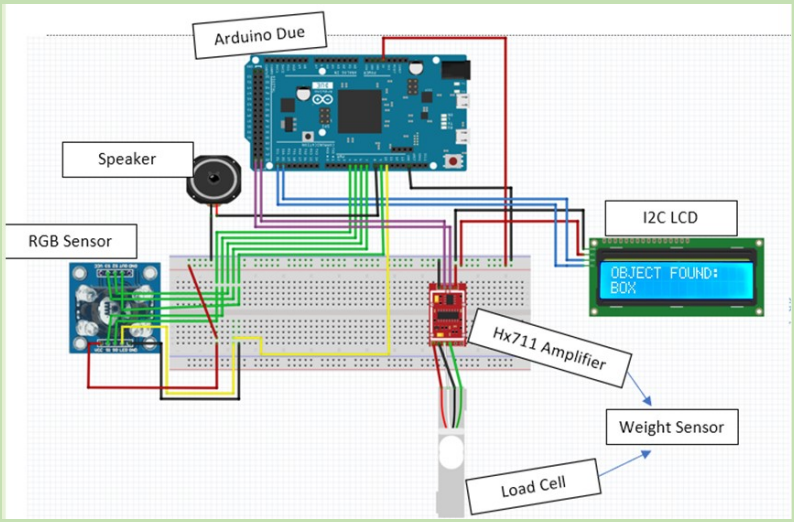
Table of Contents

Object/Property Identifier



ABSTRACT

The primary purpose of my project was to create a system that would identify a class or name of an object. The motivation for the system was focused on aiding individuals with vision disabilities, ex. Color blindness and lack or absence of vision. The final implementation, as of the time of this report, is a system that allows the identification of the weight and color of an object independently. Further, if both color and weight of a specific item are detected, the system will recognize the name of the object. In the case that color, weight, or name of an object is recognized, the system will display each of these properties visually and audibly. The visual aspect is displayed through an LCD screen, while the audio is outputted through a simple speaker. It is important to note the speaker uses morse code for the output, not a spoken language. For example, rather than the system saying "red", the speaker would use a dot/dash system ".-. . -..", where the audio of the dashes is 3x the length of the dots. This was done using an array of all of the characters in the alphabet, in conjunction with the tone function which was implemented through open-source forms. The calibration and connection of both the weight and color sensor were done through open-source forms. The connection and setup of the LCD were also done through open sources.

Continuation of this project is a viable possibility, as there are numerous ways to improve the current iteration of the system. Optimization, the inclusion of multiple modes, and Bluetooth compatibility are all things that can improve the current iteration of the system. Further, the system can be modified to create a system more focused on a variety of areas such as kids learning.

INTRODUCTION

The primary motivation for this project was to create a system that would help improve the daily life of an individual with a disability. There were many possibilities, but the final decision was to create a multi-purpose system that would aid in identifying objects and their respective properties, i.e., an object identifier.

Initially, the incorporation of ML learning into the system to help improve the accuracy and robustness of the system was considered, but implementation proved to be far too complex. Hence, to accomplish the goal, 2 sensors and actuators were used. The two sensors include the RBG and weight sensors; The two actuators include the speaker and an LCD.

To aid in the understanding of the components many different resources were used. In terms of the color sensor, DroneBot Workshop and Denjan, from How to Mechatronics, both provided excellent resources explaining the mechanics and functionality of the RGB sensor. They also explained the steps necessary to implement the RGB sensor. As for the weight sensor, Indrek Luuk and Michael Schoeffler provided great resources explaining the different components of the weight sensors. They also provided information on how to combine the different components, amplifier & loadcell, and the requirements for the module of the weight sensor. Using these resources, along with multiple other resources, the understanding and implementation of each of the different components were clear.

The potential market of this system is relatively high as the system is flexible, in terms of the wide variety of applications it can fill. For example, the system could be used as a learning system for children or a specific color detection system for painters or artists. Though the implication that the system would be of less use in aiding individuals with visual disabilities should not be considered as, through multiple iterations of the system, a system that allows all these applications to co-exist within one system is possible. This could be done using a mobile application that connects to the system, or the use of multiple modes through the implementation of a button(s).

There are similar products that have similar functionality to the O/P I identifier. For example, the Nix color sensor is a product that allows for specific and accurate detection of an array of colors. But like many other products in the market, it serves a very specific purpose. Relatively, the O/P Identifier has a much wider range of applications. But this is also the greatest disadvantage when it comes to the implementation of such a general system. The primary challenge of this project is cohesiveness. Because the system serves a more general purpose it creates more issues and bugs when programming the functionality. This is the biggest challenge holding back the O/P identifier publication into the market, the difficultly and possibly high cost in developing such a robust system without tools such as Machine Learning

I. PROPOSED PROJECT

A. *A. Program*

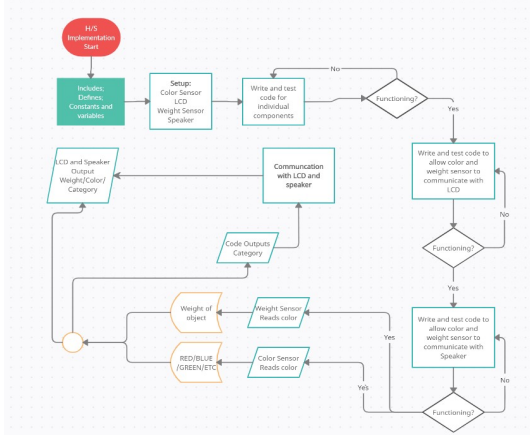


Figure 1. *Flowchart of software implementation*

The process the program follows is relatively simple. Initially, the global variables and handlers are defined. Next, the program runs through the setup function. This portion initializes the different components of the system for use. Next is the main loop of the program. This loop is what the system uses to detect any changes in the values of the color or weight sensor. Initially, the system shows a “UNKWN” for color and zero for the weight. The system waits for the detection of a specific color or the detection of a specific color and weight combination. In the former case, the system will print the color of the object on the LCD and then output the name of the color in morse code through the speaker. The color detected will stay on the screen until a new color or object is detected. In the latter case, when a specific color and weight value is detected, the system will follow the same steps it would for color detection. The only difference that the LCD will update the display on the LCD to only show the current name of the object. Once the speaker finishes outputting the name of the object, the LCD will reset to the initial state.

```
// libaries used
// Tone library was directly implemented
#include <HX711_ADC.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <stdio.h>
#include <string.h>

//List of letters to use with Tone library
int letters[ ][4] =
{{6,2,0,0}, //a
{2,6,6,6},  //b
{2,6,2,6},  //c
{2,6,6,0},  //d
{6,0,0,0},  //e
{6,6,2,6},  //f
{2,2,6,0},  //g
{6,6,6,6},  //h
{6,6,0,0},  //i
{6,2,2,2},  //j
{2,6,2,0},  //k
{6,2,6,6},  //l
{2,2,0,0},  //m
{2,6,0,0},  //n
{2,2,2,0},  //o
{6,2,2,6},  //p
{2,2,6,2},  //q
{6,2,6,0},  //r
{6,6,6,0},  //s
{2,0,0,0},  //t
{6,6,2,0},  //u
{6,6,6,2},  //v
{6,2,2,0},  //w
{2,6,6,2},  //x
{2,6,2,2},  //y
{2,2,6,6}}; //z

String color = "UNKWN"; // Default

// defines ports needed for color
#define S0 4
#define S1 5
#define S2 6
#define S3 7
#define sensorOut 9
#define LED 10

// calibrated frequencies for each

int red = 135;
int blue = 200;
int green = 150;//

float weight = 0.00;
```

Figure 2*. Libraries, handlers, and global variables*

1) *Setup*

```
void colors()
{
  //set photodiodes to read red
  digitalWrite(S2, LOW);
  digitalWrite(S3, LOW);
  //count OUT, pRed, RED
  red = pulseIn(sensorOut, digitalRead(sensorOut) == HIGH ? LOW : HIGH);

  //set photodiodes to read blue
  digitalWrite(S3, HIGH);
  //count OUT, pBLUE, BLUE
  blue = pulseIn(sensorOut, digitalRead(sensorOut) == HIGH ? LOW : HIGH);

  //set photodioes to read blue
  digitalWrite(S2, HIGH);
  //count OUT, pGreen, GREEN
  green = pulseIn(sensorOut, digitalRead(sensorOut) == HIGH ? LOW : HIGH);
}
```

Figure 3. *Libraries, handlers, and global variables*

2) *Main Loop*

```
//If statements to determine the color the sensor detects
if (red < blue && red < green && red < 300 && red > 100)
{
  setColor("RED");
}

else if (blue < red && blue < green && blue < 300 && blue > 100)
{
  setColor("BLUE");
}

else if (green < red && green < blue && green < 300 && green > 100)
{
  setColor("GREEN");
}
else if(green && blue && red < 80){
  setColor("WHITE");
}
else if(green > 400 && blue > 400 && red > 400 && green < 2000 && blue < 2000 && red < 2000){
  setColor("BLACK");
}
else{
  lcd.print(color);
  lcd.print("     ");
  lcd.print(weight);
}
delay(250);
```

Figure 4. *Series of if statements that check for a recognition of a specific color*

The main loop is the loop that the system is continually running, hence it contains the main logic of the program. It checks for any detection in color, weight, or a combination of the two. As seen in figure 4, there are multiple if statements

that continuously check for specific intensities of red, blue, and green colors, which the color sensor detects. When any of these statements are satisfied, the object will call the ***setColor()*** function. If no color has been identified the system will call the else statement, displaying "UNKWN" or the previous color detected on the LCD. The LCD will also show any changes in the values of weight. The Arduino receives this data from the weight sensor.

3) *Main Loop Cont'd*

```
95  LoadCell.update(); // retrieves data from the load cell
96  weight = LoadCell.getData(); // get output value
97
98  //If  to statements decide if weight and color match a specific object
99  if(weight > 97 && weight <99.9 && color == "BLUE" ){
100   object("CARDS");
101 }
102 else if(weight > 31.4 && weight < 32 && color == "BLACK"){
103   object("BOX");
104 }
105 else if(weight > 23 && weight < 25 && color == "WHITE"){
106   object("ERASER");
107 }
```

Figure 5. *Series of if statements that check for a specific object*

**figure does not show all statements*

A As seen in figure 5, the weight of the load cell is continuously updated using the ***LoadCell.Update()*** function and stored in the global variable weight. Further, within the main loop, there are a series of if statements that continuously check for a specific object. If a specific color and weight combination is reached the program will call the ***object()*** function. If not, nothing occurs. The reason there is a range for the weight within the if statements is due to the high sensitivity of the weight sensor. This high sensitivity can cause inconsistent results when measuring weight. This range allows improves practical usage of the weight sensor.

4) *Functions*

```
140 void object(char* object){ // Displays name of object visually and audiably when detected
141   lcd.clear();
142   lcd.setCursor(0,0);
143   lcd.print("OBJECT FOUND:");
144   lcd.setCursor(0,1);
145   lcd.print(object);
146   morseOut(object);
147   LoadCell.update();
148 }
149
```

Figure 6. *object() function*

Within the program there are multiple functions, each follows a sequence of logic when called anywhere in the code. In figure 6, the objective function can be seen. This function takes in a char* (pointer) input named *"object"* as the input. When called, it clears the LCD, sets the cursor at the top left of the LCD, and prints "object found", using the lcd.print()[1] function. It then prints "OBJECT FOUND" and sets the cursor to the bottom left of the LCD and prints the name of the object, which is taken from the input the ***object()*** function takes. It then calls the ***morseOut()*** function and updates the load cell.

```
150 void setColor(char* col){ //sets global color varible and outputs color/weight on LCD
151   color = col;
152   lcd.print(col);
153   lcd.print("     ");
154   lcd.print(weight);
155   morseOut(col);
156 }
```

Figure 7. *setColor() function*

In figure 7 the ***setColor()*** function, seen in figure 6, takes a char* (pointer) input named "*col*". When it is called it sets the global variable color to the input *"col"*. Then using the lcd.print() function it prints that color and weight on the LCD. Lastly the morseOut() function is called.

```
214 void colors()
215 {
216   //set photodiodes to read red
217   digitalWrite(S2, LOW);
218   digitalWrite(S3, LOW);
219   //count OUT, pRed, RED
220   red = pulseIn(sensorOut, digitalRead(sensorOut) == HIGH ? LOW : HIGH);
221
222   //set photodiodes to read blue
223   digitalWrite(S3, HIGH);
224   //count OUT, pBLUE, BLUE
225   blue = pulseIn(sensorOut, digitalRead(sensorOut) == HIGH ? LOW : HIGH);
226
227   //set photodioes to read blue
228   digitalWrite(S2, HIGH);
229   //count OUT, pGreen, GREEN
230   green = pulseIn(sensorOut, digitalRead(sensorOut) == HIGH ? LOW : HIGH);
231 }
```

Figure 8. *colors() function [1]*

S2	S3	Photodiode Type
L	L	Red
H	H	Green
L	H	Blue
H	L	Clear (no filter)

Table 1. The logic levels for the different photodiode types

The ***colors()*** function, seen in figure 8, is called in the main loop. It sets the S2 and S3 pins on the color sensor, which in turn determine the photodiode type on the color sensor. The photodiode type on the color sensor is what allows the color sensor to read the intensity of that specific color. For example, when S2 and S3 are both set to logic level low, the photodiode type on the color sensor is red. This is what sets the color sensor to read the intensity of red light. The other combinations can be seen above in Table 1. Within the function the initial photodiode type is set to red. This value is then stored in the global variable *"red"*. The same logic is then followed for both blue and green.

```
// Uses tone function to loop through given array of characters (input) to output the given input in morse code
void morseOut(char* input){
  for(int j = 0; j < strlen(input); j++){
      for(int i = 0; i < 4; i++){
    if(letters[(int)input[j] - 65][i] == 0){
      continue;
    }
    int noteDuration = 1000/letters[(int)input[j] - 65][i];
    tone(8, 262 , noteDuration);

    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    noTone(10);
  }
  delay(100);
  }
```

Figure 8. *morseOut() function*

The final function, ***morseOut()***, as seen in figure 8, is used to output morse code through the speaker. It takes a char* input named *"input"*. The function then loops through the array of characters given to the function called, e.x. ***morseOut(arrOfChar).*** Within the initial for loop a second loop is called, which loops through with a length of <4, as the maximum tones within a single alphabet are 4. If the value of the letter array is equal to 0, or no tone should be played, then the function will *"continue"* skipping the rest of the logic in the for loop and continuing to the next iteration. If the value within the letter array is not equal to 0, then the duration of the note is calculated and set in the variable "*noteDuration"*. The duration is either a long note, 6, or a short note, 2. This follows the more code convention as a long note is 3 times that of the short note. Once the "*noteDuration"* has been set the tone will play using the ***tone()*** function. The pitch will stay consistent as there is no fluctuation in pitch in morse code. The function will then calculate and set a variable "*pauseBetweenNotes"*. Immediate after, the ***delay()*** function is called, which will delay the next iteration of the letter. Once the first letter has been output through the speaker, the next iteration of the first for loop will be called. The same logic will then apply for the next letter, then the next, until no more letters remain in the array of character. The delay(100), causes a pause between each letter.

B. A. Hardware

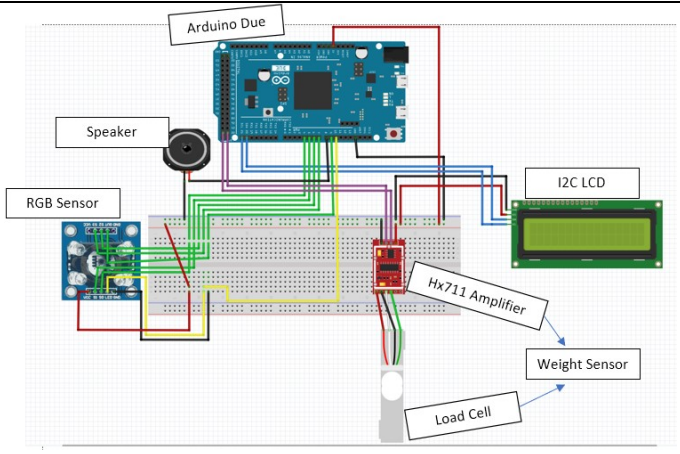


Figure 9. *Hardware diagram of project*

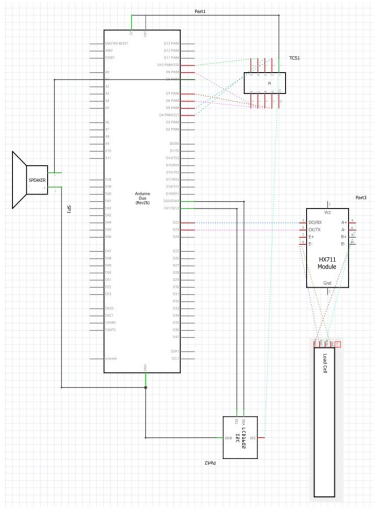


Figure 10. *Schematic of project*

The system consists of 5 total components, as seen in figures 9 and 10. All the components are connected to the Arduino Due. This connection is what allows the data the sensors receive to be read by the Arduino, which is then used to prompt the actuators to do certain tasks. The connections are relatively simple and will be described below.

Figure 11. *RGB color sensor*

1) *RGB color sensor: Connections*

The RGB color sensor can be seen in figure 11. Pins S0-S3 connect to ports 4-7 on the Arduino, respectively. The S1 and S2 pins determine the scaling of the color sensor. The S2 and S3 pins determine the Photodiode Type, as explained in the Program portion of this report. The OUT port connects to port 9. This pin is what outputs the data from the sensor to the Arduino. The LED pin on the sensor connects to port 10 on the Arduino and is responsible for turning on the LED on the color sensor. This helps the color sensor get a clearer and consistent read of the light intensities it is sensing. The VCC is connected to the 5V on the Arduino. It is what powers the color sensor. The GND and OE connect to the GND port on the Arduino. This completes the circuit.

2) *RGB color sensor: Functionality*

The primary purpose of the color sensor is to detect a color change, or more specifically the change in light intensities of the RGB colors it detects from what is in front of its photodiodes. The photodiodes are what sense the light intensities of different objects. Based on the different current it detects, it outputs a square wave, which is converted into frequency. The square wave is proportional to light intensity. This intensity is read by the Arduino.

3) *Load Cell and Hx711 Amplifier: Connections*

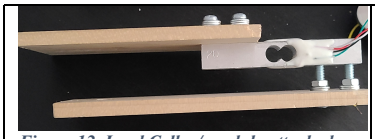


Figure 12. Load Cell w/ module attached

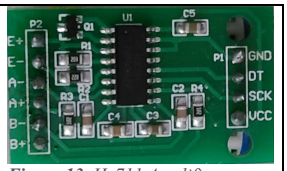


Figure 13. Hx711 Amplifier

The load cell, seen in figure 12, is connected to the amplifier through 4 wires. The red, black, green and white wires connect to pins E+, E-, A-, A+ on the Hx711 amplifier, seen in figure 12, respectively. This connection, along with the module made for the load cell, is what makes up the weight sensor. The GND on the Hx711 connects to GND on the Arduino. The DT and SCK connect to ports 22 and 23 on the Arduino respectively. The VCC is the power for the load cell, hence it connects to the 5V on the Arduino.

4) *Load Cell and Hx711 Amplifier: Functionality*

The primary function of the load cell is to measure the weight of an object. The load cell consists of 4 flexible resistors, known as strain gauges, that are in a Wheatstone bridge configuration. When an object is placed on top of the load cell, this causes the cell to deform from its original state. This causes a change, in turn, causes the voltage to change in the wheat Stonebridge configuration. This change in voltage is amplified by the hx711 amplifier. The proportionality between this stress and voltage change, which is used to calibrate weight. In this project, the calibration was done using the Hx711 library.

5) *Speaker and LCD: Connections*

Figure 14. Hx711 Amplifier

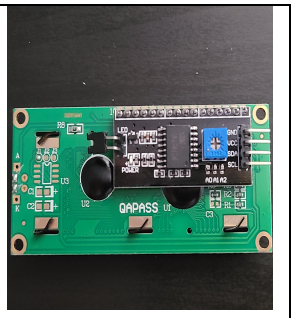


Figure 15. i2c LCD

The Speaker seen in figure 14, has two connections, the negative end and positive. The positive end connects to port 8 on the speaker and the negative to the GND port on the Arduino. The LCD, seen in figure 15, has 4 connections. The VCC connects to the 5V on the Arduino and the GND also connects to the GND port on the Arduino. As for the SDA and SCL on the LCD, the port connects to the SDA and SCL port on the Arduino respectively.

6) *Speaker and LCD: Functionality*

The main purpose of the speaker is to output the color or name of an object when detected. This is achieved through the tone library as the hardware connections for the speaker are very simple. The main purpose of the LCD is to display the color, weight, or name of an object. It utilizes the Liquid Crystal library to achieve this, but it receives its data from the SDA (serial data) and SCL connections.

II. RESULTS

A. Development Process

The first step in the development of the project was to create a setup and test each of the components individually. First, the color sensor was set up and tested using the serial monitor. Once the calibration and setup were complete for the color sensor, the next component to set up was the LCD. When both components were functional, they were used together to create a cohesive system. Once the color sensor and weight sensor functioned in a small-scale system, the speakers were added. The inclusion of the speaker into the system was much easier than both the LCD and color sensor as there were only two wires to connect. There was no setup for the speaker, but the implementation of the tone library was necessary as the official tone library is not supported on the Arduino Due. Again, a cohesive mini system was created with the color sensor, LCD, speaker. The functionality of the system was limited to detecting the color of an object, then outputting that color to the LCD and speaker. The last component that needed setup and testing was the loadcell, but this sensor, as stated before, is comprised of two components, the load cell, and the hx711 amplifier. Further, the load cell required a specific module that required physical construction. The construction of the module was done using wooden planks, screws, and washers. The key idea was to leave space in between the board and the load cell to allow the cell to deform. As explained earlier, this deformation is necessary to measure weight. Once the load cell was set up, the pins required to connect the hx711 amplifier to the circuit board were soldered on. The next step was the connect the load cell to the amplifier, which was then connected to the Arduino. The weight sensor was then calibrated and set up using the LCD, and finally implemented into the system. At this step the hardware of the system was complete. The next step was the implementation of the software, which was previously explained in the proposed project section.

1) *Functionality*

Below you will find figures showcasing the functionally of the current iteration of this project. It is important to note that the audio functionality will not be shown through this report.

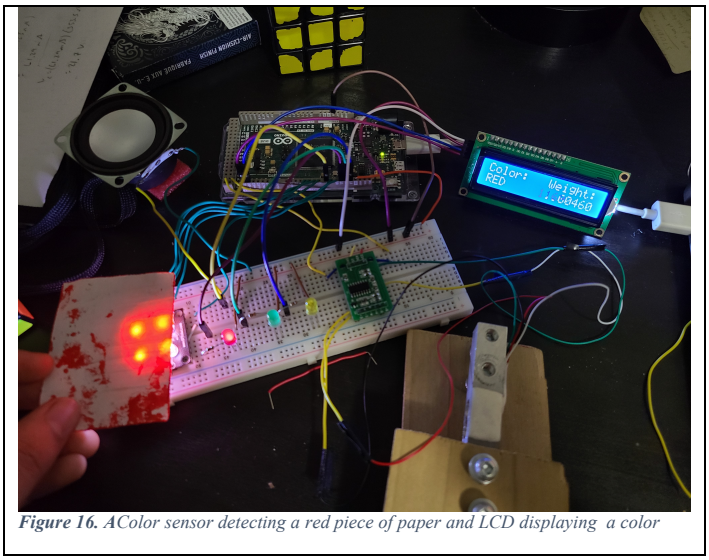


Figure 16. A*Color sensor detecting a red piece of paper and LCD displaying a color*

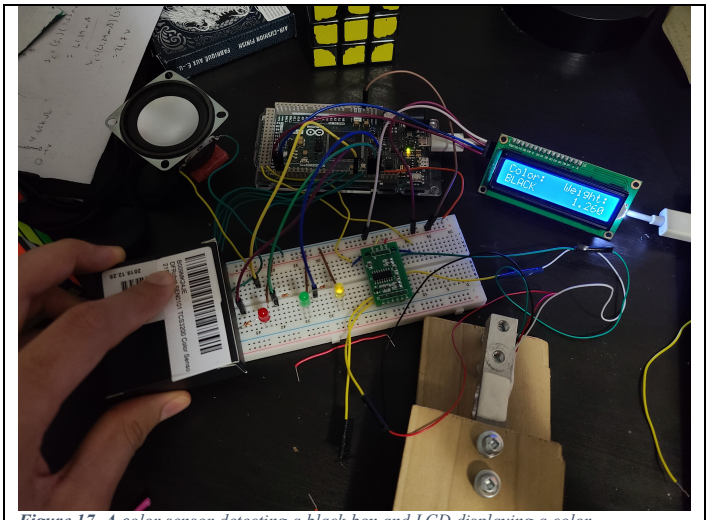


Figure 17. A color sensor detecting a black box and LCD displaying a color

2) *Functionality Cont'd*

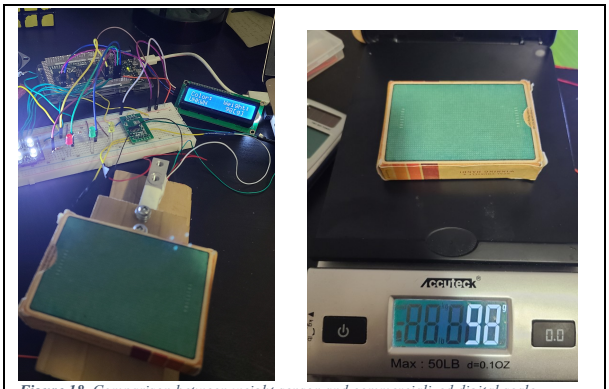


Figure 18. *Comparison between weight sensor and commercialized digital scale*

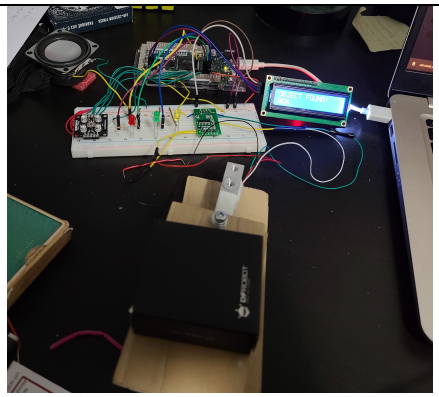


Figure 19. *Specific name/category detected and LCD displaying the name*

III. Discussion

Many technical issues occurred throughout the development process of this project, but one of the most notable issues occurred with the color sensor. The goal was to create a dynamic system; hence the use of buttons or interrupts was rejected. But this caused the color sensor to read the objects too quickly, resulting in the detection of the incorrect color, in most cases black. The solution I implemented was a simple delay. This would allow more time for the object to approach the color sensor, greatly reducing incorrect readings. A downside to this fix was the speed decrease in weight readings, i.e. it would take a noticeably longer time for the weight sensor to update the weight and the LCD to display the value. This downside of course is must less impactful when compared to incorrect readings.

Another issue was the lack of library support for the Arduino Due. The use of certain libraries such as PCM, a library that would allow me to output voice through the speaker with no external storage capacity was incompatible, forcing alternative implementations. This lack of support was what drove the audio output to change from a spoken language to morse code. Beyond that, the Arduino Due lacked compatibility with the tone library. This resulted in the manual implantation of the tone library through the use of public forms.

Finally, the last major issue occurred with the intrinsic nature of the color sensor. The color sensor was greatly sensitive to light from any source, causing incorrect readings. The solution to this was to keep a consistent light setting. To do this, a re-calibration was done in a room with the curtains closed and lights always on. This eliminated this erratic behavior of the color sensor.

IV. Conclusion

The main objective of this project was to create a system that could identify object names and their properties, then visually and audibly output that data. In terms of the project proposal, everything initially promised was implemented. There was no secret to this "success", as many sacrifices were made, and alternatives implemented. The two biggest factors that caused this were the lack of support and the cost of the project.

First, the lack of support refers to the in-person help students were unable to receive. Although it is possible to ask questions through email, it is not nearly as effective as having the ability to receive in-person help from the professor and TAs. Secondly, the factor of cost greatly inhibited the scope of the project. Every component implemented would affect the finical well-being of the student, limited the variety of components that could be used in a system. Because of these reasons, the current implementation of the system is lacking and greatly flawed in terms of practicality or commercialization. As previously discussed in the introduction, many features can be implemented and optimizations can be made. Nevertheless, the current system is far from ideal but has great potential.

Despite the conditions, I learned a great deal about embedded systems. Through this project, I have learned about the SDLC of an embedded system. Difficult ideas to grasp, such as the communication between the sensors, actuators, and microcontroller have become much clearer. In terms of software skills, my C/C++ knowledge has also improved.

All this knowledge is valuable as I may find myself working in a field requiring knowledge developing an embedded system, or more specific aspects such as the functionality of a color sensor. In terms of the near future, this knowledge will assist in any personal project I will tackle.

V. Acknowledgments

I would like to express my thanks to Professor Ebrahim Ghafar-Zadeh and his TA Fayas Ahamed Mohamed Hasan for answering any questions I had along the way.

VI. References

[1] A. Newton, "RGB Color Detector using TCS3200 Color Sensor & Arduino," *How To Electronics*, 27-Feb-2021. [Online]. Available: https://how2electronics.com/rgb-color-detector-tcs3200-color-sensor-arduino/. [Accessed: 29-Apr-2021].

[2] D. Nedelkovski , "Arduino Color Sensing Tutorial - TCS230 TCS3200 Color Sensor," *HowToMechatronics*, 05-Feb-2021. [Online]. Available: https://howtomechatronics.com/tutorials/arduino/arduino-color-sensing-tutorial-tcs230-tcs3200-color-sensor/. [Accessed: 29-Apr-2021].

[2] Fmalpartida, "fmalpartida/New-LiquidCrystal," *GitHub*, 2020. [Online]. Available: https://github.com/fmalpartida/New-LiquidCrystal. [Accessed: 29-Apr-2021].

[4] I. Luuk, "4-Wire Load Cell (1/5/10/20/200kg) with HX711 and Arduino," *4-Wire Load Cell (1/5/10/20/200kg) with HX711 and Arduino - Circuit Journal*, 15-Jun-2020. [Online]. Available: https://circuitjournal.com/four-wire-load-cell-with-HX711. [Accessed: 29-Apr-2021].

[5] Mantoui, "Arduino Due and tone()," *Arduino Forum*, 11-Dec-2012. [Online]. Available: https://forum.arduino.cc/t/arduino-due-and-tone/133302/3. [Accessed: 29-Apr-2021].

[6] M. Schoeffler, "Arduino Tutorial: HX711 Load Cell Amplifier (Weight Sensor Module) + LCM1602 IIC V1 LCD," *Michael Schoeffler*, 06-Jan-2021. [Online]. Available: https://mschoeffler.com/2017/12/04/arduino-tutorial-hx711-load-cell-amplifier-weight-sensor-module-lcm1602-iic-v1-lcd/. [Accessed: 29-Apr-2021].

[7] O. Kallhovd, "HX711_ADC," *GitHub*, 20-Feb-2021. [Online]. Available: https://github.com/olkal/HX711_ADC. [Accessed: 29-Apr-2021].

Laxit Shahi is a 3rd-year student at York University, Toronto, Ontario, currently pursuing a BEng in Computer Engineering. He has an avid interest in both hardware and software development. He hopes to gain enough knowledge and experience throughout his time in university to impact the world.

Terms—

Class: a category of things with similar properties or attributes

RGB: Red, Green, Blue

ML: Machine Learning

SDLC: System Development life cycle