TECHNOLOGICAL INSTITUTE OF THE PHILIPPINES

1338 Arlegui St., Quiapo, Manila

A Design of Noise Controlled Environment for Non-Destructive Measurement of Sugar Content in Citrullus Lanatus: A Supervised Machine Learning Approach using Acoustic Property

De Guzman, J.R. T.

Hoganas, Charisse Jane B.

Montana, Marie Aileen S.

Po, Elynne Gianna E.

Sarmiento, Shaira Faye L.

Zamora Jasper Ian A.

December 2020

TECHNOLOGICAL INSTITUTE OF THE PHILIPPINES
1338 Arlegui St., Quiapo, Manila

COLLEGE OF ENGINEERING AND ARCHITECTURE

ACCEPTANCE SHEET

The proposed system entitled **A Design of Noise Controlled Environment for Non-Destructive Measurement of Sugar Content in Citrullus Lanatus: A Supervised Machine Learning Approach using Acoustic Property** has been prepared and submitted by the proponents:

De Guzman, J.R. T.

Hoganas, Charisse Jane B.

Montana, Marie Aileen S.

Po, Elynne Gianna E.

Sarmiento, Shaira Faye L.

Zamora Jasper Ian A.

for approval to the committee for the *Project Design.*

After a thorough review and evaluation of the proposed design project, the committee has accepted the presented proposed design based on the required criteria.

The acceptance is valid to the information being presented. Accepted this **December 2020, 1st semester, School Year 2020-2021.**

Engr. Joshua S. Gulmatico Panel Member	**Engr. Alvin S. Alon** Project Adviser
Engr. Mon Arjay F. Malbog Panel Member	**Engr. Jennifer B. Enriquez** Class Adviser
Engr. Rufo I. Marasigan Jr. Panel Member	**Engr. Jennifer B. Enriquez** Chair of Computer Engineering

TECHNOLOGICAL INSTITUTE OF THE PHILIPPINES
1338 Arlegui St., Quiapo, Manila

COLLEGE OF ENGINEERING AND ARCHITECTURE

APPROVAL SHEET

The proposed system entitled **A Design of Noise Controlled Environment for Non-Destructive Measurement of Sugar Content in Citrullus Lanatus: A Supervised Machine Learning Approach using Acoustic Property** which was presented on **December 2020** by the proponents:

De Guzman, J.R. T.

Hoganas, Charisse Jane B.

Montana, Marie Aileen S.

Po, Elynne Gianna E.

Sarmiento, Shaira Faye L.

Zamora Jasper Ian A.

is hereby APPROVED by the following members of the committee:

Engr. Joshua S. Gulmatico Panel Member	**Engr. Alvin S. Alon** Project Adviser
Engr. Mon Arjay F. Malbog Panel Member	**Engr. Jennifer B. Enriquez** Class Adviser
Engr. Rufo I. Marasigan Jr. Panel Member	**Engr. Jennifer B. Enriquez** Chair of Computer Engineering

Acknowledgement

Firstly, the researchers would like to thank God for providing this opportunity to continue their studies amidst the on-going pandemic. They are grateful for the guidance, provision, wisdom and strength He is giving to sustain them all throughout the project.

The engineers would like to extend their thankfulness to their families for their continuous support morally and financially; for loving and cheering them up whenever facing difficulties at school. Where they are now wouldn't also be possible without their families' beneficial contributions.

Moreover, the completion and success of this project wouldn't be turned into reality without the help of the following people who are very much willing to impart and share their knowledge and experiences to them. They wish to convey their gratitude towards Engr. Alvin S. Alon, their Project Adviser, for constantly guiding and helping them in their project. His expertise and advice have helped them a lot to stay on track and progress in their design project.

Lastly, the engineers are grateful to their classmates and Class Adviser, Engr. Jennifer B. Enriquez, for sharing their knowledge and resources generously and unhesitantly. Their valuable inputs have helped them to improve vastly on their final output.

Table of Contents

List of Figures

List of Tables

Chapter 1

Problem and Its Background

The Project

Watermelon, scientifically known as Citrullus lanatus, is a lush fruit which originated from the tropical country of Africa. It efficiently adapts to a warm and dry environment and is described as prone, “basally branched, softly hairy vines possessing deeply and roundly indented, blue–grey-green leaves and branched tendrils”. In terms of its size, it varies relatively from 2 kilograms (kg) to over 10 kg and is up to 8 inches wide. The fruit’s shape “ranges from almost spherical to obovate”; while the color is usually green and some rare forms have distinct colors such as yellow fruits or bicolor, and mixture of yellow and green (Schaffer and Paris, 2003).

Being a highly nutritious fruit, watermelon is rich in lycopene, potassium, calcium, iron, thiamin, vitamin A and vitamin C (Chogou et al., 2019). Hence, eating the fruit is beneficial to people as it may lower inflammation and oxidative stress, improves the skin and hair, enhances digestion and heart health and keeps a person hydrated primarily (Jennings, 2018). Knowing these benefits, it is also as important as identifying the best kind of watermelon to be consumed.

The different factors to consider in determining the quality of a good watermelon are its ripeness, maturity and sweetness. These are commonly checked based on the experience of the farmers with testing the fruit (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). In the Philippines, the good watermelons are bought at the price of PHP 10 to PHP 70 per kg depending on its variety while the unripe ones are not actually sold, according to Mr. Jeffrey Pamo, an Agriculturist in the Local Government Unit (LGU) of Bani in Pangasinan.

Although, in checking its sweetness, Mr. Pamo said that the color and shape factors are the only possible indicators during manual inspection. He also added that the watermelon’s sweetness varies due to the seed, soil, brackish type of water and fertilizers used. Even though these processes are the most common, there still has been a difficulty in using these traditional methods when people have to check several fruits in hectares of land. The results may be inconsistent, subjective and slow (Sun et. al., 2010).

Additionally, the importance of sugar in food products is to provide sweetness and energy (Rosa et. al., 2009). Sugar has also a critical adequate metabolic function such as to prevent stress on the body and to avoid the depletion of critical cellular components (Vaclavik et. al., 2014). However, when sugar in food is consumed excessively, serious health problems may arise. It increases the risk for the following: obesity, type 2 diabetes, increased pressure on the heart and blood vessels, and dental decay. Thus, it is helpful to know just how much sugar is in the most consumed foods in the market to maintain and control one's sugar levels (Nordqvist, 2018). Estimatedly, a cup of watermelon balls contains about 10 grams of sugar.

To test how much sugar a fruit has, there are destructive techniques done by some researchers using experimental tools. Bhosale (2017) used a Capacitance Measuring Machine which is composed of different controllers and sensors. Refractometer is also one of the equipment that was used in that experiment to measure the fruit's sugar concentration of liquid in the unit percentage Brix wherein 1 %Brix is 1 gram of sugar in 100 grams of solution. Furthermore, there are methods called Enzymatic Method and Gas Chromatography Mass Spectrometry (GCMS) in determining the sugar content. The former is the most common quantification method for defined sugars while the latter is widely used for sugar identification and quantification (Al-Mhanna, 2018).

Also, a number of studies are made about non-destructive techniques in determining watermelon's sugar content. One of which is using acoustic property technology wherein the interaction between the fruit and sound waves creates an acoustical impulse response (Wei et al. 2012). A device was developed in that study wherein the "acoustic transmissivity at (six different) frequencies were used to set up a multi regression model for the detection of watermelon sugar content". Wei et al. (2012) concluded that the "detecting effect might be better when percussion and signal receiving parts were located at the equatorial part".

Nevertheless, there are notable downsides in using the acoustic property technology. These are (1) low detection precision; (2) poor model adaptability; and (3) interference from the environment because it is easily disturbed. This method takes more time and there is a difficulty in acquiring the internal quality of the fruit since most results were limited to the laboratory research. Hence, to complement the downsides of this technique in measuring the sugar content of watermelon, this project will use sound property to improve the internal detection while reducing unnecessary interruptions from the environment affecting the

outcome. In this way, better results are foreseen to be produced in the accuracy and precision of measurements.

Project Objectives

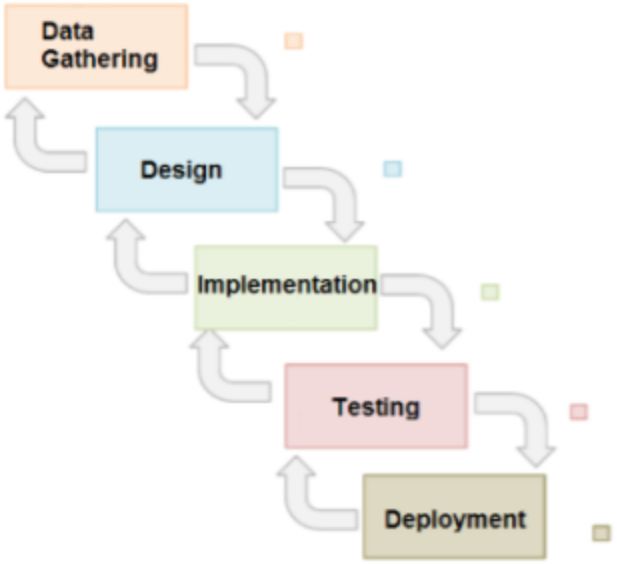
Generally, this project aims to create a device to address the stated problem when using acoustic property technique in measuring the sugar content of watermelon. Specifically, it intends to do the following (1) design a prototype that will use noise absorption materials to lessen the noise in the surroundings; (2) to use supervised machine learning algorithm to improve the accuracy and precision of detection in the sugar content of watermelon; and (3) evaluate and validate the accuracy of the system in assessing the sugar content of watermelon and noise reduction.

Project Scope and Limitations

To ensure the attainment of the objectives, this project covers the following (1) determines the sugar content of watermelon without having to destroy the fruit; (2) uses acoustic technique to improve the accuracy of detection; (3) uses Sugar Baby variety of watermelon coming from Bani, Pangasinan for testing; and (4) detects 51 pieces of small to medium-sized fruit upon harvest only and one at a time. However, due to time and financial constraints, the following limitations would have been the value addition to the project (1) unable to detect other fruit's sugar content; (2) unable to detect the maturity and ripeness level of the watermelon; and (3) unable to filter other types of noise other than what are stated.

Project Development

Figure 1.1 *Waterfall Model*



The first process is gathering information wherein the engineers identified a problem for the project that is about watermelon's ripeness. A detailed data gathering from the internet, books and online journals or from other related reference materials was done to learn the background of the problem. Also, the engineers went to the Agricultural Training Institute under the Department of Agriculture in Quezon City to validate the need of determining the ripeness of watermelon. However, upon re-validating with the Department of Agriculture in Bani, Pangasinan, it was found out that it is more important to detect the sugar content of the fruit instead of ripeness. The farmers in Bani also gave their requirements that will be used as guidelines for this project.

Second stage is Design where diagrams and figures are made to represent how the system will work. This includes the storyboard diagram, system architecture, design options and trade off analysis. The engineers will plan the flow of the whole device to be created including the software and hardware components based on the project requirements given by the client. In the trade off analysis, three design options are examined and only one will be chosen for the next stage.

Third stage is Implementation where the initial prototyping of the best design option will be carried out. The coding of the entire program for the system will also be performed in this stage. Problems may occur and error can be encountered but it should be resolved by repeating the above steps.

Fourth stage is Testing of the functional prototype. Several testings must be made on the developed design. A testing must also be made together with the client for feedback and redevelopment of the device. However, as of this writing, due to the current situation with the pandemic, testing with the client will be postponed.

Last is the Deployment stage where the final prototype will be prepared for client use. The device must already be freed from defects and bugs, and polishing of its appearance is needed to make it at industry level.

The Client

This project would mainly be for the farmers in Bani, Pangasinan led by Mr. Jolly Calisaan who is the chairman of the Municipal Fisheries and Aquatic Resources Management Council. Bani is best known for having the best watermelon in the country and the watermelon farmers in that location are confident enough that their products are of the best taste. With that, this project would help them to provide actual proof for what they are claiming.

Chapter 2

Design Inputs

Client Requirements

After discussing the problems encountered by the client, another interview was conducted to gather their requirements that will be integrated in the proposed solution. The following are the requirements given by the client.

1. The device must be portable.
2. The device must be user-friendly.
3. The device should be accurate in measuring the sugar content of watermelon.
4. The device should be non invasive and safe to watermelon when measuring sugar content.
5. The cost of the device should not exceed 10 thousand pesos, much better if it will only cost 5 thousand pesos.

Design Criteria

After convening the requirements from the client, the group derived the design criteria as shown in Table 2.1 below, and identified multiple constraints that will be considered in the development of the solution of the problem.

Table 2.1 *Design Criteria and Constraints*

Design Criteria	Design Constraints
The device should have the capability to test the sugar content of a watermelon in high-speed.	Speed
The device should have an accurate measurement of sugar content based on the noise reduction of internal and external noises that are present.	Accuracy
The device should obtain a low consumption of memory and a fast clock cycle.	Efficiency

Speed

Speed is the overall time the device takes to complete processing. The rate of the speed during the detection determines if the device is faster than the manual method of checking the sugar content of the watermelon. As per the interview conducted, it takes approximately less than a minute to complete the quality checking of a single watermelon in the manual method. The device aims to detect the sugar content of more than one fruit within a minute. This will be measured by timing the duration of detection from pressing the button to start up to displaying the output on the screen.

Accuracy

It is the degree to which a measured value agrees with the correct value for that measurement. The uncertainty in the correctness will be identified by the engineers which will be expressed as a percent of the measured value. The standard uncertainty limit according to ISO 15197:2013 is 5% which would mean a value of 95% for accuracy. However, this project will use the result of the study of Wei et al. (2012) as its reference in measuring the percentage of accuracy which is at 86%. The computation for accuracy will also be generated based on the program codes used in this project.

Efficiency

Efficiency is defined as a state or quality of being able to accomplish something with the least waste of time and effort. The efficiency of a device is the proportion of the energy supplied that is transferred in useful ways. In this constraint, the efficiency will focus on two things, the memory consumption and the Central Processing Unit (CPU) usage. The device, in terms of memory, must obtain a low consumption measured in megabytes (MB). On the other hand, another factor to be considered is the usage of CPU which is measured by percentage. Generally, the “normal CPU usage is 2-4% at idle, 10% to 30% when playing less demanding games, up to 70% for more demanding ones, and up to 100% for rendering work” (Monk, 2020).

Storyboard Diagram

Figure 2.1 *Storyboard Diagram*

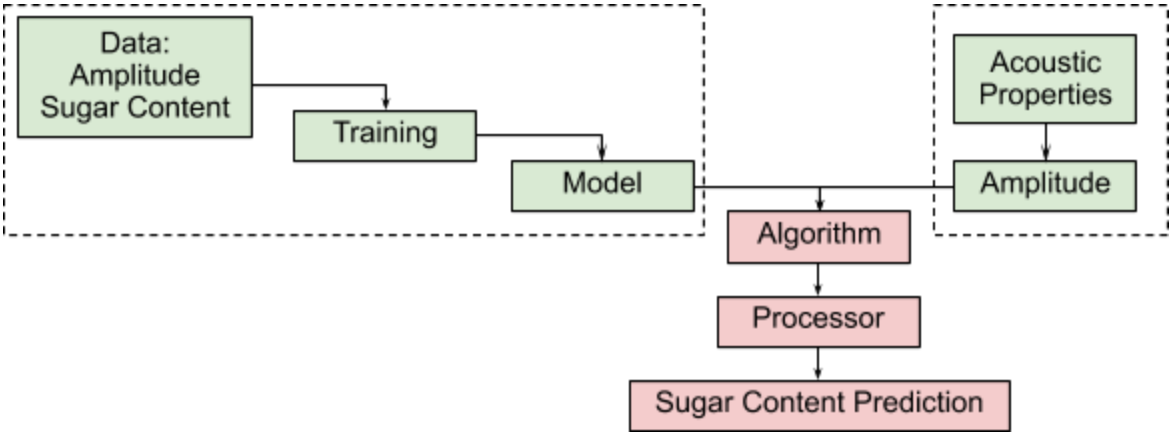


Figure 2.1 shows the storyboard of the solution that this project will implement. The amplitude and sugar level acquired from the watermelon will be trained for the model that will be applied in the detection algorithm. While the amplitude obtained from the acoustic properties of the fruit will be used directly as the input in the same algorithms. The sugar content level of the watermelon, as the output, will then be predicted by the algorithm.

Additional Information

For this project, a variety of watermelon called as Sugar Baby will be used for the testing. Nonetheless, there are other different varieties and hybrids of round watermelon that are grown in the Philippines. In Table 2.2, some of which are shown with their size measured in kilograms, the colors of their flesh and rind, and their approximate sugar content percentage (%Brix).

Table 2.2 *Watermelon Varieties*

IMAGE	VARIETY	WEIGHT (kg)	FLESH	RIND	%BRIX
	Diana	2-4	Red	Golden Yellow	12-14
	Tiffany	2-4	Creamy Yellow	Dark Green	12-14

	Sugar Baby	4-5	Dark Red	Dark Green	10-12
	Crimson Sweet	3-6	Deep Red	Striped Green	11-13
	Charleston Gray	2-3	Bright Red	Light Green	11-13
	Orchid Sweet	3-5	Bright Yellow	Striped Green	11-12

Furthermore, amplitude is the key property of sound that will be used in this project. As defined from the Standard Terminology for Nondestructive Examinations (ASTM E1316, 2014), it is “the peak voltage of the largest excursion attained by the signal waveform from an emission event”. Also, it is the maximum positive or negative waveform of acoustic emission (AE) signal excursion during an AE hit and expressed in dB (Physical Acoustic Corporation, 2005). In sound, a larger amplitude means a louder sound and more energetic vibration, and a smaller amplitude means a softer sound. Additionally, the amplitude is measured in decibels (dB) or Volts (V).

Based on the study entitled “Acoustic Detecting System for Sugar Content of Watermelon” by Wei et al. (2012), sound is produced when air responds to a disturbance, creating air waves that travel in all directions away from the disturbance. In direct relation, acoustic properties are those that govern how materials respond to sound waves, which are what perceived as sound. These acoustic properties were used in their study to interpret acoustic waves of the fruit and input the algorithm of the sugar level content.

Commonly, the problem with acoustic technique is the noise interference. Noise is defined as any unwanted form of energy that tends to interfere with proper reception and reproduction of a wanted signal. It is classified as: (1) external noise which is produced from outside of the device that cannot always be controlled; and (2) internal noise which is generated from inside the device and can be controlled or manipulated (Simon, 2011). The external noise that will be considered in this project are man-made and natural noise which both come from the outside environment of the device which are measured in decibels.

As per Mr. Calisaan, the farms of Bani, where the testing will be done, may generate noise coming from the animals and strong wind.

Aside from that, there are three types of internal noise interference that may be present in the project but are negligible. First is the electro-magnetic interference (EMI), which is an electromagnetic emission that causes a disturbance in another piece of electrical equipment. EMI can be attributed to a wide span of the electromagnetic spectrum including radio, DC and even microwave frequencies. Anything that carries rapidly changing electrical currents gives off electromagnetic emissions, thus it is quite common for one object's emissions to "interfere" with another's. EMI can be caused two ways: (1) by direct physical contact with a conductor, called conducted EMI; or (2) by induction (without physical contact), called radiated EMI. While nearly every electrical device creates emissions that can cause EMI, (i.e. electric motors, Bluetooth devices, cellphones) electric power cords and high-speed cabling are the most common sources of radiated EMI. Various ways to prevent electro-magnetic interference include keeping electronic devices away from heavy machinery, motors or generators, using shielded cables, as well as electro-magnetic interference filters (Santora, 2015).

The second type of noise interference is the electric motor vibration, which is simply defined as the back and forth movement or oscillation of machines and components in a motorized equipment caused by electrical currents. Vibration itself can be caused by the motor and its normal mechanisms, or by one or more problematic factors at any given time. The most common being imbalance, misalignment, wear and looseness. Imbalance, or having a "heavy spot" in a rotating component will cause vibration when the unbalanced weight rotates around the machine's axis, creating a centrifugal force. As machine speed increases the effects of imbalance become greater. Second is misalignment of the machine shafts. The resulting vibration can be radial or axial misalignment (in line with the axis of the machine) or both. Lastly, as components of the motor such as ball or roller bearings, drive belts or gears become worn or loose, they might cause vibration as it prevents the normal mechanisms to process (Rycroft, 2015). This type of noise interference can be prevented by having preventive maintenance and ensuring that all motor mechanisms are intact and are working smoothly to minimize the occurrence of the negative factors causing electric motor vibration (Finley and Hodowance, 2001).

The final type of noise interference is the thermal noise, which is a factor that is always present in electronic circuits and is one of the major sources of noise. It is generated as a result of thermal agitation of

the charge carriers which are typically electrons within an electrical conductor (Thompson, 2002). Thermal noise actually occurs regardless of the applied voltage because the charge carriers vibrate as a result of the temperature. This vibration is dependent upon the temperature - the higher the temperature, the higher the agitation and hence the thermal noise level. Some alternative ways to reduce the thermal noise content are to reduce the temperature of operation, or reduce the value of the resistors in the circuit (Rouphael, 2014). All of the solutions provided in each noise type can be executed in two ways: (1) by choice of components and materials for shielding, and (2) through the processor programming for noise reduction.

There are sound absorbing or reflecting materials used in the industry such as foam, wool and wood that are used as shielding. Sound absorptive materials are commonly used to soften the acoustic environment of a closed volume by reducing the amplitude of the reflected waves. Absorptive materials are generally resistive, either fibrous, porous or in rather individual cases, reactive resonators (Bell, 1994).

Foams have notable acoustic absorption properties (Smardzewski et al. 2015). Foams are formed by trapping pockets of gas within a liquid or solid. There are different types of foam like polyurethane, melamine, polyimide, honeycomb, polypropylene, etc., but this study will focus on the first type. Polyurethane foam is a widely used product and inexpensive material as a filter (Adams, 2016). A peak absorption coefficient of 0.95 was observed for foam with a 73% open-cell content and a 9 mm cavity in the study of Smardzewski et al. (2015).

Similarly, wool absorbs sound that is used in noise reduction due to its porous structures (Bell, 1994). It has low resistivity thus can achieve higher absorption coefficients for control of room reverberation. Reverberation reduces when the reflections hit a surface that can absorb sound (Elbert et al., 2018). The acoustical property of wool that helps in sound absorbing is the specific flow resistance per unit thickness of material. The lesser the air permeability present, the higher the airflow resistivity. To maximize the use of wool's acoustical performance, it must be in relatively thick layers of more than 50 millimeters (mm) or around 2 inches. According to Elbert et al. (2018), woolen materials can increase the transmission loss of stud walls by up to 6 dB or more.

On the other hand, wood is used as a sound reflector that is helpful in noise reduction. If dense enough, its structure can easily be made into surfaces for sound reflections. When wood is placed with porous absorption material, a board resonator is formed where it effectively lessens the intensity of sounds when it vibrates. The vibration of wood creates or resonates the sound (Roohnia, 2016). According to

Smardzewski et al. (2015), for frequencies between 125 and 500 Hz, the highest capability of sound absorption was determined of low surface layer density and high porosity. Hence, the denser the wood, the more it is elastic.

Datasets

The engineers will be collecting new data from the watermelon since the study of Wei et al. (2012) did not reveal the datasets they have used. The new data will be gathered using a microphone which will record the amplitude from the sound the speaker sends. Every watermelon acoustically tested will be compared to the measured and actual sugar content using a refractometer. For a single watermelon, the process will be repeated in eight different sections. However, for the initial testing of the programming codes, the following sample data from the study of Pintor et al. (2016) will be used considering frequency as the main acoustic property instead of amplitude.

Table 2.3 *Watermelon Frequency Range*

SAMPLE		**FREQUENCY (Hz)**			**SAMPLE**		**FREQUENCY (Hz)**		
		SMALL	MEDIUM	LARGE			SMALL	MEDIUM	LARGE
1	Bottom	161.5	145.35	116.64	**11**	Bottom	166.88	163.29	165.09
	Center	96.9	122.01	125.61		Center	122.02	156.12	118.43
	Top	138.17	154.32	96.9		Top	161.5	145.35	118.43
2	Bottom	159.7	96.9	150.73	**12**	Bottom	91.52	188.42	209.95
	Center	163.29	91.52	127.4		Center	129.2	113.05	86.13
	Top	183.03	172.27	91.52		Top	104.8	209.95	118.43
3	Bottom	113.05	190.21	80.75	**13**	Bottom	96.9	127.4	111.25
	Center	107.67	86.13	114.84		Center	102.28	91.52	134.58
	Top	118.43	145.35	152.53		Top	96.9	107.67	129.2

4	Bottom	220.72	95.1	217.13	**14**	Bottom	156.12	209.95	150.73
	Center	123.82	98.69	114.84		Center	123.82	113.05	129.1
	Top	166.88	209.95	148.94		Top	209.95	129.2	96.9
5	Bottom	209.95	118.43	107.67	**15**	Bottom	236.87	150.73	200.98
	Center	209.95	139.97	139.97		Center	139.97	91.52	145.35
	Top	118.43	107.67	107.67		Top	134.58	156.12	113.05
6	Bottom	107.67	147.14	161.5	**16**	Bottom	211.74	204.57	129.2
	Center	139.97	96.9	134.58		Center	215.33	123.82	129.2
	Top	148.94	91.52	107.67		Top	215.33	145.35	96.9
7	Bottom	226.1	161.5	213.54	**17**	Bottom	215.33	136.38	150.73
	Center	226.1	134.58	109.46		Center	215.33	91.52	139.97
	Top	226.1	136.38	147.14		Top	209.95	123.82	107.67
8	Bottom	123.82	113.05	105.87	**18**	Bottom	163.29	113.05	199.18
	Center	91.52	120.23	96.9		Center	156.12	107.67	150.73
	Top	98.69	113.05	123.82		Top	145.35	118.43	113.05
9	Bottom	152.53	132.79	154.32	**19**	Bottom	209.95	229.69	102.28
	Center	129.2	156.12	129.2		Center	113.05	118.43	134.58
	Top	102.28	134.58	132.79		Top	145.35	113.05	143.55
10	Bottom	226.1	193.8	127.4	**20**	Bottom	236.87	209.95	129.2
	Center	91.52	96.9	123.82		Center	139.97	113.05	129.2
	Top	226.1	127.4	113.05		Top	134.58	145.35	96.9

Table 2.3 presents the data gathered in the study conducted in the Philippines by Pintor et al. (2016). Twenty watermelons each classified by size as Small, Medium and Large were used and tested for three times in different points of the fruit namely, Bottom, Center and Top. Thus, a total of 180 frequencies were generated. The results showed that the lowest frequency is 80.75 Hz and the highest is 236.87 Hz. On the other hand, Table 2.4 below shows the averaged frequency data of Table 2.3.

Table 2.4 *Averaged Watermelon Frequency in Hz*

	SMALL	MEDIUM	LARGE
Bottom	174.3305	156.385	149.207
Center	141.8515	113.1405	125.6995
Top	154.231	139.339	117.8055

Table 2.5 *Measured Sweetness in %Brix*

SAMPLE		**SWEETNESS (%Brix)**			**SAMPLE**		**SWEETNESS (%Brix)**		
		SMALL	MEDIUM	LARGE			SMALL	MEDIUM	LARGE
1	Bottom	7.37	7.87	8.6	11	Bottom	7.37	7.97	8.4
	Center	9.1	9.23	9.97		Center	9.17	8.9	9.23
	Top	8.53	9.5	9.2		Top	8.77	7.93	8.97
2	Bottom	7.37	9.9	9.7	12	Bottom	7.83	8.1	8.63
	Center	8.4	9.43	9.37		Center	8.5	8	9.9
	Top	6.8	7.27	9.6		Top	7.73	6.5	8.9
3	Bottom	8	9.43	9.77	13	Bottom	7.77	8.5	8.4
	Center	8.67	10.07	9.57		Center	9.3	9.27	9.13
	Top	7.7	9.2	9.07		Top	7.87	8.03	7.7

4	Bottom	6.6	8.97	8.8	14	Bottom	7.87	7.77	8.5
	Center	8.03	8.8	9.4		Center	8.23	9.3	9.77
	Top	6.37	6.4	9.4		Top	6.2	8.2	8.87
5	Bottom	7.17	9.27	8.6	15	Bottom	8.63	7.37	8.03
	Center	7.93	9.4	9.1		Center	8.93	8.63	9.1
	Top	8.5	8.77	9.43		Top	6.97	7.33	8.53
6	Bottom	8	9.33	8.23	16	Bottom	6.9	8.5	8.57
	Center	8.4	10	8.57		Center	7.83	9	8.67
	Top	8.27	9.83	8.8		Top	5.7	8.3	9
7	Bottom	6.77	8.83	8.57	17	Bottom	7.4	8.03	8.03
	Center	7.63	9.07	8.63		Center	7.8	9.53	9.2
	Top	6.1	7.84	8.37		Top	6.27	8.33	8.87
8	Bottom	7.67	8.93	9.13	18	Bottom	7.87	8.83	8.9
	Center	8.7	9.4	9.17		Center	9.03	8.97	9.77
	Top	8.27	8.22	9.23		Top	7.8	7.5	8.73
9	Bottom	8.27	8.27	8.1	19	Bottom	8.13	8.3	8.3
	Center	9.43	9.53	8.5		Center	8.63	9.3	9.07
	Top	9.9	8.83	8.37		Top	7	8.2	8.37
10	Bottom	8.33	8.77	8.5	20	Bottom	8.6	7.97	8.4
	Center	9.27	9.47	8.97		Center	8.8	8.47	8.93
	Top	7.57	8.5	8.6		Top	8.07	7.27	8.33

Table 2.6 *Averaged Sugar Content (%Brix)*

	SMALL	MEDIUM	LARGE
Bottom	7.696	8.5435	8.6115
Center	8.589	9.1885	9.201
Top	7.5195	8.0985	8.817

In the same study, Table 2.5 presents the sweetness measurements of all 60 watermelons in 3 different points. The data were gathered using a refractometer. Based on the acquired results, it showed that the highest sweetness index was measured at 10.07%Brix while the lowest was measured at 5.7%Brix. Table 2.6 shows the averaged results of Table 2.5. In essence, the bigger the watermelon, the higher the sweetness in %Brix of the watermelon.

Additionally, below is Table 2.7 which shows the averaged sample sizes and weight of the 60 watermelons used in the study of Pintor et al. (2016) as stated above. According to the national size classification standard, watermelon that weighs less than 3 kg is small, 3.1 - 4.0 kg is medium, 4.1 - 7.0 kg is large and greater than 7 kg is extra large.

Table 2.7 *Average Sample and Weight of Watermelon*

Size	Vertical Diameter (cm)	Horizontal Diameter (cm)	Weight (kg)
Small	200.32	191.33	3.83
Medium	227.35	216.5	5.21
Large	256.3	235.95	7.00

According to the project, “High-Tech Melon Thumper”, a watermelon ripeness sensor that was developed by Behr et al. (1999), a student from the University of Delaware, the frequency of the signal, when normalized using volume, has shown a promising correlation to the actual sugar content of watermelon. The size of the watermelon influences the frequency of its signal. Watermelon characteristic

frequencies have ranged from 100 to 250 Hz, corresponding to the desired sugar content of 8 to 12 percent.

Table 2.8 *Sweetness Level Categorization*

%Brix	Dissolved Sugar (g/L)	Sweetness Level
8	82.55	Low
12	125.81	Medium Low
14	147.96	Medium Low
16	170.47	High

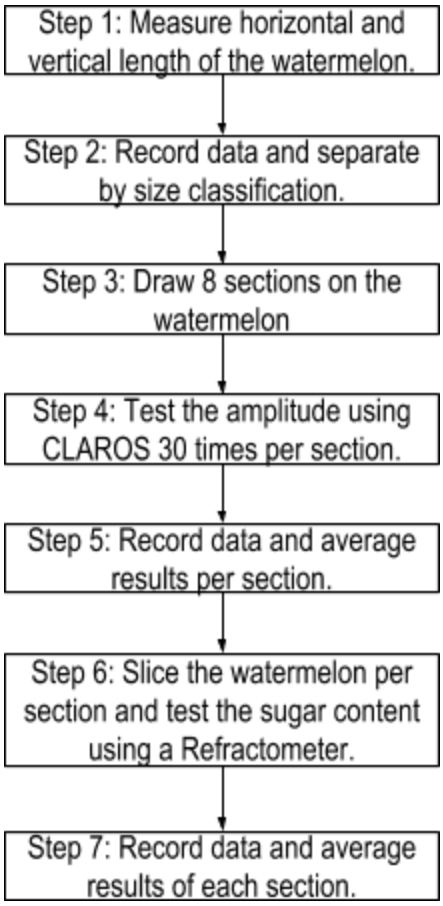
Moreover, Table 2.8 presents the standard for the sweetness level based on its actual sugar content measurement according to the study conducted by Reams (2013). The data obtained shows that the sugar content with measurement of 8, 12, 14, and 16 %Brix has a sweetness level of Low, Medium Low, Medium High and High, respectively. This will be the basis of the measurements to be made in this project.

Procedures

From Figure 2.2, there are seven steps to complete the collating of datasets. This would require the use of materials such as the Refractometer and the device to be created for this project named as CLARO (Citrullus LAnatus sugaR cOntent) testing facility. The former will obtain the sugar content of the watermelon; while the latter will obtain the amplitude.

Beforehand, the different watermelon size classification has been defined. Steps 1 and 2 from the same figure are the first two methods the engineers will perform to determine which watermelons are considered as Small, Medium and Large.

Figure 2.2 *Data Collection Process*



Afterwards, Steps 3 and 4 are to be done to increase the accuracy of the measurement of amplitude and sugar content. The watermelons will be sectioned into 8 parts (See Appendix A), and their amplitudes will be taken 30 times each section using the fabricated testing facility. The results will be recorded and averaged to be used as one of the two variables for the final datasets as referred in Step 5.

In Step 6, the watermelons will be sliced and segmented physically per section to test for their actual sugar content using a Refractometer. While in Step 7, the results for every part will also be recorded and averaged to be used as the second variable for the datasets.

Supervised Machine Learning

The most common sub branch of Machine Learning is the Supervised Machine Learning. The name *supervised* originates from the idea that training this type of algorithm is like having a teacher supervising or monitoring the whole learning process of a particular system. To simplify, there are correct outputs, training of the algorithm will run repeatedly and will only stop when the algorithm meets the acceptable level of result/output. When training a supervised learning algorithm, the training data will consist of inputs paired with correct outputs. During training, the algorithm will search for patterns in the data that correlate with the desired outputs. After training, a supervised learning algorithm will take in new unseen inputs and will determine which label the new inputs will be classified as based on prior training data. The objective of a supervised learning model is to predict the correct label for newly presented input data. At its most basic form, a supervised learning algorithm can be written simply as: Y=f(x); where Y is the predicted output that is determined by a mapping function that assigns a class to an input value x. The function used to connect input features to a predicted output is created by the machine learning model during training.

To apply the said algorithm, the developers will use python programming language in training the datasets for the detection of the watermelon's sugar content. In technical terms, Python is an interpreted, object-oriented, high-level programming language with dynamic semantics that can be used in a wide variety of applications. It is very attractive for Rapid Application Development, because of its high-level built in data structures, combined with dynamic typing and dynamic binding. Also, because of high-level data and many more features, it is useful for scripting or glue language to connect existing components together.

In line with the discussion above, in training the new datasets, the engineers will use different supervised machine learning algorithms to test which one is more accurate in detecting the watermelon's sugar content. First is the Support Vector Machine (SVM). It has been initially used since 1990 and has been applied to many applications related to engineering (Gholami et al., 2017). It uses a linear classification to classify data into two categories non-probabilistically. It is capable of countering the problem of overfitting in artificial networks (Peter et al., 2019). For linear classification SVM, a data point is viewed as a dimensional vector and a separate data point with a dimensional hyperplane.

Second known supervised machine learning algorithm is the Decision Tree. Hallinan (2012) stated that Decision Tree is a type of classifier that groups items into a set of categories. Those single items are

described through the taken sets of inputs. This type of algorithm is trained using a set of input examples which are labelled individually with the category to which it is included. It “examines the input data to determine which variable best distinguishes between the categories, and which values of this variable are informative. This variable forms the root of the tree. The remaining variables are scrutinized for the next-most-informative variable, which generates the second level of the tree”. The process repeats until maximum separation between the output categories is reached.

Third is Random Forest. It is a supervised machine learning algorithm which is also a classifier that evolves from many decision trees. It gathers the classifications and chooses the result that is the most voted prediction (Mao et al., 2012). According to Pal (2017), Forest prediction uses the randomized feature selection process wherein the tree is fitted based on the bootstrap sample generated from the training data.

Chapter 3

Design Options

After defining the solutions to the identified problems as presented in the previous chapter, these are now translated into a design. In this chapter, three system designs will be created and implemented while fulfilling the different constraints.

System Architecture

Figure 3.1 *System Architecture*

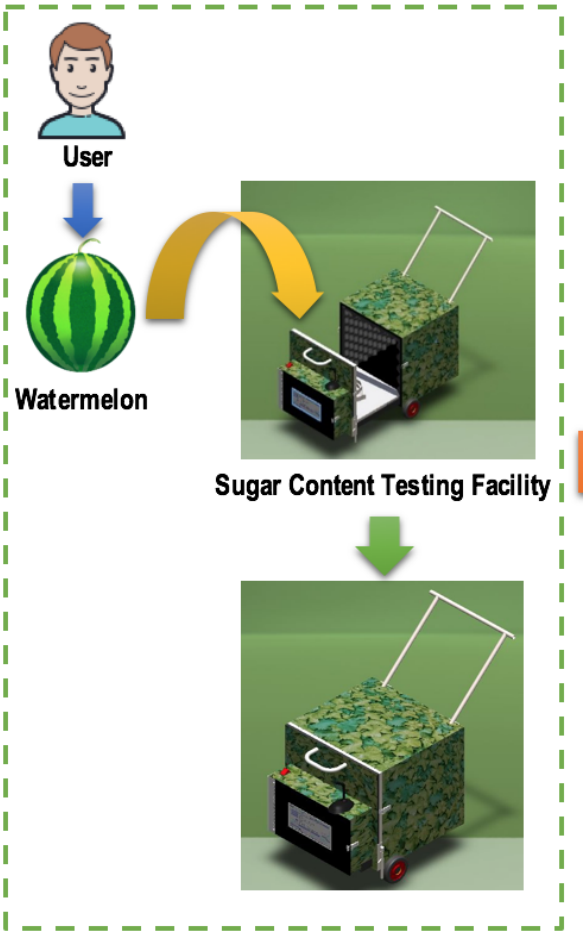


Figure 3.1 shows the system architecture of the project. The farmer, who is the user of the device, will take a watermelon for testing and will place it inside the 'sugar content testing facility'. The facility basically consists of a speaker, microphone, microcontroller, and a microprocessor. The microphone will be used as the input device to record the sound given off by the speaker in the system. Then, the recorded

frequency will be used for the sugar content prediction. The output, which is the sugar content of the watermelon (in percentage format using units of %Brix) will be displayed together with the sweetness level indicator from low to high. This will help the farmers to categorize their watermelons.

Block Diagram

Figure 3.2 *Block Diagram*

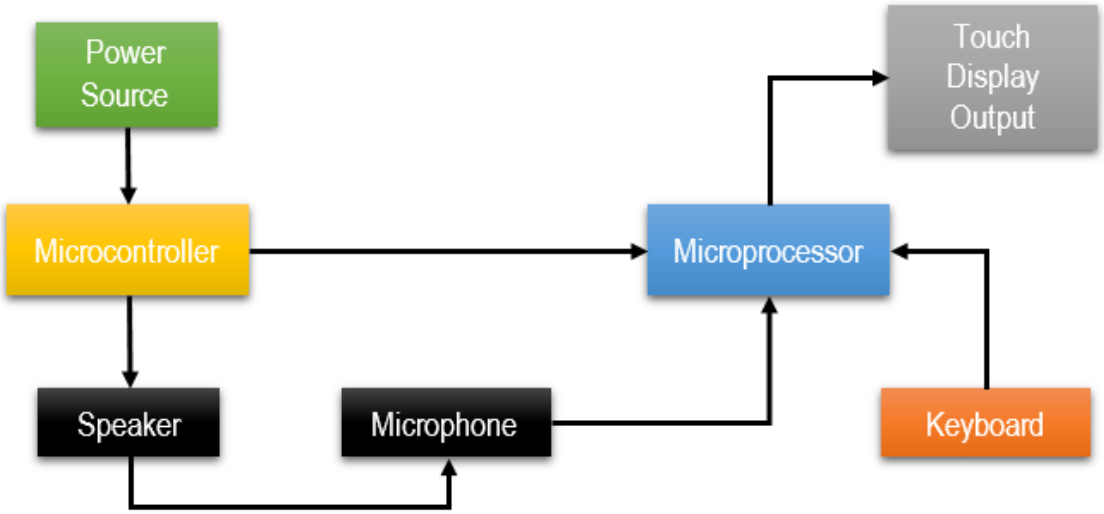


Figure 3.2 presents the block diagram of the relationship of each of the major hardware components to be used in the system. The process will start on the power source as it simply supplies the power for the microcontroller and the processor. As shown in the figure above, the microcontroller is tethered to the speaker as the microcontroller takes the sound frequency from the speaker and sends the received frequency to the microprocessor. The sound frequency that the speaker emits during the process that emanates by thumping a side of watermelon will be recorded by a microphone which is placed on the opposite side of the subject. During the process, the microprocessor will receive the frequency data gathered from the microcontroller and a desired output will be exhibited through the medium of the connected microprocessor.

Furthermore, the figure presented above demonstrated that a keyboard was connected to the processor. The keyboard is vital for the whole process because the system will require a registration before the process starts. The input devices to be used are an external keyboard and microphone; while the

output devices are speaker and LCD. All these input and output devices will be processed by the microcontroller.

Frequency Signal Generator

Figure 3.3 *Flowchart for Frequency Generator*

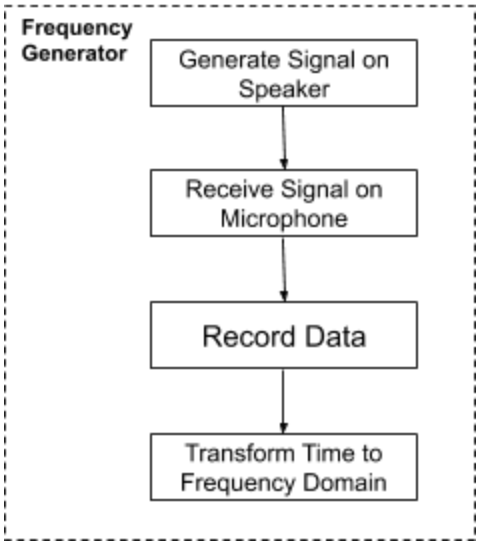


Figure 3.3 shows the process of generating frequency signals. A fixed 1600 Hz signal will be fed or transmitted to the speaker that will be received by the microphone. This will be recorded, and since the signal is in time domain, it will be transformed into frequency domain by means of coding in the program. The 1600 Hz frequency will be adapted from the study of Adalla et al. (2017) which they tested for three different frequencies (100 Hz, 150 Hz and 200 Hz) and concluded based on Analysis of Variance (ANOVA) data analysis that the frequency with the lowest variance will be utilized. The engineers have added 1500 Hz to the 100 Hz in their design since the prototype is enclosed in a box and requiring higher frequency.

Using these processes, the initial testing was performed. Three trials were made per constraint and design option and Table 3.1 below shows the values as a result of the testing which will be used later for the computations.

Table 3.1 *Testing Results*

	SPEED (sec.)	ACCURACY (%)	CPU USAGE (%)	MEMORY (MB)
Support Vector Machine	10.72	49.72	13	129.4
	10.6	51.27	15	130.7

	10.51	52.57	14	131.7
	10.4	43.53	15	130.6
	10.37	47.08	17	131.6
Decision Tree	10.32	47.24	15	132.6
	10.51	59.33	14	133.3
	10.57	61.75	14	134.4
Random Forest	10.5	62.81	15	135.4

These acquired data would be a great contributing factor in deciding which design option would be applied for the project. Also, the mean value for the trials performed will be computed using the formula:

$$\bar{x} = \frac{\Sigma\, trials}{number\ of\ trials} \qquad eq.\ 1$$

Design Option 1

The first Design Option uses the Support Vector Machine (SVM) algorithm. It is a supervised machine learning model that uses a linear classification which means that it divides the data into two groups, this is to separate the classes of the data to be used in a system.

Figure 3.4 *Design Option 1 Flowchart*

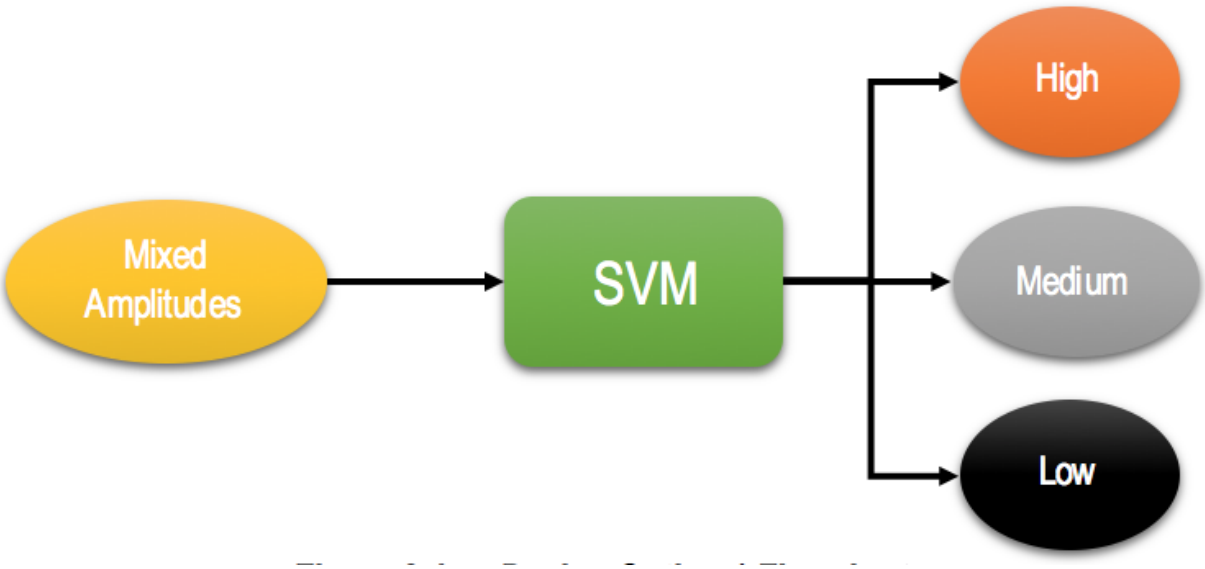


Figure 3.4 shows the process flowchart of a Support Vector Machine (SVM) algorithm. The idea of the algorithm for SVM is creating a line or hyperplane which separates data into classes. During the

process of watermelon sugar content detection, the mixed amplitudes that are present during the process will be categorized by the SVM algorithm into three sugar levels which are low, medium and high.

Now, computing for the mean values per constraint based on the initial testing and using this algorithm:

Speed

$$\overline{x_{speed}} = \frac{10.71 + 10.6 + 10.51}{3} = 10.61\ seconds$$

Accuracy

$$\overline{x_{accuracy}} = \frac{49.72 + 51.27 + 52.57}{3} = 51.19\%$$

CPU Usage

$$\overline{x_{CPU}} = \frac{13 + 15 + 14}{3} = 14\%$$

Memory Consumption

$$\overline{x_{memory\ consumption}} = \frac{129.4 + 130.7 + 131.7}{3} = 130.6\ MB$$

Table 3.2 *Design Criteria for Design Option 1*

Speed		10.61 sec
Accuracy		51.19%
Efficiency	CPU Usage	14%
	Memory Consumption	130.6 MB

Based on the table and computations shown above using SVM, the developers concluded that this algorithm has an average speed of 10.61 seconds, accuracy of 51.19%, CPU usage of 14%, and Memory Consumption of 130.6 MB. This is from adding all the output of the test. After getting the sum of each result, it is then divided by three or the number of times the test was conducted. Basically, the mean formula is used in each constraint.

Design Option 2

The second design option uses the Decision Tree algorithm. It is a supervised machine learning technique for inducing a decision tree from training data. A decision tree (also referred to as a classification tree or a reduction tree) is a predictive model which is a mapping from observations about an item to conclusions about its target value. Moreover, in the tree structures, leaves represent classifications (also referred to as labels), non-leaf nodes are features, and branches represent conjunctions of features that lead to the classifications (Mitchell et al., 1997).

Figure 3.5 *Design Option 2 Flowchart*

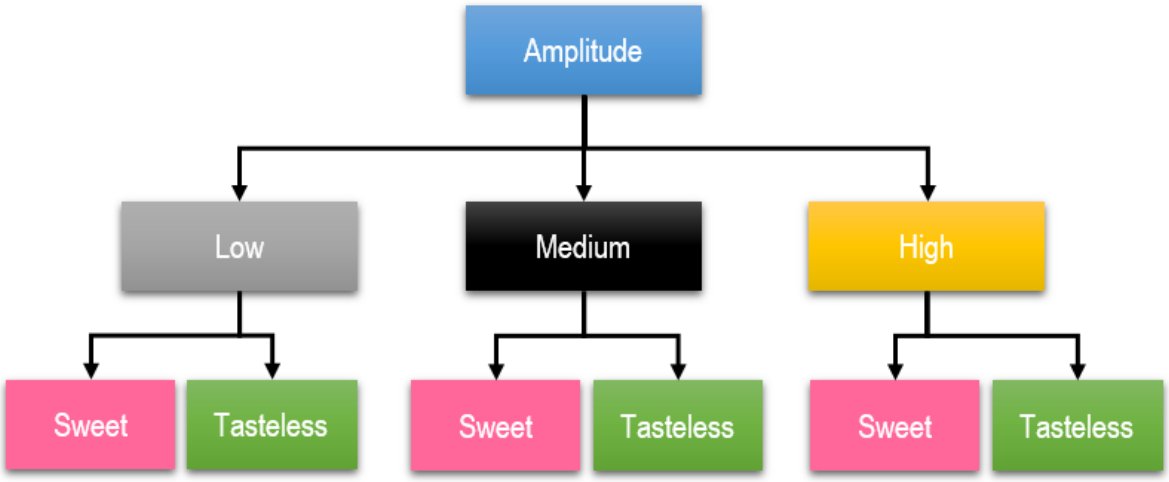


Figure 3.5 shows the flowchart of the Decision Tree algorithm that will be implemented to the system. At the top of the flowchart is the amplitude which is the input data. On that data there's a class of amplitude and that is the low, medium, and high class of amplitude. Lastly, each class corresponds with the category of sweet and tasteless. Based on collected data, the engineers were able to determine which class of amplitude has a sweet and tasteless category.

Now, computing for the mean values per constraint based on the initial testing and using this algorithm:

Speed

$$\overline{x}_{speed} = \frac{10.4 + 10.37 + 10.32}{3} = 10.36\ seconds$$

Accuracy

$$\overline{x_{accuracy}} = \frac{43.53 + 47.08 + 47.24}{3} = 45.95\%$$

CPU Usage

$$\overline{x_{CPU}} = \frac{15 + 17 + 15}{3} = 15.67\%$$

Memory Consumption

$$\overline{x_{memory\ consumption}} = \frac{130.6 + 131.6 + 132.6}{3} = 131.6\ MB$$

Table 3.3 *Design Criteria for Design Option 2*

Speed		10.36 sec
Accuracy		45.95%
Efficiency	CPU Usage	15.67%
	Memory Consumption	131.6 MB

As shown on Table 3.3, the results above present the mean value results using the DT algorithm. The output for the average speed shows 10.36 seconds, accuracy of 45.95%, CPU usage of 15.67%, and Memory Consumption of 131.6 MB. Although this algorithm detects quicker than the first, its accuracy is lower and it is less efficient.

Design Option 3

The third design option uses the Random Forest. It is an algorithm that develops from Decision Trees. Mao et al. (2012) mentioned in their study that the classification of data inputs are provided by each decision tree. Then, the random forest algorithm gathers the classifications and selects the best predicted result.

Figure 3.6 *Design Option 3 Flowchart*

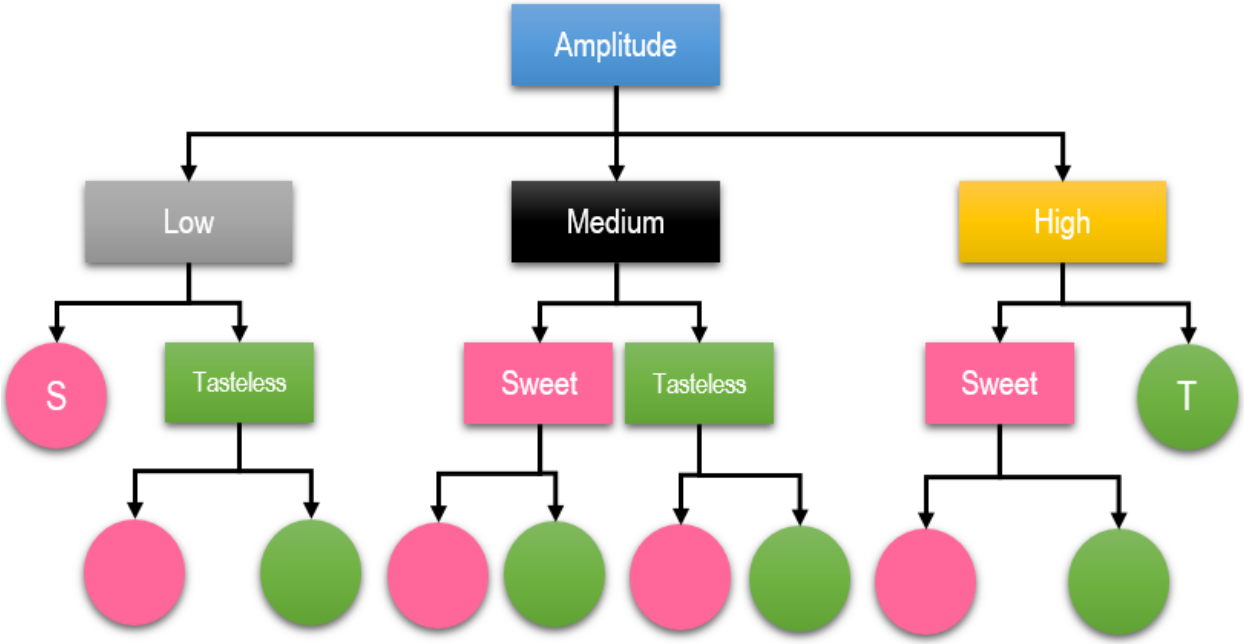


Figure 3.6 shows the process flowchart of a Random Forest algorithm. This is similarly close to the previous design option and actually contains several decision trees with additional subset features. These are the highest and lowest sugar content values from the testing done and based on their amplitude classification. Each tree has its predictions and on its final predictions, the outcomes predict if it is sweet or tasteless. Using the algorithm, three trials were also made to test for the system's speed, accuracy, CPU usage and memory consumption.

These values are averaged as shown on the equations below:

$$\overline{x}_{speed} = \frac{10.51 + 10.57 + 10.5}{3} = 10.53\ seconds$$

$$\overline{x}_{accuracy} = \frac{59.33 + 61.75 + 62.81}{3} = 61.3\ \%$$

$$\overline{x}_{CPU} = \frac{14 + 14 + 15}{3} = 14.33\ \%$$

$$\overline{x}_{memory} = \frac{133.3 + 134.4 + 135.4}{3} = 134.37\ MB$$

Table 3.4 *Design Criteria for Design Option 3*

Speed		10.53 seconds
Accuracy		61.3%
Efficiency	CPU Usage	14.33 %

	Memory Consumption	134.37 MB

Table 3.4 shows the result of the sample and initial testings performed using the mean values of the trials made per design constraint. Final result showed that the Random Forest algorithm has an average speed of 10.53 seconds from the start of run time to the display of results, detection accuracy of 61.3%, CPU usage of 14.33% and memory consumption of 134.37 MB. Hence, making it the second fastest detection and efficient algorithm, and the highest accuracy among the three.

Table 3.5 *Result Summary of the Design Options Trials*

DESIGN OPTION	CRITERIA		TRIAL 1	TRIAL 2	TRIAL 3	AVERAGE
	SPEED		10.72	10.6	10.51	10.61
	ACCURACY		49.72	51.27	52.57	51.19
	EFFICIENCY	CPU USAGE	13	15	14	14
SVM		MEMORY CONS.	129.4	130.7	131.7	130.6
	SPEED		10.4	10.37	10.32	10.36
	ACCURACY		43.53	47.08	47.24	45.95
	EFFICIENCY	CPU USAGE	15	17	15	15.67
DT		MEMORY CONS.	130.6	131.6	132.6	131.6
	SPEED		10.51	10.57	10.5	10.53
	ACCURACY		59.33	61.75	62.81	61.3
	EFFICIENCY	CPU USAGE	14	14	15	14.33
RF		MEMORY CONS.	133.3	134.4	135.4	134.37

As shown on Table 3.5, the design criterion: speed, accuracy, CPU usage and memory consumption are considered in choosing the best design option. The averaged results present that for speed, the second design option (DT) has the lowest detection time at 10.36 seconds. For accuracy, the third design option (RF) has the highest detection rate at 61.3%; while for the efficiency criteria, the first design option (SVM) has the preferable values. Under efficiency, the CPU usage only takes up 14% and the consumed memory is 130.6 MB.

Chapter 4

Constraints, Standards, and Trade-Offs

As previously mentioned, the criteria that were considered on this design project are: speed, accuracy and efficiency. In this chapter, comparisons and calculations of those three will be done to make an unbiased decision for what will be the best design for implementation.

Constraints and Standards

A. Speed

This criterion is regarding the detection speed of the overall system which has the second highest importance factor. The manual method for testing the sugar content of a single watermelon estimately takes a minute, and this project would be able to detect about 3-5 watermelons within 60 seconds.

B. Accuracy

This criterion has the highest importance factor among all constraints which is concerned about the value of correctness of the sugar content measurement. This testing facility's accuracy would be based on the 86% correctness value shown on the related study by Wei et al. (2012).

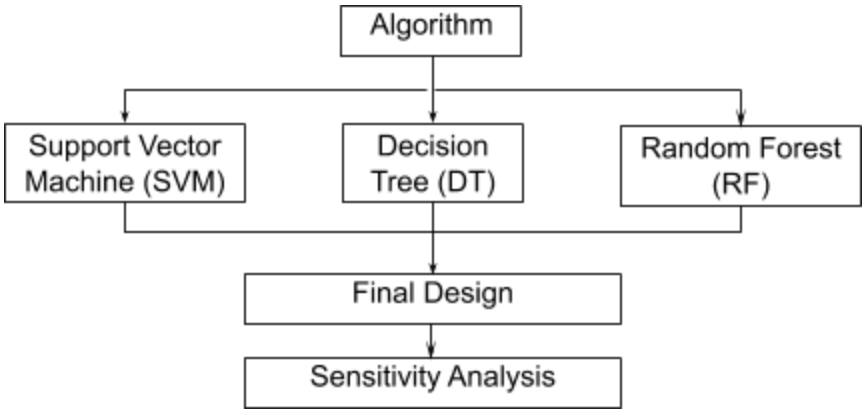
C. Efficiency

This criterion has the least importance factor and no definite standards has been provided. This deals with the algorithm's quality analysis in terms of memory consumption and CPU usage. These sub-constraints aim to have the algorithm consume the lowest possible memory, and CPU usage at regular percentage between 10 and 30% .

Trade-Off Development

Below, Figure 4.1 represents the process that will be done for the selection of the final design. From the trade-off analysis of the three algorithms, the final design will be selected. The algorithms will be applied to the different analyses to establish how the ranking of each design criteria is affected based on the changes from the criterion importance factor.

Figure 4.1 *Trade-Off Development*



In choosing the final design to implement, each design and criteria will be evaluated using the ranking equation as shown below:

$$Rank = (1 - |\frac{Higher\ Value - Lower\ Value}{Higher\ Value}|)\ x\ 5. \quad \text{eq. 2}$$

On the other hand, to compute for the overall ranking value, the equation to be used is:

$$Overall\ Ranking\ Value = \sum (i\ x\ (ranking)), \quad \text{eq. 3}$$

where i = Criterion Importance Factor.

Table 4.1 *Trade-Off Analysis*

TRADE-OFF ANALYSIS									
Design Criteria		**Unit**	**Criterion Importance Factor**	**Trade-Off Between**					
				SVM		**DT**		**RF**	
				VALUE	RANK	VALUE	RANK	VALUE	RANK
Speed		Second	20	10.61	4.88	10.36	5	10.53	4.92
Accuracy		%	60	51.19	4.18	45.95	3.75	61.3	5
Efficiency	CPU Usage	%	10	14	5	15.67	4.47	14.33	4.88
	Memory Consumption	Mega Byte	10	130.6	5	131.6	4.96	134.37	4.86
OVERALL RANK					448.4		419.3		495.8

Table 4.1 shows the trade-off analysis for the different design options. The criterion importance factor is based on the importance level the client mentioned during the interview. Using equations 2 and 3 above, RF has the highest overall ranking at 495.8, SVM has the second at 448.4, and DT has the least at 419.3. Hence, RF will be used for the final design.

Aside from the Trade-Off Analysis, Sensitivity Analysis will also be applied to the table information above to show how the rankings are affected by the random changes in each criteria importance factor.

Table 4.2 *Sensitivity Analysis 1*

Design Criteria		Unit	Criterion Importance Factor	SENSITIVITY ANALYSIS 1					
				SVM		DT		RF	
				VALUE	RANK	VALUE	RANK	VALUE	RANK
Speed		Second	60	10.61	4.88	10.36	5	10.53	4.92
Accuracy		%	20	51.19	4.18	45.95	3.75	61.3	5
Efficiency	CPU Usage	%	10	14	5	15.67	4.47	14.33	4.88
	Memory Consumption	Mega Byte	10	130.6	5	131.6	4.96	134.37	4.86
OVERALL RANK					476.4		469.3		492.6

Table 4.2 presents the first Sensitivity Analysis with changes in the importance factors: 60, 20, 10, 10, respectively. The overall ranking shows that RF has the highest at 492.6, SVM has the second at 476.4 and DT has the least at 469.3. Similar to Trade-Off Analysis, higher ranking means better design for the project.

Table 4.3 *Sensitivity Analysis 2*

Design Criteria		Unit	Criterion Importance Factor	SENSITIVITY ANALYSIS 2					
				SVM		DT		RF	
				VALUE	RANK	VALUE	RANK	VALUE	RANK
Speed		Second	20	10.61	4.88	10.36	5	10.53	4.92
Accuracy		%	20	51.19	4.18	45.95	3.75	61.3	5
Efficiency	CPU Usage	%	30	14	5	15.67	4.47	14.33	4.88
	Memory Consumption	Mega Byte	30	130.6	5	131.6	4.96	134.37	4.86
OVERALL RANK					481.2		457.9		490.6

Table 4.3 shows the second Sensitivity Analysis performed for each of the design options. SVM's overall rank shows a result of 481.2, DT with a result of 457.9; while RFwith 490.6. From these, the third design option, RF, has again the highest value implying a preferable design for the testing facility.

Table 4.4 *Sensitivity Analysis 3*

Design Criteria		Unit	Criterion Importance Factor	SENSITIVITY ANALYSIS 3					
				SVM		DT		RF	
				VALUE	RANK	VALUE	RANK	VALUE	RANK
Speed		Second	20	10.61	4.88	10.36	5	10.53	4.92
Accuracy		%	30	51.19	4.18	45.95	3.75	61.3	5
Efficiency	CPU Usage	%	25	14	5	15.67	4.47	14.33	4.88
	Memory Consumption	Mega Byte	25	130.6	5	131.6	4.96	134.37	4.86
OVERALL RANK					473		448.25		491.9

Figure 4.4 exhibits the results of the third Sensitivity Analysis conducted for the three design options. The overall rank for SVM is 473, for DT is 448.25, and for RF is 491.9. Given those values, it can be conveyed upon that RF is the top choice for implementation.

Table 4.5 *Sensitivity Analysis 4*

Design Criteria		Unit	Criterion Importance Factor	SENSITIVITY ANALYSIS 4					
				SVM		DT		RF	
				VALUE	RANK	VALUE	RANK	VALUE	RANK
Speed		Second	25	10.61	4.88	10.36	5	10.53	4.92
Accuracy		%	25	51.19	4.18	45.95	3.75	61.3	5
Efficiency	CPU Usage	%	25	14	5	15.67	4.47	14.33	4.88
	Memory Consumption	Mega Byte	25	130.6	5	131.6	4.96	134.37	4.86
OVERALL RANK					476.5		454.5		491.5

Table 4.5 shows the last Sensitivity Analysis applied to the design options to decide on which would be the final design. The overall rankings for SVM, DT and RF are shown as 476.5, 454.5 and 491.5, respectively. That being so, the most appropriate design would be RF as it has the highest ranking.

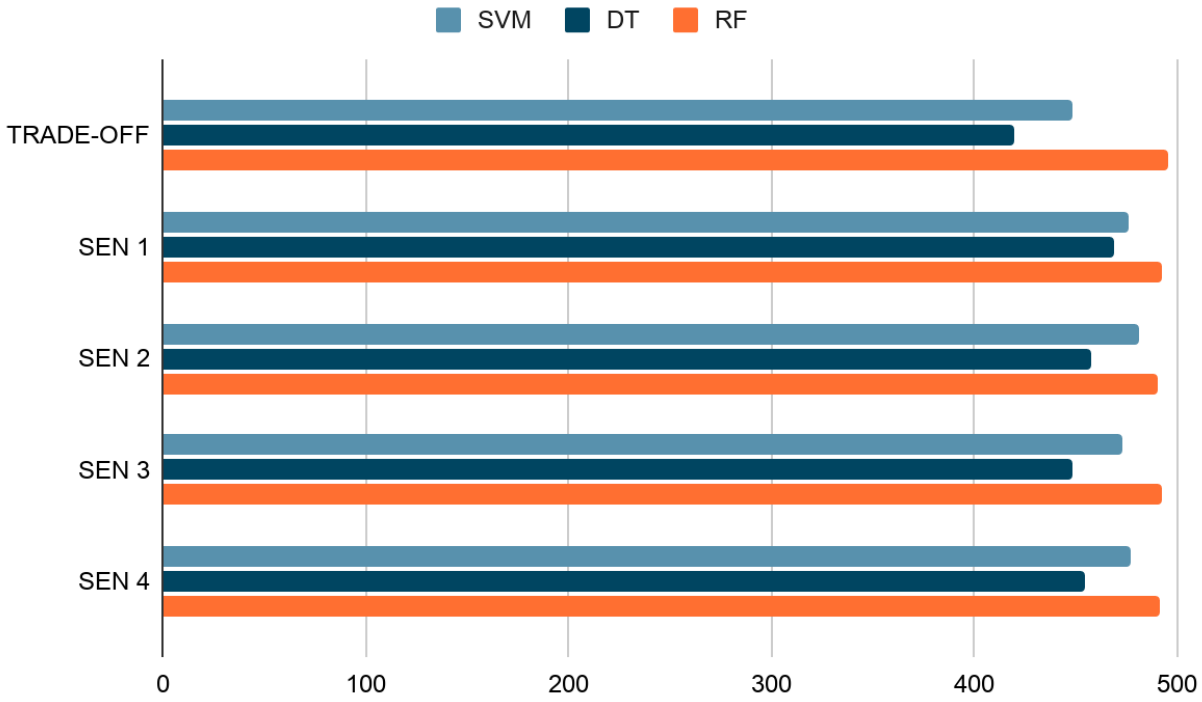
Table 4.6 *Summary of Analyses*

ANALYSIS	SPEED %	ACCURACY %	EFFICIENCY		SVM	DT	RF
			CPU USAGE %	MEMORY CONSUMPTION %			
Trade-Off	20	60	10	10	448.4	419.3	495.8
SEN 1	60	20	10	10	476.4	469.3	492.6
SEN 2	20	20	30	30	481.2	457.9	490.6

SEN 3	20	30	25	25	473	448.25	491.9
SEN 4	25	25	25	25	476.5	454.5	491.5

Table 4.6 shows the summary of the overall rankings of the analyses done for every design option (SVM, DT and RF) and criterion importance factor. The results reveal that RF is dominating even with the arbitrary variation in importance factors. Thus, it can be concluded that RF is the best design to be implemented in the project.

Figure 4.2 *Sensitivity Analysis Results Graph*



As shown in Figure 4.2, there are 4 Sensitivity Analyses (SEN) performed aside from the Trade-Off Analysis for the Design Options: SVM, DT and RF which are represented on the y-axis of the figure. On the other hand, the x-axis represented the overall rankings. Also, the graphical representation above presents the summary results of Table 4.6. From this bar chart, one can easily see that for every analysis, RF is leading and holds the highest overall ranking.

Chapter 5

Final Design

After the analysis performed to identify the best algorithm to use, this chapter provides the information about what the actual and implemented design for the project.

Final System Architecture

Figure 5.1 *System Architecture of Overall System*

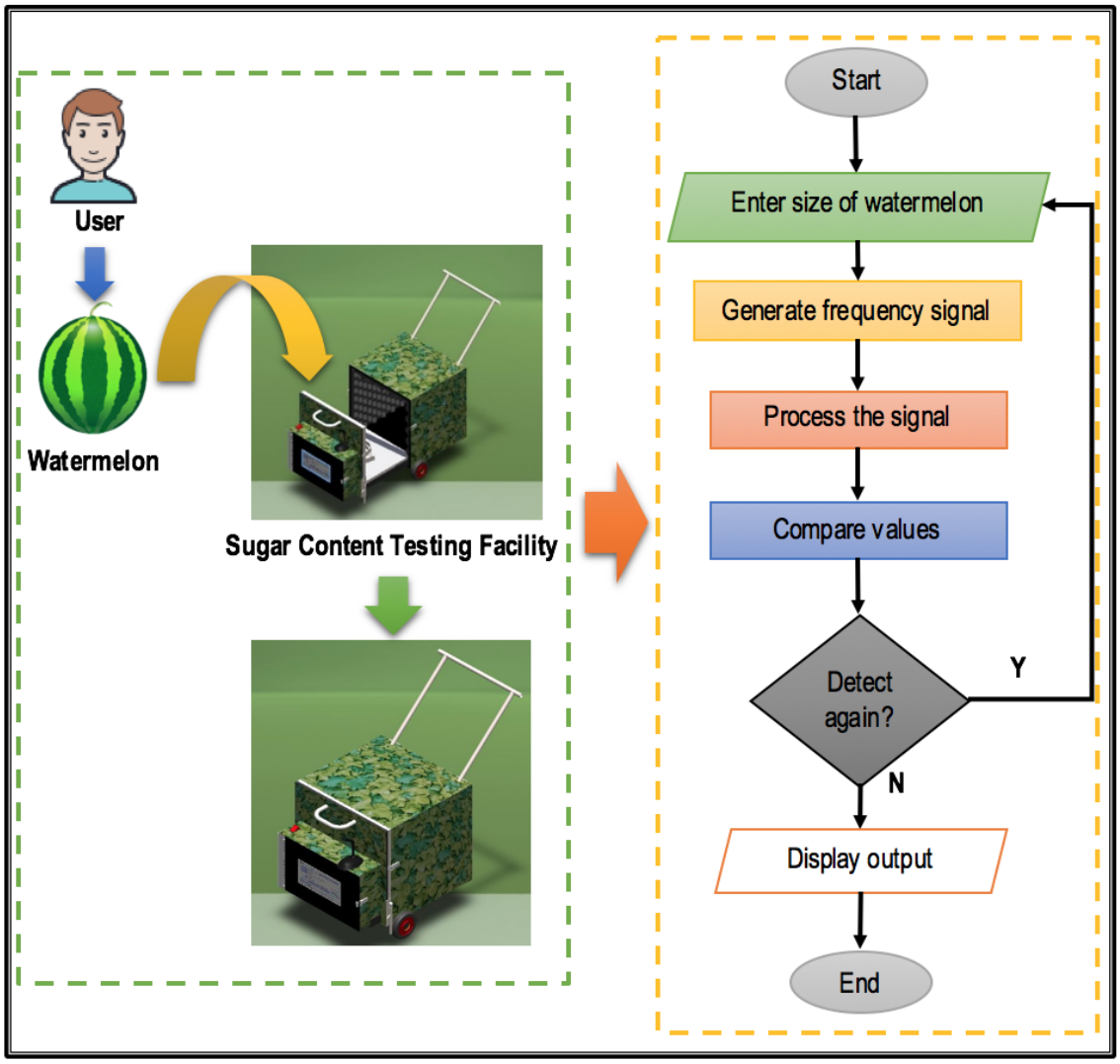
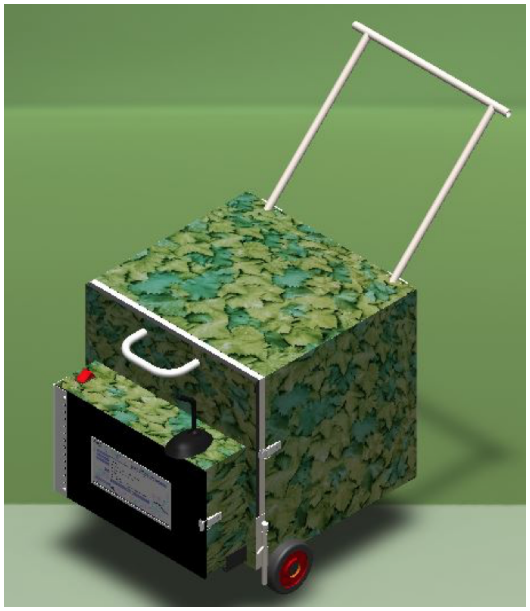


Figure 5.1 presents the overall System Architecture for the project. This is the aforementioned architecture in the previous chapter. Aside from the hardware part, it also shows the detection process that happens inside the testing facility. The process contains a flowchart representation of how the system

would operate. To begin, the watermelon size must be determined first. Then the frequency signal will be generated by the emitted sound of the speaker, and the recorded signal acquired by a microphone will be processed by the microcontroller. The microcontroller will also be responsible for comparing the detected value from the reference value as stated in this project using different algorithms. Last process is a decision whether to start a new detection or not. If yes, it will loop back to identifying the watermelon size. Otherwise, the output will be displayed on the LCD screen and the programs will end.

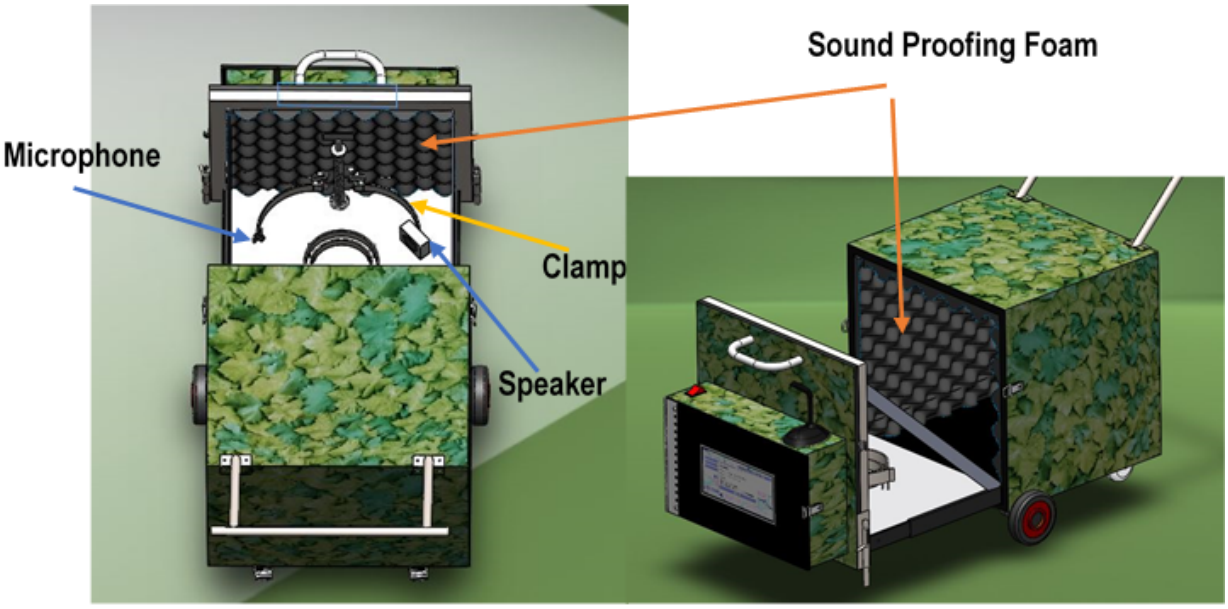
Prototype 3D Design

Figure 5.2 *Prototype Exterior Design*



To solve for the external noise intercepted by the system during frequency recording of the watermelon, Figure 5.2 shows that the watermelon testing will be done while enclosed in a wooden box type facility which measures 19x19x19 inches. To make it easier to use and more accessible in the farm, there will be wheels attached on it. In front of the device, a Raspberry Pi Touch Display is to be placed.

Figure 5.3 *Prototype Interior Design*



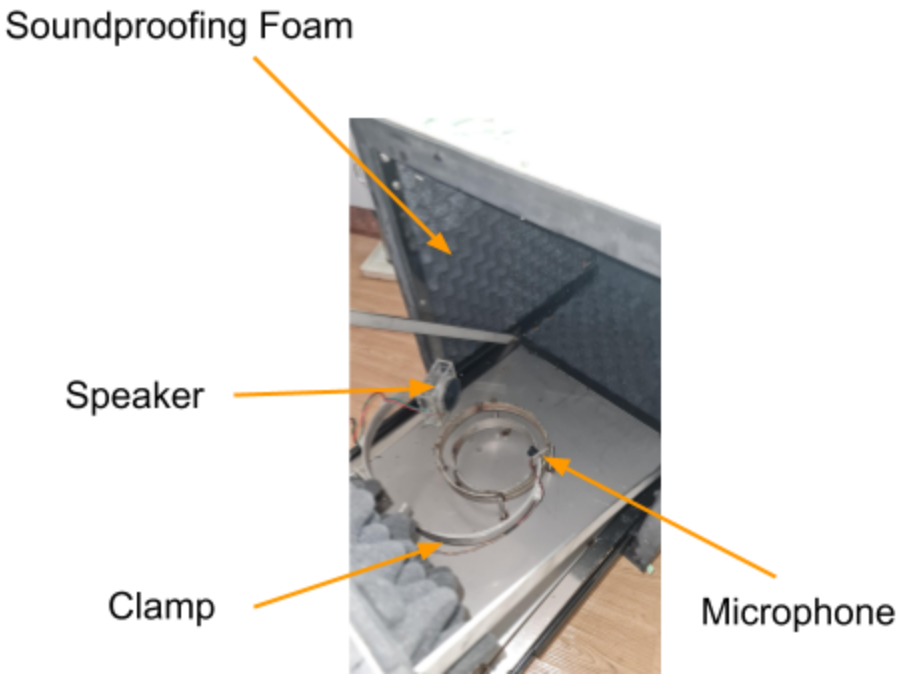
On the other hand, to reduce or eliminate the unwanted total internal noise, Figure 5.3 shows the internal design for this project. When the watermelon is entered into the insulator box, the clamp will be used to hold the fruit. At the same time, one side of the clamp will be used as a holder for the speaker which will transmit a signal. While the other clamp will be used for the microphone that will act as the receiver for the transmitted signal. The inside of the box will be covered by an egg crate foam which measures two inches on each surface.

Actual Prototype

Figure 5.4 shows the actual testing facility fabricated for the sugar content detection. Similar to the 3D design, the wooden box facility has a 19x19x19 inches measurement with LCD touch display for easy interactive navigation, outside microphone to record outside noise, wheels and handle for portability.

Figure 5.4 *Actual Exterior Design*

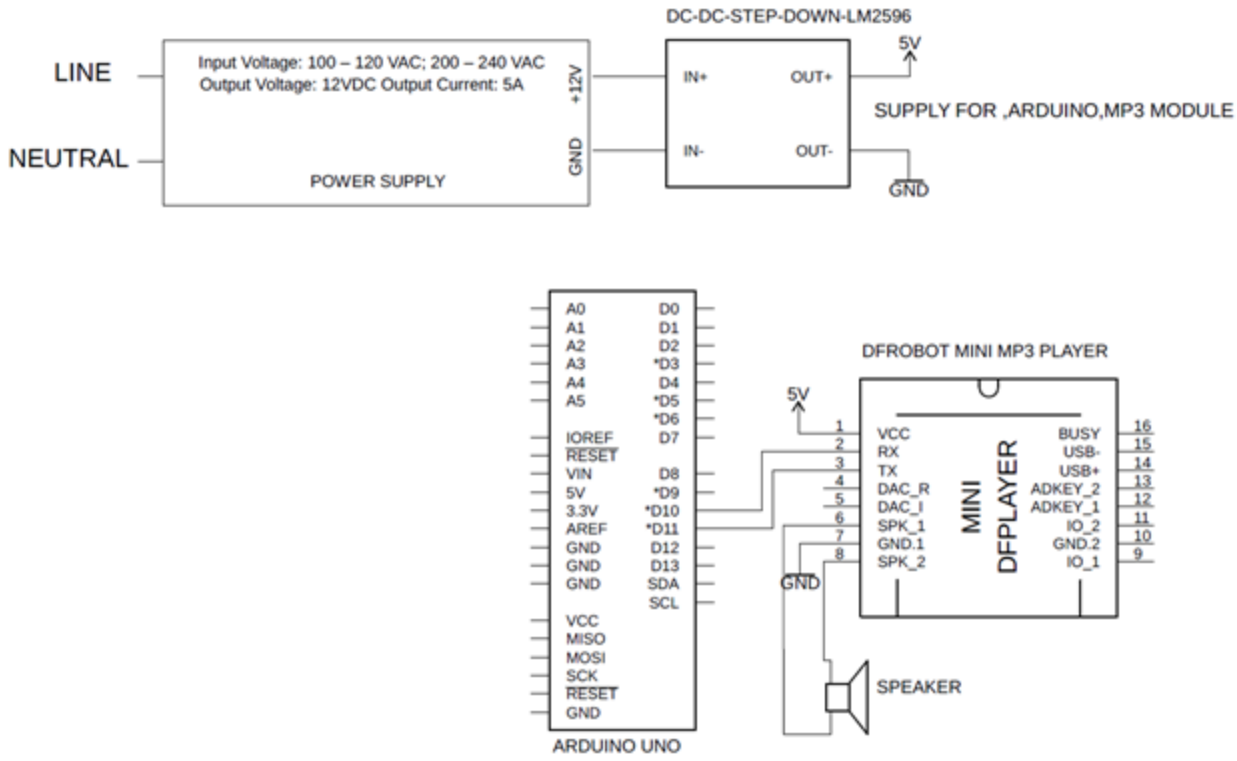
Figure 5.5 *Actual Interior Design*



Likewise, Figure 5.5 presents the actual interior of the prototype produced. The inside materials shown are the microphone to record the sound given off by the speaker, the clamp to hold the watermelon in place and the egg crate foam for soundproofing. The component box is also shown above which contains the power supply, microcontroller and microprocessor.

Schematic Diagram

Figure 5.6 *Schematic Diagram 1*



In Figure 5.6, it shows the schematic diagram of the connections in the power supply, and the Arduino. For the prototype to work, it needs a 200 - 240 VAC. This power supply is connected to the LM2596 which will be connected to the arduino. The input connections of this device are both connected to the power supply which releases 12V. From 12V, the LM2596 will have an output of 5V. LM2596 is also called a step-down switching regulator. It is capable of driving a 3-A load with excellent line and load regulation. These kinds of devices are available in fixed output voltages of 3.3V, 5V, 12V, and an adjustable

output version. Hence, the structure of the schematic diagram. The pins D10 and D11 in Arduino were used to connect to the DFROBOT MINI MP3 PLAYER, this player is a small MP3 module with a simplified output directly to the speaker. The module can be used as a stand alone module with attached battery, speaker and push buttons or can be used with an Arduino Uno connected to it, just as shown in the schematic diagram.

Figure 5.7 *Schematic Diagram 2*

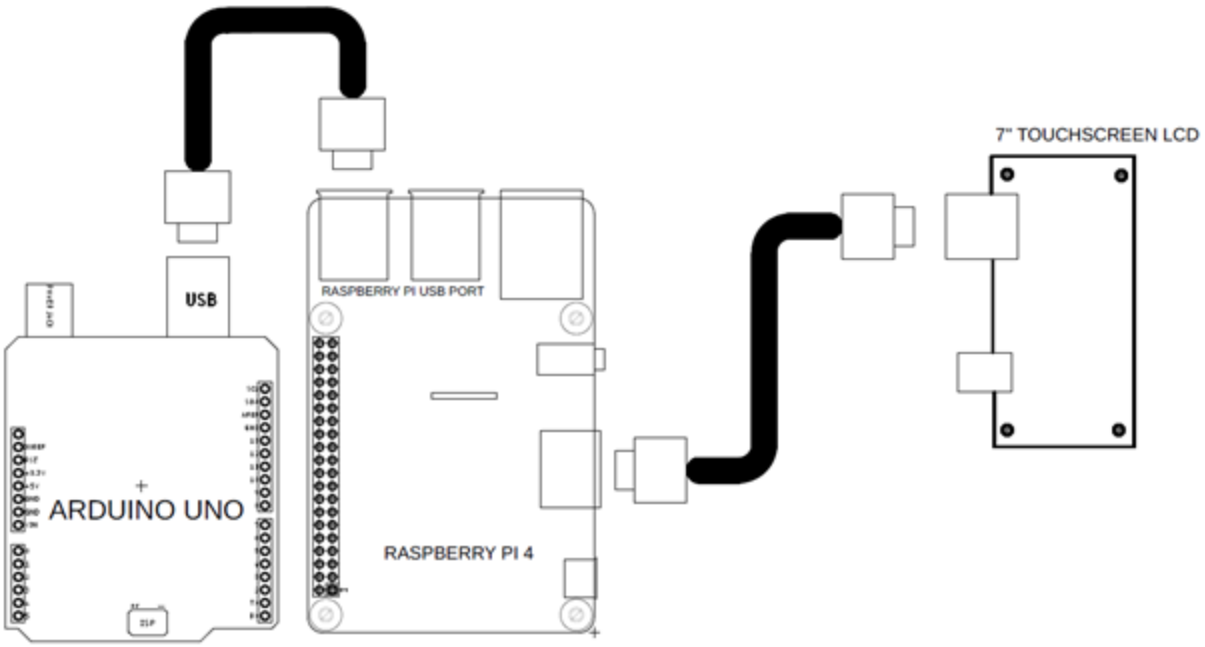
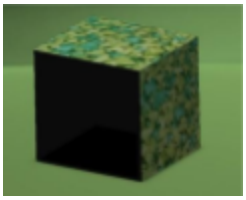
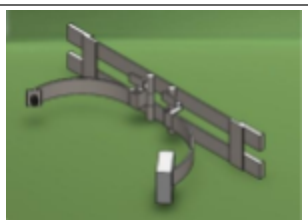
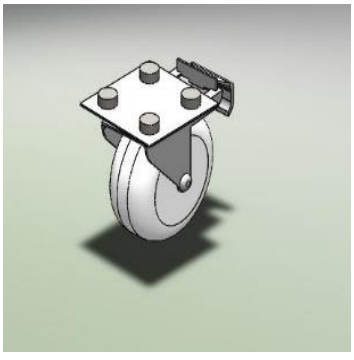
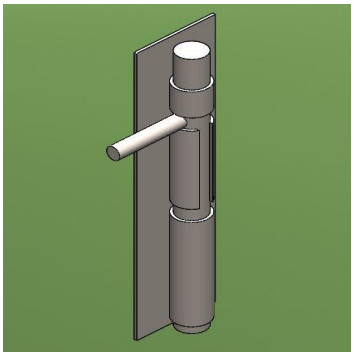


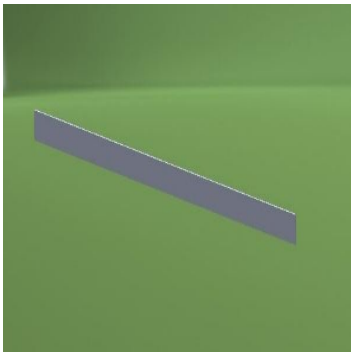
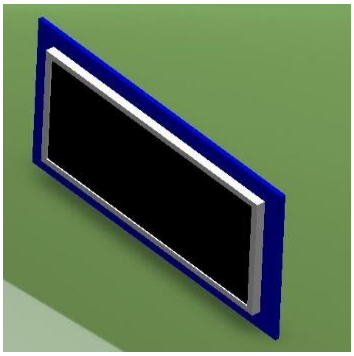
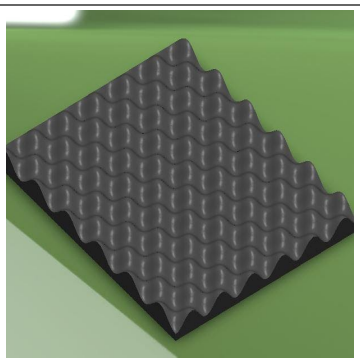
Figure 5.7 shows the diagram connection of Arduino Uno to Raspberry Pi to the LCD. Two different microcontrollers were used for the connections of the different components. Arduino boards is an open-source electronics platform based in easy-to-use hardware and software. In this case, the raspberry pi is connected to the Arduino because the touchscreen LCD works better in this microcontroller. The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. Raspberry pi has more USB ports that can be used in connection with anything the user can use to control the machine like keyboard and LCD.

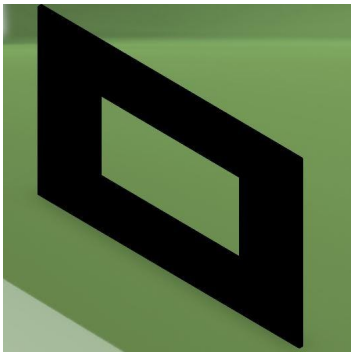
List of Materials

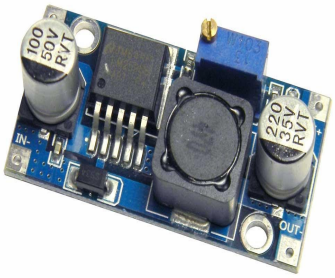
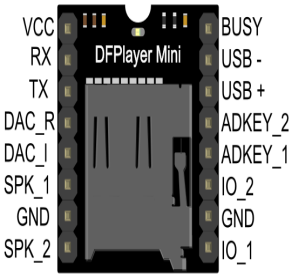
Table 5.1 *Materials List*

Quantity	Material	Image
1	Wooden Box	
1	Adjustable Clamp Tester	
2	Microphone	
1	Speaker	
1	Caster Wheel (Back wheel)	

2	Drawer Slides	
1	HInges	
2	Box Supporter Stand	
3	Lock	

2	Flat Bar	
2	Front Wheel	
1	Raspberry Pi Touch Display	
4	Soundproofing Foam	

1	Silicon Glass (covered by color black)	
1	Arduino Uno	
2	Raspberry Pi 4 Model B	
2	Power Supply for Raspberry Pi 4 Model B	

1	GSM Module	
1	Step-down Voltage Regulator	
1	Extension cord	
1	DFPlayer Mini	

1	I/O Switch	
1	Refractometer	

Table 5.1 shows that list of materials used in building the prototype. Here are the uses of each component listed above:

1. Wooden Box
    - The wooden box is used as the base of the whole prototype. The engineers chose the wooden box because it is lightweight compared to metal, and wood can also be used to avoid unwanted noise.
2. Adjustable Clamp Tester
    - This material is placed inside the box, and connected in it is the microphone and the speaker. In between these two components is where the watermelon will be placed.
3. Microphone
    - This component is used to record sound signals coming from the surroundings (external) and sound emitted by the speaker inside the box (internal).
4. Speaker
    - This component is used to transmit sound signals and located inside the wooden testing facility.

5. Caster Wheel (Back Wheel)
    - To make the project easier and more convenient to use, the engineers added the caster wheel so that the prototype can be transferred from one place to another without the need of carrying the whole prototype itself.
6. Drawer Slides
    - Drawer Slides are used to support the opening of the box. Since the opening of the box is in front and not on top, the engineers decided to make the opening a sliding one, this is also not to ruin the connection of the wires inside the box.
7. Hinges
    - This material is used to support the opening of where all the components are placed such as the microcontrollers and the screen.
8. Box Supporter Stand
    - This material is used to support the opening/door of the box. Since it is sliding, and attached to the opening are the different components, this part is a bit heavy. To support this part of the box, a box supporter stand was added so that when it is opened, all components are safe and will not be affected and moved.
9. Lock
    - Locks were added on the part where the components, such as microcontrollers are placed so that it is protected and will not be affected when we're opening the box. Some locks were also placed at the back wheel to support the box in a flat surface.
10. Flat bar
    - Flat bar is used as the clamp of the watermelon where each end has the microphone and the speaker.
11. Front Wheel
    - Just like the back wheel, this material is also added so that the user can move the prototype from one place to another.

12. Raspberry Pi Touch Display
    - This material is one of the most important components in the project. This is where the output is displayed and this is where the user will input their login information, and run the program.
13. Soundproofing Foam
    - This material was added to reduce the noise inside the box. Since one of the problems encountered in the related study of this project is the noise, the use of this material was maximized that will help prevent this problem.
14. Silicon Glass
    - Silicon Glass is placed around the Raspberry Pi Touch Display to support and protect it.
15. Arduino Uno
    - Arduino is a microcontroller, this is where some components were connected for the project like the microphone that's placed outside the box.
16. Raspberry Pi Model B
    - Just like Arduino, Raspberry Pi is also a microcontroller. A Raspberry Pi was used because this component has some parts that are limited to Arduino, like the USB port. Raspberry Pi has more USB ports than Arduino, that is why Raspberry Pi was found out to be very useful in building the prototype. One of the components connected to it is the Touch Screen Display.
17. Power Supply for Raspberry Model B
    - A power supply was added for the Raspberry Pi to function.
18. GSM Module
    - GSM Module is a chip or device that establishes a connection between a computing device and a GSM system. This module was used to easily connect to the device. It is very useful especially when some updates were added to the program and wanted to test it in the prototype.

19. Step-down Voltage Regulator
    - This component is used to provide appropriate supply of voltage to the microprocessor; it decreases high voltages to what the CPU can only handle.
20. Extension Cord
    - Since there are different connections, the engineers decided to add an extension cord to make the connection for the power supply not scattered.
21. DFPlayer Mini
    - This component is connected to the Arduino that is used to supply sound/output for the speaker to release.
22. I/O Switch
    - A switch is added to control the use of the whole prototype and to prevent the components from overheating.
23. Refractometer
    - This material is used to be able to see the sweetness of the watermelon, manually. This device can measure the sweetness level of the watermelon by just adding a small amount of liquid of watermelon in it. This was used for gathering of data for all the 30 watermelons.

Graphical User Interface

Figure 5.8 *Homepage*

In Figure 5.8, it shows the Homepage user interface of Citrullus LAnatus sugaR cOntent (CLARO) and the logo. This page consists of the functional buttons: Login, Register and Shut Down. Login button allows existing users to gain access through the system. The Register button allows a user to create a new account; while the Shut Down button enables the user to terminate the program.

Figure 5.9 *Registration Page*

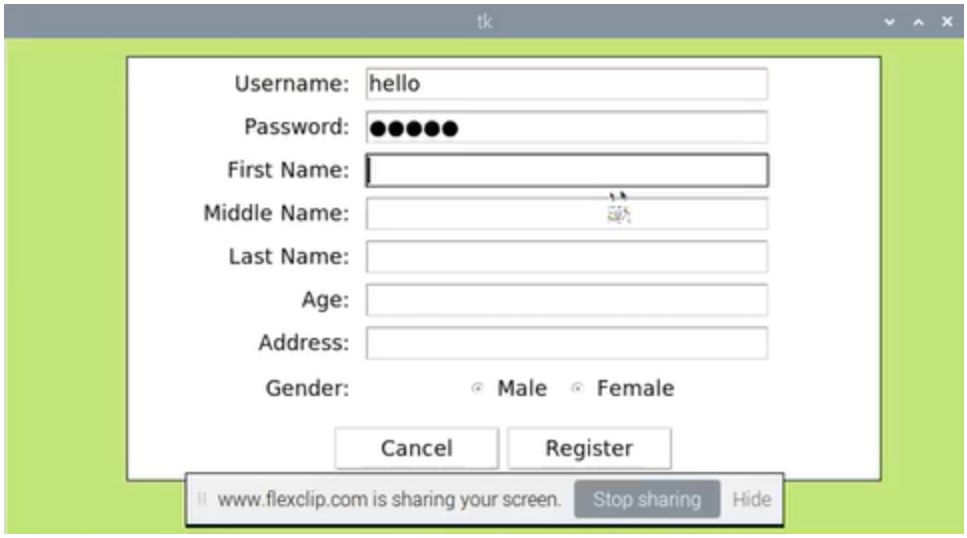


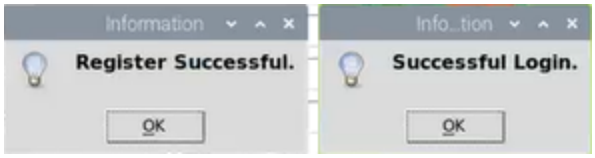
Figure 5.9 shows the Registration page of the system. It will ask the user to input a unique username, password, first name, last name, age, address and gender. It also contains the Register and

Cancel functional buttons. The former enables the user to successfully create a new account; while the latter disregards any input and exits out the window.

Figure 5.10 *Login Page*

Figure 5.10 presents the Login page for existing users. They will be required to input their username and passwords. It contains the Login and Cancel functional buttons. The former enables the user to successfully login and gain access to the system; while the latter disregards any input and exits out the window.

Figure 5.11 *Popup Boxes*



In Figure 5.11, it shows samples of user interface for popup boxes used to indicate if the registration or login has been successful or has failed.

Figure 5.12 *Startup Page*

Figure 5.12 presents the page where it takes the user once login is successful. It has two functional buttons, namely, Start and Log Out. The former takes the user to the Detection page; while the latter logs out the user from the system and loses access.

Figure 5.13 *Detection Page*

Figure 5.13 shows the user interface for the Detection Page. It will display the details of the person logged in such as the user's full name, age, address and gender. Due to limited resources of the fruit, the watermelons are no longer segregated based on its sizes. Hence, the size options are non-functional. The

page also consists of two working buttons named as Process and Back. The former starts the detection of the sugar content from sending off the sound signal to displaying the results; while the latter brings the user back to the Startup page.

Figure 5.14 *Results Window*

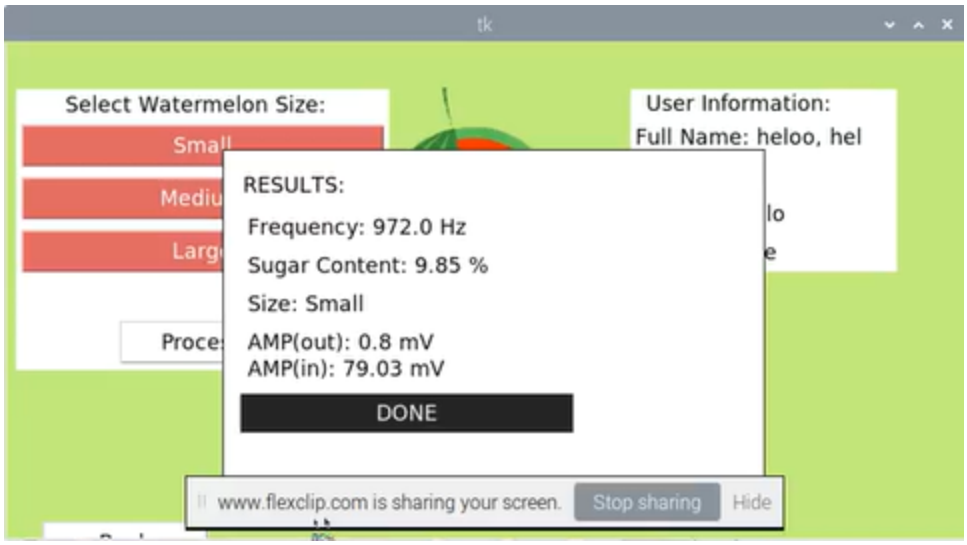


Figure 5.14 presents the window for the detection results. It contains the information like the frequency given off by the speaker, the predicted sugar content, the sound's amplitude computed from the microphone outside and inside. The Done button is selected to get back to the Detection Page.

System and Prototype Testing

System Testing

Table 5.2 *Test Cases Information*

TEST CASE NAME	TEST CASE ID
Creating/registering a user	01
User login	02
User logout	03
View User Information	04

Start Processing/Detection	05
Shut Down	06
Error Messages/ Pop-up windows	07

Table 5.2 shows the list of test cases assigned in our unit testing with their corresponding Test Case ID for better and easier identification. It consists of seven test cases that will be used as a checklist for the developer's actual testing.

Table 5.3 *Developer's Test*

Test Case Number	Test Condition/ Scenarios	User's Name	Expected Output	Actual Output	Remarks
01	Registering with incomplete information	Admin	Pops up an error message; failed registration	Pops up an error message; failed registration	N.A.
01	Creating a taken username	Admin	Pops up an error message; failed registration	Pops up an error message; failed registration	N.A.
02	User login correct	Admin	Successful login; Directs to Startup Page	Successful login; Directs to Startup Page	N.A.
02	Incorrect username/ password	Admin	Unsuccessful login; Popup error message	Unsuccessful login; Popup error message	N.A.
03	User logout	Admin	Successful logout; Directs to Login Page	Successful logout; Directs to Login Page	N.A.
04	Matching user Information	Admin	Shows matched details as the user registered	Shows matched details as the user registered	N.A.
05	Starting the Process/Detection	Admin	Pops up Results window	Pops up Results window	N.A.

05	Display of Detection Results	Admin	Shows complete information per label	Shows complete information per label	N.A.
06	Shut Down function	Admin	Entire system/ window successfully closes	Entire system/ window successfully closes	N.A.
07	Pop-up windows	Admin	All functional buttons showing desired information	All functional buttons showing desired information	N.A.

In Table 5.3, various scenarios and conditions were given per test case to find out the different outputs it will generate considering Admin's access. However, for CLARO's program, there are no contrasting and definite access limitations given to the administrator or any user. Hence, all the results as shown above are applicable to any person who is logged in. Lastly, it turns out that the actual outputs are the same with the expected ones.

Prototype Testing

This testing was made to identify how much of the external noise is reduced using the soundproofing foam and wooden device. The microphones from outside and inside of the box recorded the natural noise simultaneously; then results were compared to know the noise reduced. Given this, the engineers found out that the value of the sound signal or noise doesn't change internally at 26.9 dB.

Table 5.4 *Prototype Testing*

TEST NUMBER	EXTERNAL NOISE (dB)	NOISE REDUCED (dB)
01	65.2	38.3
02	64.3	37.4
03	60.8	33.9
04	65.9	39
05	59.1	32.2
06	63.2	36.3

07	64.6	37.7
08	61.9	35
09	63.0	36.1
10	65.1	38.2
	AVERAGE	36.41

As shown in Table 5.4, the external natural and man-made noise were recorded for ten times under the same environment. These averaged external noise results were acquired within the duration of ten seconds. The Noise Reduced column was determined by subtracting the constant internal sound value of 26.9 dB from the third column values. In average, the results present that 36.41 dB is the amount of noise reduced using the device.

Summary of Findings

The primary objective of this project is to build a device that determines watermelon's sugar content with Supervised Machine Learning using Acoustic Property. This study relates to "Review on the recent progress of non-destructive detection technology for internal quality of watermelon of Dengfei Jie and Xuan Wei"; however, the encountered problems in that study were considered in this study. Those are (1) hard to obtain internal quality (2) low precision, and (3) easily disturbed by the environment. With the said problem, the developers came up with the objectives of (1) determining the sugar content of watermelon without having to destroy the fruit; (2) uses acoustic technique to improve the accuracy of detection. To understand further the problem stated and to know if there really is a need to build such a device, the engineers did some research about watermelon, and interviewed people that have broad knowledge about the main topic, which is the watermelon. With the interviews and researches conducted, they concluded that creating this project will be beneficial to the farmers, specifically to the farmers in Bani, Pangasinan, where the best watermelon in the country is produced.

First of all, for the engineers to know what materials to use and algorithms to apply in the said project, they did some research and studies on how to reduce the noise surrounding the watermelon and the best algorithm that will accurately detect the sweetness of the watermelon. To cancel the noises that will

affect the detection, the engineers decided to make a wooden box and place a foam inside it, and since the project will use acoustic property, they also placed the speaker and the microphone inside the box. The watermelon will be placed between these two components, the speaker will generate a sound, and the microphone is the receiver of the sound. Through this, the device must be able to detect the sugar content of the watermelon. To make this possible, they applied a Supervised Machine Learning in the prototype. Then they tested different algorithms namely, Support Vector Machine, Decision Tree, and Random Forest. Each algorithm was tested in 30 watermelons to be able to know which one gives the highest speed, accuracy, and efficiency. In Chapter 4 of the documentation is the trade-off analysis that shows the ranking of each algorithm. With that, the engineers concluded that the Random Forest (RF) algorithm gives the best result and will be used in detecting the sweetness of the watermelon.

Conclusion

The project initially sought to create a device using acoustic property technique to measure the sugar content of watermelon. Incorporating sound properties made noise a problem. Hence, the engineers thoroughly thought of how to reduce the noise, and it was tested that the isolation wooden box and soundproofing foam are effective materials in reducing the noise from the sound recorded. An average value of 36.41 dB is reduced from the unwanted noise in the surroundings.

Also, among the algorithms used to improve the quality of detection in the sugar content of watermelon, Random Forest is the best design to be implemented. Winning from all of the analyses conducted made it more reliable to use. However, due to limited time and resources for data gathering, the engineers weren't able to reach the 86% accuracy rate of the related study of Wei et al. (2012).

Moreso, the results show that there are no significant changes with the sugar content depending on the watermelon size tested. It can also be concluded that from the 8 sections made during testing, the sweeter part with the higher sugar content is from Sections 1 to 4 located above the yellow splotch. Lastly, Sugar Baby watermelons from Bani, Pangasinan can be classified as having a medium low sweetness level.

Recommendation

The following are the suggestions of the engineers to the beneficiaries of this project. Given the lack of accuracy of the output, it would be advisable to increment the number of watermelons to be tested to increase the correctness value of the sugar content measurement. To improve the datasets, it would also be better to use all classifications of watermelon sizes and other varieties for further research to compare and see if there will be any significant changes in the values of measurement.

REFERENCES

Abhang, P. A., Gawali, B. W., & Mehrotra, S. C. (2016, April 15). Technical Aspects of Brain Rhythms and Speech Parameters. Retrieved January 15, 2020, from https://www.sciencedirect.com/science/article/pii/B9780128044902000038

Acoustical surfaces. (n.d.). Reverberation: Reverberation time: Reverb effect. Retrieved November 29, 2019, from https://www.acousticalsurfaces.com/acoustic_IOI/reverberation.htm

Monk, A. (2020, July 27). How Much CPU Usage is Normal? Retrieved January 12, 2020, from https://howmonk.com/how-much-cpu-usage-is-normal/

Ali, M. M., Hashima, N. M., Bejo, S. M., & Shamsudin, R. M. (2017). Watermelon.pdf. Retrieved from https://drive.google.com/file/d/1-D6wyDtucsNQ9WaLYxeuNGDyi6Ad76BD/view

Ballagh, K. (1999, May 03). Acoustical properties of wool. Retrieved November 28, 2019, from https://www.sciencedirect.com/science/article/abs/pii/0003682X95000428

BBC. (n.d.). Efficiency - Work, power and efficiency - AQA - GCSE Physics (Single Science) Revision - AQA - BBC Bitesize. Retrieved December 18, 2019, from https://www.bbc.co.uk/bitesize/guides/zp8jtv4/revision/3

Bryant, J., Jung, W., & Kester, W. (2007, September 02). EMI/RFI Considerations. Retrieved December 04, 2019, from https://www.sciencedirect.com/science/article/pii/B9780750678414500464

Bureau of Products Standards. (n.d.). Philippine National Standard. Retrieved January 14, 2020, from http://www.bafs.da.gov.ph/2017-10-12-00-46-55/standard-formulation/philippine-national-standards?download=74:pns-bafs-56-fresh-fruits-watermelon-grading-and-classification&start=40

Carr, J. J. (2007, September 28). Filtering against EMI/RFI. Retrieved December 04, 2019, from https://www.sciencedirect.com/science/article/pii/B9780750648448500177

Casquilho, M. (n.d.). Uncertainty, Precision and Accuracy. Retrieved November 22, 2019, from http://web.tecnico.ulisboa.pt/mcasquilho/acad/errors/ftp/udel.edu_pchem_C446_error.pdf

Chogou, S. K., Assogba, R., Degbey, H., Abokini, E., & Achigan-Dako, E. G. (2019, January 31). Market structure and performance of watermelon (Citrullus lanatus) in Benin. Retrieved September 23, 2019, from https://www.sciencedirect.com/science/article/pii/S2468227618302448

Christensen, E. (2020, May 29). The Best Way to Pick a Watermelon. Retrieved September 13, 2019, from https://www.thekitchn.com/the-best-way-to-pick-a-watermelon-172375

Cocal, C. J., Lopez, R. M., & Frias, G. G. (2017, April 26). Culture and management practices of watermelon farmers in Western Pangasinan, Philippines. Retrieved September 13, 2019, from http://apjarba.apjmr.com/wp-content/uploads/2018/02/APJARBA-2017.3.08.pdf

Dataforth. (n.d.). Protecting Signal Lines Against Electromagnetic Interference. Retrieved November 27, 2019, from https://www.dataforth.com/protecting-signal-lines-against-electromagnetic-interference.aspx

Ding, P. (2017). Tropical Fruit. Retrieved September 13, 2019, from https://www.sciencedirect.com/topics/food-science/tropical-fruits

Electronics Notes. (n.d.). RF Thermal Noise: Johnson-Nyquist Noise. Retrieved November 27, 2019, from https://www.electronics-notes.com/articles/basic_concepts/electronic-rf-noise/thermal-johnson-nyquist-basics.php

Figura, L. O., & Teixeira, A. A. (2010). Food physics: Physical properties - measurement and applications. doi:https://doi.org/10.1007/978-3-540-34194-9_12

Finley, W. R., & Hodowanec, M. M. (2001, April 01). Please Enable Cookies. Retrieved November 27, 2019, from https://www.ecmweb.com/content/motor-vibration-analysis-keeping-it-simple

Fluke. (2013, June 19). Top Causes of Machine Vibration. Retrieved November 27, 2019, from https://www.fluke.com/en-ph/learn/blog/vibration/most-common-causes-of-machine-vibration

Freckmann, G., Schmid, C., Baumstark, A., Rutschmann, M., Haug, C., & Heinemann, L. (2015, April 14). Analytical performance requirements for systems for self-monitoring of blood glucose with focus on system accuracy. Retrieved January 14, 2020, from http://ncbi.nlm.nih.gov/pmc/articles/PMC4525642/

Ghanem, M., & Harphoush, S. (2018, October). Sugar: Types and their functional properties in food and human health. Retrieved September 13, 2019, from https://www.culinarymd.org/uploads/2/0/4/0/2040875/sugars.pdf

Grant, A. (n.d.). Please Enable Cookies. Retrieved November 22, 2019, from https://www.gardeningknowhow.com/edible/fruits/watermelon/sugar-baby-cultivation.htm

Hanbury, A. (2008, May 30). Mathematical Morphology Applied to Circular Data. Retrieved January 15, 2020, from https://www.sciencedirect.com/science/article/pii/S1076567003800642

International Organization for Standardization. (n.d.). Accuracy (trueness and precision) of measurement methods and results. Retrieved January 14, 2020, from https://www.iso.org/obp/ui/#iso:std:iso:5725:-1:ed-1:v1:en

Jie, D., & Wei, X. (2018). Review on the recent progress of non-destructive detection technology for internal quality of watermelon. Retrieved September 13, 2019, from https://sci-hub.se/https://doi.org/10.1016/j.compag.2018.05.031

Kerr, B. (2013, September 23). Getting watermelons ready for the market. Retrieved September 23, 2019, from https://www.farmersweekly.co.za/crops/vegetables/getting-watermelons-ready-for-the-market

Known-You Seed (n.d.). Vegetable catalog No. 21. Retrieved January 14, 2020, from http://www.knownyou.com/en_index.jsp?bodyinclude=PRODUCTLIST&classid=9C8B16EC5695090B8AB8E3428C1AEBDBC037

Large, D., & Farmer, J. (2009, March 18). Coaxial RF Technology. Retrieved December 06, 2019, from https://www.sciencedirect.com/science/article/pii/B9780123744012000024

Li, Y., Zhao, H., Chang, D., & Han, D. (2012, June). Maturity Qualitative Discrimination of Small Watermelon Fruit. Retrieved September 23, 2019, from https://www.researchgate.net/publication/230631226_Maturity_Qualitative_Discrimination_of_Small_Watermelon_Fruit

Mao, W., & Wang, F. (2012, June 08). Cultural Modeling for Behavior Analysis and Prediction. Retrieved November 20, 2019, from https://www.sciencedirect.com/science/article/pii/B9780123972002000087

Meyman, C. (2014, June 17). Aluminum Foil's Role in Sound Insulation. Retrieved December 04, 2019, from https://ezinearticles.com/?Aluminum-Foils-Role-in-Sound-Insulation&id=8569853

Old Farmer's Almanac. (n.d.). Watermelons. Retrieved September 13, 2019, from https://www.almanac.com/plant/watermelons

Oyedepo, S., Adeyemi, G., Fayomi, O., Fagbemi, O., Solomon, R., Adekeye, T., . . . Nwanya, S. (2018, December 19). Dataset on noise level measurement in Ota metropolis, Nigeria. Retrieved December 06, 2019, from https://www.sciencedirect.com/science/article/pii/S2352340918315889

Peng, L. (2017). Sound Absorption Coefficient. Retrieved November 20, 2019, from https://www.sciencedirect.com/topics/engineering/sound-absorption-coefficient

Peter, S. C., Dhanjal, J. K., Malik, V. U., Radhakrishnan, N. U., Jayakanthan, M. U., & Sundar, D. U. (2018, September 06). Quantitative Structure-Activity Relationship (QSAR): Modeling Approaches to Biological Applications. Retrieved January 15, 2020, from https://www.sciencedirect.com/science/article/pii/B9780128096338201970

Pfliegel, P., Augusztinovicz, F., & Granát, J. (n.d.). Noise analysis and noise reduction of small DC motors. Retrieved December 6, 2019, from https://pdfs.semanticscholar.org/470a/26d6c52901af1aae09a69ad55ef178413f72.pdf

Prasadh, S.K., Natrajan, S. S. & Kalaivani, S. (2018, May 28). Efficiency analysis of noise reduction algorithms: Analysis of the best algorithm of noise reduction from a set of algorithms, pp. 1137-1140. doi: 10.1109/ICICI.2017.8365318. Retrieved January 14, 2020, from https://ieeexplore.ieee.org/document/8365318

Python. (n.d.). What is Python? Executive Summary. Retrieved May 17, 2020, from https://www.python.org/doc/essays/blurb/

Qiu, X. (n.d.). Flow Resistivity. Retrieved November 28, 2019, from https://www.sciencedirect.com/topics/engineering/flow-resistivity

Reich, G. (2005). Science direct. Near-infrared Spectroscopy and Imaging: Basic Principles and Pharmaceutical Applications. doi:https://doi.org/10.1016/j.addr.2005.01.020

Roberts, C. (2020, April 28). What is reverberation time and how it is calculated? Retrieved November 04, 2019, from https://www.cirrusresearch.co.uk/blog/2018/04/what-is-reverberation-time-and-how-it-is-calculated/

Roohnia, M. (2016, January 01). (PDF) Wood: Acoustic Properties. Retrieved January 05, 2020, from https://www.researchgate.net/publication/301262772_Wood_Acoustic_Properties

Rycroft, M. (2015, November 13). Causes and control of electrical machine vibration. Retrieved November 27, 2019, from https://www.ee.co.za/article/electrical-machine-vibration-causes-controls-2.html

Santora, M. (2015, April 15). What is EMI and how can you prevent it? Retrieved November 27, 2019, from https://www.wireandcabletips.com/what-is-emi-and-how-can-you-prevent-it/

Sarian, Z. B. & Taculao, P. B. S. (2019, May 17). Watermelon production guide. Retrieved January 14, 2020, from https://www.agriculture.com.ph/2019/07/22/watermelon-production-guide/

Schaffer, A. A., & Paris, H. S. (2003). Watermelon. Retrieved September 23, 2019, from https://www.sciencedirect.com/topics/medicine-and-dentistry/watermelo

Smardzewski, J., Kamisiński, T., Dziurka, D., Mirski, R., Majewski, A., Flach, A., & Pilch, A. (2015, May 01). Sound absorption of wood-based materials. Retrieved January 05, 2020, from https://www.degruyter.com/view/j/hfsg.2015.69.issue-4/hf-2014-0114/hf-2014-0114.xml

Sun, T., Huang, K., Xu, H., & Ying, Y. (2010, May 31). Research advances in nondestructive determination of internal quality in watermelon/melon: A review. Retrieved September 23, 2019, from https://www.sciencedirect.com/science/article/abs/pii/S0260877410002773

Tang, X., & Yan, X. (2017, July 03). Acoustic energy absorption properties of fibrous materials: A review. Retrieved November 28, 2019, from https://www.sciencedirect.com/science/article/abs/pii/S1359835X17302609

Thompson, R. A. (2002). Thermal Noise. Retrieved November 27, 2019, from https://www.sciencedirect.com/topics/engineering/thermal-noise

Vasilescu, G. (2005). Electronic noise and interfering signals: Principles and applications. Berlin: Springer. doi:https://doi.org/10.1007/3-540-26510-4_8

Ward, J. L., & Shewry, P. R. (2009). Near Infrared Spectroscopy. Retrieved September 23, 2019, from https://www.sciencedirect.com/topics/food-science/near-infrared-spectroscopy

Wei, Y., Rao, X., & Qi, B. (2012, February). Acoustic detecting system for sugar content of watermelon. doi:10.3969/j.issn.1002-6819.2012.03.049. Retrieved September 23, 2019, from https://www.researchgate.net/publication/286292175_Acoustic_detecting_system_for_sugar_content_of_watermelon

Wei-Chou, D. (2000). High-Tech Melon Thumper Developed by Student Designers. Retrieved December 04, 2019, from https://me.udel.edu/wp-content/uploads/2018/08/ME_2000-Newsletter-Winter.pdf

Williamson, T. (1993, December). Designing Microcontroller Systems for Electrically Noisy Environments. Retrieved November 19, 2019, from https://ecee.colorado.edu/~mcclurel/iap125.pdf

Wilson, A. (2019, October 01). A Brief Introduction to Supervised Learning. Retrieved May 17, 2020, from https://towardsdatascience.com/a-brief-introduction-to-supervised-learning-54a3e3932590

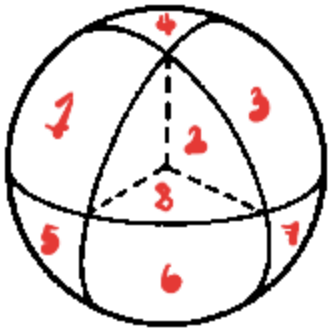
Wu, J. H., & Alamo, J. A. (2013, May 03). Design and modeling of Faraday cages for substrate noise isolation. Retrieved November 22, 2019, from https://www.sciencedirect.com/science/article/pii/S0038110113001494

Zingale, C., Ahlstrom, V., & Kudrick, B. (2005, November). Human Factors Guidance for the Use of Handheld, Portable, and Wearable Computing Devices. Retrieved October 22, 2019, from https://hf.tc.faa.gov/publications/2005-human-factors-guidance-for-the-use-of-handheld/full_text.pdf

APPENDICES

Appendix A

WATERMELON SECTIONING



Appendix B

TABLE OF AVERAGED AMPLITUDE RESULTS PER SECTION OF A SAMPLE

	AMPLITUDE (mV)								
	Section 1	Section 2	Section 3	Section 4	Section 5	Section 6	Section 7	Section 8	Average
Sample 1	7.4	7.15	7.49	7.78	7.68	7.39	7.54	7.51	7.49
Sample 2	8.72	7.39	7.999	7.55	8.02	7.93	7.93	7.22	7.84
Sample 3	6.70	5.88	5.97	6.89	5.36	5.94	6.34	5.93	6.16
Sample 4	6.49	6.35	8.06	6.495	7.27	7.93	7.007	6.48	7.01
Sample 5	6.21	5.33	5.84	5.44	5.92	6.195	7.18	7.48	6.2
Sample 6	6.75	7.06	6.18	5.48	5.58	6.63	5.53	6.26	6.18
Sample 7	6.31	5.88	5.85	6.53	6.37	6.97	5.86	6.498	6.28
Sample 8	6.703	5.88	5.97	6.89	5.36	5.94	6.34	5.93	6.13
Sample 9	6.297	5.15	5.71	6.54	7.25	6.67	5.598	4.94	6.02
Sample 10	6.95	6.57	6.31	6.65	6.201	6.89	6.84	7.019	6.68
Sample 11	7.04	6.503	6.33	6.62	7.04	6.77	7.04	7.14	6.81
Sample 12	6.23	5.795	6.76	5.92	6.73	7.12	6.96	6.33	6.48
Sample 13	5.85	6.07	6.77	5.99	6.76	8.34	7.49	6.84	6.76
Sample 14	6.11	6.15	7.56	7.47	7.43	7.25	6.83	6.06	6.86
Sample 15	4.87	4.59	5.36	5.31	5.703	6.39	5.62	6.67	5.56
Sample 16	5.064	5.47	5.35	5.25	6.09	5.69	6.204	7.46	5.82
Sample 17	6.44	5.899	7.309	6.44	6.701	6.92	5.94	5.52	6.4
Sample 18	5.37	6.21	6.16	5.58	5.55	5.09	5.26	5.63	5.61
Sample 19	7.33	7.09	6.067	6.81	6.58	6.32	5.11	5.78	6.39
Sample 20	4.11	4.79	5.38	6.19	6.63	11.32	6.82	7.03	6.53

Sample 21	3.782	4.29	4.19	3.91	3.89	4.21	4.11	3.45	3.98
Sample 22	5.114	5.096	5.67	5.14	5.69	5.75	6.04	5.36	5.48
Sample 23	5.11	5.08	5.21	5.28	4.99	5.13	5.18	5.31	5.16
Sample 24	3.62	3.87	3.28	3.41	3.302	3.41	3.25	3.02	3.4
Sample 25	3.82	6.46	4.99	3.86	4.599	5.19	3.98	4.16	4.63
Sample 26	4.74	4.35	4.59	4.04	3.98	4.58	3.43	4.12	4.23
Sample 27	5.28	4.99	4.44	4.68	4.403	4.95	4.48	4.31	4.69
Sample 28	5.89	5.73	5.39	5.97	5.07	4.89	4.62	4.95	5.31
Sample 29	4.09	4.58	4.74	4.71	4.53	5.99	5.01	5.002	4.83
Sample 30	6.06	5.51	5.52	5.62	5.81	4.899	4.82	5.26	5.44

Appendix C

TABLE OF AVERAGED SUGAR CONTENT MEASUREMENTS PER SECTION OF A SAMPLE

	SUGAR CONTENT (%Brix)								
	Section 1	Section 2	Section 3	Section 4	Section 5	Section 6	Section 7	Section 8	Average
Sample 1	8.9	8.2	9	8.9	8.1	8	8.6	8.2	8.49
Sample 2	8.1	7.3	5.9	8.2	7.9	7	8	6.9	7.41
Sample 3	10	10	10.1	10	9.5	9	8.5	9	9.51
Sample 4	10	10	10	9.5	9	9	9.5	9.5	9.56
Sample 5	10	10	11	10	10	9	9	9	9.75
Sample 6	9	10	9.2	9.8	10	8.8	10.1	9	9.49
Sample 7	9.8	10	10	10	9	8.5	8.5	10	9.48
Sample 8	11	10	12	10	11	11	9.5	10	10.56
Sample 9	9	9	11	10	9	8.5	8	9	9.19
Sample 10	10.1	8.9	9.9	9.2	9.5	11	10	8.9	9.69
Sample 11	10.9	12.3	11.2	10.2	10	10.9	10	10	10.69
Sample 12	11	11	11	11	11	11	11	10	10.88
Sample 13	10.2	10.1	10.2	12.8	10.1	10.1	10.1	5.9	9.94
Sample 14	8.2	8.9	6.8	7.8	8.2	8.8	8.9	10	8.45
Sample 15	10	10	10	10	10	9	11	8.5	9.81
Sample 16	10.2	10.2	10.2	10.2	11	11	10	11	10.48
Sample 17	10.1	10.1	10.1	10.1	10	10	10	10	10.05
Sample 18	11	10.2	10.2	12	11	10	12	11	10.93
Sample 19	11	10	10	10	10	10	10	10	10.13
Sample 20	12	10.4	11	11	10.2	10	10	10	10.58
Sample 21	10.1	10.1	10.1	10.12	10.1	10.1	10	10	10
Sample 22	10	10.1	10.1	10.1	10.1	7	10	10	9.68
Sample 23	10.01	9.8	11	10.7	10.3	9	9.8	11.2	10.23

Sample 24	11.1	10.4	10	10.9	11	10.8	10.2	11.2	10.7
Sample 25	10	9	9.3	9.8	9.2	9	9	9	9.29
Sample 26	10	8.8	10	9.8	10	9	8	9	9.33
Sample 27	10.12	10.1	10.1	10.12	9.8	10.09	10.8	10.8	10.24
Sample 28	10	10.12	10.1	10.2	10.11	10	10.15	10.12	10.1
Sample 29	10.1	10.02	10.03	10	9.2	9	10	9	9.67
Sample 30	3.2	9.3	10	10	10.1	10.2	10.3	10	9.76

Appendix D

MAIN PROGRAM

```
import cv2 as cv2
import time as time
import argparse
import sys

import imutils
import tkinter as tk
import tkinter.filedialog
import tkinter.messagebox
import tkinter.font
import PIL
from PIL import Image
from PIL import ImageTk

import numpy as np
import serial
import re

import time
import datetime
import runpy
import ImportantFunctions as IF
from datetime import datetime
import sqlite3

from scipy.io import wavfile
import sounddevice as sd
from scipy.io.wavfile import write

import pandas as pd
import numpy as np
from sklearn.model_selection
import train_test_split
from sklearn.preprocessing import
StandardScaler
from sklearn.ensemble import
RandomForestRegressor
from sklearn import metrics
from sklearn import svm
from sklearn.neighbors import
KNeighborsRegressor
from sklearn.tree import
DecisionTreeRegressor
import pickle

dataset =
pd.read_csv('/home/pi/Desktop/data
sets_watermelon (1).csv')
dataset.head()
X = dataset.iloc[:, 0:1].values
y = dataset.iloc[:, 1].values
X_train, X_test, y_train, y_test =
train_test_split(X, y,
test_size=0.2, random_state=0)

## Mode = 1 RF
## Mode = 2 SVM
## Mode = 3 DT

f = open("Mode.txt", "r+")
Mode = int(f.read())
Mode = 1

if Mode != 1 and Mode != 2 and
Mode !=3:

tk.messagebox.showinfo("Error",
"Mode can only be 1,2 or 3.")
    quit()
f.close()

##RFs =
RandomForestRegressor(n_estimators
=20, random_state=0)
##RFs.fit(X_train, y_train)
##
##SVMs = svm.SVR()
##SVMs.fit(X_train, y_train)
##
```

```
##DTs =
DecisionTreeRegressor(random_state
=0)
##DTs.fit(X_train, y_train)
##


# Fit the model on training set
RFs = RandomForestRegressor()
RFs.fit(X_train, y_train)

SVMs = svm.SVR()
SVMs.fit(X_train, y_train)

DTs = DecisionTreeRegressor()
DTs.fit(X_train, y_train)




# Save the trained model as a
pickle string.
RF_model = pickle.dumps(RFs)
SV_model = pickle.dumps(SVMs)
DT_model = pickle.dumps(DTs)


# Load the pickled model
RF_from_pickle =
pickle.loads(RF_model)
SVM_from_pickle =
pickle.loads(SV_model)
DT_from_pickle =
pickle.loads(DT_model)

# Use the loaded pickled model to
make predictions
RF_from_pickle.predict(X_test)

SVM_from_pickle.predict(X_test)

DT_from_pickle.predict(X_test)
```

```
print("Handshaking with other
r-pi.")
SerialData =
serial.Serial("/dev/ttyS0", 9600,
timeout=0.2)
time.sleep(2)
SerialData2 =
serial.Serial("/dev/ttyACM0",
9600, timeout=0.2)
time.sleep(2)
print("Handshaking successful...")

def cal_average(num):
    sum_num = 0
    for t in num:
        sum_num = sum_num + t

    avg = sum_num / len(num)
    return avg

def freq(file, start_time,
end_time):
    sample_rate, data =
wavfile.read(file)
    start_point = int(sample_rate
* start_time / 1000)
    end_point = int(sample_rate *
end_time / 1000)
    length = (end_time -
start_time) / 1000
    counter = 0
    counter2 = 0
    Sum_Amplitude = 0
    Amplitude = max(data)
    print(max(data))
    for i in range(start_point,
end_point):
        Sum_Amplitude =
Sum_Amplitude + abs(data[i])
        counter2 += 1
        if data[i] < 0 and
data[i+1] > 0:
            counter += 1
    print(Sum_Amplitude, counter2)
```

```
    return counter/length,
round((Sum_Amplitude/counter2),5)*
1000
    return counter/length,
round((Amplitude)*1000,2)

def DetFreq():
    fs = 44100  # Sample rate
    seconds = 3  # Duration of
recording

    myrecording =
sd.rec(int(seconds * fs),
samplerate=fs, channels=1)
    sd.wait()  # Wait until
recording is finished
    write('output.wav', fs,
myrecording)  # Save as WAV file

    Freq, Amp = freq("output.wav",
1000 ,2000)
    Freq, Amp = str(Freq),
str(Amp)

    return Freq, Amp

db =
sqlite3.connect("Database.db")

def GetDatabase(): ##Get Database
in a form of list
    global db
    cur = db.cursor()
    cur.execute('SELECT * FROM
Databases ORDER by "_rowid_" ASC')
    rows = cur.fetchall()
    return rows

def
InsertIntoDatabase(US,PW,FN,AGE,AD
,G):
    global db
    cur = db.cursor()
    cur.execute('SELECT * FROM
Databases ORDER by "_rowid_" ASC')
    rows = cur.fetchall()
    cur.execute('INSERT INTO
"main"."Databases"("Username","Pas
sword","Extra1","Extra2","Extra3",
"Extra4","Extra5","Extra6","Extra7
","Extra8","Extra9","Extra10")
VALUES
('+US+','+PW+','+FN+','+AGE+','+AD
+','+G+',NULL,NULL,NULL,NULL,NULL,
NULL);')
    db.commit()
    return rows

def Login():
    Frame1.place(x=165,y=130)

def Cancel():
    Frame1.place_forget()

User = 0

def GoToLoginPage():
    Username = UsernameEntry.get()
    Password = PasswordEntry.get()
    Database = GetDatabase()
    if Username == "" or Password
== "":

tk.messagebox.showinfo("Informatio
n", "Please, fill up all fields.")
        return
    else:
        global User
        User = 0

        for row in Database:
            if Username == row[0]
and Password == row[1]:
                print("Success")

UsernameEntry.delete(0,tk.END)
```

```
PasswordEntry.delete(0,tk.END)

Frame1.place_forget()

tk.messagebox.showinfo("Informatio
n", "Successful Login.")

WelcomePage.forget()
                HomePage.pack()
                return
            User = User + 1

tk.messagebox.showinfo("Informatio
n", "User not found.")

def GoToRegisterPage():
    WelcomePage.forget()
    RegisterPage.pack()

def MainLoop():
    global User

TimeLabel.configure(text=datetime.
now().strftime("%H:%M:%S %p %B %d,
%Y"))
    root.after(5,MainLoop)

root=tk.Tk()    #Welcome Screen
root.state('normal')
root.configure(background='white')
root.attributes('-fullscreen',
True)
subFont1 =
tkinter.font.Font(family='Century
Gothic', size = 13)
subFont2 =
tkinter.font.Font(family='Times
New Roman', size = 13, weight =
"bold")
subFont3 =
tkinter.font.Font(family='Segoe
UI', size = 13)

WelcomePage = tk.Frame(root)
WelcomePage.pack()

Background = tk.Label(WelcomePage,
text='',font = subFont1,
bg='white',bd=0)
Background.pack()
IF.tkShow(Background, "final.png",
1)

LoginButton =
tk.Button(WelcomePage,
text='Login', font = subFont1,
width = 10, bg='white',
fg='black', command=Login)
LoginButton.place(x=180, y=330)

RegisterButton =
tk.Button(WelcomePage,
text='Register', font = subFont1,
width = 10, bg='white',
fg='black',
command=GoToRegisterPage)
RegisterButton.place(x=340, y=330)

def Shuts():
    SerialData.write(bytes("B",
"utf-8"))
    time.sleep(2)
    import os
    os.system("sudo shutdown -h
now")

ShutButton =
tk.Button(WelcomePage, text='Shut
Down', font = subFont1, width =
10, bg='white', fg='black',
command=Shuts)
ShutButton.place(x=500, y=330)

##################################
##################################
##################################
############################
```

```
IF.Create_White_Screen("Frame1.png
",450,230)

Frame1 =
tk.Frame(WelcomePage,highlightthic
kness=1,
highlightbackground='black',
borderwidth=0) #Login Page

Background = tk.Label(Frame1,
text='',font = subFont1,
bg='cyan',bd=0)
Background.pack()
IF.tkShow(Background,
"Frame1.png", 1)

UsernameLabel = tk.Label(Frame1,
text='Username: ',font = subFont1,
bg='white',bd=0)
UsernameLabel.place(x=70,y=45)

UsernameEntry = tk.Entry(Frame1,
bg = 'white', font = subFont1,
width = 23)
UsernameEntry.place(x=170,y=45)

PasswordLabel = tk.Label(Frame1,
text='Password: ',font = subFont1,
bg='white',bd=0)
PasswordLabel.place(x=70,y=95)

PasswordEntry = tk.Entry(Frame1,
bg = 'white', font = subFont1,
width = 23, show="●")
PasswordEntry.place(x=170,y=95)

CancelButton1 = tk.Button(Frame1,
text='Cancel', font = subFont1,
width = 10, bg='white',
fg='black', command=Cancel)
CancelButton1.place(x=235, y=150)

LoginButton1 = tk.Button(Frame1,
text='Login', font = subFont1,
width = 10, bg='white',
fg='black', command=GoToLoginPage)
LoginButton1.place(x=90, y=150)

##################################
##################################
##################################
############################
def RegisterBack():
    RegisterPage.forget()
    WelcomePage.pack()

def RegisterProcess():
    Username =
UsernameEntry2.get()
    Password =
PasswordEntry2.get()
    FirstName =
FirstNameEntry2A.get()
    MiddleName =
MiddleNameEntry2.get()
    LastName =
LastNameEntry2.get()
    Age = AgeEntry2.get()
    Address = AddressEntry2.get()
    Gender = "H"
    Gender = GenderSelection.get()

    FullName = "!" + FirstName +
"@" + MiddleName + "#" + LastName
+ "%"

    if Username == "":

tk.messagebox.showinfo("Informatio
n", "Please input a username.")
    elif Password == "":

tk.messagebox.showinfo("Informatio
n", "Please input a password.")
    elif FirstName == "":
```

```
tk.messagebox.showinfo("Informatio
n", "Please input your first
name.")
    elif MiddleName == "":

tk.messagebox.showinfo("Informatio
n", "Please input your middle
initial.")
    elif LastName == "":

tk.messagebox.showinfo("Informatio
n", "Please input your last
name.")
    elif Age == "":

tk.messagebox.showinfo("Informatio
n", "Please input your  Age.")
    elif Address == "":

tk.messagebox.showinfo("Informatio
n", "Please input your Address.")
    elif Gender == "":

tk.messagebox.showinfo("Informatio
n", "Please select your gender.")
    else:
        GenderSelection.set("")

MaleButton.configure(indicatoron=1
)

FemaleButton.configure(indicatoron
=2)
        InsertIntoDatabase('"'+
Username +'"','"'+ Password
+'"','"' + FullName + '"','"'+ Age
+'"','"' + Address + '"','"' +
Gender + '"')

print(Username,Password,FirstName,
MiddleName,LastName,Age,Address,re
pr(Gender))

tk.messagebox.showinfo("Informatio
n", "Register Successful.")
        RegisterBack()
RegisterPage = tk.Frame(root)

Background =
tk.Label(RegisterPage,
text='',font = subFont1,
bg='white',bd=0)
Background.pack()
IF.tkShow(Background, "final.png",
1)

Frame2 =
tk.Frame(RegisterPage,bg="white",h
ighlightthickness=1,
highlightbackground='black',
borderwidth=0)
Frame2.place(x=100, y=15)

Frame2A = tk.Frame(Frame2,
bg="white")
Frame2A.grid(row=1,column=1,
pady=4)

PadX1 = 3
PadY1 = 4
WidthLabels1 = 17
WidthLabels2 = 23
UsernameLabel2 = tk.Label(Frame2A,
text='Username: ',font = subFont1,
bg='white',bd=0,width=WidthLabels1
, anchor=tk.E)
UsernameLabel2.grid(row=1,column=1
,padx=PadX1, pady=PadY1)

UsernameEntry2 = tk.Entry(Frame2A,
bg = 'white', font = subFont1,
width = 30)
UsernameEntry2.grid(row=1,column=2
,padx=PadX1, pady=PadY1)
```

```
space = tk.Label(Frame2A,
text='',font = subFont1,
bg='white',bd=0,width=8)
space.grid(row=1,column=3)

PasswordLabel2 = tk.Label(Frame2A,
text='Password: ',font = subFont1,
bg='white',bd=0,width=WidthLabels1
, anchor=tk.E)
PasswordLabel2.grid(row=2,column=1
,padx=PadX1, pady=PadY1)

PasswordEntry2 = tk.Entry(Frame2A,
bg = 'white', font = subFont1,
width = 30, show="●")
PasswordEntry2.grid(row=2,column=2
,padx=PadX1, pady=PadY1)

FirstNameLabel2 =
tk.Label(Frame2A, text='First
Name: ',font = subFont1,
bg='white',bd=0,width=WidthLabels1
, anchor=tk.E)
FirstNameLabel2.grid(row=3,column=
1,padx=PadX1, pady=PadY1)

FirstNameEntry2A =
tk.Entry(Frame2A, bg = 'white',
font = subFont1, width = 30)
FirstNameEntry2A.grid(row=3,column
=2,padx=PadX1, pady=PadY1)

MiddleNameLabel2 =
tk.Label(Frame2A, text='Middle
Name: ',font = subFont1,
bg='white',bd=0,width=WidthLabels1
, anchor=tk.E)
MiddleNameLabel2.grid(row=4,column
=1,padx=PadX1, pady=PadY1)

MiddleNameEntry2 =
tk.Entry(Frame2A, bg = 'white',
font = subFont1, width = 30)
MiddleNameEntry2.grid(row=4,column
=2,padx=PadX1, pady=PadY1)

LastNameLabel2 = tk.Label(Frame2A,
text='Last Name: ',font =
subFont1,
bg='white',bd=0,width=WidthLabels1
, anchor=tk.E)
LastNameLabel2.grid(row=5,column=1
,padx=PadX1, pady=PadY1)

LastNameEntry2 = tk.Entry(Frame2A,
bg = 'white', font = subFont1,
width = 30)
LastNameEntry2.grid(row=5,column=2
,padx=PadX1, pady=PadY1)

AgeLabel2 = tk.Label(Frame2A,
text='Age: ',font = subFont1,
bg='white',bd=0,width=WidthLabels1
, anchor=tk.E)
AgeLabel2.grid(row=6,column=1,padx
=PadX1, pady=PadY1)

AgeEntry2 = tk.Entry(Frame2A, bg =
'white', font = subFont1, width =
30)
AgeEntry2.grid(row=6,column=2,padx
=PadX1, pady=PadY1)

AddressLabel2 = tk.Label(Frame2A,
text='Address: ',font = subFont1,
bg='white',bd=0,width=WidthLabels1
, anchor=tk.E)
AddressLabel2.grid(row=7,column=1,
padx=PadX1, pady=PadY1)

AddressEntry2 = tk.Entry(Frame2A,
bg = 'white', font = subFont1,
width = 30)
AddressEntry2.grid(row=7,column=2,
padx=PadX1, pady=PadY1)
```

```
GenderLabel2 = tk.Label(Frame2A,
text='Gender: ',font = subFont1,
bg='white',bd=0,width=WidthLabels1
, anchor=tk.E)
GenderLabel2.grid(row=8,column=1,p
adx=PadX1, pady=PadY1)

GenderSelection = tk.StringVar()

SubFrame1 = tk.Frame(Frame2A,
bg="white")
SubFrame1.grid(row=8,column=2,padx
=PadX1, pady=PadY1)

MaleButton =
tk.Radiobutton(SubFrame1,
text="Male",  font=subFont1,
state=tk.NORMAL, bg='white',

variable=GenderSelection,
value="Male", activebackground =
'white',

highlightthickness = 0,
borderwidth = 0)
MaleButton.grid(row=1,column=1,pad
x=PadX1, pady=PadY1)

FemaleButton =
tk.Radiobutton(SubFrame1,
text="Female",  font=subFont1,
state=tk.NORMAL, bg='white',

variable=GenderSelection,
value="Female", activebackground =
'white',

highlightthickness = 0,
borderwidth = 0, relief =
tk.RIDGE)
FemaleButton.grid(row=1,column=2,p
adx=PadX1, pady=PadY1)

Frame2B = tk.Frame(Frame2,
bg="white")
Frame2B.grid(row=2,column=1,
pady=4)

CancelButton1 = tk.Button(Frame2B,
text='Cancel', font = subFont1,
width = 10, bg='white',
fg='black', command=RegisterBack)
CancelButton1.grid(row=1,
column=1, padx=PadX1, pady=PadY1)

RegisterButton1 =
tk.Button(Frame2B,
text='Register', font = subFont1,
width = 10, bg='white',
fg='black',
command=RegisterProcess)
RegisterButton1.grid(row=1,
column=2, padx=PadX1, pady=PadY1)

##################################
##################################
##################################
############################
def Logout():
    HomePage.forget()
    WelcomePage.pack()

def GoToAboutPage():
    HomePage.forget()
    AboutPage.pack()

def GoToStartPage():
    global User
    Database = GetDatabase()
    FullName = Database[User][2]
    Age = Database[User][3]
    Address = Database[User][4]
    Gender = Database[User][5]
    print(FullName)
    FirstName     =
IF.GetINFO(FullName, "!","@")
    print(FullName)
```

```
    MiddleInitial =
IF.GetINFO(FullName, "@","#")
    print(FullName)
    LastName      =
IF.GetINFO(FullName, "#","%")

    FullNameLabel3.configure(text=
"Full Name: "+ LastName + ", " +
FirstName + " " + MiddleInitial +
".")
    AgeLabel3.configure(text="Age:
"+ str(Age))

AddressLabel3.configure(text="Addr
ess: " + str(Address))

GenderLabel3.configure(text="Gende
r: " + str(Gender))
    HomePage.forget()
    StartPage.pack()

HomePage = tk.Frame(root) #Login
Page
##HomePage.pack()

Background = tk.Label(HomePage,
text='',font = subFont1,
bg='white',bd=0)
Background.pack()
IF.tkShow(Background, "final.png",
1)

ShutButton = tk.Button(HomePage,
text='Shut Down', font = subFont1,
width = 10, bg='white',
fg='black', command=GoToLoginPage)
ShutButton.place(x=20, y=420)

StartButton = tk.Button(HomePage,
text='Start', font = subFont1,
width = 10, bg='white',
fg='black', command=GoToStartPage)
StartButton.place(x=340, y=330)

AboutButton = tk.Button(HomePage,
bg="white",
fg="white",font=subFont1,
highlightthickness=0,
highlightbackground='white',
borderwidth=0,
highlightcolor="white",
activebackground="white",
command=GoToAboutPage)
##AboutButton.place(x=20, y=20)
##IF.tkShow(AboutButton,
"Info1.png", 1)

TimeLabel = tk.Label(HomePage,
text=datetime.now().strftime("%H:%
M:%S %p %B %d, %Y"), font =
subFont1, bg='white')
TimeLabel.place(x=445,y=420)

LogOutButton = tk.Button(HomePage,
text='Log Out', font = subFont1,
width = 10, bg='white',
fg='black', command=Logout)
LogOutButton.place(x=620, y=20)

GUI3 = tk.Frame(root) #Register
Page

Background = tk.Label(GUI3,
text='',font = subFont1,
bg='white',bd=0)
Background.pack()
IF.tkShow(Background, "bg.png", 1)

##################################
##################################
##################################
############################
def AboutBack():
    AboutPage.forget()
    HomePage.pack()

AboutPage = tk.Frame(root) #Login
Page
```

```
Background = tk.Label(AboutPage,
text='',font = subFont1,
bg='white',bd=0)
Background.pack()
IF.tkShow(Background, "final.png",
1)

AboutLabel = tk.Label(AboutPage,
text='About Page',font = subFont1,
bg='white', fg="black",bd=0,
anchor=tk.E)
AboutLabel.place(x=70, y=55)

BackButton = tk.Button(AboutPage,
text='Back', font = subFont2,
width = 10, bg='white',
fg="black", command=AboutBack)
BackButton.place(x=335, y=370)

###################################
###################################
###################################
############################
MelonSize = ""
def StartProcess():
    global MelonSize
    global Mode
    if MelonSize == "":

tk.messagebox.showinfo("Informatio
n", "Please, select a size.")
        return
    else:

SmallButton.configure(bg="#db6161"
, fg='white')

MediumButton.configure(bg="#db6161
", fg='white')

LargeButton.configure(bg="#db6161"
, fg='white')
        Frequency1, Amplitude1 =
DetFreq()
        print("Amplitude (No
Sound): ", Amplitude1)

SerialData2.write(bytes("1",
"utf-8"))
        time.sleep(1)
        Frequency, Amplitude =
DetFreq()
        print("Amplitude (With
Sound): ", Amplitude)

        Date =
datetime.now().strftime("%B %m,
%Y")

SerialData.write(bytes("A",
"utf-8"))
        time.sleep(2)
        x =
SerialData.readline().decode("utf-
8")
        Start = time.time()
        while(x == ''):
            x =
SerialData.readline().decode("utf-
8")
            if (time.time() -
Start) > 10:
                x = "None"
                break

        if Mode == 1:
            global RF_from_pickle
            SugarContent =
RF_from_pickle.predict([[float(Amp
litude1)]])
            print("Sugar
Content(RF):", SugarContent)
        elif Mode == 2:
            global SVM_from_pickle
```

```
            SugarContent =
SVM_from_pickle.predict([[float(Am
plitude1)]])
            print("Sugar
Content(SVM):", SugarContent)
        elif Mode == 3:
            global DT_from_pickle
            SugarContent =
DT_from_pickle.predict([[float(Amp
litude1)]])
            print("Sugar
Content(DT):", SugarContent)

        SugarContent =
round(SugarContent[0],2)

        InsertIntoDatabase2(Date,
MelonSize, str(Frequency) + " Hz"
, str(SugarContent) + " %")

FrequencyLabel.configure(text="
Frequency: "+ str(Frequency) + "
Hz")

SugarLabel.configure(text=" Sugar
Content: "+ str(SugarContent) + "
%")
        SizeLabel.configure(text="
Size: "+ MelonSize)
        AmpLabel.configure(text="
AMP(out): " + str(x) + " mV\n
AMP(in): " +
str(round(float(Amplitude1),2)) +
" mV")
        Frame5.place(x=180, y=90)
        MelonSize = ""
    pass

def BackStartPage():
    StartPage.forget()
    HomePage.pack()

def SmallButFunc():
    global MelonSize

SmallButton.configure(bg="#db6161"
, fg='white')

MediumButton.configure(bg="white",
fg='black')

LargeButton.configure(bg="white",
fg='black')
    MelonSize = "Small"

def MediumButFunc():
    global MelonSize

SmallButton.configure(bg="white",
fg='black')

MediumButton.configure(bg="#db6161
", fg='white')

LargeButton.configure(bg="white",
fg='black')
    MelonSize = "Medium"

def LargeButFunc():
    global MelonSize

SmallButton.configure(bg="white",
fg='black')

MediumButton.configure(bg="white",
fg='black')

LargeButton.configure(bg="#db6161"
, fg='white')
    MelonSize = "Large"

StartPage = tk.Frame(root) #Login
Page
##StartPage.pack()

Background = tk.Label(StartPage,
text='',font = subFont1,
bg='white',bd=0)
```

```
Background.pack()
IF.tkShow(Background, "final.png",
1)

Frame3 =
tk.Frame(StartPage,bg="white",high
lightthickness=1,
highlightbackground='white',
borderwidth=0)
Frame3.place(x=8, y=40)

PadX2 = 3
PadY2 = 4

SelectLabel = tk.Label(Frame3,
text='  Select Watermelon Size:
',font = subFont3, bg='white',
fg="black",bd=0, anchor=tk.E)
SelectLabel.grid(row=1,column=1)

SmallButton = tk.Button(Frame3,
text='Small', font = subFont1,
width = 25, bg='#db6161',
fg='black', command=SmallButFunc)
SmallButton.grid(row=2,column=1,pa
dx=PadX1, pady=PadY1)

MediumButton = tk.Button(Frame3,
text='Medium', font = subFont1,
width = 25, bg='#db6161',
fg='black', command=MediumButFunc)
MediumButton.grid(row=3,column=1,p
adx=PadX1, pady=PadY1)

LargeButton = tk.Button(Frame3,
text='Large', font = subFont1,
width = 25, bg='#db6161',
fg='black', command=LargeButFunc)
LargeButton.grid(row=4,column=1,pa
dx=PadX1, pady=PadY1)

tk.Label(Frame3, text='', font =
subFont2,
bg="white").grid(row=5,column=1,pa
dx=PadX1, pady=PadY1)

ProcessButton = tk.Button(Frame3,
text='Process', font = subFont1,
width = 10, bg='white',
fg='black', command=StartProcess)
ProcessButton.grid(row=6,column=1,
padx=PadX1, pady=PadY1)

Frame4 =
tk.Frame(StartPage,bg="white",high
lightthickness=0,
highlightbackground='black',
borderwidth=0)
Frame4.place(x=520, y=40)

AboutLabel = tk.Label(Frame4,
text='  User Information:
',font = subFont3,
bg="white",fg="black",bd=0,
anchor=tk.E)
AboutLabel.grid(row=1,column=1)

FullNameLabel3 = tk.Label(Frame4,
text='Full Name: ',font =
subFont3,
bg='white',bd=0,width=WidthLabels1
,
wraplength=300,anchor=tk.W,justify
= tk.LEFT)
FullNameLabel3.grid(row=2,column=1
,padx=PadX1, pady=PadY1,
sticky=tk.W)

AgeLabel3 = tk.Label(Frame4,
text='Age: ',font = subFont3,
bg='white',bd=0,width=WidthLabels1
, anchor=tk.W)
AgeLabel3.grid(row=3,column=1,padx
=PadX1, pady=PadY1, sticky=tk.W)

AddressLabel3 = tk.Label(Frame4,
text='Address: ',font = subFont3,
```

```
bg='white',bd=0,width=WidthLabels1
,
anchor=tk.W,wraplength=250,justify
= tk.LEFT)
AddressLabel3.grid(row=4,column=1,
padx=PadX1, pady=PadY1,
sticky=tk.W)

GenderLabel3 = tk.Label(Frame4,
text='Gender: ',font = subFont3,
bg='white',bd=0,width=WidthLabels1
, anchor=tk.W)
GenderLabel3.grid(row=5,column=1,p
adx=PadX1, pady=PadY1,
sticky=tk.W)

BackButton = tk.Button(StartPage,
text='Back', font = subFont1,
width = 10, bg='white',
fg='black', command=BackStartPage)
BackButton.place(x=30, y=400)

######PROCESS DIALOG BOX

Frame5 =
tk.Frame(StartPage,bg="white",high
lightthickness=1,
highlightbackground='black',
borderwidth=0)
##Frame5.place(x=175, y=130)

IF.Create_White_Screen("Frame5.png
",450,310)

Background = tk.Label(Frame5,
text='',font = subFont1,
bg='white',bd=0)
Background.pack()
IF.tkShow(Background,
"Frame5.png", 1)

SubFrame5 = tk.Frame(Frame5,
bg="white")
SubFrame5.place(x=10,y=10)

ResultsLabel = tk.Label(SubFrame5,
text='RESULTS: ',font = subFont1,
bg='white',bd=0,width=WidthLabels2
, anchor=tk.W)
ResultsLabel.grid(row=1,column=1,p
adx=5, pady=7, sticky=tk.W)

FrequencyLabel =
tk.Label(SubFrame5, text='
Frequency (Hz): ',font = subFont1,
bg='white',bd=0,width=WidthLabels2
, anchor=tk.W)
FrequencyLabel.grid(row=2,column=1
,padx=PadX1, pady=PadY1,
sticky=tk.W)

SugarLabel = tk.Label(SubFrame5,
text=' Sugar Content (%): ',font =
subFont1,
bg='white',bd=0,width=WidthLabels2
, anchor=tk.W)
SugarLabel.grid(row=3,column=1,pad
x=PadX1, pady=PadY1, sticky=tk.W)

SizeLabel = tk.Label(SubFrame5,
text=' Size: ',font = subFont1,
bg='white',bd=0,width=WidthLabels2
, anchor=tk.W)
SizeLabel.grid(row=4,column=1,padx
=PadX1, pady=PadY1, sticky=tk.W)

AmpLabel = tk.Label(SubFrame5,
text=' Inside Amplitude: ',font =
subFont1,
bg='white',bd=0,width=WidthLabels2
, anchor=tk.W, justify=tk.LEFT)
AmpLabel.grid(row=5,column=1,padx=
PadX1, pady=PadY1, sticky=tk.W)

def BackSubframe5Func():
    Frame5.place_forget()
```

```
BackSubframe5 =
tk.Button(SubFrame5,
text='DONE',font = subFont1,
bg='#242424',
fg='white',bd=0,width=WidthLabels2
, anchor=tk.CENTER,
justify=tk.CENTER,
command=BackSubframe5Func)
BackSubframe5.grid(row=6,column=1,
padx=PadX1, pady=PadY1,
sticky=tk.W+tk.E)

#### HISTORY

def HistoryBack():
    HistoryPage.forget()
    HomePage.pack()

HistoryPage = tk.Frame(root)
#Login Page
##HistoryPage.pack()

Background = tk.Label(HistoryPage,
text='',font = subFont1,
bg='white',bd=0)
Background.pack()
IF.tkShow(Background,
"bgnew1.png", 1)

from tkinter import ttk
tree=ttk.Treeview(HistoryPage,heig
ht=18)
tree["columns"]=("1","2","3")
tree.column("#0", width=185,
minwidth=160, stretch=tk.NO)
tree.column("1", width=185,
minwidth=160, stretch=tk.NO)
tree.column("2", width=185,
minwidth=160, stretch=tk.NO)
tree.column("3", width=185,
minwidth=160, stretch=tk.NO)

tree.heading("#0",text="Date",anch
or=tk.W)
tree.heading("1",
text="Size",anchor=tk.W)
tree.heading("2",
text="Frequency",anchor=tk.W)
tree.heading("3", text="Sugar
Content",anchor=tk.W)

tree.place(x=30, y=30)

vsb = ttk.Scrollbar(HistoryPage,
orient="vertical",
command=tree.yview)
vsb.place(x=765,y=30, height=381)

def GetDatabase1(): ##Get Database
in a form of list
    global db
    cur = db.cursor()
    cur.execute("SELECT * FROM
History ORDER by _rowid_")
    rows = cur.fetchall()
    return rows

def CreateTable():
    for i in GetDatabase1():
        folder1=tree.insert("", 1,
text=i[0],
values=(i[1],i[2],i[3]))

BackButton =
tk.Button(HistoryPage,
text='Back', font = subFont2,
width = 10, bg='#242424',
fg='white', command=HistoryBack)
BackButton.place(x=623, y=430)

CreateTable()

def
InsertIntoDatabase2(Date,Size,Freq
uency,Sugar):
    global db
    cur = db.cursor()
```

```
    cur.execute("SELECT * FROM
History ORDER by _rowid_")
    rows = cur.fetchall()
    cur.execute('INSERT INTO
`History`(`Date`,`Size`,`Frequency
`,`Sugar Content`,\

`Extra1`,`Extra2`,`Extra3`,`Extra4
`,`Extra5`,`Extra6`) ' + \
                'VALUES ("' + Date
+ '","' + Size + '","' + Frequency
+ '","' + Sugar + \
'",NULL,NULL,NULL,NULL,NULL,NULL);
')
    db.commit()
    return rows

root.after(5,MainLoop)
root.mainloop()
```

Appendix E

ENDORSEMENT LETTER

TECHNOLOGICAL INSTITUTE OF THE PHILIPPINES
1338 Arlegui St., Quiapo, Manila

August 2, 2019

To whom it may concern,

We, a team of bonafide Computer Engineering students from the Technological Institute of the Philippines - Manila, would like to humbly request a moment of your time for us to conduct a simple interview and have prepared a series of questions, regarding the different proceedings in a argricultural farm specifically in a watermelon farm. The resulting interview and information gathered will be used to support and further our research that will potentially improve certain conditions or enhance the efficiency of processes involved in agricultural farms. We assure you that any conditions that the interviewee/management would like to be implemented while inside the vicinity will strictly be adhered to. Also, any information the interviewee/management would not like to be disclosed publicly will be kept privately by the interviewers.

We hope for you kind consideration

Respectfully yours,

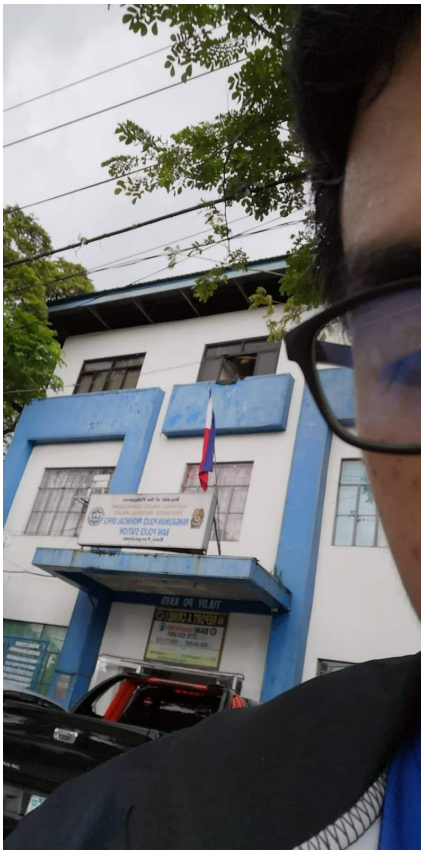
J.R. T. De Guzman
Group Representative

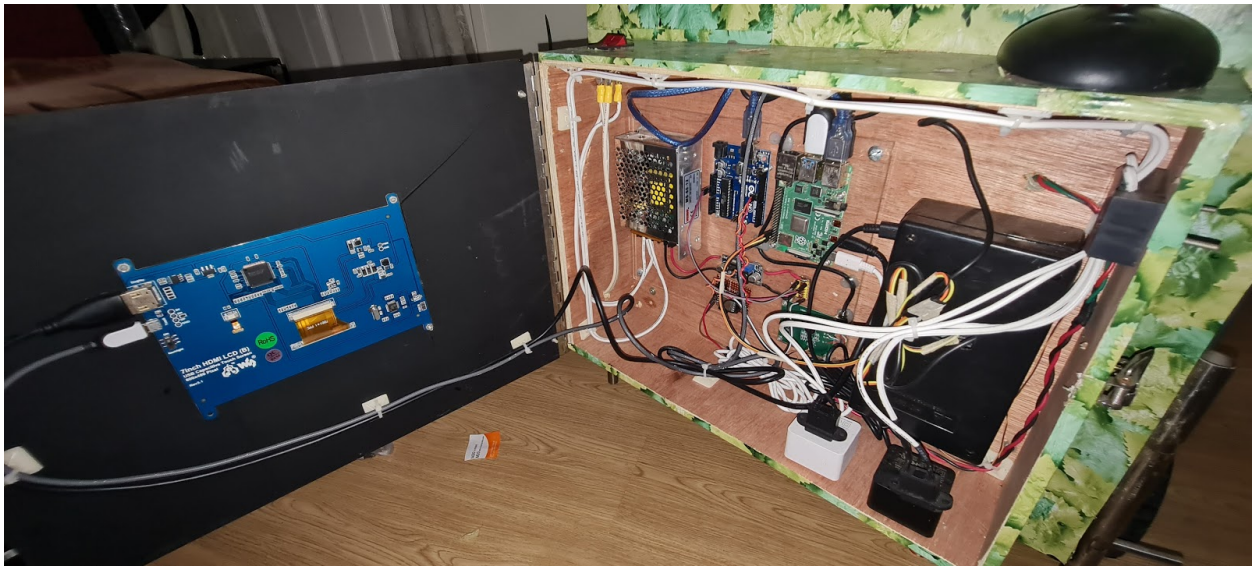
Noted by:

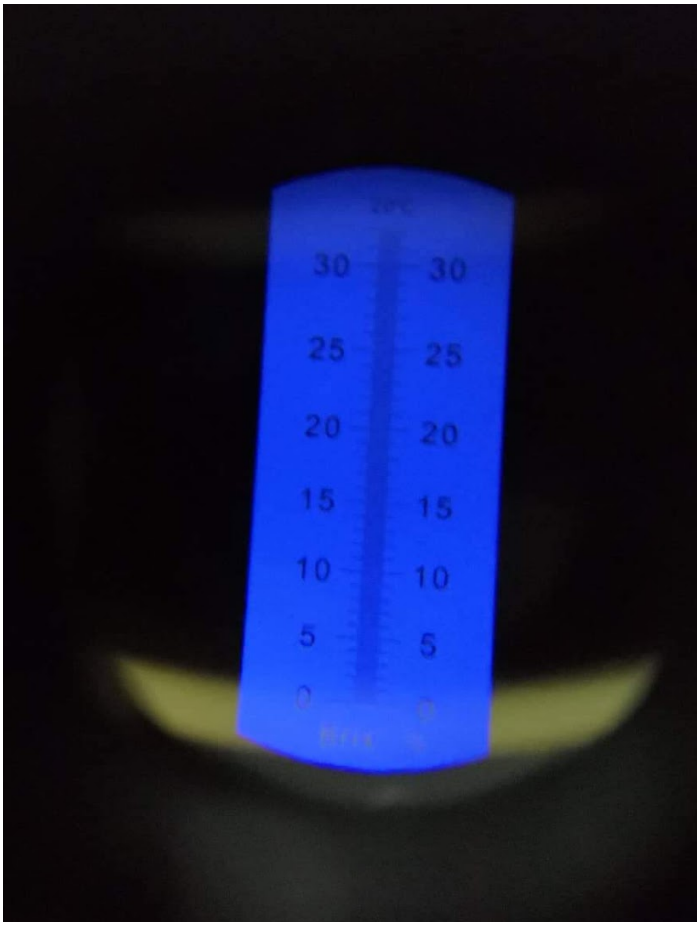
Engr. Jennifer B. Enriquez
Chair of Computer Engineering

Appendix F

PHOTOS






30
25
20
15
10
5
0
30
25
20
15
10
5
0