Color Based: Maturity Identification of Citrullus Lanatus using Arduino

By

**Aquino, Jessa C.**

**De Guzman, J.R. T.**

**Javier, Adrian Walter C.**

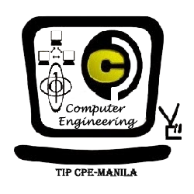
**Pacate, J-C James C.**

**Sanchez, John Robie A.**

Technological Institute of the Philippines

A Final Documentation Submitted to the Department of Computer Engineering, in Partial Fulfillment of the Requirements of Microprocessor Systems Course

September 2019

******TECHNOLOGICAL INSTITUTE OF THE PHILIPPINES**

**1338 Arlegui St., Quiapo, Manila**

**DEPARTMENT OF COMPUTER ENGINEERING**

**APPROVAL SHEET**

The proposed project entitled **Color Based: Maturity Identification of Citrullus Lanatus using Arduino** which was presented on **Date of Presentation** by the proponents:

**Aquino, Jessa C.**

**De Guzman, J.R. T.**

**Javier, Adrian Walter C.**

**Pacate, J-C James C.**

**Sanchez, John Robie A.**

is hereby APPROVED.

|  |
| --- |
| **Engr. Rufo I. Marasigan Jr.** |
| Class Adviser |

**Table of Contents**

[**Acknowledgement** 4](#_Toc22297248)

[**Abstract** 5](#_Toc22297249)

[**Chapter 1** 6](#_Toc22297250)

[**Introduction** 6](#_Toc22297251)

[**Chapter 2** 8](#_Toc22297252)

[**Review of Related Literature** 8](#_Toc22297253)

[**Chapter 3** 10](#_Toc22297254)

[**Methodology** 10](#_Toc22297255)

[**Chapter 4** 24](#_Toc22297256)

[**Project Details** 24](#_Toc22297257)

[**A.** **System Architecture** 24](#_Toc22297258)

[**B.** **Software** 25](#_Toc22297259)

[**C.** **Hardware** 42](#_Toc22297260)

[**Chapter 5** 48](#_Toc22297261)

[**Testing’s, Results, & Interpretations** 48](#_Toc22297262)

[**Chapter 6** 51](#_Toc22297263)

[**Conclusions and Recommendations** 51](#_Toc22297264)

# **Acknowledgement**

This project would not have been possible without the support of many people. We would also like to convey our thanks to our class adviser Engr. Rufo Marasigan Jr. for providing valuable information as the guidance of the project. Most importantly, none of this could have happened without our support from our family. Our parents, who offered their encouragement– despite our own limited devotion to correspondence. Every time we were ready to quit, you did not let us down and we are forever grateful. This dissertation stands as a testament to your unconditional love and encouragement.

For this achievement, I give back all the glory and praises to the omnipotent **Father Almighty.**

Aquino, Jessa C.

De Guzman, J.R. T.

Javier, Adrian Walter C.

Pacate, J-C James C.

Sanchez, John Robie A.

# **Abstract**

This study proposed a method to determine the watermelon’s rind for categorization of maturity level. In this study, we’re going to use an Arduino Microcontroller and a color sensor to determine the RGB value of the watermelon. The main purpose of this study is to determine ripeness of the watermelon base on it’s rind color. The scope of this study are: (1) The system will detect the best spot of the watermelon ripeness identification; (2) The system will idenify whether the watermelon is ripe or unripe based on it’s rind color. In this study we used the following components to finish up the project, an Arduino Mega Microcontroller, TCS3200 color sensor, stepper motor, L298N Dual H-Bridge Driver Motor Module, keypad, and LCD. TCS3200 color sensor is used to detect the rind color of watermelon and a L298N H-Bridge Driver is used to control the stepper motor.

**Keywords:**  Arduino Microcontroller, RGB, TCS3200 color sensor, stepper motor, L298N Dual H-Bridge Driver Motor Module, watermelon ripeness identification.

# **Chapter 1**

## **Introduction**

Watermelon or Citrullus Lanatus was found growing wild by Livingstone in 1854 (John, 2003). Generally, watermelons are round in shape. However, most of them are not perfectly round. Moreover, the rind color distributions are also not uniform. It is hard to find two watermelons which have similar uniformity in the rind color. In order to establish a parameter of the rind for ripeness categorization, it is crucial to find the best spot or position of capturing that can provide the most reliable anddistinguishable features between each ripeness stage. Watermelons need to be ripe when harvested. Watermelon typically has 10-11% soluble solids when ripe. Because individual fruits are pollinated at different times, multiple harvests are usually necessary (Syazwan, 2012).

Monitoring and controlling ripeness is becoming a very important issue in the fruit industry since the state of ripeness during harvest, storage, and market distribution determines the quality of the final product measured in terms of customer satisfaction. Many methods to monitor the ripeness of fruit have already been proposed (Correig, 2000). There’s three ways to determine the ripeness of the watermelon, the first one is base on it’s round shape; second, is searching for it’s yellow spot; third, is thumping the skin of watermelon.

The different factors to consider in determining the quality of a good watermelon are the ripeness, maturity and sweetness to name a few. These are commonly tested by human experience. (Sun et. al., 2010) In considering the ripeness and maturity, the best fruits possess the following characteristics: the tendril (the cup-shaped leaf) is dried out, weight should feel heavy for its size, color has creamy yellow splotch under the fruit and the thumping sound must be hollow. (Christensen, 2012).

There are some problem with the existing studies and these are:: (1) They are using manual positioning of watermelon; (2) Hard to detect the best spot of watermelon ripeness identification. The objectives of this study are: (1) To determine the ripeness of watermelon base on it’s rind color; (2) develop a system that will detect the best spot of the watermelon ripeness identification; (3) The system will use the RGB value to determine the ripeness of watermelon.

The scope of this study are: (1) The system will detect the best spot of the watermelon ripeness identification; (2) The system will identify whether the watermelon is ripe or unripe based on it’s color. And these are the limitations: (1) The system will only process one green watermelon at a time; (2) The system will only process green watermelon with the diameter of 10 inches; (3) the position of color sensor is steady it cannot be adjusted.

# **Chapter 2**

## **Review of Related Literature**

According to Susanna Buratti (2006) evaluated the capability of a commercial electronic nose based on a 10 sensor array (PEN2, Airsense Analytics Inc.) to classify the fruits belonging to different cultivars and to monitor their ripening stage during shelf life from the day after harvest until fruits were senescent in their research. The performance was compared using the sensorarray with the results of classical and non-destructive techniques such as the evaluation of colour and ethylene production. The electronic nose data are analysed by PCA, LDA and CART analysis and were used to build a mathematical model for the evaluation of the ripening stage.

Further, a study using an artificial olfactory system as a non-destructive instrument to measure fruit ripeness. This system comprised of an array of semiconductor gas sensors as well as data acquisition and analysis components. Based on the study conducted, it has been proven that this electronic nose system is capable of determining fruit ripeness. The sensor array successfully left unique characteristic pattern or fingerprint for each stage of ripeness (Nordiyana, 2005).

A studied on a development of a disposable sensor used to monitor the ripeness process and to investigate the different maturity stages of jackfruit by chemometric treatment. Response of the sensor strip fabricated using screen printing technology was analyzed using Principal Component

Analysis (PCA) and the classification model constructed by means of Canonical Discriminant Analysis (CDA) enable unknown maturity stages of jackfruit to be identified (Sim, 2003).

Further, (J. Brezmes et al, 2000) developed an Electronic Nose for non-destructively for monitoring the fruit ripening process. Based on a tin oxide chemical sensor array and neural network-based pattern recognition techniques, the olfactory

system designed is able to classify fruit samples into three different states of ripeness such as green, ripe and overripe with very good accuracy. Based on the proposed method, classification of peaches and pears attained success rate above 92% while a slightly worse accuracy for classification of apples. An additional feature of the system is its ability to accurately predict the number of days the fruit has been in storage since harvest.

# **Chapter 3**

## **Methodology**

The system is designed for the purpose of the following functions; detecting the rind color of watermelon and identify if it’s ripeness level based on it’s rind color. **Figure 3.1** showed the overall process in determining the ripeness level of watermelon based on it’s rind color.

START

.

Placement of Watermelon on a curved surface

Experimental Setup and Calibration

Press asterisk button to start and the Stepper Motor rotates (this will be done in 8 steps)

Detecting the rind color of Watermelon using color sensor every step

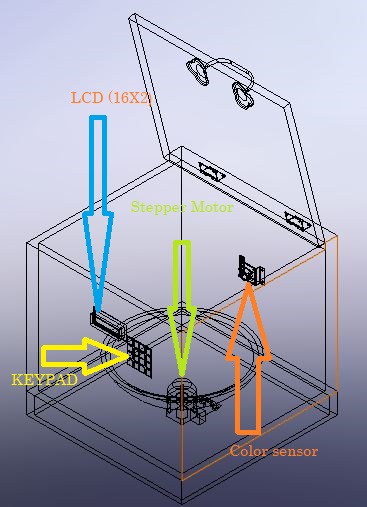
Average of RGB values

Determination of Watermelon’s level of ripeness.

END

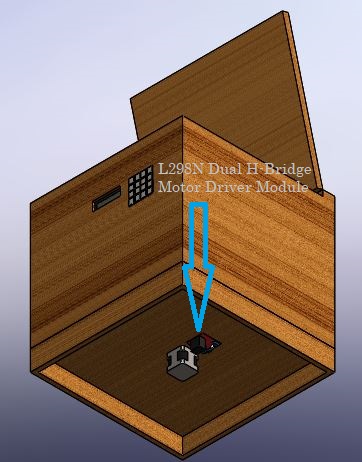
*Figure 3.1: Overall process in determining the ripeness level of watermelon based on it’s rind color.*

*Experimental setup and calibration*

 In this study, a total of 6 samples of watermelons are used. The experimental setup is as shown in Figure 3.1.

*Figure 3.2: Experimental Setup 1*

A cube setup as shown in the Figure 3.1, it consists of color sensor, stepper motor, keypad, LCD, and Arduino Mega. And in Figure 3.2 a L298N Dual H-Bridge Driver Motor Module. The used of L298N Dual H-Bridge Driver Motor is to drive inductive loads that requires forward and reverse function with speed control such as DC Motors, and Stepper Motors. This Dual H-Bridge driver is capable of driving voltages up to 46V and continuous current up to 2A in each channels.



*Figure 3.3: Experimental Setup 2*

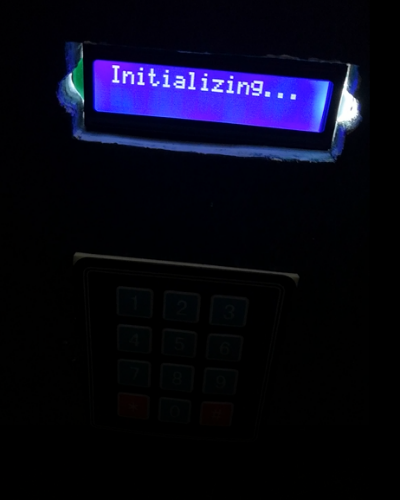
The sensor is a single monolithic CMOS integrated circuit that combines a configurable silicon photodiode and a current-to-frequency converter. In this study we used the TCS3200 color sensor, this TCS3200 is ideal for RGB LED confrol, industrial, and medical diagnostic equipement. This RGB sensor chip and four white LEDs into an easy-to-use, breadboard friendly breakout that can be powered by 2.7 V to 5.5 V.

A stepper motor is a type of DC motors that move in increment or steps, they move at a known interval for each pulse of power. These pulses of power are provided by a motor driver and is referred to as a step. As each step moves a known distance it makes them handy devices for repeatable positioning. In this study the used of stepper motor is to rotate the watermelon inside a container.



*Figure 3.4: Actual Setup*

As shown in **Figure 3.3** the actual setup of the prototype, the watermelon is placed in a bowl or curved surface that makes the watermelon carried, below of the curved surface is the stepper motor which will rotate the watermelon. At the back view is the color sensor; the front view is the LCD and the keypad. The Arduino Microcontroller is placed near the color sensor. The operation of this prototype requires 12V supply to the stepper motor for have to work and a 9V supply for the Arduino Microcontroller.



1. (b) (c)

(d) (e) (f)

*Figure 3.5: Reading the Watermelon’s RGB*

In **Figure 3.4** as you can see the flow of the program on how to read the RGB color of the watermelon, (a) the startup; (b) initializing or reading the RGB color of the watermelon (in this process it has 8 cycle, each cycle have it’s reading); (c) after the reading, the RGB output will display (this process is connected to (b) ); (d) snd (e) after collecting the RGB color in 8 cycle, will compute for its average; (f) after getting the average it will determine the ripeness level of the watermelon if it is ripe or unripe.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S | W1 | | | W2 | | | W3 | | | W4 | | | W5 | | |
|  | Red | Green | Blue | Red | Green | Blue | Red | Green | Blue | Red | Green | Blue | Red | Green | Blue |
| 1 | 210 | 233 | 204 | 233 | 238 | 216 | 227 | 244 | 210 | 199 | 227 | 182 | 210 | 233 | 199 |
| 2 | 216 | 238 | 204 | 244 | 244 | 227 | 227 | 238 | 204 | 204 | 221 | 176 | 210 | 233 | 193 |
| 3 | 210 | 233 | 199 | 283 | 260 | 260 | 221 | 238 | 204 | 199 | 221 | 176 | 204 | 233 | 193 |
| 4 | 210 | 233 | 204 | 277 | 255 | 255 | 221 | 238 | 204 | 199 | 221 | 182 | 209 | 233 | 193 |
| 5 | 210 | 233 | 199 | 260 | 244 | 233 | 216 | 238 | 204 | 199 | 277 | 182 | 204 | 233 | 193 |
| 6 | 244 | 233 | 199 | 277 | 238 | 216 | 216 | 238 | 204 | 199 | 260 | 176 | 204 | 233 | 193 |
| 7 | 210 | 227 | 187 | 227 | 244 | 210 | 221 | 266 | 193 | 199 | 133 | 193 | 204 | 227 | 182 |
| 8 | 210 | 227 | 187 | 210 | 238 | 199 | 216 | 227 | 187 | 204 | 233 | 193 | 199 | 227 | 182 |
|  | Average | | | Average | | | Average | | | Average | | | Average | | |
|  | 215 | 232 | 197 | 252 | 245 | 227 | 220 | 240 | 201 | 200 | 230 | 182 | 204 | 231 | 190 |
|  | R | | | R | | | R | | | UR | | | UR | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S | W6 | | | W7 | | | W8 | | | W9 | | | W10 | | |
|  | Red | Green | Blue | Red | Green | Blue | Red | Green | Blue | Red | Green | Blue | Red | Green | Blue |
| 1 | 210 | 233 | 199 | 204 | 233 | 187 | 210 | 233 | 199 | 210 | 227 | 187 | 233 | 238 | 199 |
| 2 | 210 | 233 | 199 | 204 | 233 | 193 | 210 | 233 | 199 | 204 | 227 | 182 | 221 | 233 | 199 |
| 3 | 210 | 233 | 193 | 204 | 233 | 187 | 210 | 238 | 199 | 210 | 227 | 182 | 221 | 233 | 199 |
| 4 | 210 | 233 | 193 | 199 | 233 | 187 | 210 | 233 | 199 | 204 | 227 | 187 | 211 | 233 | 199 |
| 5 | 210 | 233 | 193 | 204 | 233 | 193 | 210 | 233 | 193 | 204 | 227 | 187 | 216 | 233 | 199 |
| 6 | 210 | 233 | 193 | 204 | 266 | 187 | 210 | 233 | 199 | 204 | 227 | 182 | 210 | 233 | 193 |
| 7 | 210 | 227 | 187 | 204 | 221 | 221 | 210 | 227 | 221 | 204 | 233 | 193 | 204 | 233 | 193 |
| 8 | 204 | 260 | 187 | 204 | 227 | 182 | 210 | 227 | 187 | 204 | 233 | 193 | 204 | 233 | 193 |
|  | Average | | | Average | | | Average | | | Average | | | Average | | |
|  | 209 | 236 | 193 | 203 | 234 | 192 | 205 | 288 | 186 | 210 | 232 | 199 | 214 | 233 | 196 |
|  | UR | | | UR | | | UR | | | UR | | | UR | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| COLOR RANGES | | | | | |
| RIPE | | | **UNRIPE** | | |
| Red | Green | Blue | Red | Green | Blue |
| 210 – 283 | 227 – 266 | 187 – 260 | 199 – 210 | 221 – 277 | 176 – 227 |

*Table 3.1: Gathered Data from Watermelon*

Based on the gathered data in **Table 3.1**, shows different values of RGB for ripeness. Ripe watermelons have highest RGB values of 283, 266, 260 respectively, and lowest RGB values of 210,227,187. For unripe watermelons, RGB values of 210,277,227 are the highest while 199,221,176 are the lowest values. This makes the ranges for ripe and unripe watermelons that can be used in coding for Arduino.

**Procedure on how to develop the system**

1. Preparation of Materials and Assembling



**Figure 3.6 -** Preparing 6 pcs of plywood 15x15 inches diameter



**Figure 3.7 –** Assembling them with nails,handle, and hinges, then forming a box**.**



**Figure 3.8 –** Putting wood glue on edges of the box for stability and durability



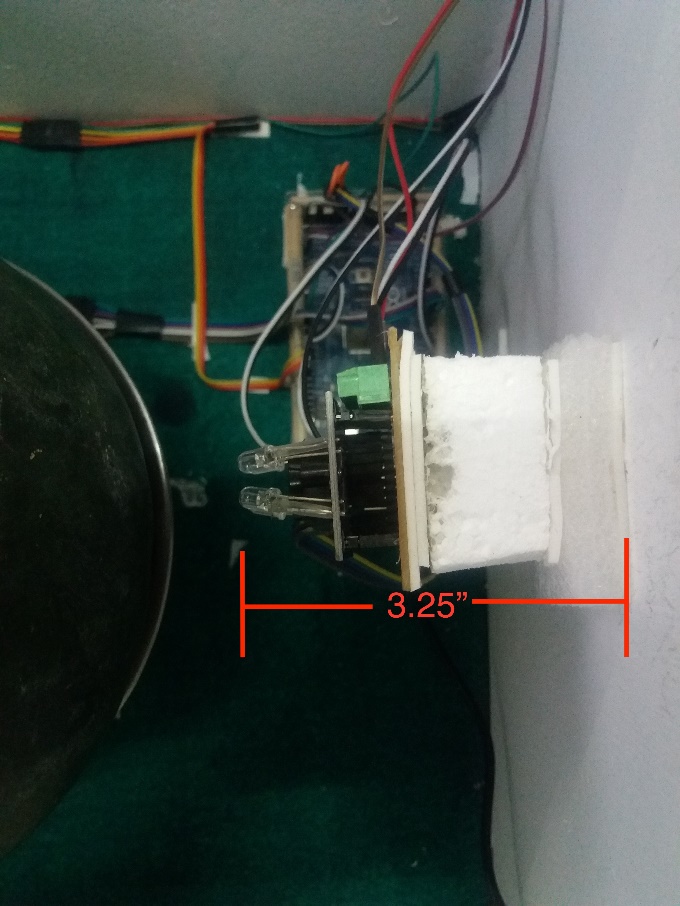
**Figure 3.9 –** Making hole that will fit the LCD and keypad with size of 3x1 inch on LCD and 3x2.75 inches on keypad.



**Figure 3.10 –** Making another hole that will fit the stepper motor with 1.75x1.75 inches.



**Figure 3.11 –** Preparing watermelon mount with 6.75 inches diameter

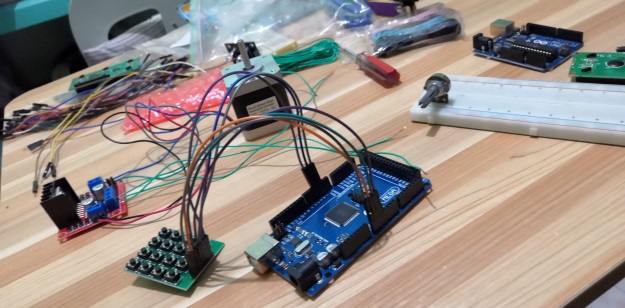


**Figure 3.12 –** Placing the sensor.

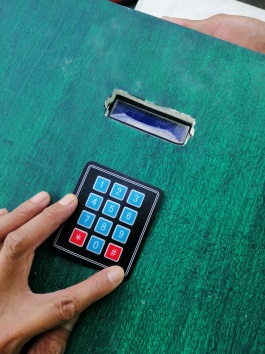


**Figure 3.13 –** Finishing the box with putty and latex paint

1. Setting up circuits and devices.

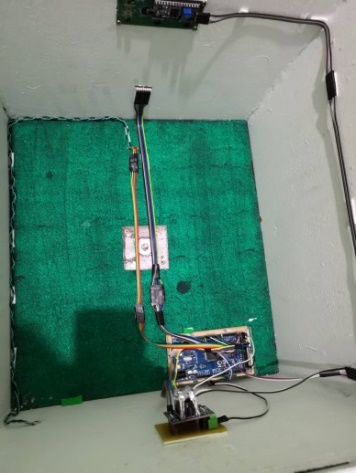


**Figure 3.14 –** Checking of devices and arduino.



**Figure 3.15 –** Putting lcd display and keypad.



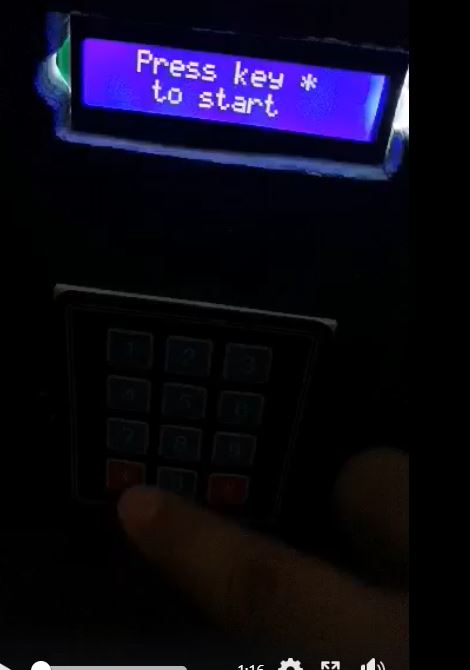
 **Figure 3.16 –** Setting up stepper motor with container.

**Figure 3.17 –** Setting up connections

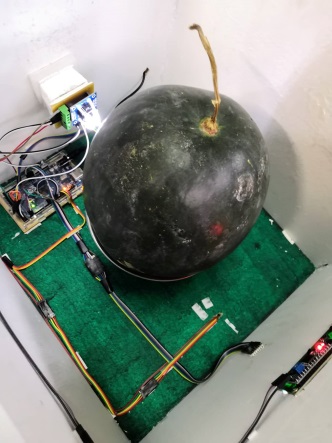
1. Data gathering and analysis



**Figure 3.18 –** Preparing watermelons to be tested.



**Figure 3.19 –** Starting to test watermelons.



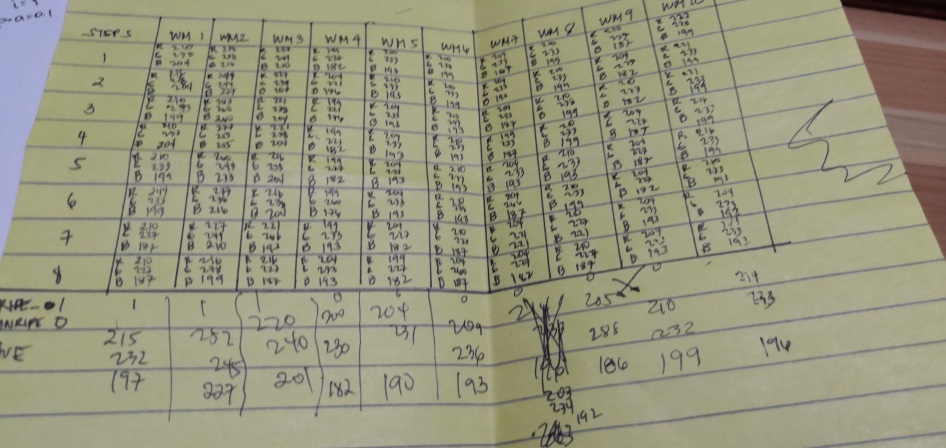
**Figure 3.20 –** Testing of watermelons. The stepper motor rotates in 8 steps.



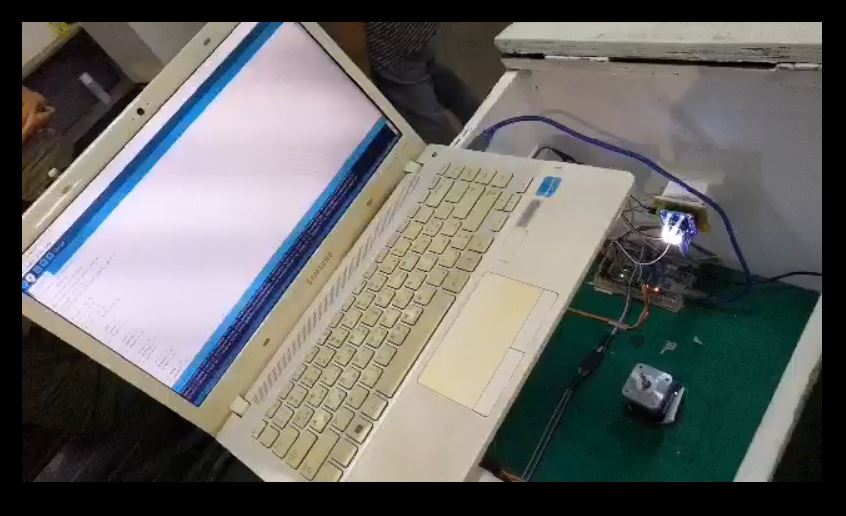
**Figure 3.21 –** Checking of RGB values in the LCD display.



**Figure 3.22 –** Checking watermelons if ripe or not.



**Figure 3.23 –** Tabulating results after testing 10 watermelons.

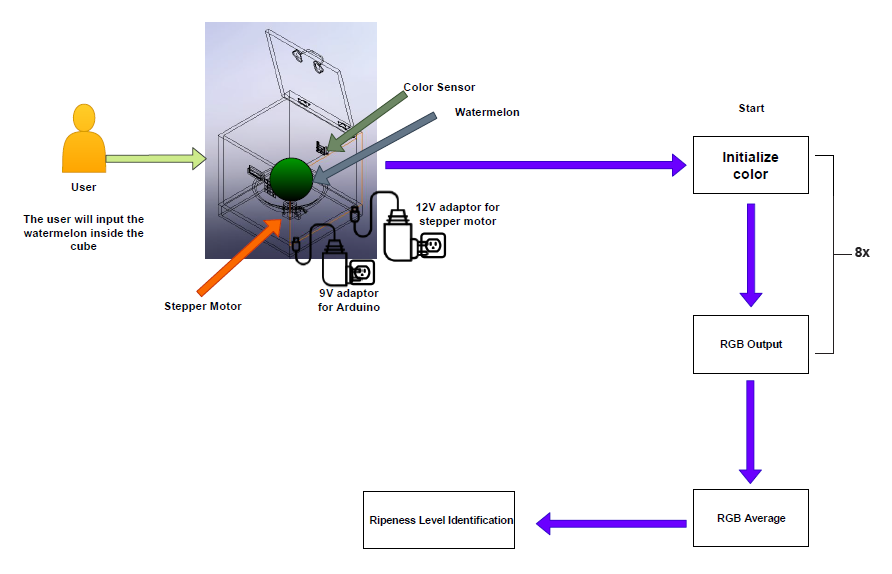


**Figure 3.24 –** Re-coding and re-uploading program in Arduino for final setting.

# **Chapter 4**

## **Project Details**

### **System Architecture**



*Figure 4.1 – System Architecture*

In **Figure 4.1** shows the System Architecture of the project. First, the user will put the watermelon inside the cube, second after putting the watermelon the user will press the asterisk (“\*”) button to start the process. After pressing the button asterisk the stepper motor will start to rotate this will be done in 8 steps. In every step the color sensor will get the RGB data from the watermelon and it will average the gathered data. The average RGB data will be the basis of the output whether the watermelon is ripe or unripe.

### **Software**

#### System Flowchart

START

Experimental Setup and Calibration

Placement of Watermelon on a curved surface

Detecting the rind color of Watermelon using color sensor every step

Determination of Watermelon’s level of ripeness.

Press asterisk button to start and the Stepper Motor rotates (this will be done in 8 steps)

END

Average of RGB values

*Figure 4.2 – System Flowchart*

In **Figure 4.1** it shows the System Flowchart of the project, first is setting-up the prototype on its proper place. Second, the watermelon will placed in a curved surface where the watermelon will rotate. Third, after putting the watermelon in a curved surface the stepper motor will rotate when pressing the asterisk (“\*”) button (this will be done in 8 steps). Fourth, when the stepper motor rotates the color sensor will initialize the watermelon’s color and then will output the corresponding RGB value in every steps. Fifth, the gathered RGB value from every step will average. Sixth, after getting the average RGB value the system will determine whether the wateremelon is ripe or unripe.

#### Source Code

//====================

// SENSOR

This part of the source code is the setting-up the library for LCD (16x2) I2C and defining the pins for S0, S1, S2, S3 and sensorOut of the TCS3200 Color Sensor.

//====================

#include <LiquidCrystal\_I2C.h>

#include <Wire.h>

LiquidCrystal\_I2C lcd (0x3F, 16,2);

#define S0 33

#define S1 2

#define S2 3

#define S3 30

#define sensorOut 32

int red = 0;

int green = 0;

int blue = 0;

int red1 = 0;

int green1 = 0;

int blue1 = 0;

int red2 = 0;

int green2 = 0;

int blue2 = 0;

int red3 = 0;

int green3 = 0;

int blue3 = 0;

int red4 = 0;

int green4 = 0;

int blue4 = 0;

int red5 = 0;

int green5 = 0;

int blue5 = 0;

int red6 = 0;

int green6 = 0;

int blue6 = 0;

int red7 = 0;

int green7 = 0;

int blue7 = 0;

int redAve = 0;

int greenAve = 0;

int blueAve = 0;

Library of stepper motor, defining the following pins and setting an angle of 360 degrees

//====================

// STEPPER MOTOR

//====================

#include <Stepper.h>

const int angle = 360;

Stepper myStepper(angle, 34,9,10,11); //IN1, IN2, IN3, IN4

//====================

// KEYPAD

//====================

#include <Keypad.h>

This part of source code is for the Keypad. Keypad (3x4) will be used here.

const byte ROWS = 4;

const byte COLS = 3;

char keys[ROWS][COLS] = {

{'1','2','3'},

{'4','5','6'},

{'7','8','9'},

{'\*','0','#'}

};

byte rowPins[ROWS] = {A0, A1, A2, A3};

byte colPins[COLS] = {7,6,24};

Keypad keypad = Keypad( makeKeymap(keys), rowPins,

colPins, ROWS, COLS );

void setup() {

myStepper.setSpeed(60);

Serial.begin(9600);

pinMode(S0, OUTPUT);

pinMode(S1, OUTPUT);

pinMode(S2, OUTPUT);

pinMode(S3, OUTPUT);

pinMode(sensorOut, INPUT);

// Setting frequency-scaling to 20%

digitalWrite(S0,HIGH);

digitalWrite(S1,HIGH);

lcd.init(); // initialize the lcd

lcd.init();

// Print a message to the LCD.

lcd.backlight();

lcd.setCursor(3, 0);

lcd.print("Watermelon");

lcd.setCursor(1, 1);

lcd.print("Color Detector");

delay(1000);

lcd.clear();

lcd.setCursor(2,0);

lcd.print("\*\*\* Using \*\*\*");

lcd.setCursor(1,1);

lcd.print(">>> Arduino <<<");

delay(1000);

lcd.clear();

lcd.setCursor(2,0);

lcd.print("Press key \* ");

lcd.setCursor(3,1);

lcd.print("to start");

Serial.begin(9600);

}

void loop() {

char key = keypad.getKey();

if (key == '\*')

{

delay(1000);

myStepper.step(angle / 4);

RGB();

delay(1000);

myStepper.step(angle / 3.6);

RGB1();

delay(1000);

myStepper.step(angle / 3);

RGB2();

delay(1000);

myStepper.step(angle / 2.4);

RGB3();

delay(1000);

myStepper.step(angle / 2);

RGB4();

delay(1000);

myStepper.step(angle / 1.5);

RGB5();

delay(1000);

myStepper.step(angle / 1.1);

RGB6();

delay(1000);

myStepper.step(angle);

RGB7();

delay(1000);

average();

delay(300);

result();

}

else if(key == '1' || key == '2' || key == '3' || key == '4' || key == '5' || key == '6' || key == '7' || key == '8' || key == '9' || key == '0' || key == '#')

{

lcd.clear();

delay(1000);

lcd.setCursor(4,0);

lcd.print("Error!");

lcd.setCursor(2,1);

lcd.print("Wrong Input");

delay(1000);

setup();

}

}

void RGB()

{

color();

lcd.clear();

lcd.setCursor(0,0);

lcd.print("Initializing...");

delay(4000);

lcd.clear();

lcd.setCursor(0,0);

lcd.print("R=");

lcd.print(red);

delay(700);

lcd.setCursor(6,0);

lcd.print("G=");

lcd.print(green);

delay(700);

lcd.setCursor(0,1);

lcd.print("B=");

lcd.print(blue);

delay(700);

lcd.clear();

}

void RGB1()

{

color1();

lcd.clear();

lcd.setCursor(0,0);

lcd.print("Initializing...");

delay(4000);

lcd.clear();

lcd.setCursor(0,0);

lcd.print("R=");

lcd.print(red1);

delay(700);

lcd.setCursor(6,0);

lcd.print("G=");

lcd.print(green1);

delay(700);

lcd.setCursor(0,1);

lcd.print("B=");

lcd.print(blue1);

delay(700);

lcd.clear();

}

void RGB2()

{

color2();

lcd.clear();

lcd.setCursor(0,0);

lcd.print("Initializing...");

delay(4000);

lcd.clear();

lcd.setCursor(0,0);

lcd.print("R=");

lcd.print(red2);

delay(700);

lcd.setCursor(6,0);

lcd.print("G=");

lcd.print(green2);

delay(700);

lcd.setCursor(0,1);

lcd.print("B=");

lcd.print(blue2);

delay(700);

lcd.clear();

}

void RGB3()

{

color3();

lcd.clear();

lcd.setCursor(0,0);

lcd.print("Initializing...");

delay(4000);

lcd.clear();

lcd.setCursor(0,0);

lcd.print("R=");

lcd.print(red3);

delay(700);

lcd.setCursor(6,0);

lcd.print("G=");

lcd.print(green3);

delay(700);

lcd.setCursor(0,1);

lcd.print("B=");

lcd.print(blue3);

delay(700);

lcd.clear();

}

void RGB4()

{

color4();

lcd.clear();

lcd.setCursor(0,0);

lcd.print("Initializing...");

delay(4000);

lcd.clear();

lcd.setCursor(0,0);

lcd.print("R=");

lcd.print(red4);

delay(700);

lcd.setCursor(6,0);

lcd.print("G=");

lcd.print(green4);

delay(700);

lcd.setCursor(0,1);

lcd.print("B=");

lcd.print(blue4);

delay(700);

lcd.clear();

}

void RGB5()

{

color5();

lcd.clear();

lcd.setCursor(0,0);

lcd.print("Initializing...");

delay(4000);

lcd.clear();

lcd.setCursor(0,0);

lcd.print("R=");

lcd.print(red5);

delay(700);

lcd.setCursor(6,0);

lcd.print("G=");

lcd.print(green5);

delay(700);

lcd.setCursor(0,1);

lcd.print("B=");

lcd.print(blue5);

delay(700);

lcd.clear();

}

void RGB6()

{

color6();

lcd.clear();

lcd.setCursor(0,0);

lcd.print("Initializing...");

delay(4000);

lcd.clear();

lcd.setCursor(0,0);

lcd.print("R=");

lcd.print(red6);

delay(700);

lcd.setCursor(6,0);

lcd.print("G=");

lcd.print(green6);

delay(700);

lcd.setCursor(0,1);

lcd.print("B=");

lcd.print(blue6);

delay(700);

lcd.clear();

}

void RGB7()

{

color7();

lcd.clear();

lcd.setCursor(0,0);

lcd.print("Initializing...");

delay(4000);

lcd.clear();

lcd.setCursor(0,0);

lcd.print("R=");

lcd.print(red7);

delay(700);

lcd.setCursor(6,0);

lcd.print("G=");

lcd.print(green7);

delay(700);

lcd.setCursor(0,1);

lcd.print("B=");

lcd.print(blue7);

delay(700);

lcd.clear();

}

void color()

{

// Setting red filtered photodiodes to be read

digitalWrite(S2,LOW);

digitalWrite(S3,LOW);

// Reading the output frequency

red = pulseIn(sensorOut, LOW);

//Remaping the value of the frequency to the RGB Model of 0 to 255

red = map(red, 25,70,255,0);

delay(100);

// Setting Green filtered photodiodes to be read

digitalWrite(S2,LOW);

digitalWrite(S3,HIGH);

// Reading the output frequency

green = pulseIn(sensorOut, LOW);

//Remaping the value of the frequency to the RGB Model of 0 to 255

green = map(green, 25,70,255,0);

delay(100);

// Setting Blue filtered photodiodes to be read

digitalWrite(S2,HIGH);

digitalWrite(S3,HIGH);

// Reading the output frequency

blue = pulseIn(sensorOut, LOW);

//Remaping the value of the frequency to the RGB Model of 0 to 255

blue = map(blue, 25,70,255,0);

delay(100);

}

void color1()

{

// Setting red filtered photodiodes to be read

digitalWrite(S2,LOW);

digitalWrite(S3,LOW);

// Reading the output frequency

red1 = pulseIn(sensorOut, LOW);

//Remaping the value of the frequency to the RGB Model of 0 to 255

red1 = map(red1, 25,70,255,0);

delay(100);

// Setting Green filtered photodiodes to be read

digitalWrite(S2,LOW);

digitalWrite(S3,HIGH);

// Reading the output frequency

green1 = pulseIn(sensorOut, LOW);

//Remaping the value of the frequency to the RGB Model of 0 to 255

green1 = map(green1, 25,70,255,0);

delay(100);

// Setting Blue filtered photodiodes to be read

digitalWrite(S2,HIGH);

digitalWrite(S3,HIGH);

// Reading the output frequency

blue1 = pulseIn(sensorOut, LOW);

//Remaping the value of the frequency to the RGB Model of 0 to 255

blue1 = map(blue1, 25,70,255,0);

delay(100);

}

void color2()

{

// Setting red filtered photodiodes to be read

digitalWrite(S2,LOW);

digitalWrite(S3,LOW);

// Reading the output frequency

red2 = pulseIn(sensorOut, LOW);

//Remaping the value of the frequency to the RGB Model of 0 to 255

red2 = map(red2, 25,70,255,0);

delay(100);

// Setting Green filtered photodiodes to be read

digitalWrite(S2,LOW);

digitalWrite(S3,HIGH);

// Reading the output frequency

green2 = pulseIn(sensorOut, LOW);

//Remaping the value of the frequency to the RGB Model of 0 to 255

green2 = map(green2, 25,70,255,0);

delay(100);

// Setting Blue filtered photodiodes to be read

digitalWrite(S2,HIGH);

digitalWrite(S3,HIGH);

// Reading the output frequency

blue2 = pulseIn(sensorOut, LOW);

//Remaping the value of the frequency to the RGB Model of 0 to 255

blue2 = map(blue2, 25,70,255,0);

delay(100);

}

void color3()

{

// Setting red filtered photodiodes to be read

digitalWrite(S2,LOW);

digitalWrite(S3,LOW);

// Reading the output frequency

red3 = pulseIn(sensorOut, LOW);

//Remaping the value of the frequency to the RGB Model of 0 to 255

red3 = map(red3, 25,70,255,0);

delay(100);

// Setting Green filtered photodiodes to be read

digitalWrite(S2,LOW);

digitalWrite(S3,HIGH);

// Reading the output frequency

green3 = pulseIn(sensorOut, LOW);

//Remaping the value of the frequency to the RGB Model of 0 to 255

green3 = map(green3, 25,70,255,0);

delay(100);

// Setting Blue filtered photodiodes to be read

digitalWrite(S2,HIGH);

digitalWrite(S3,HIGH);

// Reading the output frequency

blue3 = pulseIn(sensorOut, LOW);

//Remaping the value of the frequency to the RGB Model of 0 to 255

blue3 = map(blue3, 25,70,255,0);

delay(100);

}

void color4()

{

// Setting red filtered photodiodes to be read

digitalWrite(S2,LOW);

digitalWrite(S3,LOW);

// Reading the output frequency

red4 = pulseIn(sensorOut, LOW);

//Remaping the value of the frequency to the RGB Model of 0 to 255

red4 = map(red4, 25,70,255,0);

delay(100);

// Setting Green filtered photodiodes to be read

digitalWrite(S2,LOW);

digitalWrite(S3,HIGH);

// Reading the output frequency

green4 = pulseIn(sensorOut, LOW);

//Remaping the value of the frequency to the RGB Model of 0 to 255

green4 = map(green4, 25,70,255,0);

delay(100);

// Setting Blue filtered photodiodes to be read

digitalWrite(S2,HIGH);

digitalWrite(S3,HIGH);

// Reading the output frequency

blue4 = pulseIn(sensorOut, LOW);

//Remaping the value of the frequency to the RGB Model of 0 to 255

blue4 = map(blue4, 25,70,255,0);

delay(100);

}

void color5()

{

// Setting red filtered photodiodes to be read

digitalWrite(S2,LOW);

digitalWrite(S3,LOW);

// Reading the output frequency

red5 = pulseIn(sensorOut, LOW);

//Remaping the value of the frequency to the RGB Model of 0 to 255

red5 = map(red5, 25,70,255,0);

delay(100);

// Setting Green filtered photodiodes to be read

digitalWrite(S2,LOW);

digitalWrite(S3,HIGH);

// Reading the output frequency

green5 = pulseIn(sensorOut, LOW);

//Remaping the value of the frequency to the RGB Model of 0 to 255

green5 = map(green5, 25,70,255,0);

delay(100);

// Setting Blue filtered photodiodes to be read

digitalWrite(S2,HIGH);

digitalWrite(S3,HIGH);

// Reading the output frequency

blue5 = pulseIn(sensorOut, LOW);

//Remaping the value of the frequency to the RGB Model of 0 to 255

blue5 = map(blue5, 25,70,255,0);

delay(100);

}

void color6()

{

// Setting red filtered photodiodes to be read

digitalWrite(S2,LOW);

digitalWrite(S3,LOW);

// Reading the output frequency

red6 = pulseIn(sensorOut, LOW);

//Remaping the value of the frequency to the RGB Model of 0 to 255

red6 = map(red6, 25,70,255,0);

delay(100);

// Setting Green filtered photodiodes to be read

digitalWrite(S2,LOW);

digitalWrite(S3,HIGH);

// Reading the output frequency

green6 = pulseIn(sensorOut, LOW);

//Remaping the value of the frequency to the RGB Model of 0 to 255

green6 = map(green6, 25,70,255,0);

delay(100);

// Setting Blue filtered photodiodes to be read

digitalWrite(S2,HIGH);

digitalWrite(S3,HIGH);

// Reading the output frequency

blue6 = pulseIn(sensorOut, LOW);

//Remaping the value of the frequency to the RGB Model of 0 to 255

blue6 = map(blue6, 25,70,255,0);

delay(100);

}

void color7()

{

// Setting red filtered photodiodes to be read

digitalWrite(S2,LOW);

digitalWrite(S3,LOW);

// Reading the output frequency

red7 = pulseIn(sensorOut, LOW);

//Remaping the value of the frequency to the RGB Model of 0 to 255

red7 = map(red7, 25,70,255,0);

delay(100);

// Setting Green filtered photodiodes to be read

digitalWrite(S2,LOW);

digitalWrite(S3,HIGH);

// Reading the output frequency

green7 = pulseIn(sensorOut, LOW);

//Remaping the value of the frequency to the RGB Model of 0 to 255

green7 = map(green7, 25,70,255,0);

delay(100);

// Setting Blue filtered photodiodes to be read

digitalWrite(S2,HIGH);

digitalWrite(S3,HIGH);

// Reading the output frequency

blue7 = pulseIn(sensorOut, LOW);

//Remaping the value of the frequency to the RGB Model of 0 to 255

blue7 = map(blue7, 25,70,255,0);

delay(100);

}

void average()

{

redAve = (red+red1+red2+red3+red4+red5+red6+red7) / 8;

greenAve = (green+green1+green2+green3+green4+green5+green6+green7) / 8;

blueAve = (blue+blue1+blue2+blue3+blue4+blue5+blue6+blue7) / 8;

delay(1000);

lcd.setCursor(4,0);

lcd.print("Average");

delay(1000);

lcd.clear();

lcd.setCursor(0,0);

lcd.print("Red=");

lcd.print(redAve);

delay(1000);

lcd.setCursor(7,0);

lcd.print("Green=");

lcd.print(greenAve);

delay(1000);

lcd.setCursor(1,1);

lcd.print("Blue=");

lcd.print(blueAve);

delay(1000);

lcd.clear();

}

void result()

{

if((redAve >= 210 && redAve <= 283) && (greenAve >= 227 && greenAve <= 266) && (blueAve >= 187 && blueAve <= 260))

{

delay(1000);

lcd.setCursor(2,0);

lcd.print("Ripeness Level");

lcd.setCursor(5,1);

lcd.print("Ripe");

delay(1000);

}

else if((redAve >= 199 && redAve <= 210) && (greenAve >= 221 && greenAve <= 277) && (blueAve >= 176 && blueAve <= 227))

{

delay(1000);

lcd.setCursor(2,0);

lcd.print("Ripeness Level");

lcd.setCursor(5,1);

lcd.print("Unripe");

delay(1000);

}

else

{

delay(1000);

lcd.setCursor(1,0);

lcd.print("Out of range");

}

}

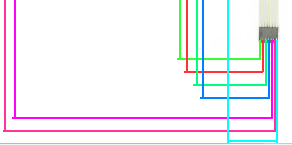
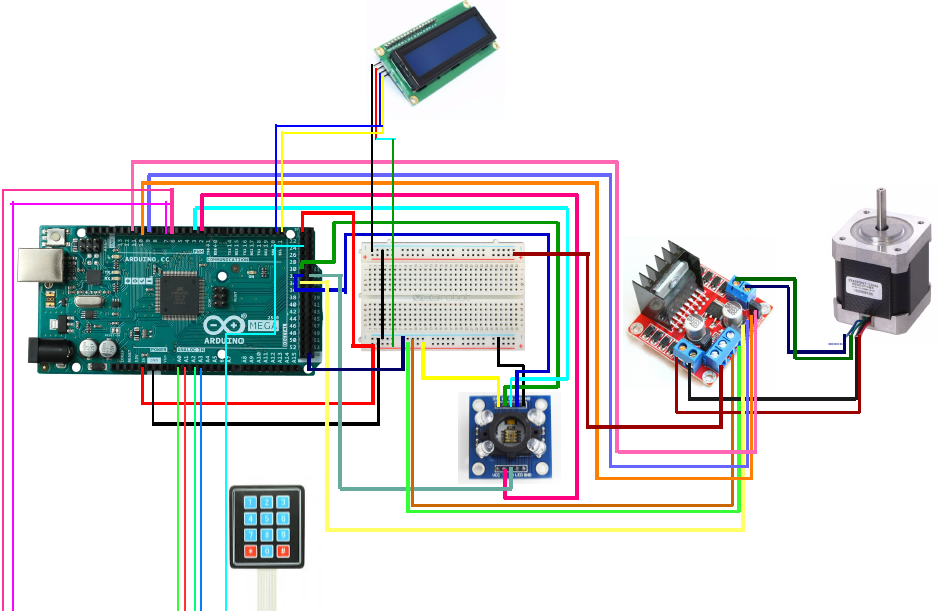
### **Hardware**

#### Block Diagram



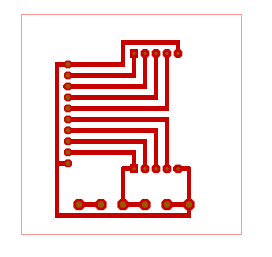
*Figure 4.3 – Block Diagram*

#### Schematic Diagram



*Figure 4.4 – Schematic Diagram*

#### PCB Layout



S1

S0

LED

OUT

S2

S3

GND

5V

GND

5V

GND

5V

*Figure 4.5 – PCB Layout for Color Sensor*

#### Prototype Layout (with sizes)



1 inch

2.2 inches

*Figure 4.6 – Distance of Color Sensor to Watermelon*



15 inches diameter (all sides)

*Figure 4.7 – Size of the Cube (all sides)*



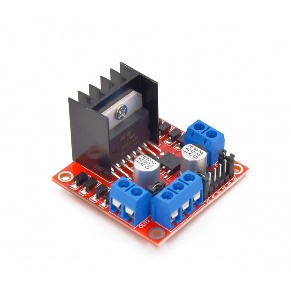
17 inches (Top view)

*Figure 4.8 – Size of the Cube (Top view)*

#### Components



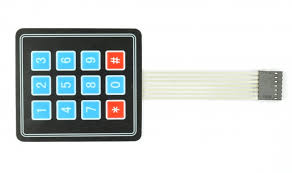
1. TCS3200 Color Sensor



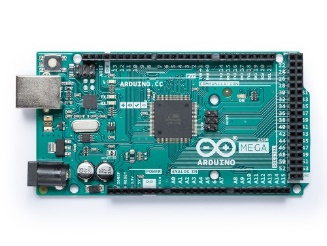
1. L298N Dual H-Bridge Driver Motor Module



1. Stepper Motor Nema 17
2.  LCD I2C (16x2)



1. Keypad (3x4)



1. Arduino Mega 2560 Rev3



1. Connecting Wires (Jumper Wire):

* Male to male
* Male to female
* Female to female

1.  Presensitized PCB



1. Terminal Block



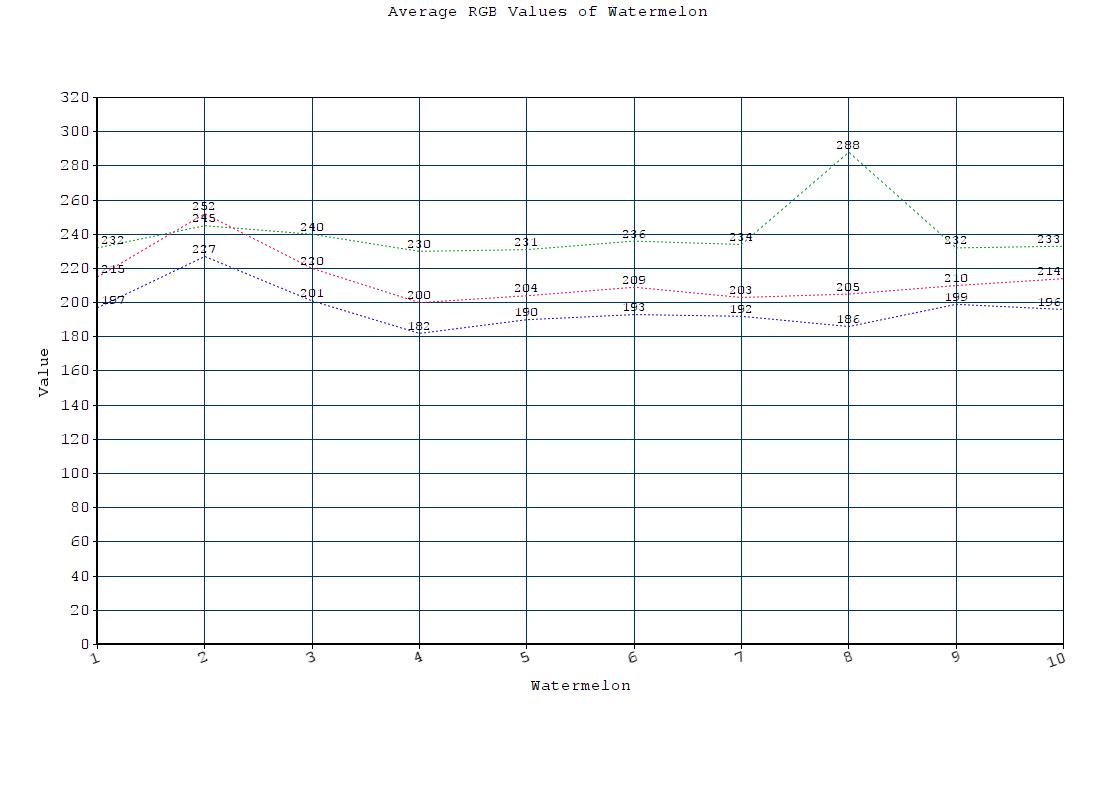
1. Header:

* Male and Female

# **Chapter 5**

## **Testing’s, Results, & Interpretations**

Below is the statistical analysis for data gathered in testing 10 watermelons:

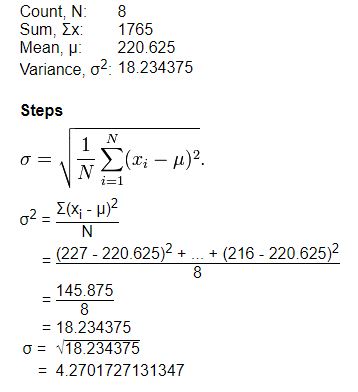
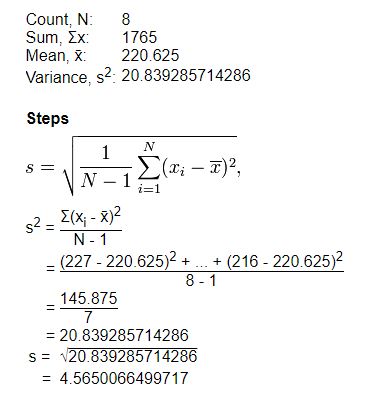


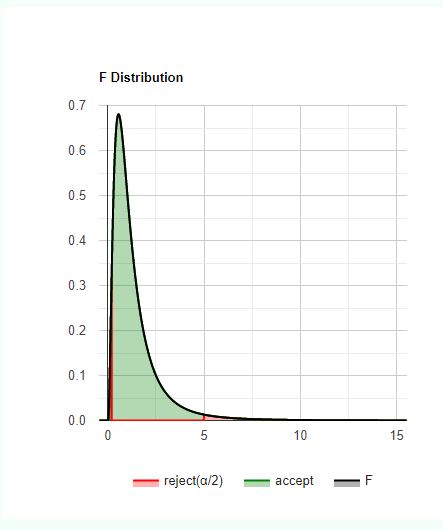
*Figure 5.1 – Graph of Average RGB values of Watermelon*

Calculations:

**F – Test**

Standard deviation (S1 and S2)





*Figure 5.2 – F – distribution of testing Watermelons*

F= S12 = (4.57)2 = 1.02222

S22 (4.52)2

H0 hypothesis

.P-value: p-value equals 1.92663e-7, ( p(x≤F) = 1.00000 ). This means that the chance of type1 error (rejecting a correct H0) is small: 1.927e-7 (0.000019%). The smaller the p-value the more it supports H1. Since p-value < α, H0 is rejected. The sample standard deviation (S1) of Group1's population is considered to be not equal to the sample standard deviation (S2). of the Group2's population. In other words, the difference between the sample standard deviation (S) of the Group1 and Group2 populations is big enough to be statistically significant.

# **Chapter 6**

## **Conclusions and Recommendations**

**Conclusion**

On this study, we therefore conclude that the ripeness and maturity level of the watermelon (*with the scientific name of Citrullus Lanatus*) can be detected with the use of a program using Arduino Mega and a RGB Color Sensor for Arduino. Through our testing, we have concluded the average RGB values that would separate the ripe from unripe samples. Ripe watermelons have the range of color red values from 210 to 283; range of color green values are 227 to 266; and the range for color blue value is 187 to 260. Whilst for the Unripe watermelons, the value for color red rangers from 199 to 210; range value of green is 221 to 277 for color green; and range value of 176 to 227 for the color blue.

**Recommendation**

This study was able to detect the ripeness and maturity level of the watermelon using an Arduino Mega, a RGB Color Sensor for Arduino and a NEMA 17 Stepper Motor successfully. However, there are still factors that can improved by future researchers. We have encountered limitations with the NEMA 17 Stepper Motor regarding the weight of the samples. Samples over 6 kilogram are hard for the stepper to carry and rotate. The following are the recommended factors that can be improved for future researchers: 1.) use of a better model of a stepper motor that can accommodate the maximum possible weight of watermelon samples; 2.) use of better RGB Color Sensor for improved color detection and accuracy; 3.) use of more samples to improve the accuracy of reading; 4.) use of better approach regarding the formulation of codes for a better reading classification of ripe and unripe samples of water melon.