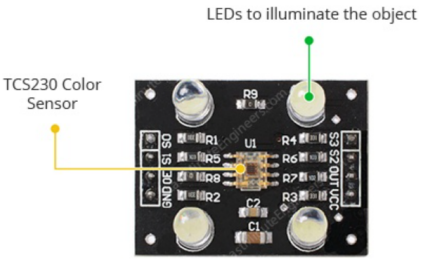


Figure 4 - Color sensor

The TCS3200 color sensor is a light-to-frequency converter, consisting of silicon photodiodes that can sense light, and a current-to-frequency converter. Its operating voltage is 5V. A total of 64 sensors are arranged in a 8x8 matrix forming arrays of photodiodes. The photodiodes are covered by one of four different filters: either red, green, blue, or clear filter. Each color covers 16 of the sensors. This makes sense since all colors can essentially be broken down into these three primary colors (RGB values).

Followingly, as soon as the TCS3200 is exposed to light the color filters will only allow a certain range of wavelengths associated with the respective color to pass through. If for example a red object is placed below the sensor the photodiodes covered by a red filter will sensor a high intensity of light that passes through the filter whereas the blue and green filters do not allow most of the light to go through leading to a lower intensity.

A representation of this process as well as the arrangement of the color filters can be found in the image below.

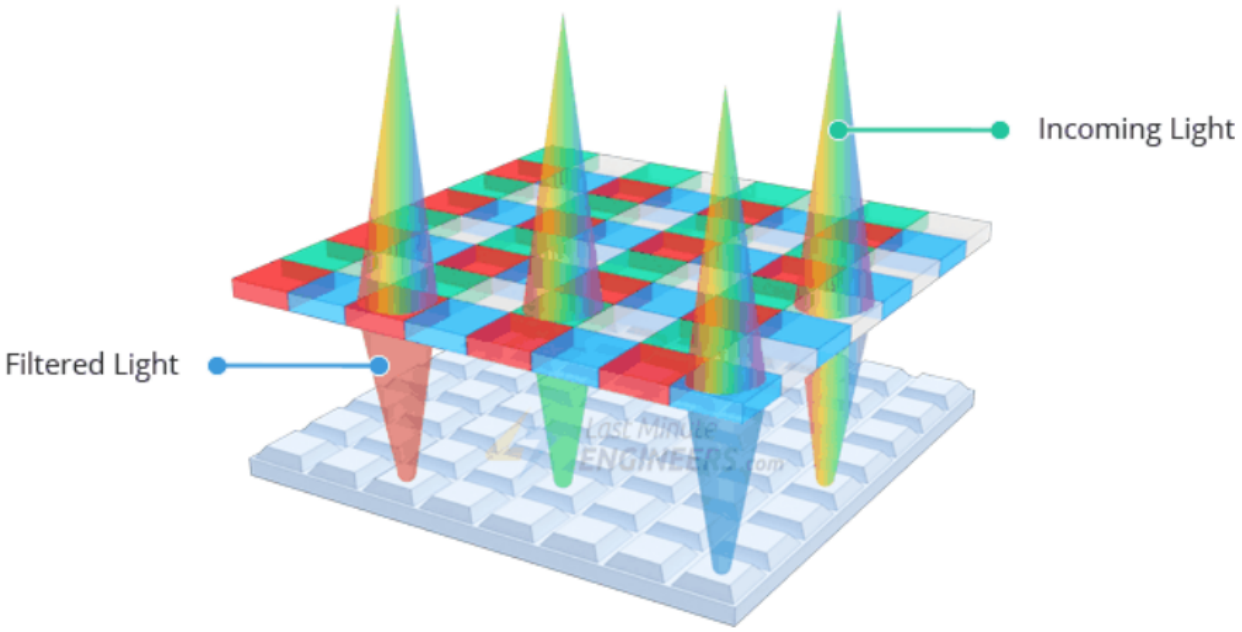


Figure 5 - Light filtering process

In order to know which color filter should be used to detect the light, the TCS3200 color sensor has every 16 photodiodes of a color connected in parallel using two control pins S2 and S3. By setting the pins to high or low you can select which color to focus on. The following table shows us which color is accessed by which setting:

S2	S3	PHOTODIODE TYPE
L	L	RED
L	H	BLUE
H	L	Clear (no filter)
H	H	GREEN

Figure 6 - Configuration diagram of pin S2 and S3

The filtered light then reaches the photodiodes which essentially are electric components generating small electric currents when triggered by incoming light. The amount of current it produces is dependent on the intensity of light falling onto the photodiode. These currents are characterized by a distinct pattern that alternates between two voltage levels: High and low. Thus the frequency output is considered to be a square wave signal.

The generated square wave output corresponds to a color and light intensity. The typical range of output frequency is 2HZ-500KHZ. The sensitivity of the TCS3200 sensor can be adjusted by changing the scaling

factor. The scaling factor is a parameter one can adjust to modify the relationship between the detected color or light intensity and the resulting output frequency. For example, an output frequency scaling of 20% means that the frequency of the square wave signal generated by the color sensor is reduced to 20% of its original value, which is mostly used when working with Arduino microcontrollers. The output frequency scaling values can be changed by using different combinations of S0 and S1. The chart below shows the different settings.

S0	S1	OUTPUT FREQUENCY SCALING (fo)
L	L	Power down
L	H	2%
H	L	20%
H	H	100%

Figure 7 - Configuration diagram of pin S0 and S1

These square wave signals are analog signals which will then be converted into digital values using an analog-to-digital (ADC) converter. These output values represent the intensity of light detected for each color (red, green, and blue) and can range from zero to the maximum value supported by the sensor. The color composition is determined by the values of the three different colors (red, green, and blue). As a next step, one has to calibrate the sensor which involves determining the relationship between the sensor's output value and the corresponding colors. Once the calibration is done the digital output values for each color channel can be obtained.

Further possibilities will then entail the conversion of the frequency values into RGB values. A brief description of this process can be found below. This project however will not make use of such a conversion. The obtained numbers will enable us to do scaling and normalization in order to convert the values into a specific numerical range (0-255) which is used in RGB representations.

A mapping algorithm could be used to convert the calibrated, scaled, and normalized output values into RGB values. To do so the algorithm applies mathematical calculations to determine the appropriate RGB values based on the given output values. The outcome is an RGB value representing the color composition detected by the sensor. Each RGB value stands for the intensity of red, blue, and green light needed to create the detected color.

5.2.2 Module 2 Implementation

The color sensor is connected to Arduino pins: S0 - digital pin 4, S1 - digital pin 5, S2 - digital pin 7, S3 - digital pin 6, OUT - digital pin8, the pinout is presented on the schematic below:

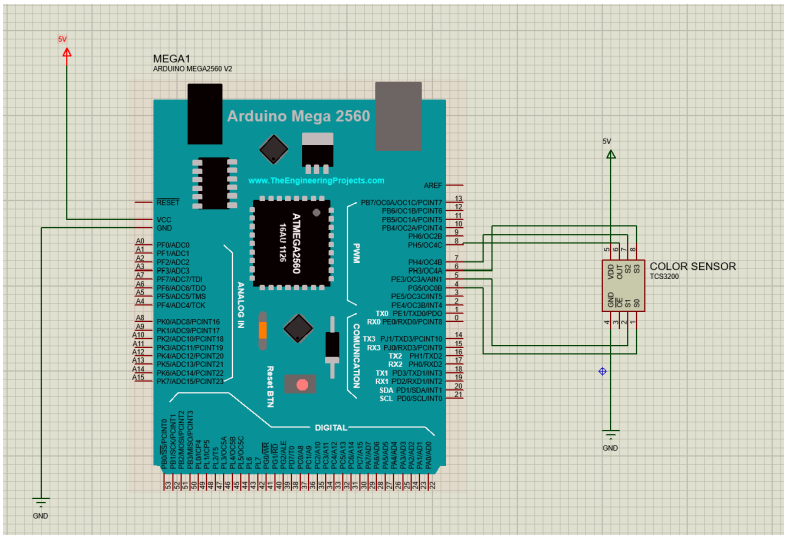


Figure 8 - Schematic1

5.2.3 Module 2 Test

To ensure that the color sensor is working, the first step was to check if it is able to provide readings. The output readings should provide a color frequency which refers to the wavelength of light and thus to the detected color. In color frequency representation, colors are described using their wavelength or frequency value. After conducting research on the sensor implementation, there are several possible Arduino libraries for TCS230 sensors. However, I decided to use the “pulseIn” function, which measures the pulse of a digital signal. The first test was conducted in order to check the functionality of the sensor. This was done using black and white colors.

```
const int S0 = 4;
const int S1 = 5;
const int S2 = 7;
const int S3 = 6;
const int OUT = 8;
int red = 0;
int green = 0;
int blue = 0;

void setup() {

  Serial.begin(9600);

  pinMode(OUT, INPUT);
  pinMode(S0, OUTPUT);
  pinMode(S1, OUTPUT);
  pinMode(S2, OUTPUT);
  pinMode(S3, OUTPUT);
  digitalWrite(S0, HIGH);
  digitalWrite(S1, LOW);// using 20% of frequency scaling

}

void loop() {

  digitalWrite(S2, LOW);
  digitalWrite(S3, LOW);
  red = pulseIn(OUT, LOW);

  digitalWrite(S2, HIGH);
  digitalWrite(S3, HIGH);
  green = pulseIn(OUT, LOW);

  digitalWrite(S2, LOW);
  digitalWrite(S3, HIGH);
  blue = pulseIn(OUT, LOW);

  Serial.print(" Red: ");
  Serial.print(red);
  Serial.print(" Green: ");
  Serial.print(green);
  Serial.print(" Blue: ");
  Serial.println(blue);

  delay(500);
}
```

Figure 9 - basic functionality test

The test proved that the sensor provides constant measurements. When working with frequency values high numbers represent less intense colors such as black whereas low numbers represent colors of high intensity such as white. Those values can be also converted to the RGB scale as mentioned in the

description. However, this is not of importance for this device since the sensor will be installed in a fixed position and the used colors of the candy will be easily distinguishable. Thus, the range of frequency values of each color will be rather static and predictable.

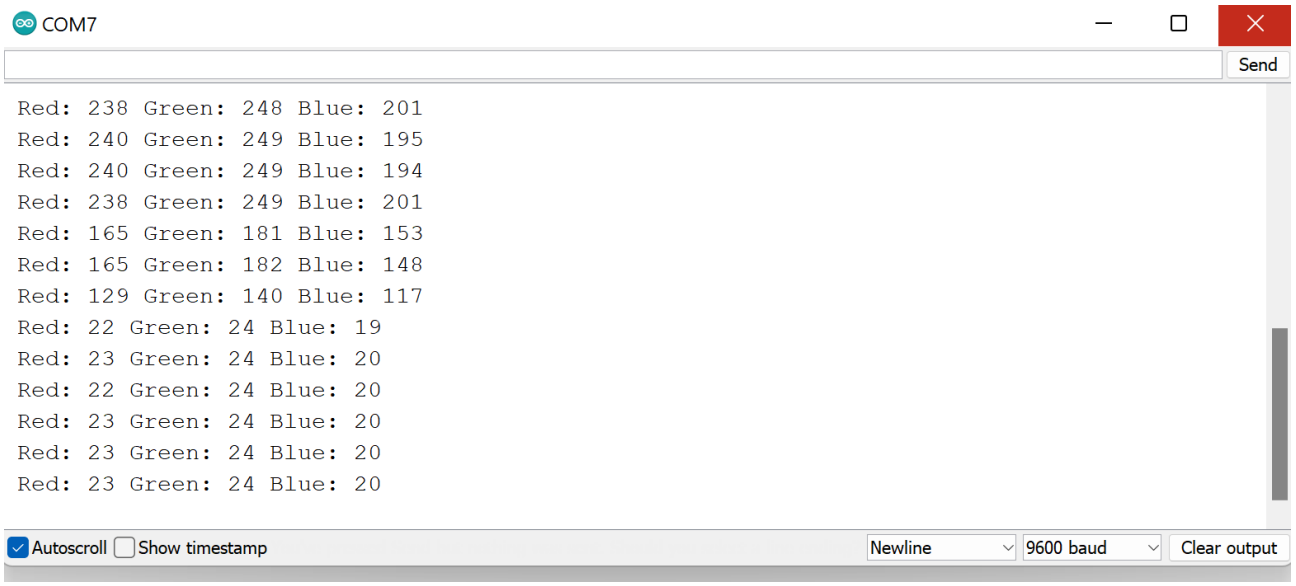
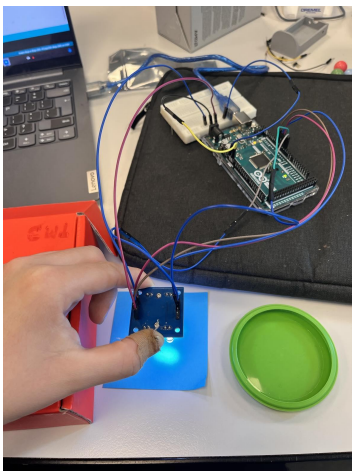


Figure 10 - Measurements for Black and White colors

In the next step, the same test was conducted with different colors. The first distance to be checked was according to the datasheet the optimal distance to read the measurements. Later on, distances of 3 and 5 cm were checked as well to see if the values were still accurate.

This testing was important in order for the Sensor to be installed at a distance that will allow for accurate results but yet does not interfere with the candy transporting mechanism of the turning top servo. After conducting a few experiments with different objects, backgrounds, surfaces, and distances I realized that the sensor is very sensitive and even a small difference in the color intensity of the object changes the measurements significantly. Therefore, the first prototype of the project will only use one brand of candies (smarties) and will focus on the three main colors. However, it is to be said that different candies and colors would be possible as well given more time to accurately define frequency ranges and higher quality material to install a better functioning transporting mechanism.

Pictures of different testing stages of the color blue can be found below:



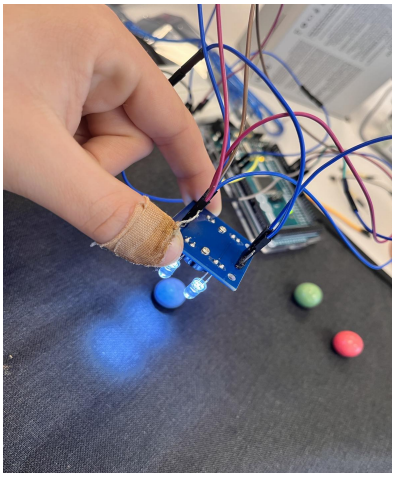


Figure 11 - Test of the color sensor

For the main code the color sensor will be used in a function that returns an integer: 1 for red, 2 for blue, and 3 for green color, that way the function can be called every time before servo will make a move to know what angle to shift depending on the read value - detected color of the candy.

When doing measurements it is important to notice that even when a certain color is measured there will be an output for all three colors. An output thus always consists of frequency values for red, green, and blue. How my code determines the color based on the frequency value outputs will be shown below.

To do so I started by taking sample measurements of frequency for red, blue, and green candy. The results of these sample measurements are shown in the screenshot below. To simplify the code and improve its aesthetics the color names were replaced by the letters R (red), G (green) and B (blue).

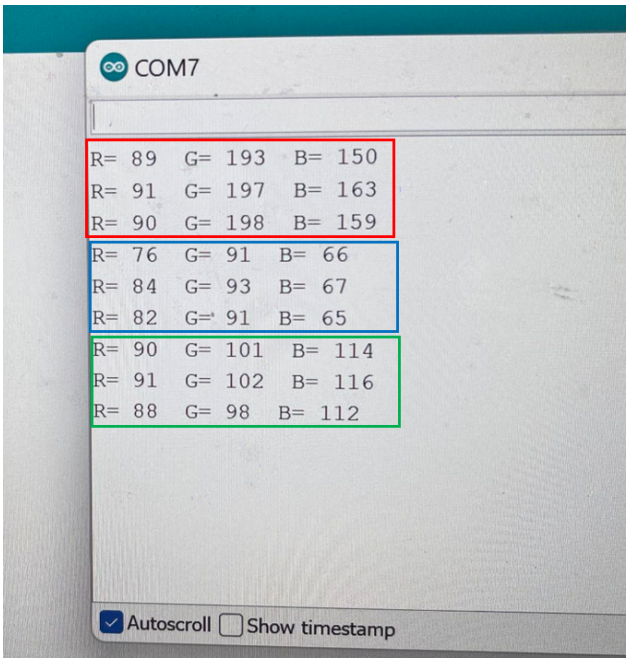


Figure 12 - Measurements for 3 different colors

The picture above shows us nine different sample values of measurements. The first three samples were done to detect a suitable frequency value range of the color red by using a red candy (red box). The following three measurements correspond to the color blue (blue box) and the last three to the green (green box) using the respective colored candy.

Within the main code, I made use of these frequency values in order to detect the right color. Since every time the frequency values for all three colors are measured, the sensor will be installed in a fixed position and the candies of each respective color should be relatively similar; one can use the detected sample values to make estimates of a range of frequency values that can be expected in future color detections.

The code will followingly look like this:f (R > 80 && R < 100 && G > 160 && G < 220 && B > 120 && B < 175) {

```
  color = 1; // Red color
 }
 if (G > 90 && G < 115 && R > 82 && R < 97 && B > 90 && B < 125) {
  color = 2; // Green color
 }
 if (B > 40 && B < 80 && G > 75 && G < 120 && R > 55 && R < 95) {
  color = Blue; //Blue color
```

The first line entails the range of values I estimated for the color red. After having relatively stable sample values when detecting red candies of 89, 91, and 90 (see picture above) I set the frequency value range for light detected by red-filter-covered photodiodes to be between 80 and 100. Furthermore, I derived frequency value ranges for the other two color-filtered photodiode types, in the same manner, having ranges of 160 - 200 for green and 120 - 175 for blue.

So whenever a new candy is placed underneath the sensor the new values will be compared to the defined color ranges. When one set of frequency ranges is hit for all three detected color frequencies the code will give us an output of the corresponding color.

For example, if a new candy will be placed underneath the color sensor and the sensor will read frequency numbers of 95 for green, 90 for red, and 100 for blue the code checks whether these values fit into one of the three colors with its respectively defined frequency ranges. It should then give us an output of 2 (green color) since all three values fall within the respective range. For green the range is from 90 to 115 and 95 is in between these numbers. The same goes for the other two colors red (82<90<97) and blue (90<100<125).

A corresponding range for each output color has to be defined for every possible color I want to be able to detect. The applied logic is identical to the earlier described red line of coding.

5.3 Module 3 – Servo Micro Motor

5.3.1 Module 3 Design

To build the sorting mechanism of the machine two Micro Servos SG90 Towerpro were used. These Servos can rotate approximately 180 degrees (90 in each direction). The servos are controlled with Pulse Width Modulation. The Arduino board generates a PWM signal on the specified pin, and the servo interprets the duration of the pulse to determine the angle of rotation. By adjusting the duration of the pulse, the servo can be positioned at different angles. The range of motion is crucial for the machine to move precisely 45 degrees to the right or left in order to position candies at right angles based on their detected color.

5.3.2 Module 3 Implementation

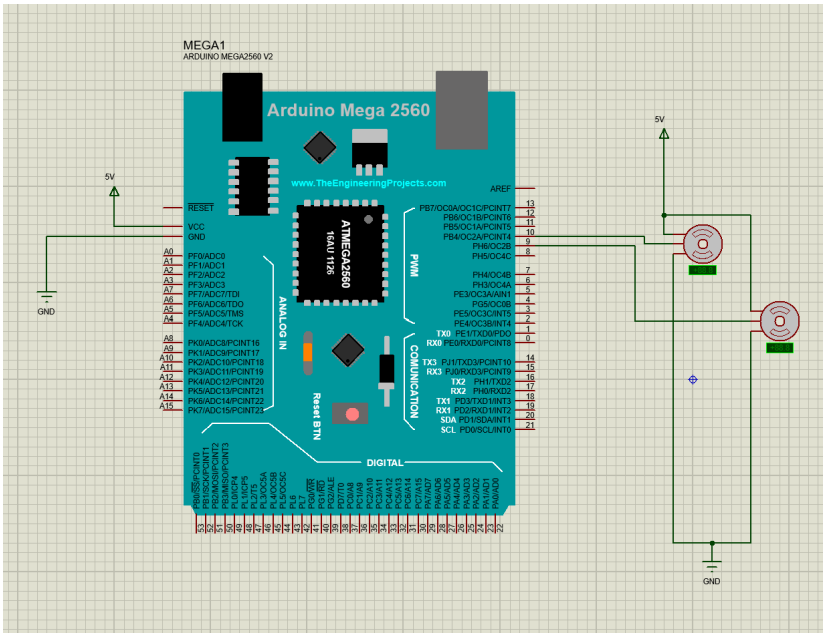


Figure 13 - Schematic 2

Micro Servos were connected according to their pinout, to the shared power source of 5V, common ground, and digital pins 9 and 10. In order to implement servos into the machine, the right angles have to be measured, for both top and bottom servos.

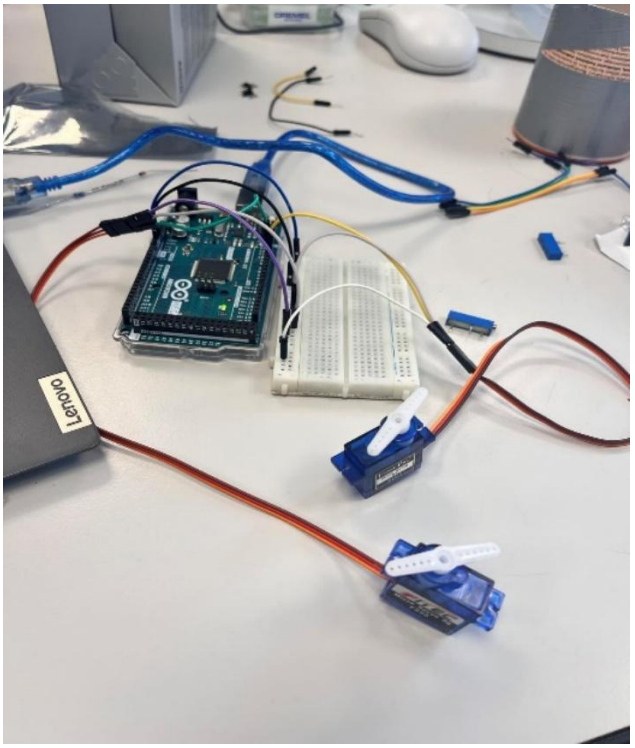
The first servo needs to be adjusted to the position where the candy drops down from the tube. It then moves its head which will transport the candy. It first stops underneath the color sensor. This angle will be the same every time the loop iterates since the color sensor is mounted in a fixed position.

Next, the color sensor function will return the value 1,2 or 3 depending on the detected color. Each of these values corresponds to another angle of the second servo. Thus, the bottom servo will shift to one of 3 possible angles (corresponding to the 3 different colors). Once the color is detected, the first servo needs to move again to drop down the candy. The angles were adjusted according to the hardware installations.

The whole implementation of the code with comments for servo motors and color sensor can be found in the Appendix 1.

5.3.3 Module 3 Test

The first test of the servos was made to check their basic functionality and the pins to which they were connected. Thus, a short code was written and uploaded to the Arduino board. The test checked if the servos moved to chosen angles and for any possible errors with the implementation of the code. The Servo library was implemented for precise and efficient control of both motors.



servo_test

```
#include <Servo.h>

Servo servo;
Servo servo1;

void setup() {
  servo.attach(9);
  servo1.attach(10);
  servo.write(0);
  servo1.write(0);
  delay(1000);
}

void loop() {
  servo.write(45);
  delay(1000);
  servo.write(0);
  delay(1000);
  servo1.write(90);
  delay(1000);
  servo1.write(0);
  delay(1000);
}
```

Figure 14 - Implementation of both Servo motors
Figure 15 - Basic functionality test of Servo's motors

After conducting a successful basic functionality test, the servos were implemented into the prototype of the machine in order to choose the right scope of movement for each servo and to adjust the suitable angles, and delays necessary for the main code of the program. It is important to mention that each servo will perform a different task within the sorting process. The top servo will catch the candy, move it underneath the color sensor and then move it even further to drop it into the next tube. This tube will be directly connected to the second Servo which will be sorting the candy according to the output of the color sensor.

Figure 16 - Servo’s installation

After uploading the code (Appendix 1) to the Arduino board, a test of the mechanism of color sorting was conducted. The test focused on both the two servos as well as the color sensor. The code was successfully uploaded and the machine is capable of sorting candies according to red, green, and blue colors.

5.4 Module 4 - VL53L1X Time-of-Flight Distance Sensor

5.4.1 Module 4 Design

The VL53L1X Time-of-Flight Distance Sensor mainly consists of three different parts: an emitter, a receiver, and a control circuit. These are arranged on top of a 2x2 cm board and can detect objects from 4cm to 4m distance with a frequency range of 50 Hz. Its small size, operating voltage of 5V, and compatibility with Arduino microcontrollers matched the candy dispenser's design and aspired functionality perfectly. This feature contributes to the candy dispensers' touchless usage and unique functionality.

The VL53L1X Time-of-Flight Distance Sensor relies on a basic and simple functionality that can be used for gesture recognition as well as obstacle detection.

The sensor emits short pulses of high-frequency, near-infrared laser light which will then travel through the air and eventually reach an object in their path. As soon as the object's surface is hit, the light will be reflected and start traveling back to the sensor.

The receiver contains a photodiode that detects the reflected laser pulses and converts them into an electrical signal that is sent to the control circuit. The control circuit measures the time it takes for the laser pulses to travel from the emitter to the object and back to the receiver, thus the name “time-of-flight”. By knowing the speed of light and how much time it took the signal to travel there and back the control circuit is able to calculate the distance that was covered within this timeframe and thus knows how far away the hit object is by dividing the total distance covered by 2.

The sensor then provides the distance measurement as a digital output which can be read by the Arduino microcontroller. In the case of this candy dispenser, this device brings an innovative function that only activates the sorting process when it detects a hand within 10 cm or closer. If the hand moves further away, the dispenser will stop working. To achieve this, the time-of-flight sensor was mounted on the external side of the box.

5.4.2 Module 4 Implementation

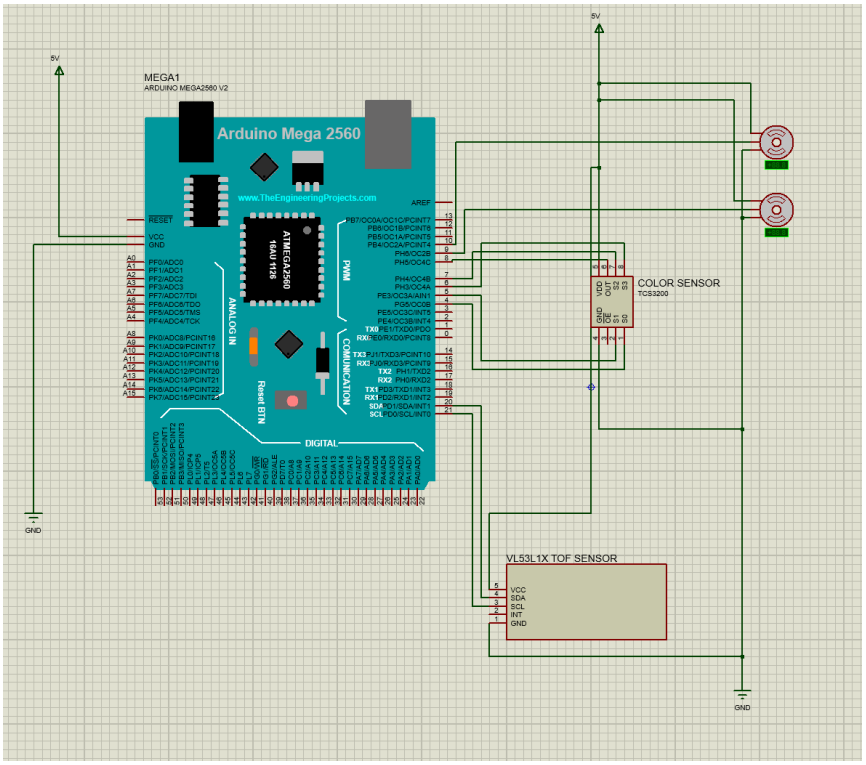


Figure 17 - Schematic 3

The VL53L1X sensor got connected to the Arduino board to corresponding SDA (serial data pin) and SCL (clock pin) communication pins, as the device operates on 5V, it got connected to a shared power source and common ground.

The sensor was implemented into the program to make it work while an object is detected at a distance of 10 cm. To implement the sensor to the program VL53L1X.h library was included, to make the interaction with the sensor easier. The library provides functions like setDistanceMode(). There are 3 modes to choose from: short, medium, or long. For this project “short distance mode” was chosen since the detection of short distances will be sufficient. Therefore the sensor can also give more precise values.

SetMeasurementTimingBudget() - sets up a maximum time possible for every measurement, this value got set to 20 milliseconds in order to get accurate measurements of time for a short distance.

The Start continuous () function sets up a given period of time in milliseconds between measurements. This will keep the sensor constantly updating and checking for objects. The period got set up to 50 milliseconds between every measurement.

The read() - function is necessary for reading every single measurement from the sensor and is used to create if statements that start the sorting process after detecting an object within 10 cm. Moreover, the Wire.h library was added in order to provide a function for I2C communication in this case for communication of Arduino with the ToF sensor.

The library provides basic functions like Wire. begin() - a function that allows IC2 communication between the Arduino and the sensor as well as Wire.setClock() - a function that sets the clock speed for devices communication. The maximum speed of 400 kHz for Arduino was chosen in order to set the communication to the fast mode.

The whole implementation of the code with comments for servo motors and color sensor can be found in the Appendix 1.

5.4.3 Module 4 Test

At first to check the basic functionality of the distance sensor a short code was implemented to ensure accurate measurements of the distance of an object in millimeters. Subsequently, the obtained value was printed on the serial monitor. The test was successful as the sensor consistently provided accurate readings when a hand was placed in close proximity to it.

```
#include <Wire.h>
#include <VL53L1X.h>

VL53L1X sensor;

void setup()
{
  Serial.begin(9600);
  Wire.begin();
  Wire.setClock(400000);

  sensor.setTimeout(500);
  if (!sensor.init())
  {
    Serial.println("Failed to detect and initialize sensor");
    while (1);
  }

  sensor.setDistanceMode(VL53L1X::Short);
  sensor.setMeasurementTimingBudget(20000);
  sensor.startContinuous(50);

}

void loop()
{
  uint16_t distance = sensor.read();

  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" mm");

  delay(500);

}
```

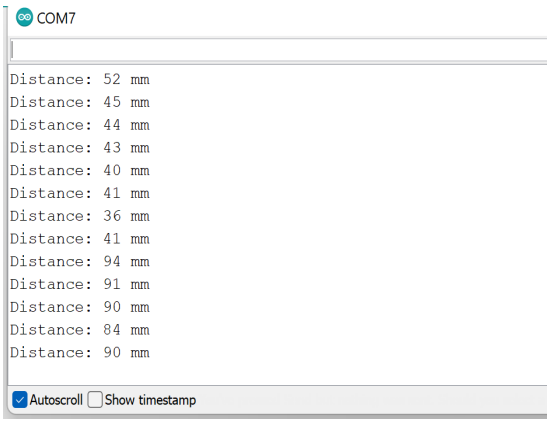


Figure 18 - Basic Functionality test of distance sensor
Figure 19 - screenshot of the serial monitor

After conducting a successful functionality test, the sensor was implemented throughout the entire machine and tested in conjunction with a color sensor and servos. When a hand was detected at a distance of 10 cm, the program started running, indicating a successful test.

5.5 Module 5 - WS2811 LED Strip

5.5.1 Module 5 Design

The WS2811 LED strip allows controlling of the color brightness and behavior of every single individual 5050 SMD LED on the strip. By generating a PWM signal from the microcontroller and varying the intensity of the red, green, and blue full-color RGB mixing is possible. For this project, the main focus is on those 3 colors, however, in the future an adjustment of a program entailing a wider range of different colors can be aspired. The strip can be cut or extended which makes it easily adaptable to the size of the box. 70 LEDs are used in the current project. The operating voltage is 5V, therefore it's connected to the shared power source, common ground, and digital pin 2 on the board.

5.5.2 Module 5 Implementation

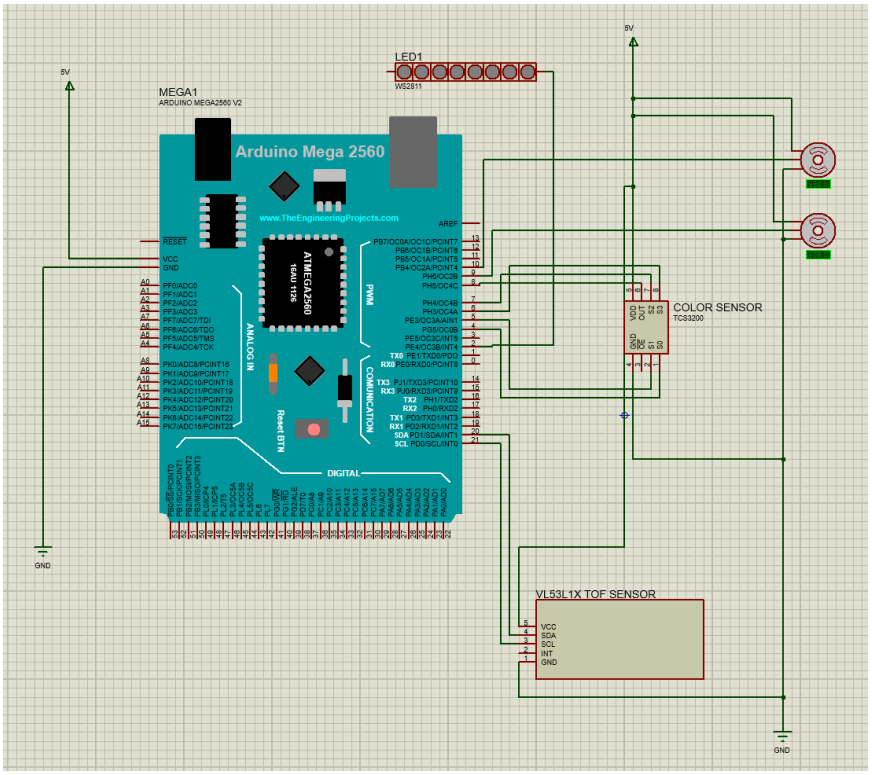


Figure 20 - Schematic 4

There are commonly used libraries for LED strips on Arduino IDE "NeoPixel" or "FastLED". “FastLED” was implemented for this project. This easy-to-use library allows for adjusting of the LED strip by specifying LED type, pin number, and color order. To configure colors on the led strip the FastLED.show() function is used.

In addition to the FastLEDs function, the setBrightness function enables the adjustment of the brightness level of the LEDs.

For every recognized color of candy, the LED strip is set to the same color within the code implementation. Furthermore, by implementing the FastLED.show() function the LEDs are updated every time a different color is detected. This feature not only provides the candy dispenser with a visual advantage but also enhances its interactivity and overall appeal.

Below there is a piece of code presented for the red color. The whole code containing the LED strip implementation with the comments can be found in Appendix 3.

```
Case 1:
  servo1.write(45); // move the second servo to an angle of 45 degrees
  for (int i = 0; i < NUM_LEDS; i++) {
    leds[i] = CRGB::Red;   // Set the LED color to red
  }
  FastLED.show();         // Update the LED strip with the new color
  delay(500);
  break;
```

5.5.3 Module 5 Test

To conduct a basic functionality test of the LED strip a short code was implemented to light up the 70 led, included in the final project. By using a “for” loop that iterates over every specified LED in the strip and assigns the RGB color value for red to each element one could test the LED strip. The test was successful and 70 LEDS were lit up in the red color.

```
#include <FastLED.h>
#define LED_PIN 2
#define NUM_LEDS 70

CRGB leds[NUM_LEDS];

void setup() {
  FastLED.addLeds<WS2811, LED_PIN, GRB>(leds, NUM_LEDS);
  FastLED.show();
}
void loop() {
  // Set all LEDs to red
  for (int i = 0; i < NUM_LEDS; i++) {
    leds[i] = CRGB::Red;
  }
  FastLED.show();
  delay(500);
}
```

Figure 21 - Basic Functionality test of WS2811 LED strip
Figure 22 - proof of functionality of the WS2811 LED strip

After conducting a successful functionality test, the LED strip was implemented throughout the entire machine and tested in conjunction with a color sensor, servos, and a distance sensor. Once the color sensor detected each different candy color, the LED strip lit up in red, blue, or green accordingly. Therefore, the final component of the project has been successfully implemented.

6. System Integration

After the construction of the machine prototype, the system was gradually integrated by adding the Arduino MEGA board, TCS230 color sensor, two servo Micromotors, VL53L1X Time-of-Flight sensor, and finally the WS2811 LED strip.

The first round of installments entailed the distance sensor which was attached to the left side of the container. Next, the top servo needed to be installed and properly integrated into the candy moving mechanism so the candy could be caught, transported to the color sensor, and eventually dropped in the right spot. As a third device, the color sensor was placed into its designated place, tested, and adjusted according to the new position. With the bottom servo, the last part of the sorting process chain needed to be installed. I managed to install it in its designated spot where it was able to catch candy with its attached tube and move it to the right place according to the detected color.

The implementation and testing of the LED module was the last step in the integration of the entire system. The LED test was successful in illuminating the lamps with the same color as the candy and the bottom servo correctly positioned itself to the desired angle. The system test confirmed that all the components were functioning properly together. The code for integrating the entire system can be found in Appendix 3 and in Appendix 4 a schematic diagram illustrating all the connections to the pins on the Arduino board can be found.

7. Proof

Proof of a functional candy dispenser can be found on a YouTube video by clicking the link provided below:

https://www.youtube.com/shorts/lN5HF4YUYCY

8. Perspective/ Reflections

With this candy dispenser I made use of both theoretical and practical, in-class acquired, knowledge about several different devices. I managed to logically install a distance sensor, color sensor, and two micro servos in order to sort candy by its color using C++ coding. Through the implementation of an Arduino microcontroller, the code was executed accordingly. As a visual feature LED stripes were installed in order to show the recipient of the candy what color was detected. Overall the dispenser runs smoothly and manages to fulfill its task as supposed to.

In the following part of the paper, I present points of improvement and mention some of the ways in which a device like this could be further expanded.

One of the dispenser's main restrictions is due to its current hardware installation. As of right now does the catching of candy not allow several different candies to be in the tube at the same time. Even though the code, and the devices executing it, would be able to handle several candies right after one another the physical installment does not allow the dispenser to do so. This is because the hole within the catching mechanism is too large. Since the type of candy used had to be switched after the building process due to some external factors more than one candy would be caught if filled into the tube simultaneously.

Another point of improvement would be for the distance sensor to act as a trigger that only needs to be activated once instead of having to constantly hold something in front of it. This is because of the coding loop which has not been fixed due to limited time resources.

For future perspectives, one can think of a candy dispenser that makes use of the installed devices in additional ways. For example, could several different distance sensors be placed according to the different colors. The idea would then be to put your hand for example under a distance sensor named “red” and the device will then keep sorting candies until it will be able to provide you with the wished-for red candy.

Additionally, more colors could be added to the sorting repertoire. Additional colors can be implemented by measuring the respective values and defining the frequency ranges distinctively. Colors that are rather far away from the three used ones should work quite well. In theory, however, almost every color of candy should be able to be realized, especially since within the same type of candies the colors usually are easily distinguishable anyways.

A future perspective could also be to improve the hardware installation of the dispenser in terms of material as well as aesthetics and design. Even though the basic idea of the mechanics behind the sorting is well executed by the device it is rather fragile and does not allow for rough handling which might be expected especially due to its high appeal to children. Furthermore, some of the wiring and devices are outside of the container making it prone to damage and malfunction. In the next step, one could think of a design to hold all its technical devices within a closed container only revealing the contact sensor as well as candy in and output from the exterior. Higher quality material will also lead to improved aesthetics and durability.

9. Conclusion

With this color-sorting candy dispenser I managed to install several different devices according to the proper IT process of designing, implementing, and testing. Once all the devices were installed and running correctly the original, self-developed C + + code was uploaded to an Arduino MEGA 2560 microcontroller. Finally, the dispenser managed to successfully be activated via a VL53L1X Time-of-Flight Distance Sensor, moving candy using a Servo Micro Motor, detecting colors using a TCS3200 Color Sensor, and sorting them correctly by the use of a second Servo Micro Motor. Additionally, WS2811 LED strips were installed as a visual feature. This report holds information about the process of designing, testing, prototyping, building, installing, and discussing future perspectives of a color-sorting candy dispenser.

10. Sources

https://ardustore.dk/produkt/servo-micro-motor-9g-sg90-1-0kg?gclid=CjwKCAjwvdajBhBEEiwAeMh1U_Y_DmwK7L5x4QDei7ynusT1ZGTxU_x6u01QC4yN4ImaKD8z8HxvRBoC4oQQAvD_BwE

https://howtomechatronics.com/how-it-works/how-servo-motors-work-how-to-control-servos-using-arduino/

http://wiki.sunfounder.cc/index.php?title=TCS3200_Color_Sensor_Module

https://www.mouser.com/catalog/specsheets/tcs3200-e11.pdf

https://www.britannica.com/science/color/Physical-and-chemical-causes-of-colour

https://arduinotech.dk/shop/color-sensor-recognition-module-tcs230-tcs3200/

https://www.arduinolibraries.info/libraries/tcs3200

https://www.rapidtables.com/web/color/RGB_Color.html

https://www.youtube.com/watch?v=CPUXxuyd9xw

https://www.youtube.com/watch?v=MwdANEcTiPY&t=1618s

https://www.youtube.com/watch?v=SWBI6_rxmT8&t=1s

https://www.youtube.com/watch?v=dBtBj434VEk

https://www.youtube.com/watch?v=pEQDG7iNwKY&t=296s

https://roboticsbackend.com/arduino-pulsein-function/

https://randomnerdtutorials.com/arduino-color-sensor-tcs230-tcs3200/

https://ardustore.dk/produkt/ws2811-5050-rgb-led-strip-12v-dc-waterproof

https://www.sdiplight.com/flexible-ws2811-led-strip/

https://www.syncrolight.co.uk/datasheets/WS2811-60%20Specifications.pdf

https://cdn.shopify.com/s/files/1/0174/1800/files/vl53l1x.pdf?3713516543837568345

https://shop.pimoroni.com/products/vl53l1x-breakout?variant=12628497236051

https://makersportal.com/blog/2019/4/10/arduino-vl53l1x-time-of-flight-distance-measurement

https://www.utmel.com/components/vl53l1x-laser-ranging-sensor-datasheet-pinout-and-schematic?id=288

https://www.pololu.com/blog/774/now-available-vl53l1x-library-for-arduino

11. Figures