POLYTECHNIC UNIVERSITY OF THE PHILIPPINES

PAINT MIXER WITH MACHINE VISION-BASED COLOR SCANNER

A Design Project
Presented to the Faculty of
Computer Engineering Department
Polytechnic University of the Philippines
Sta. Mesa, Manila

In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science in Computer Engineering

by

Charlene Mae DG. Esteban
Enrico Camilo P. Gomez Jr.
Justin Earl L. Guevarra
Ami V. Morita

MARCH 2020

ACKNOWLEDGEMENTS

The success and outcome of this project required a lot of guidance and assistance from many people and we are extremely privileged to have got this all along the completion of our project. The accomplishment of this project is only by virtue of such supervision and support and we would not forget to thank them.

Most importantly, we sincerely thank the Almighty God for giving us strength throughout the preparation of this project. We also wish to extend our sincere and heartfelt gratitude to our family for providing our needs.

We respect and thank our Design Project advisers, Engr. Rolito L. Mahaguay and Engr. Ronald Fernando, for giving us an opportunity to do the project work and giving us all support and guidance, which made us complete the project duly.

We owe our deep gratitude to the staff of Finemix Paint Center, especially to Mr. Henry, who took keen enthusiasm on our project proposal and allow us to observe and improve their processes.

We are indebted to and fortunate enough to get knowledge and wisdom from all the teaching staffs of the Department of Computer Engineering of the Polytechnic University of the Philippines which helped us in successfully completing our project by having the lessons they shared with us.

CERTIFICATION OF ORIGINALITY

This is to certify that the research work presented in this thesis/dissertation, PAINT MIXER WITH MACHINE VISION-BASED COLOR SCANNER for the degree Bachelor of Science in Computer Engineering at the Polytechnic University of the Philippines embodies the result of original and scholarly work carried out by the undersigned. This dissertation does not contain words or ideas taken from published sources or written works that have been accepted as basis for the award of a degree from any other higher education institution, except where proper referencing and acknowledgment were made.

CHARLENE MAE DG. ESTEBAN
Researcher

ENRICO CAMILO P. GOMEZ JR.
Researcher

JUSTIN EARL L. GUEVARRA
Researcher

AMI V. MORITA
Researcher

_____
Date Signed

ABSTRACT

Title	:	Paint Mixer with Machine Vision-Based Color Scanner
Researchers	:	Charlene Mae DG. Esteban, Enrico Camilo P. Gomez Jr., Justin Earl L. Guevarra, Ami V. Morita
Degree	:	Bachelor of Science in Computer Engineering
Institution	:	Polytechnic University of the Philippines
Year	:	2020
Advisers	:	Engr. Rolito L. Mahaguay, Engr. Ronald D. Fernando

Paint color replication is not an easy feat that can be done instinctively; it requires knowledge in color representation and mixing theory. Color replication in paint centers is usually done manually that took a lot of time and effort which also requires talent and skill.

Series of experiments are done to develop the machine. Different camera and image processing techniques are tested to select which suits the best for getting the accurate color of a sample material. Formula for color mixing theory are also tested to see which of the color theory representations would work best in real life paint mixing application. Processing an image, captured by microscopic camera to detect the color of a sample and generate the required ratio of paint to be mixed, shortened the scanning time, lessen the amount of effort and ensure that paints are not wasted. CMYK color theory are applied to get the ratio of colors needed to replicate a color sample.

Developing a machine which automates the scanning and mixing of paints eliminates the negative factors that are present in manual mixing.

Keywords: paint mixer, machine vision, color scanner, color mixing theory, image processing, color replication, RGB conversion

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

THE PROBLEM AND ITS BACKGROUND

1.1 Introduction

The paint manufacturing business in the Philippines produces about 250 million liters of various paint products a year. The demand for coatings and paints in the Philippines is sure to increase due to various construction projects and continues development in the community.

Majority of the Filipino households uses paint to hide walls' flaws and keep dirt to a minimum, boosting their house's visual appeal corresponding to the owner's style.

Though there is a wide variety of paint colors available in the market, it is still not enough for people's desire of other shades. To answer this problem, paint experts uses available paint colors to come up with another color following the color mixing principle. Ordinary Filipino customers usually go to paint centers in the Philippines to achieve their desired paint color. Some does their experiment on their own. The common way of paint color mixing is done through manual estimation and trial and error which needs a lot of patience and accuracy. However, the manual estimation and trial and error done in paint color mixing do not always result to a correct production of desired color shade. Also, in manual paint mixing process, paints are wasted when the desired output is not met. This compromises money, paints, accuracy, time and efficiency.

In this paper, the proponents propose the design of the automation of paint color mixing to minimize waste of paint, time and effort in mixing of paint color. This also includes color scanning ability through machine vision technology to make mixing of paint color be precise and accurate as possible by imitating the color sample desired with minimum human intervention. With this machine, manual estimation and trial and error that causes waste of money, paint and time will be solved.

1.2 Theoretical Framework

Color Mixing

Color mixing has two types which are additive and subtractive. Additive color mixing happens when a set of wavelengths is added to another set of wavelengths. This is what happens when lights of different wavelengths are mixed. However, additive color mixing does not correspond to the mixing of physical substances such as paint, it is basically mixing of lights and is applied in digital world. Subtractive color mixing happens when a new color is created by removing wavelengths from a broad spectrum of wavelengths. This occurs when paints, dyes, or pigments are mixed.

It is important to understand the differences between hue, saturation, value, tint, tone, and shade in order to easily mix the desired colors. Hue is the color itself wherein saturation corresponds to its intensity, and value is the lightness or the darkness of it. When hue is added with white to lighten it, tint is created, while shade is created when black is added to a color to darken it. Tone is produced when both black and white is added to hue. (craftsy: Color Categories, 2013)

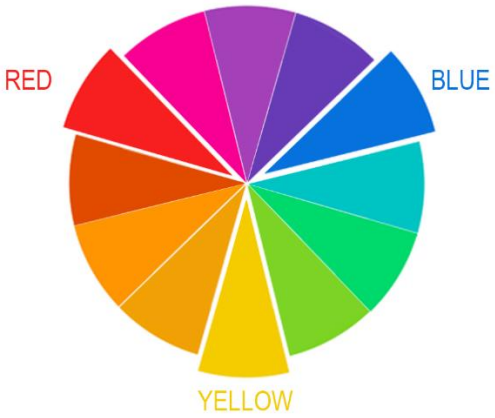
Mixing of primary colors can create a wide range of new colors. By using different quantities of RYB colors, where additional depth can be added by including white and black, following a proportional color mixing formula guarantees that a desired output can be created.

Color Wheel

The three primary colors used in paint mixing is RYB (Red-Yellow-Blue) which is also the primary colors in the color wheel. (emptyeasel, 2007) It is primarily used in painting. When the three primary colors are mixed, secondary colors are created which makes up violet, orange, and green. Another set of colors will then be created as primary and secondary colors

are mixed. Though RYB are primarily used in painting, it does not cover the widest range of colors unlike CMY (Cyan-Magenta-Yellow).

Figure 1. Color Wheel



RGB, CMYK and HSV Color Models

RGB stands for Red, Green and Blue and is used in additive color mixing which is for display technology. Using RGB is not applicable in paint and pigment, digital screens and computers usually process RGB. The colors used in subtractive color mixing is CMYK which stands for Cyan, Magenta, Yellow, and Key (Black). These are the colors used by computer printers to achieve the desired combination of colors generated from a computer screen which is usually from RGB. However, CMYK paint colors cannot be easily found on the market. Neither RGB nor CMYK color models seem ideal for this project. Thus, another model is needed which is the HSV. HSV stands for Hue, Saturation and Value. Hue could first be matched by mixing of the primary colors, then saturation and value could be matched by adding in white and black respectively. (Lindgren, 2017)

Machine Vision Technology

Machine vision systems analyze images from cameras to generate image feature data that guides robotic and automation machines in the understanding of the physical world depicted in the image. (K-state Polytechnic, 2017)

Machine vision uses both software and hardware. The system acquires images of an object using a camera and then sends the information to the computer to analyze and measure the characteristics of the object. One of the characteristics that machine vision can be analyzed is the object's color. The algorithm of color machine vision will be further studied and analyzed by the proponents throughout the research.

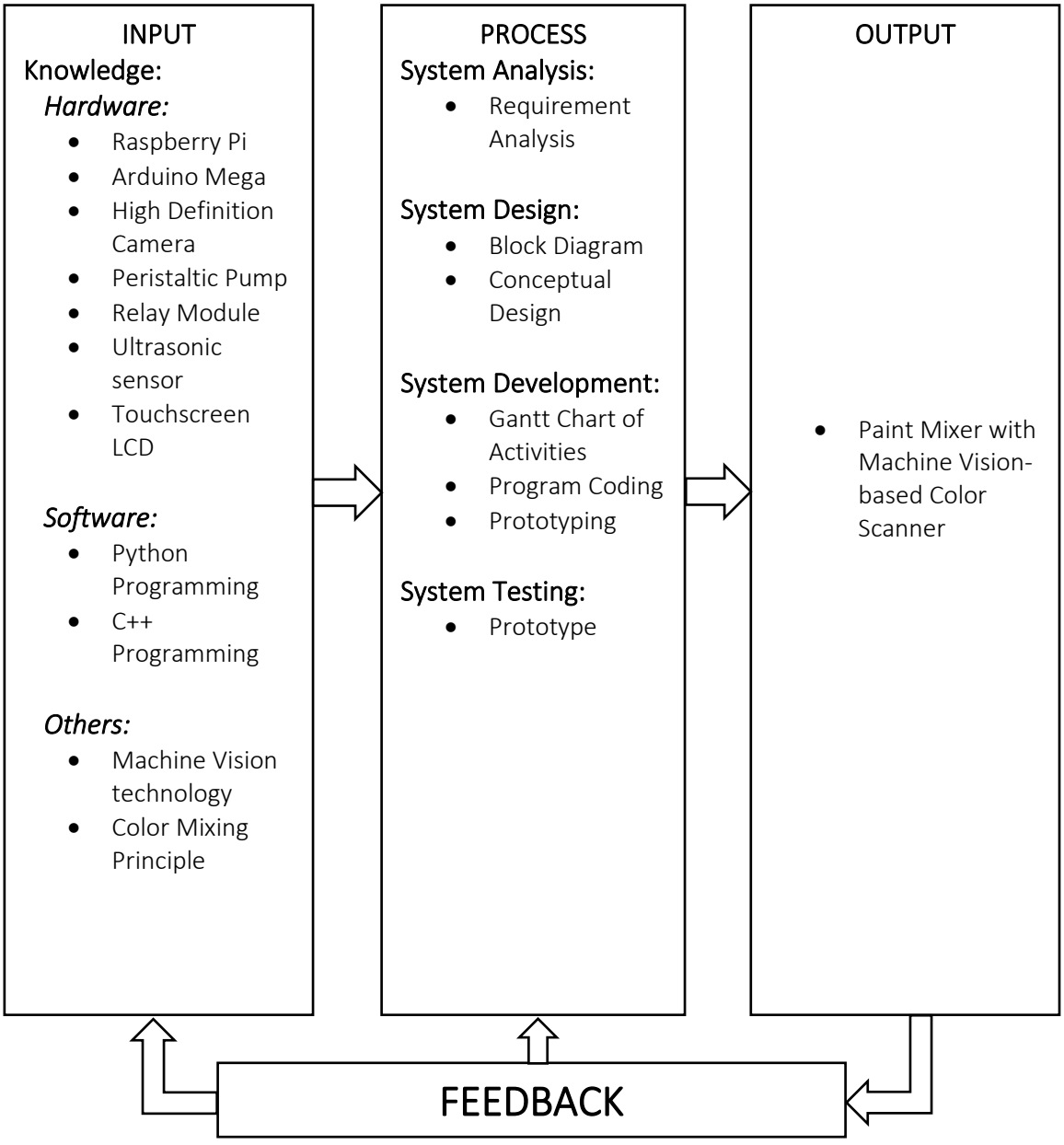
Single-board Computers

A single-board computer is a complete computer built on a single circuit board, with microprocessors, memory, input/output (I/O) and other featured required on a functional computer. Unlike a modern desktop computer, an SBC does not need additional memory or storage in order to boot. SBCs serve in a range of educational, commercial and industrial applications. (circuito: Single-board Computer, 2018)

The most famous of all SBCs is the Raspberry Pi. Raspberry Pi is a mini computer originally designed for education. This is a low-cost device that would improve programming skills and hardware understanding. Raspberry Pi can be used for many tasks that a computer does like image processing. This minicomputer will be used for machine vision application and automation of paint color mixing by controlling the components connected to it.

1.3 Conceptual Framework

Figure 2. Conceptual Framework of Paint Mixer with Machine Vision-based Color Scanner



The development of the proposed machine will be based on the conceptual framework shown in Figure 2. The knowledge needed in this project is stated in the input part of the framework which includes the hardware components to be used, software that will be utilized

and other principles and theories that will support the project. The process is the actual development of the system which serves as the guide in building the project. The expected output of the system is the paint mixer with machine vision-based color scanner.

1.4 Statement of the Problem

The study aims to develop a paint mixer with machine vision-based color scanner that specifically seeks to answer the following:

1. How will the proposed machine copy a desired color in terms of:
    1.1 image scanning?
    1.2 color processing?
2. How accurate does the proposed machine copy a desired paint color in terms of:
    2.1 machine vision technology?
3. How will the machine utilize five paint colors to produce different color shades and tints in terms of:
    3.1 RGB to CMY-Wht-Blk?
    3.2 RGB to RYB-Wht-Blk?
4. What are the advantages of using the proposed machine compared to manual paint mixing in terms of:
    4.1 production time?
        4.1.1 scanning time?
        4.1.2 mixing time?
    4.2 accuracy?

1.5 Scope and Limitations

This project covers the production of a desired color derived from the three primary colors in the color wheel which are red, blue and yellow, together with white and black to add saturation and value. The machine will run with a minicomputer board with Python programming language. The machine has high-definition camera for color sample input. The machine also saves the color samples input for future use. Peristaltic pumps are used on each of the paint color containers to control the dispensing of the paints. Mixer motor will be used to mix the dispensed paint colors. Another high-definition camera is set on the output container to compare whether the machine achieved the correct color. Sensors will also be placed inside the paint containers to monitor the level of the paints. There will be a self-cleaning option which will be done on the paint tubes and mixer of the machine.

The color scanner is limited to detect a surface of monochromatic color only. The machine will utilize red, blue, yellow, white and black paint colors only. The machine is limited only to water-based paints for easy maintenance of containers. Water-based paint has different types that can be used in concretes, woods, etc. The standard output paint volume will be one liter (1L) only. The refilling process of paints will be manually done by the users. The self-cleaning feature of the machine will only be applicable on the paint tubes and mixer.

1.6 Significance of the Study

With the development of this study, many people will take advantage of its functions. Upon completion of this research, the machine will be beneficial to the following:

Paint Users. This study will help paint users to ease the time consuming and manual paint mixing. This will help them lessen the inconvenience of trial and error in achieving a desired paint color output.

Paint Centers. This study will help paint centers in producing wide range of color paints. This will help them make more color paints as this machine is time efficient in producing different colors.

Proponents. This study will help the proponents apply and improve their acquired knowledge in the chosen field.

Future Researchers. This study will help the future researchers to develop the project and make it more productive. This will benefit the future researchers by providing them the facts needed to their related study.

1.7 Definition of Terms

Arduino Mega. A microcontroller board that has 54 digital input/output that was used to control the motors, ultrasonic sensor and peristaltic pump.

Color Scanner. This term refers to the part of the machine that captures the sample image for the replication of color.

Machine Vision. This term refers to the technology used in color scanning that processes images to get its accurate color values.

Paint Mixer. This refers to the machine that automates the mixing of paint from the generated paint ratio calculated by the machine vision.

Raspberry Pi. A single board computer that is used for the image processing of scanned color to generate the paint ratio needed to replicate the paint color.

Chapter 2

REVIEW OF RELATED LITERATURE AND STUDIES

2.1 Foreign and Local Literatures and Studies

Production of Paints in the Philippines

In a national report in 2013 by Calonzo, et.al, the Philippine paint manufacturing business is producing some 250 million liters of various paint products per year and enjoying at least a 5 percent growth rate as the nation's economy steadily flourishes. This means that the demand for coatings and paints is sure to grow as residential and commercial construction projects continue to expand. This concludes that majority of Filipino prefers to use paint to decorate and color their residential and commercial spaces. Stated in a blog posted by Dela Cruz from her site blancnotes.com, she experienced purchasing paints in a hardware store near their house and did a manual mixing in which they took a lot of hits and mixes before they got their desired color almost right. However, they were not satisfied with the results that is why they tried the computerized paint mixing machine offered by Ace Hardware. A code from the chosen color was inputted to the computer then the machine automatically mixed it. The machine's innovative technology did not require them to have an eye for color or be good in the art of mixing colors in order to achieve their desired look for their house. Davies Paints Philippines in 2016 also offers a fully computerized paint mixing service in the Philippines. They stated that having the said machine will eliminate time consuming hit or miss trials and errors with consistent color accuracy. However, this is also used by choosing a shade in a thousand of ready to mix colors.

An article by Suraj Moraje stated Philippines will face profound changes in the nature of work and jobs. This refers to the automation opportunity in the Philippines. The effects of

automation in the Philippines could be significant in which at 61%, manufacturing has the highest proportion of automatable work of any sector in the country.

Manual Paint Mixing

In an article written by Stephen Pierce in 2014, he narrated that back in the day, mixing paint would have you to take the lid off the container, grab a wooden stick, and get it. However, inventions came around that would allow people to mix the paint using an electric mixer but still would need to take the lid off and manually do the ratios. An article written by Propertyware in 2017, mixing in the past was up to the paint associate. However, the associate could mix more or less of some colors that affects the shade of the match.

With an ever-growing palette of colors, customers are increasingly in need of customized paints and paint colors. (Milhorn, 2014) Many hardware stores and paint centers allow customers to purchase customized paints that are not readily available in their color selection. Customized paints are achieved by mixing a full-size container of the selected color paint. This requires help of the store personnel to identify and obtain the correct base paint, open the lid, and operate the color mixer. It was also stated that customers spend significant amount of time and money given the number of paint colors the customer tries out. In 2014, Muftah et. al, it is stated that the paint industry is one kind of process that depends on the automatic control techniques. The paint composition is the most important issue in the paint industry that is why a number of techniques have been developed to overcome such problem. In the past, paint composition was done by manual mixing which often does not meet the desired or the required color.

Automation

Automation is defined as the application of machines to tasks once performed by human beings or, increasingly, to tasks that would otherwise be impossible. Although the term

mechanization is often used to refer to the simple replacement of human labor by machines, automation generally implies the integration of machines into a self-governing system. (Groover, 2019)

According to Michael Chui et al., automation technology can already match, or even exceed, the median level of human performance required and it is likely to change the vast majority of occupations—at least to some degree—which will necessitate significant job redefinition and a transformation of business processes.

In 2018, Clint A. Bowers et al. stated that automated systems have further altered the type of task performed by operators from one involving direct control of the system to one requiring monitoring and supervisory control and management of the systems. In fact, many automated systems might have the unintended consequence of interfering with team communication and coordination. Automated systems with decision-aiding capabilities are also expected to dramatically alter the organization and allocation of tasks in complex systems. As task load increased, greater workload was reported by operators using automated systems. The workload for captains may be reduced because the automated systems eliminated tasks associated with overall system and crew management. In comparison, the workload for first officers may be increased because of the requirement to operate the automated systems and to monitor the performance of the automated systems.

As stated in the article of Dr. Florian Kongoli, automation is closely related to the modern need for sustainable development in the 21st century. One of the principles of sustainability is "Doing More with Less" which in other words, is also one of the goals of automation. By replacing the routine part of human labor with the use of machines, automation not only increases productivity and the quality of products beyond what can be achieved by humans but also frees space, time and energy for humans to deal with the new, non-routine

challenge of developing innovative and more advanced technologies. This magnificent cycle in which established developments are automated and the free resources achieved by this automation are used to develop newer technologies that are subsequently automated is one of the most successful recipes for the human race towards the goal of sustainable development.

Automation of Paint Mixer

Graco in 2015 stated a number of reasons why painting should be automated. (1) Cost, while years pass by, the cost of automation is decreasing. The initial investment can still be large, but the finishing process will increase output while reducing costs and providing benefits for years to come. This can be considered as developing a long-term plan for manufacturers. (2) Quality, even the most skilled painters cannot ensure the consistency and precision that robots and automated machine can do. Better consistency means less wasted material. (3) Technology, accurate mixing and ratio assurance will increase overall productivity of the machine. (4) Less waste, paint automation can reduce material consumption up to 30% as it accurately calculates the ratios in mixing. (5) Flexibility, automated machines and robots can easily be programmed to do new jobs. Automation also provides flexibility to modify materials and colors, and production planning. And (6) Workforce, automated machines and robots benefit the workforce in doing repetitive and tedious tasks that reduces on-the-job hazards. According to the International Federation of Robotics (IFR), the number of industrial painting robots sold worldwide is expected to reach 400,000 machines a year by 2018. They predicted that global sales will grow at a year-over-year average rate of 15%.

For this reason, the automatic paint preparation techniques take place to solve the

manual paint composition. In this research, an automatic color machine depends on the LabVIEW techniques as a control algorithm is designed and fabricated. The proposed control technique is one kind of graphical control techniques that uses a block diagram and icons as a control algorithm. The LabVIEW program is divided in two parts. The first is the front panel (user interface) and the second is Block diagram (algorithm code). The LabVIEW provided the user with user interface panel which will help the user to feed the quantity of the desired new color. The precision of the output color was acceptable compared to the original color.

By automating the process of paint mixing, the duration will be less than the manual paint mixing and the accuracy of the output will be more certain. Using only four colored colorants: red, blue, yellow and black, the system can produce a number of complex-colored paints as an output. The production of paints will be also enhanced by adding a feature that can scan the color of any peeled off paints or papers and produce the paint with the same color as the scanned object. (Bunquin, et.al, 2017)

Dispensing System

A dispenser is a system that is able to blend colours to achieve the specific colour and viscosity you need for the job and at the same time, produce the exact amount of printing ink/paint required. As stated in the blog of Base Products in 2017, there are several benefits and advantages of a dispenser, it includes: (1) Very accurately dispense the correct amounts of ingredients for a batch, reducing overmakes due to colour tinting at the quality control stage. (2) Consistently produces the exact same product, batch after batch due to the precise and accurate measurements of ingredients. (3) Improved lead times due to the speed of dispensing, and always “right first-time batches” – saving money and time. (4) Reduce environmental impact by reducing wastage [overmakes] and having the ability to recycle or rework old stock/returns back into new blends. (5) Reduce labour costs due to minimal staff

needed to operate the dispenser, and prevents errors in manual weighing. (6) High quality dispenser components ensure a long operating life and maximum reliability. (7) Systems can be custom designed to meet the needs of the customer. (8) Dispenser can be used for solvent, water based or oil-based liquids. Typically, dispenser also has a recirculation loop to maintain the materials suspension and prevent any settling out of products, etc.

Color Recognition using Machine Vision Technology

Machine vision and automated visual inspection are domains of technology with a steadily increasing economical relevance. Although the related industry is notably expanding since the past two decades, only a part of today's visual inspection tasks have been automated. This is why there is a great potential for economization in high income countries which may lead to both reduced costs and increased quality of the produced goods. (Beyerer, León, & Frese, 2015).

Image recognition nowadays is bringing revolutionary changes to the ways in which we consume and process information online (Golemanova, 2017). The applications of image recognition are diverse and empowering; this includes color extraction. The identification and analyzation of colors in images gives numerous possibilities to businesses.

There are studies in the Philippines regarding the approaches that can be used in color recognition and identification. Luta, Baldovino and Bugtai in 2017 said that color information is broadly used in several applications with today's current technology. Color machine vision (CMV) consists of any application of vision process involving color of images which is useful in determining material property, inspection and sorting, searching and locating, and measuring and matching. CMV can perform repeated tasks and can be calibrated to different scenarios unlike that human eye. Color machine vision will be suitable in color scanning ability in automation of paint mixing. However, CMV algorithms may not be very effective for images

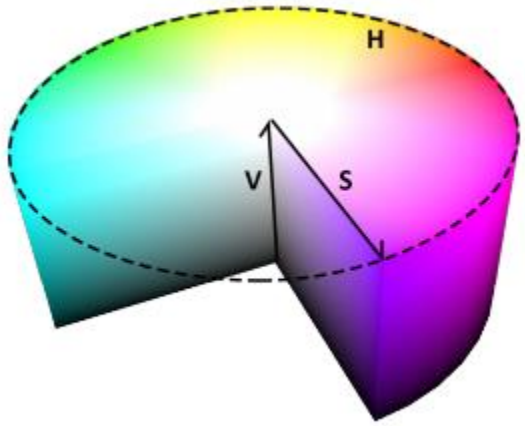
where the color of an object is present in different shades, hues or luminosity. Images taken with poor lighting, bad camera angles or low quality cameras may be interpreted by the system differently from what it was designed to do. This is often the problem that can be encountered in color scanning. This why the researchers enumerated common image processing techniques. (1) Image enhancement, a type of image preprocessing the focuses on improvement of the image interpretability. This includes noise filtering, brightness manipulation, and sharpness improvement. Noise filtering is one of the common necessary steps in image processing because most raw images are influenced by background noise. (2) Color space conversion, this refers to conversion of images to another color space before going to image processing techniques. One example is converting RGB color space into HSV color space. These techniques are mainly used in images with different colors. Color recognition of plain colors using these techniques will be more accurate understanding these principles and techniques.

Color Mixing Implementation

According to Raj & Jena in 2014, color mixing is an important process which has a wide application in several fields. There are various kinds of color mixing. One is the additive color mixing which refers to blending of colors of light which has three essential colors: red, green and blue. When there are no colors indication, the outcome is dark; otherwise, the outcome is white. This is utilized as a part of TV and workstation screens to generate an extensive variety of shades from just three colors. Another is the subtractive color mixing which is carried out by evacuating certain shades. The three essential colors in this kind are yellow, magenta, and cyan. This is utilized to make a mixture of colors when printing on paper and when painting. However according to Lindgren in 2017, CMYK paint colors cannot be easily found on the market. Neither RGB nor CMYK color models seem ideal for this project.

Thus, another model is needed which is the HSV. HSV stands for Hue, Saturation and Value. Hue could first be matched by mixing of the primary colors, then saturation and value could be matched by adding in white and black respectively.

Figure 3. Graphical representation of HSV color model by Lindgren and Thiel



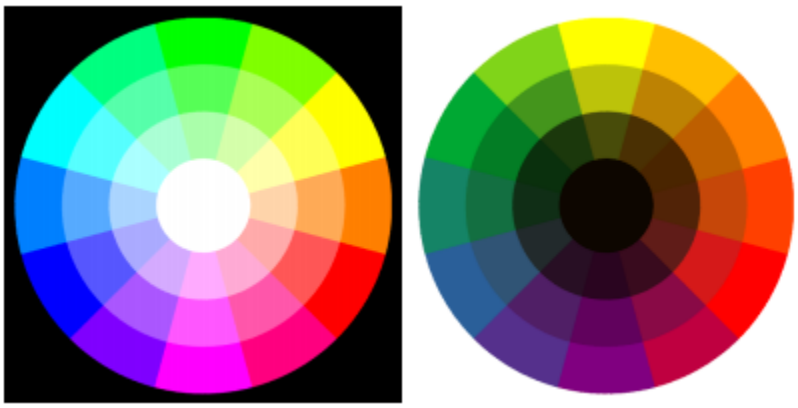
The figure shown above is a remapping of RGB color model into cylindrical coordinates and was created to mimic how an artist mixes color. This maps the hue as an angle on a color circle. The saturation becomes the distance from the center and the value becomes the height along the cylinder.

HSV is often used in image processing and computer vision fields. The RGB model was also discussed in this study where it is stated that RGB employs the principle of human eye functionality like imitating the sensitivity of three types of cones in retina to specific light spectra making visible colors reproduced by adding various intensities of red, green and blue lights. (Chernov et. al., 2015).

As observed from Gossett's and Baoquan's study, it would be difficult to try to dene a rigorous mathematical conversion from RYB to RGB for the purpose of display. However, a reasonable approximation may be obtained by dening a cube with each axis representing either Red, Yellow or Blue. By dening appropriate RGB values for each of the eight colors

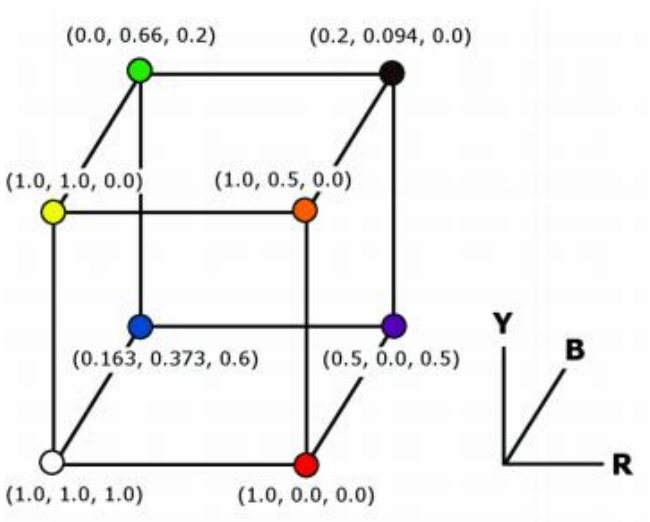
represented by the corners of the cube, we can use trilinear interpolation to obtain suitable RGB values for any color dened in RYB. For instance, if a color were to be dened as 100% Red, 50% Yellow and 25% Blue, it could be represented as (1.0, 0.5, 0.25) in RYB coordinates. By interpolating the RGB values dened at the 8 corners of the cube, this color would have RGB coordinates of (0.8375, 0.19925, 0.0625), producing a slightly muddy orange color as expected.

Figure 4. RGB and RYB Color Wheel Representation



Left: RGB color wheel. 12 o'clock is pure Green, 4 o'clock is pure Red, and 8 o'clock is pure Blue. 2, 6 and 10 o'clock are two primaries at 100%. The rest of the positions are produced with one primary at 100% and another at 50%. Colors are saturated at the outside of the circle and desaturated at the center. Right: RYB with non-linear interpolation. This wheel is produced by using the RGB values from the left image as RYB values. The colors produced are approximately the colors suggested by Itten with the exception of Blue-Green (9 o'clock) which is darker in our model.

Figure 5. RYB interpolation cube



RGB coordinates are given for each corner of the RYB cube. RGB colors are chosen based on Itten's suggestions.

2.2 Synthesis of the Reviewed Literature and Studies

The gathered literatures and studies have similarities on the subject matter of the present research material of the proponents. The literatures and studies that include facts and information about the production of paint in the Philippines and the manual mixing of paint support the proponent's idea of automating paint mixer with color scanning ability. As narrated by Dela Cruz, manual paint mixing took a lot of hits and mixes before getting the desired color. This problem is also what this study is focusing to. There are also studies about the type of color mixing in which will support the development of this study. The proponents will be using the HSV model in calculating the color ratios. According to Lindgren in 2017, HSV model which stands for Hue, Saturation, and Value will be ideal for this kind of project. There are also studies about color machine vision in which different factors were considered in getting the accurate values of the color. These similar studies will give the proponents additional and new knowledge in order to develop this project.

There are already existing automated paint mixing machine in the Philippines like those of Ace Hardware and Davies Philippines' computerized paint mixing. However, these machines limit people to color swatches pre-defined on the system. These machines do not allow customers to directly scan the color sample they desired. Color recognition with paint mixer machines are also existing in other countries however, they used color sensors in which the proponents deemed it inaccurate for color scanning. The study will focus on machine vision technology in which image processing is applied and is hardcoded.

In consonance with Graco's and Base Products' statements, creating automated dispensing systems and paint machines will surely benefit the workforce in doing repetitive and tedious tasks. Also, the benefits from the automated system such as efficiency, quality, conservation of paint and safeness from harmful toxins was observe from the study of Raj Tapas and Jena Ashirvad.

These studies and literatures enumerated different theories and principles that will possibly be used to support the proposed machine. Different problems and factors may be encountered during the study but will be solved practically and theoretically through these supporting studies.

Chapter 3

METHODOLOGY

3.1 Methods of Research

To reach the main objectives of this study, the proponents used an experimental research method. Moore and McCabe stated that the best method of establishing causation is to conduct a carefully designed experiment in which the effects of possible lurking variables are controlled. To experiment means to actively change x and to observe the response in y. Manipulation and controlled testing will be done to understand the processes in getting the desired output.

The proponents find experimental research method suitable for this study because it was used to solve practical problems and to maintain theoretical assumptions in accordance to color recognition and paint mixing. A series of experiments were conducted to test the designed prototype to achieve the purpose of the project. These tests were done to determine the most accurate algorithm for the RGB to RYB-Wht-Blk values and CMY-Wht-Blk.

In certain conditions, qualitative data gathered by the proponents were converted to numerical data and analyzed by quantitative methods so that there is more analytic comparison established.

3.2 Data Gathering Procedure

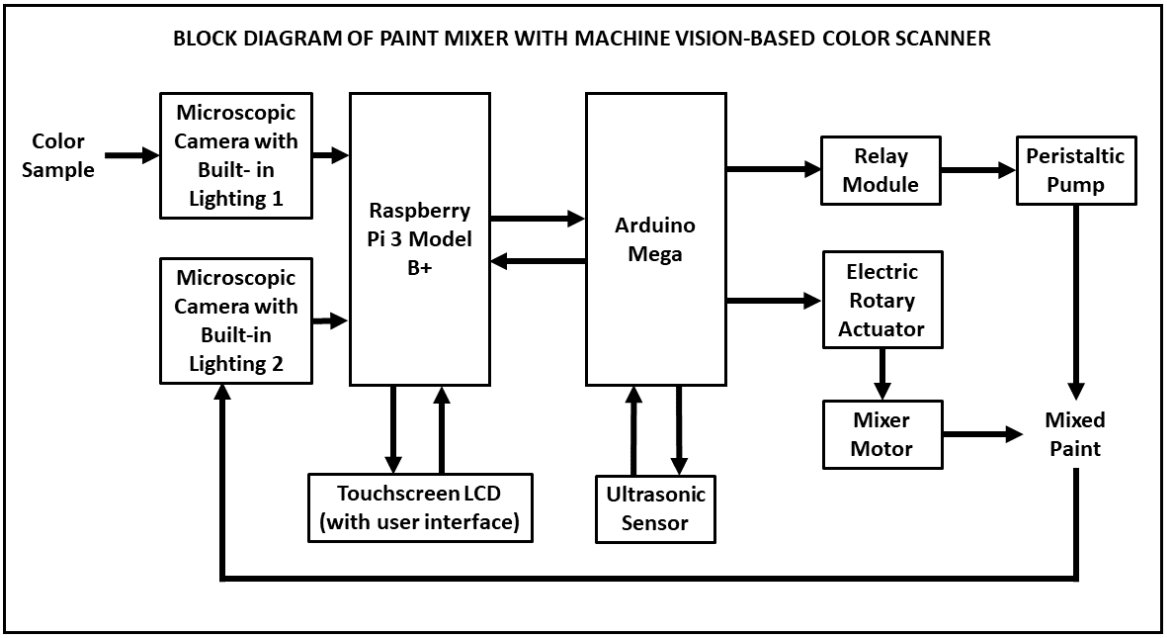
The proponents used different research journals, studies, and articles to acquire relative knowledge about the project. The data are mainly about machine vision technology, conversion of RGB to color models, and the color mixing principle. The proponents created different algorithms to establish the most accurate way of accomplishing the desired output.

The data gathering is conducted through series of experiments. The proponents analyzed different algorithms in order to get the most accurate way of color recognition through machine vision which will determine the paint volume ratio to achieve the desired paint mixture. The algorithm runs in series of tests, and data is recorded to solve practical problems and improve any sufficiency. The proponents need to evaluate and determine the precise way in paint mixing to achieve the exact shade of the desired color.

Also, an interview to different paint centers in NCR-Philippines was conducted by the proponents to gather data. This includes the purpose, significance and benefit of the study through the questions that seek specific answers. The questions were related to the automation of paint mixing with color scanning ability. The data gathered is considered as quantitative data in order to have a numerical comparison. This data gathering procedure will help the proponents to conclude with the cost-benefit analysis of the project.

3.3 Block Diagram

Figure 6. Block Diagram of Paint Mixer with Machine Vision-based Color Scanner



The machine will be controlled by a mini computer board which is Raspberry Pi. Once turned on, the user can capture a color sample using a high-definition camera installed on the machine. The user can interact on the user interface part of the system. The camera will send the information to the minicomputer in which image will be processed. This is where the machine vision technology goes. After determining the values of the color sample, the algorithm made by the proponents will convert RGB to RYB-Wht-Blk values in order to determine the proportions of the standard colors in the paint containers. The minicomputer will send signal to the Arduino Mega to control the pumps connected to the paint containers in order to dispense paints to the output container. While needed paints are being dispensed, signal to mixer motor are sent from the minicomputer. Mixing process will stop after a period of time. There will be ultrasonic sensors in each container to monitor the paint levels.

3.4 Schematic Diagram

Figure 7. Schematic Diagram of Paint Mixer with Machine Vision-Based Color Scanner

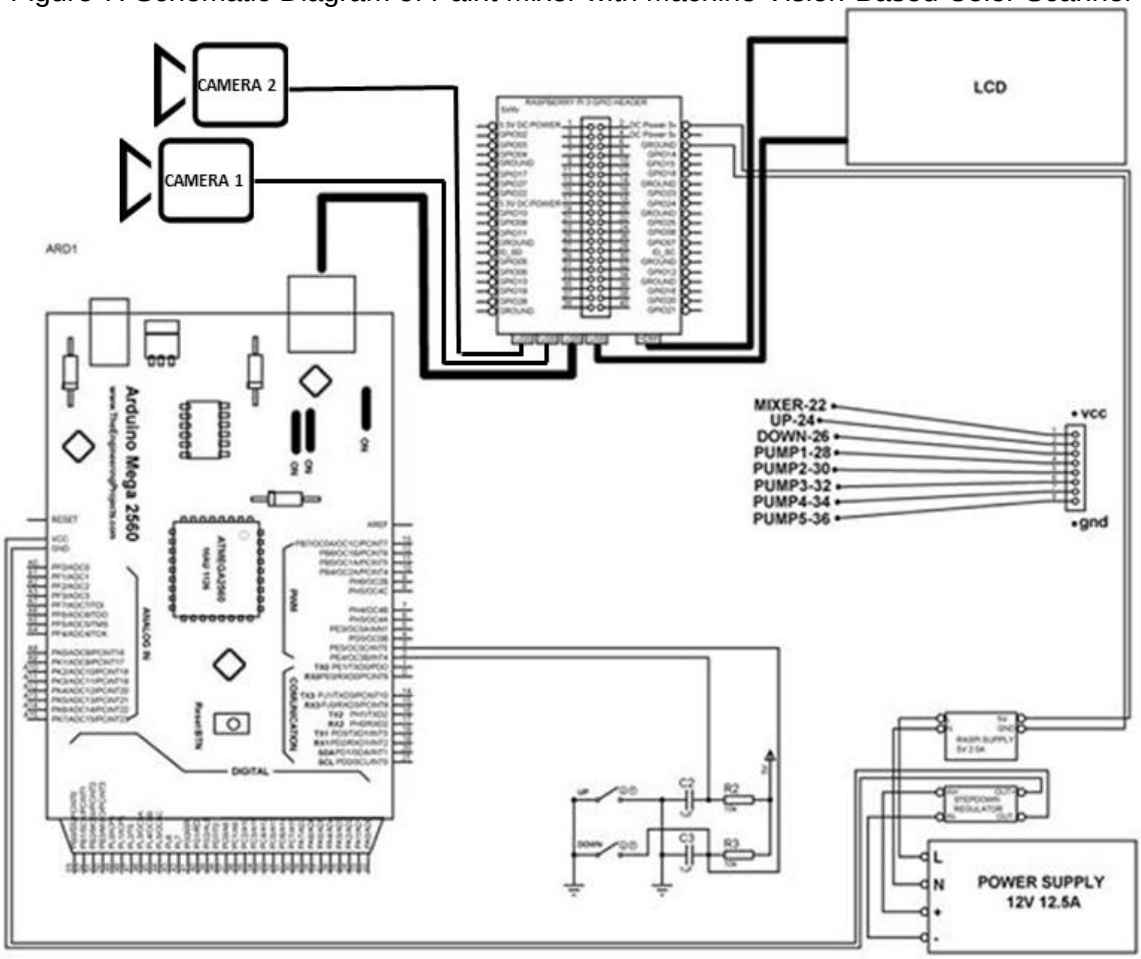
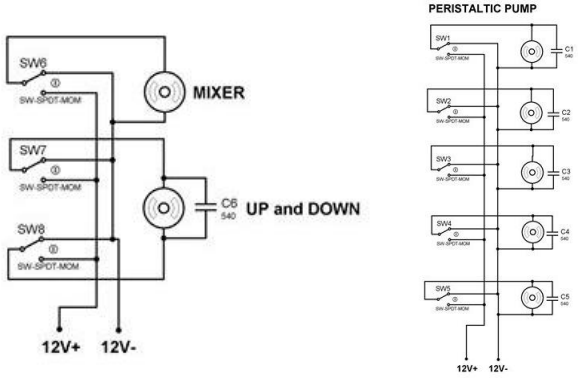


Figure 8. Schematic Diagram for Peristaltic Pumps and Paint Mixer of the Machine



3.5 Flowchart

Figure 9. Flowchart of Paint Mixer with Machine Vision-Based Color Scanner

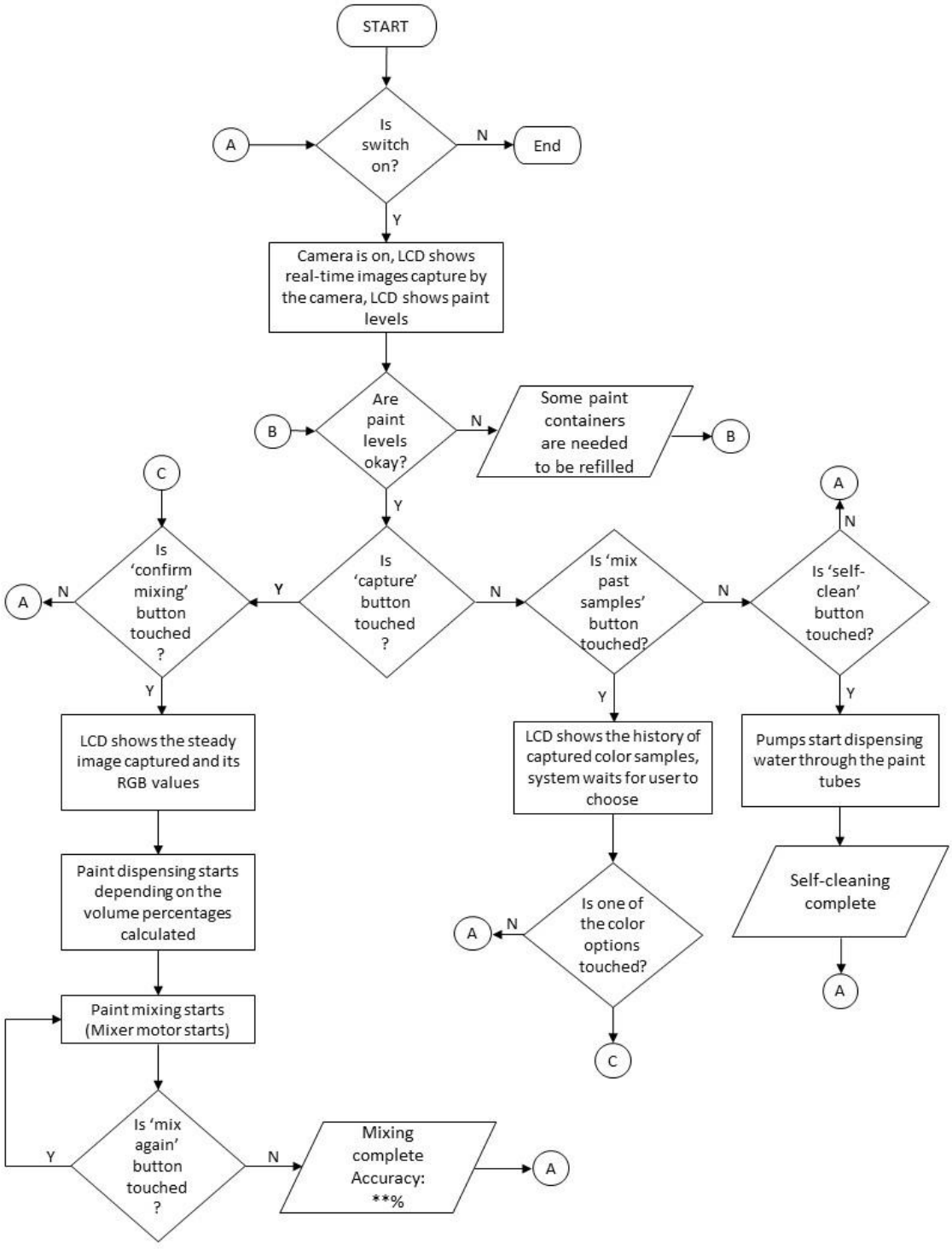


Figure 9 shows the process of the prototype. As the machine is turned on, the graphical user interface shows the images that the camera is getting, and the paint levels. The prototype will not let the user to start capturing images if the one or more of the paint levels are below normal. Otherwise, options will be available such as capture, mix past samples and self-clean.

If the ‘capture’ button is touched, the system will be asking the user for confirmation, and if confirmed, the machine will start the dispensing and mixing process of the paints. There is also a ‘mix again’ option after the mixing process that can be used if ever the user is not satisfied with the paint mixture. The mixer will be turned on for a period of time.

The machine has a feature that saves the captured color samples for future use. If the ‘mix past sample’ button is touched, the display will show the history of the color samples mixed by the machine. The machine will let the user choose which of the color options the user wants to mix again. After choosing a desired color, the machine will then be starting the dispensing and mixing process of the paints. A ‘mix again’ option can also be used.

If the ‘self-clean’ button is touched, the machine will start dispensing water through the paint tubes and mixer motor will be turned on.

Chapter 4

PRESENTATION, ANALYSIS AND INTERPRETATION OF DATA

This chapter presents the findings obtained from observation and experimentations conducted for the study. It discusses the parameters gathered from the experiments and the analysis done in each group of data. The experiments were done with the manual paint mixing and automated paint mixing for the comparison.

The proponents used same color samples in every experiment for accurate comparison of the results. The data results are analyzed in order to determine the best method in replicating the desired color using the machine.

1. How does the machine copy a desired color in terms of:

1.1 Image Scanning

The proponents first used a 480p HD Web Camera compatible to Raspberry Pi Model B+ in scanning a sample color. The desired sample color must be in a proper lighting when scanned by the camera to be able to get its true color value. The proponents then decided to try if a camera with higher quality would increase the accuracy of the image scanning. The error will be computed by averaging the differences of scanned RGB values from true RGB values. The accuracy will be computed by subtracting error from 100%.

$$accuracy = \left(1 - \frac{\frac{|R1 - R2|}{255} + \frac{|G1 - G2|}{255} + \frac{|B1 - B2|}{255}}{3}\right) \times 100\%$$

This accuracy formula will be used in all of the accuracy computations in this chapter.

Table 1

480p HD Web Camera with 1/6" CMOS Sensor

Color	Actual RGB Value	Scanned Color	Scanned RGB Value	Accuracy
	121,105,71		169,137,165	77.25%
	95,211,128		135,170,151	86.41%
	1,195,255		83,180,254	87.19%
	255,163,0		185,144,147	69.15%
	244,150,255		167,132,217	82.61%
			Average Accuracy	**80.52%**

Table 2

1080p HD PC Camera

Color	Actual RGB Value	Scanned Color	Scanned RGB Value	Accuracy
	121,105,71		115,121,116	91.24%
	95,211,128		126,131,122	84.71%
	1,195,255		48,145,243	85.75%
	255,163,0		179,110,20	80.52%
	244,150,255		112,115,112	59.48%
			Average Accuracy	**80.34%**

Table 3

Microscopic Camera with 1/3" CMOS Sensor and Built-In Lighting

Color	Actual RGB Value	Scanned Color	Scanned RGB Value	Accuracy
	121,105,71		235,172,156	65.23%
	95,211,128		106,191,125	95.56%
	1,195,255		55,161,222	84.18%
	255,163,0		253,173,70	89.28%
	244,150,255		224,141,170	85.10%
			Average Accuracy	**83.87%**

The data tables above show that having a camera with higher quality is a factor in image scanning. Same color values are scanned by different quality of camera to compare the accuracy produced. The proponents observed that the camera with higher image sensor (CMOS sensor) capture the almost true color of the sample because this sensor converts light into electrons. Also, proper lighting is a factor in getting the accurate color of the sample. Using the microscopic camera with higher CMOS sensor and built-in lighting is an advantage when getting a value of a desired color. However, the disadvantage of the microscopic camera is that due to the built-in lighting in the camera, the dark color (brown, for example) registered light in the camera. But with light colors, the registration of color values accuracy is high. Table 4 shows the proof of lighting being a factor in capturing the color of a sample.

Table 4

Microscopic Camera with 1/3" CMOS Sensor and Built-In Lighting (Low Lighting)

Color	Actual RGB Value	Scanned Color	Scanned RGB Value	Accuracy
	121,105,71		123,89,43	93.99%
	95,211,128		95,166,118	92.81%
	1,195,255		103,110,107	56.21%
	255,163,0		225,130,70	82.61%
	244,150,255		205,142,162	81.7%
			Average Accuracy	**81.46%**

The colors in table 4 was captured using the same camera as in table 3 but with less lighting. It is shown that the dark color brown has been captured with higher and better accuracy than in table 3. It is advised to calibrate the lighting of the camera to get its true color.

1.2 Color Processing

After image scanning, the machine must process the color values of the scanned images to determine the color ratio needed to mix the paint. The proponents decided to compare which method will best process the color values of the image scanned. It is considered that the image scanned will not be plain due to some factors like lighting, shadows, and noise. Therefore, the proponents come up with two ways to get the nearest or even the actual RGB value of the scanned image.

The proponents first thought of getting the dominant color value of the scanned image. The machine will look for the color value that is produced mostly by the scanned image. This means that the RGB color value with the greatest number of pixels will be the RGB value to be converted into subtractive color values.

Another way is getting the average RGB value of all the RGB values of each pixel in the scanned image. The machine will go through each pixel of the scanned image and identify its color value then average it all. The average will be the RGB value to be converted to subtractive color model like RYB or CMYK. The tables below are the test samples to compare the accuracy of the two methods stated above.

Table 5

Getting the dominant color value

Color	Actual RGB Value	Scanned Color	Dominant color value (RGB)	Accuracy
	121,105,71		198,50,50	80%
	95,211,128		120,220,111	93.33%
	1,195,255		45,170,199	83.66%
	255,163,0		230,175,80	71.63%
	244,150,255		230,180,160	81.83%
			Average Accuracy	**82.09%**

Table 6

Averaging the color value of each pixel

Color	Actual RGB Value	Scanned Color	Average color value (RGB)	Accuracy
	121,105,71		123,89,43	93.99%
	95,211,128		106,191,125	95.56%
	1,195,255		55,161,222	84.18%
	255,163,0		253,173,70	89.28%
	244,150,255		224,141,170	85.10%
			Average Accuracy	**89.61%**

Table 4 and 5 shows that by averaging the color value of the scanned image, the accuracy is greater than getting the dominant color value.

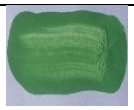
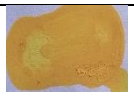
2. How accurate does the proposed machine copy a desired paint color in terms of:

2.1 Machine Vision Technology

To determine the color replication accuracy, the tests are conducted with the help of cameras to compare the RGB values of the input color and the output color. The output color pertains to the mixed paint produced by the machine.

Table 7

Input Color and Output Paint Mixture Comparison

Input Color Sample (RGB)	Scanned Input Color with RGB value	Scanned Output Color with RGB value	Accuracy
121,105,71	123,89,43	 186,171,164	70.72%
95,211,128	106,191,125	 151,211,108	89.28%
1,195,255	55,161,222	 90,191,207	89.54%
255,163,0	253,173,70	 254,182,57	96.99%
244,150,255	224,141,170	 214,164,170	95.69%
		Average Accuracy	**88.44%**

The accuracy is computed by comparing the RGB values of the scanned input color and the RGB values of the scanned output paint color. Accuracy is calculated by the machine and is shown on the screen real-time. Table 7 shows that the machine can produce paint color mixture with an accuracy up to 96.99% based on the test samples.

3. How will the machine utilize five paint colors to produce different color shades and tints in terms of:

3.1 CMYK Color Model

The proponents referred the formula for the conversion of RGB to CMY-Wht-Blk from the related studies they have gathered. The formula is written below:

$$white = \max\left(\frac{red}{255}, \frac{green}{255}, \frac{blue}{255}\right)$$

$$cyan = \frac{(white - \frac{red}{255})}{white}$$

$$magenta = \frac{(white - \frac{green}{255})}{white}$$

$$yellow = \frac{(white - \frac{blue}{255})}{white}$$

$$black = (1 - white)$$

However, this formula is only applicable to additive color mixing which is applied on digital screens. When all color values are high, white is produced. While for actual color mixing, the concept of subtractive color mixing is needed in which black is produced when all color values are high. The proponents adjusted the formula based on the subtractive color mixing and used the following formulas to establish the most accurate way to replicate the RGB values scanned by the input camera.

$$black = \min[\left(1 - \frac{R}{255}\right), \left(1 - \frac{G}{255}\right), \left(1 - \frac{B}{255}\right)]$$

If black is equal to 1, then:

$$cyan = 0; magenta = 0; yellow = 0;$$

Otherwise,

$$cyan = \frac{\left(1 - \frac{R}{255}\right) - black}{1 - black}$$

$$magenta = \frac{\left(1 - \frac{G}{255}\right) - black}{1 - black}$$

$$yellow = \frac{\left(1 - \frac{B}{255}\right) - black}{1 - black}$$

This formula is derived from the presentation of RGB to CMY and CMY to CMYK conversions. The CMY equivalent of RGB values is determined by getting the difference of the percentage values of RGB from white. It is stated that when all color values are high in subtractive mixing, black is produced. Therefore, the minimum value in CMY is distinguished to be able to determine how much will the black (or the key, K) be. Then, the black value will be subtracted by the remaining color value to determine the CMY with the K value.

However, this formula uses only CMYK and did not consider white in the mixture since it treats white as its base color. Subtractive color mixing pertains to the subtraction of lightness to white. Thus, the proponents treated white as its base color to get its ratio to the mixture. White is computed as written below:

White is initialized as equal to zero, then the statement below is iterated four times. The variable ‘color’ is replaced with values of cyan, magenta, yellow and key in each iteration.

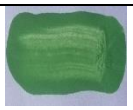
If color is not equal to zero, then:

$$white += 1 - color$$

Experiments based on the theory above is done and shown on the table below.

Table 8

Conversion of RGB to CMY-Wht-Blk

Input Color (RGB)	Scanned Color (RGB)	Cyan	Magenta	Yellow	White	Black	Output Mixture	Accuracy
121,105,71	123,89,43	0	13.82	32.52	50.22	3.44		87.42%
95,211,128	106,191,125	22.5	0	13.125	60	4.375		89.28%
1,195,255	55,161,222	69.8	4.31	0	25.88	0		89.54%
255,163,0	253,173,70	0	15.84	36	48	0.16		90.99%
244,150,255	224,141,170	0	18.5	10.56	69.5	1.44		88.69%
							Average Accuracy	**89.18%**

3.2 RYB Color Model

Since RGB is an additive color mixing model, high values of red, green and blue produce white. Whiteness component must be removed from the RGB values. White can be calculated by getting the minimum value from RGB and subtract it from red, green and blue values to get the new RGB. The formula is written below.

$$white = \min(R, G, B)$$

$$r = R - white$$

$$g = G - white$$

$$b = B - white$$

$$black = \min(1 - R, 1 - G, 1 - B)$$

Based on the data gathered by the proponents, RYB values are obtained from the red, green, blue values calculated by the equation above.

$$red = r - \min(r, g)$$

$$yellow = \frac{g + \min(r, g)}{2}$$

$$blue = \frac{b + g - \min(r, g)}{2}$$

Black is produced when all color values of subtractive color mixing are high. So, black components are added for subtractive color mixing. With this, the real values of red, yellow and blue with black are obtained with the formula given below.

$$Red = red + black$$

$$Yellow = yellow + black$$

$$Blue = blue + black$$

The formulas stated are theory to convert RGB color values to primary colors RYB with white and black. Experiments based on this are shown in table 9.

Table 9

Conversion of RGB to RYB-Wht-Blk

Input Color (RGB)	Scanned Color (RGB)	Red	Yellow	Blue	White	Black	Output Mixture	Accuracy
121,105,71	123,89,43	19.38	9.85	6.4	64.37	0	130,110,80	76.47%
95,211,128	123,211,142	4.28	15.74	34.44	45.53	0	48,180,90	79.35%
1,195,255	78,173,233	6.65	17.6	19.96	55.79	0	216,130,56	53.20%
255,163,0	253,179,54	24.71	41.46	0.41	33.42	0	70,180,100	69.93%
244,150,255	228,145,168	18.02	4.42	6.31	71.52	0	198,100,170	89.93%
						Average Accuracy		**73.78%**

The experiments conducted above test the ability of the paint colors to be mixed that gives greater accuracy. Though the proponents expected the RYB-Wht-Blk to be the most convenient to use in this project, table 8 shows greater accuracy than in table 9. This means that CMY-Wht-Blk produces better output than using RYB-Wht-Blk in mixing actual colors.

4. What are the advantages of using the proposed machine compared to manual paint mixing in terms of:

4.1 Production Time

4.1.1 Scanning Time

Scanning time refers to the decision-making of color composition of the color to be replicated.

Table 10

Manual Color Scanning and Automated Color Scanning

Color	Manual		Automated	
	Color Composition	Scanning Time	Color Composition	Scanning Time
121,105,71	Magenta, yellow, white, black	4 mins 35s	13.82% Magenta, 32.52% Yellow, 50.22% white, 3.44% Black	< 1min
95,211,128	Yellow, blue, white	2 mins	13.125% Yellow, 22.5% Cyan, 60% White, 4.375% Black	< 1min
1,195,255	Blue, white	3 mins 20s	4.31% Magenta, 69.8% Blue, 25.88% White	< 1min
255,163,0	Red, yellow, white	3 mins	15.84% Magenta, 36% Yellow, 48% White, 0.16% Black	< 1min
244,150,255	Red, white	2 mins	18.5% Magenta, 10.56% Yellow. 69.5% White, 1.44% Black	< 1min
Average Time	**2 mins 58s**		**< 1 min**	

Table 10 shows the comparison of scanning time between human effort and machine effort. The proponents asked the paint expert of the project beneficiary what the color composition of the sample colors in the table is. The color composition given by the paint expert was not based on the primary colors, he based his judgment by the availability of the nearest color to the sample. The proponents asked the paint expert for the color composition of the sample colors if there were only five paint colors available which are blue, red, yellow, white and black. This is to fairly compare the ability of human judgment and machine judgment in utilizing only five paint colors to produce different shades.

The table shows that the manual scanning takes time in deciding which colors to use especially when there is a limited paint colors available to mix. Also, the paint expert could not

quantify how much of the colors are needed in the mixture. He based his mixture on estimation. The paint expert first mixed on a hard paper to test his assumptions. The machine was able to scan the color immediately and calculate the values of the paints to be used as the camera landed on the sample color.

4.1.2 Mixing Time

Table 11

Manual Color Mixing and Automated Color Mixing

Color	Manual		Automated	
	Color Composition	Mixing Time	Color Composition	Mixing Time
121,105,71	Red, yellow, white, black	13 mins 30s	13.82% Magenta, 32.52% Yellow, 50.22% white, 3.44% Black	5 mins
95,211,128	Yellow, blue, white	4 mins	13.125% Yellow, 22.5% Cyan, 60% White, 4.375% Black	4 mins
1,195,255	Blue, white, black	5 mins	4.31% Magenta, 69.8% Blue, 25.88% White	4 mins
255,163,0	Red, yellow, white	7 mins	15.84% Magenta, 36% Yellow, 48% White, 0.16% Black	4 mins
244,150,255	Red, white, blue	6 mins 40s	18.5% Magenta, 10.56% Yellow. 69.5% White, 1.44% Black	4 mins
Average Time	**7 mins 13s**		**4 mins 2s**	

Table 11 shows the comparison of mixing time between manual and automated. After scanning the color samples, the paint expert mixed the colors based on his judgment and experience. However, there were some cases that the anticipated color composition was not correct, thus, the expert would have to adjust and use more or other color. There were also instances that if the paint expert mixed the paint too dark or too light, he would decrease the

volume of the paint he was mixing then add more light or dark colors to achieve the desired color. This means that there are times that there would be paint wastage. The table shows that the machine can mix the paint with less time than the manual mixing. This is because the time for adjustment is included in the mixing process of the manual way.

4.2 Accuracy

Table 12

Manual Paint Mixing and Automated Paint Mixing

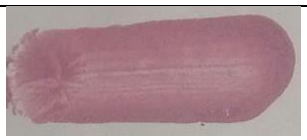
Color	Manual		Automated	
	Output Color	Accuracy	Output Color	Accuracy
121,105,71		84.05%		87.42%
95,211,128		89.54%		89.28%
1,195,255		83.14%		89.54%
255,163,0		90.2%		90.99%
244,150,255		81.18%		88.69%
Average Accuracy	**85.62%**		**89.18%**	

Table 12 shows the comparison of the color replication accuracy between manual and automated. The proponents used the camera of the machine to determine the accuracy of

each paint mixed manually and by machine. The table shows a little amount of difference with the accuracy between manual and automated. However, the average accuracy of automated is greater than manual, concluding to an improvement of color replication in the paint industry.

Chapter 5

SUMMARY OF FINDINGS, CONCLUSIONS AND RECOMMENDATIONS

This chapter presents the findings, ideas, concepts and added recommendations for future development regarding the composition of the whole system.

Summary of Findings

The automation of paint mixing, integrated with machine vision technology, will be able to improve the color replication in the paint industry. The most common way in achieving a desired paint color is through manual estimation and trial and error that often takes up time and huge amount of paint to squander.

The design project "Paint Mixer with Machine Vision-Based Color Scanner" made it possible to reduce the time consumed in manual paint mixing and to save more paints through a system that could automate the process using the five colors.

In order to test the effectiveness of the tools used in the project, the proponents experimented on different approach in color replication. Considering the color identification accuracy, machine vision technology can be implemented in determining the properties of the scanned color. With this, a high definition camera is needed in order to capture the color of the sample. By processing the image captured, color composition will be derived which will set the colors needed for the mixing of paint to replicate the color. After having able to determine which colors are to be used, the color mixing theory is applied to come up with the best method to convert the scanned color value into an actual paint color ratio.

Based on the data gathered during the experiments, the proponents were able to identify which will be the appropriate way to replicate a desired paint color. The proponents were also able to signify that the proposed project is effective compared to the traditional and

manual paint mixing. The system was able to enhance the color replication and paint mixing process in terms of its production time and accuracy.

Conclusions

There were different approaches tested to know which concepts and theories would work best for the development of the machine.

The application of machine vision technology is a vital part of the machine. Using a camera with good CMOS and proper lighting, the machine will be able to recognize the color properties of input sample that will be replicated. By averaging the color value of each pixel in the scanned image, the machine will be able to conclude the color value of the scanned image. After which, this RGB values must be converted into RYB-Wht-Blk values for actual paint color mixing. However, after accomplishing the experiments conducted, the proponents discovered that the CMYK concept is better to use in color mixing than using the primary colors in the color wheel which are red, yellow and blue. Thus, the proponents decided to use CMY-Wht-Blk in producing the desired color.

The proponents managed to accomplish constructing a machine that can be able to determine the color values of a certain sample and mixing it based on the subtractive color values computed by the machine. The machine is able to produce variety of paint colors using water-based paints only. The proponents were able to put a self-cleaning feature on the machine.

Recommendations

Based on the data gathered and drawn conclusions, the following recommendations are hereby made:

The development of Paint Mixer with Machine Vision-Based Color Scanner shows a success in automation of paint mixing process. However, the proponents believe that there are more rooms for improvement that the developed machine has.

The proponents recommend the future researchers for the automation of paint mixing to integrate the color scanning process to install a higher quality camera which may be more expensive. These kinds of cameras may give more vibrance and scan the truest color of the sample. Also, considering the fast-paced technology we have today, the camera may also be thru a smartphone that is connected to the machine. Since the machine scans color of plain surface only, the feature of the color scanner can be improved by allowing it to scan every surface available. Enhancing the operation of the machine using the Internet can also be possible.

Also, this machine uses five paint colors such as shades of cyan, magenta, yellow, black and white. The proponents think that it is possible if the machine will be able to function with input paints of different shade of blue, red and yellow.

The proponents believe that the number of volume produced by the machine can be increased and can be able to have a variety of output volume like the standard volume of paints in the market today.

REFERENCES

Emptyeasel.com (2007). *Using the Color Wheel: Colot Theory Tips for Artists and Painters.* Retrieved from https://emptyeasel.com/2007/03/16/using-the-color-wheel-color-theory-tips-for-artists-and-painters/

Lindgren, M., & Thiel, M. (2017). *Automatic Color Mixer: A method for automated color recognition and replication.* Retrieved from www.diva-portal.se › smash › get › diva2:1201176 › FULLTEXT01

Circuito.ioblog (2018 December 19). *Single Board Computers.* Retrieved from https://www.circuito.io/blog/single-board-computer/

Calonzo, M., Guarino, J., Tolibas, M., Lucero, A., Brosche, S., Emeritus, S., Denney, V., & Weinberg V. (2013 November). *Lead in New Enamel Household Paint in the Philippines.* Retrieved from https://ipen.org/sites/default/files/documents/Philippine%20Paint%20Report%202013%20(FINAL).pdf

Pierce, S. (2014 October 14). *Paint Mixer*. Retrieved from http://blog.mixerdirect.com/blog/paint-mixer/

Propertyware (2017 July 31). *Paint Matching Technology: An Interview with The Home Depot.* Retrieved from https://www.propertyware.com/blog/paint-matching-technology-interview-home-depot/

Milhorn, K. E. (2014). *Color Dispensing System and Method.* Retrieved from https://patents.google.com/patent/US8666540B2/en

Muftah, M. A., Albagul, A. M., & Faraj, A. M. (2014). *Automatic paint mixing process using LabView.* Retrieved from https://www.researchgate.net/publication/280039677_Automatic_paint_mixing_process_using_LabView

Groover, M. P. (2019). *Automation, production systems, and computer-integrated manufacturing. Prentice Hall Press.* Retrieved from https://dl.acm.org/citation.cfm?id=1537195

Bowers, C. A., Oser, R. L., Salas, E., & Cannon-Bowers, J. A. (2018). *Team performance in automated systems. In Automation and Human Performance (pp. 243-263). Routledge.* Retrieved from https://www.taylorfrancis.com/books/e/9781315137957/chapters/10.1201/9781315137957-12

Chui, M., Manyika, J., & Miremadi, M. (2015). *Four fundamentals of workplace automation. McKinsey Quarterly, 29(3), 1-9.* Retrieved from https://roubler.com/au/wp-content/uploads/sites/9/2016/11/Four-fundamentals-of-workplace-automation.pdf

Graco (2015). *5 Reasons to Say Yes.* Retrieved from https://www.graco.com/content/dam/graco/ipd/literature/misc/automation_5_reasons/Automation_5_reasonsEN-A.pdf

Macro Source Media (2017 October). *Robotics Industry Market Report.* Retrieved from http://www.macrosourcemedia.com/uploads/6/2/1/8/62183211/robotics_industry_october_2017.pdf

Base Products (2017 December 4). *Ink and Paint Dispensing Systems.* Retrieved from http://www.baseproducts.co.za/author/admin/

Beyerer, J., León, F. P., & Frese, C. (2015 October 1). *Machine vision: Automated visual inspection: Theory, practice and applications. Springer.* Retrieved from https://books.google.com.ph/books?id=UAmkCgAAQBAJ&dq=(Beyerer,+Le%C3%B3n,+%26amp%3B+Frese,+2015).&lr=&source=gbs_navlinks_s

Golemanova, R. (2017, March 23). *The Top 5 Uses of Image Recognition.* Retrieved from https://imagga.com/blog/the-top-5-uses-of-image-recognition/

Luta, R. B. G., Baldovino, R. G., & Bugtai, N. T. (2017, December). *Image preprocessing using quick color averaging approach for color machine vision (CMV) systems. In Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM), 2017 IEEE 9th International Conference on (pp. 1-4). IEEE.* Retrieved from https://ieeexplore.ieee.org/document/8269475/

Raj, T., & Jena, A. (2014). *Design and development of an automated paint mixing machine (Doctoral dissertation).* Retrieved from http://ethesis.nitrkl.ac.in/5899/1/110ID0273-1.pdf

Chernov, V., Alander, J., & Bochko, V. (2015 August). *Integer-based accurate conversion between RGB and HSV color spaces. Computers & Electrical Engineering, 46, 328-337.* Retrieved from https://www.sciencedirect.com/science/article/pii/S0045790615002827

Gossett, N., & Chen, B. (2004, October*). Paint inspired color mixing and compositing for visualization. In Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on (pp. 113-118). IEEE.* Retrieved from http://web.siat.ac.cn/~baoquan/papers/InfoVis_Paint.pdf

Appendix 1

COMPONENTS OF THE PROPOSED SYSTEM

1. Raspberry Pi 3 Model B+

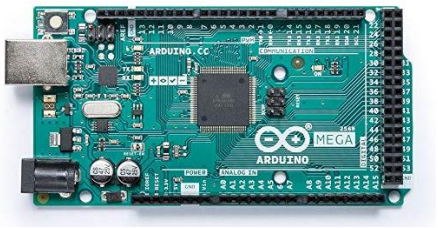
Raspberry Pi is a small computer with an ARM processor that can run Linux. This has 1 GB of RAM, dual-band WiFi, Bluetooth 4.2, Bluetooth Low Energy (BLE), an Ethernet port, HDMI output, audio output, RCA composite video output (through the 3.5 mm jack), four USB ports, and 0.1″-spaced pins that provide access to general purpose inputs and outputs (GPIO). This requires a microSD card with an opening system on it.

This is used to process the image scanned by the camera.

2. Arduino Mega 2560

The Mega 2560 is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins, 16 analog inputs, 4 UARTs, a 16MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything to support the microcontroller.

This is used to control the hardware components of the machine.



3. CISNO USB 2.0 Digital Microscope

Digital 2.0 USB microscope allows you to work quickly and effectively at the microscopic level. 0X-1000X Magnification range make it can be used on classroom exploration, stamp or coin analysis, micro-soldering, garden parasite identification, and much more.

4. Maxon 12V DC Motor

Electrical machine that converts direct current into mechanical energy.

5. Peristaltic Pump (12V DC)

A peristaltic pump is a pump, operated by a motor, that is able to uptake a liquid through one tube and drip it out through another tube. This is used for pumping

fluids such as waste sludges, lime and cement mortar, adhesives, and shear-sensitive fluids such as latex paints.

This is used for paint dispensing.

6. Ultrasonic Sensor

An ultrasonic sensor is a device that can measure the distance to an object by using sound waves. It measures distance by sending out a sound wave at a specific frequency and listening for that sound wave to bounce back.

This is used to check the level of the input paints.

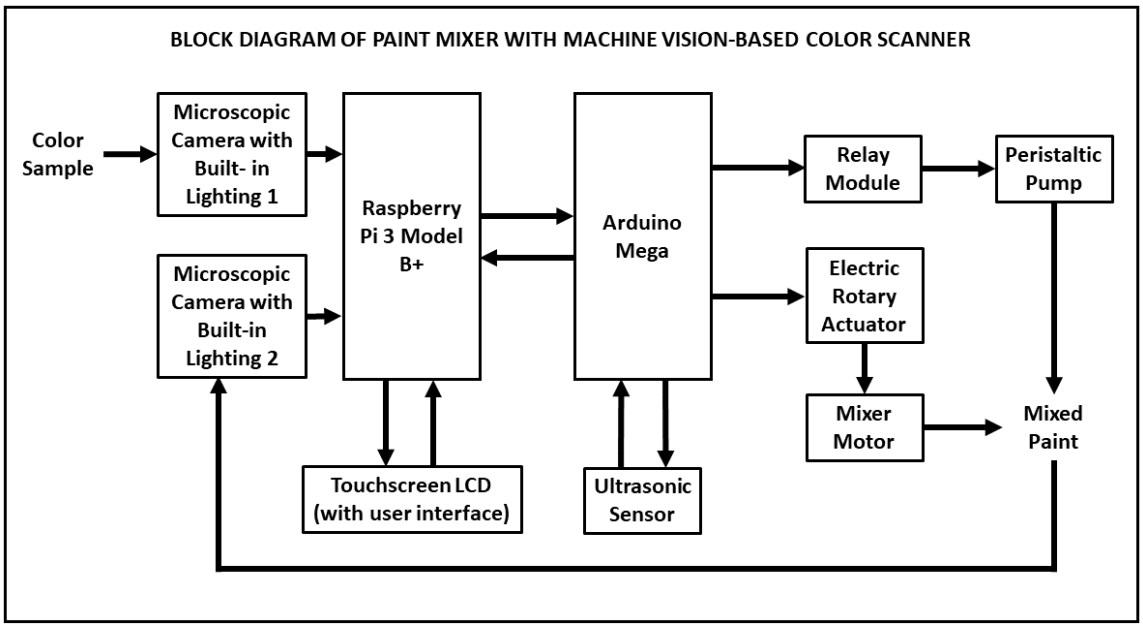
Appendix 2

BILL OF MATERIALS / COSTING

Quantity	Description	Unit Cost (Php)	Total Cost (Php)
1	Power Supply	950	950
1	Raspberry Pi 3 Model B+	2,500	2,500
1	Raspberry Pi Official Supply	450	450
1	USB Type B Cable	100	100
1	Relay Module	500	500
1	PCB	15	15
1	12-pin Pin header	45	45
3	Capacitor 104 uf	3	6
5	10k Resistor	1	5
2	20-meter Solid Wire (22)	20	40
40	Male to Female Wire	5	200
1	1-meter Soldering Lead	20	20
1	7” Standard LCD	2,610	2,610
1	USB Micro B Cable	100	100
1	HDMI Cable	200	200
1	12V DC Motor – 470 RPM	500	500
1	Maxon Motor 12 V	700	700
5	Peristaltic Pump	2,500	12,500
1	10-meter Polyethylene Tubes	233	233
1	Stainless Wire Whisk	300	300
5	Ultrasonic Sensor	320	1,600
1	Bolt & Nut	7	7
6	Latch Hasp Lock	24	144
4	Caster Wheel	200	800
1	Rocker Switch	55	55
1	Plug	20	20
2	16 MP (480p) HD Web Camera	1,500	3,000
2	1080p HD PC Camera	450	900
2	2 MP Microscopic Camera	635	1,270
20	Cable Tie	0.5	10
1	16 Gb SD Card	495	495
1	Square Tube Stainless 1/2x1	550	550
1	Arduino Mega	810	810
1	Limit Switch	35	175
1	Flatsheet Stainless 2B 1.5 MM	2,925	2,925
	Machine Casing Fabrication	40,000	40,000
		TOTAL	**74,375**

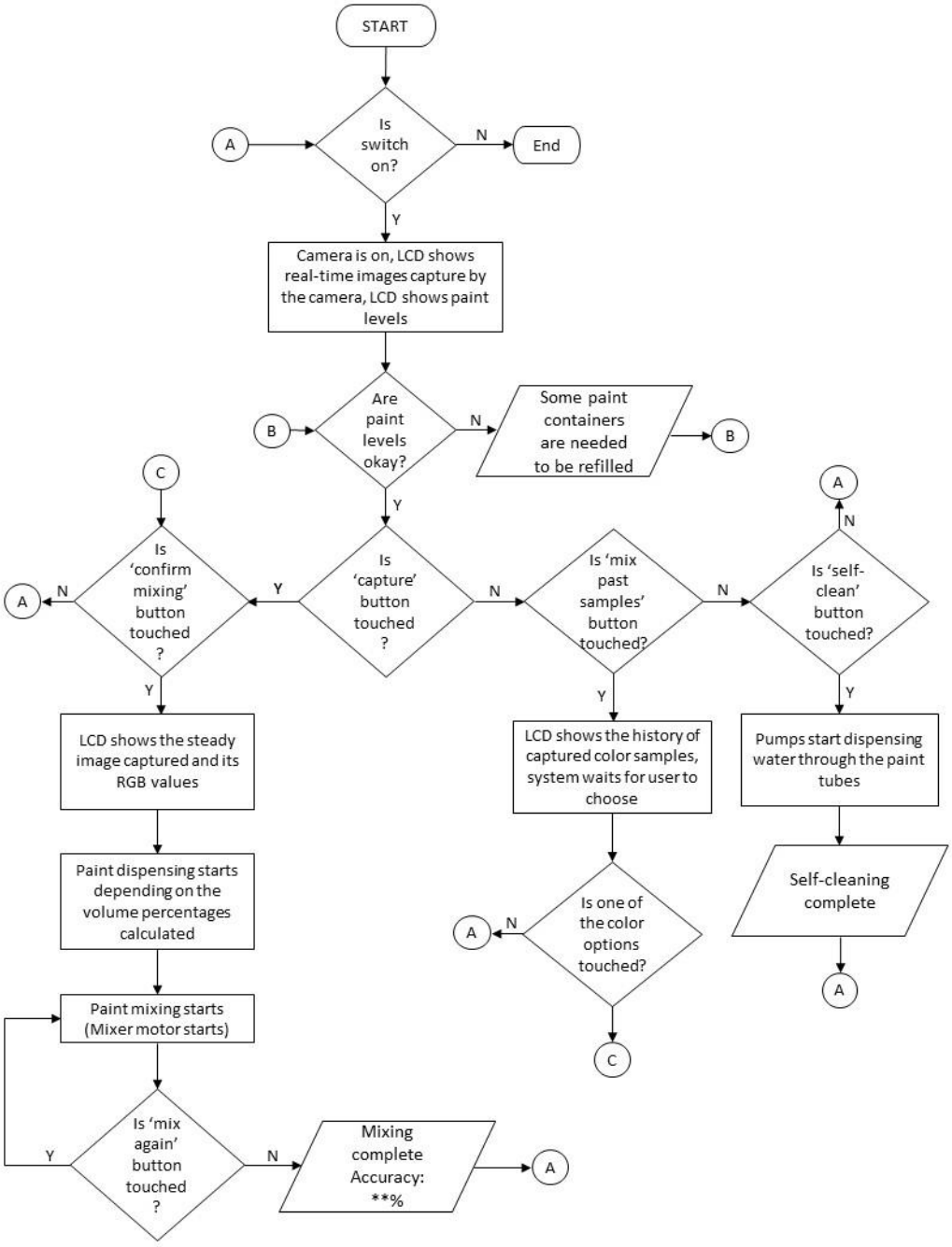
Appendix 3

BLOCK DIAGRAM



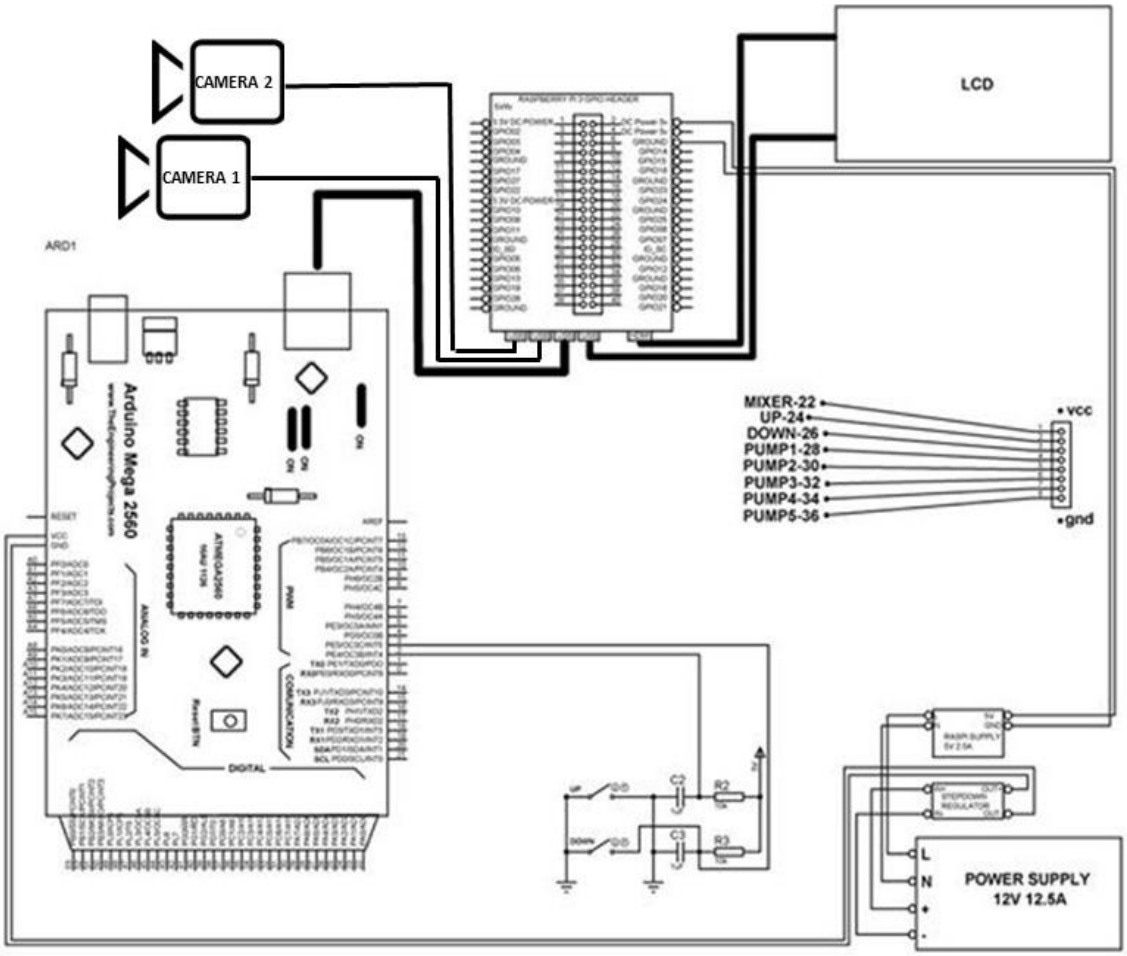
Appendix 4

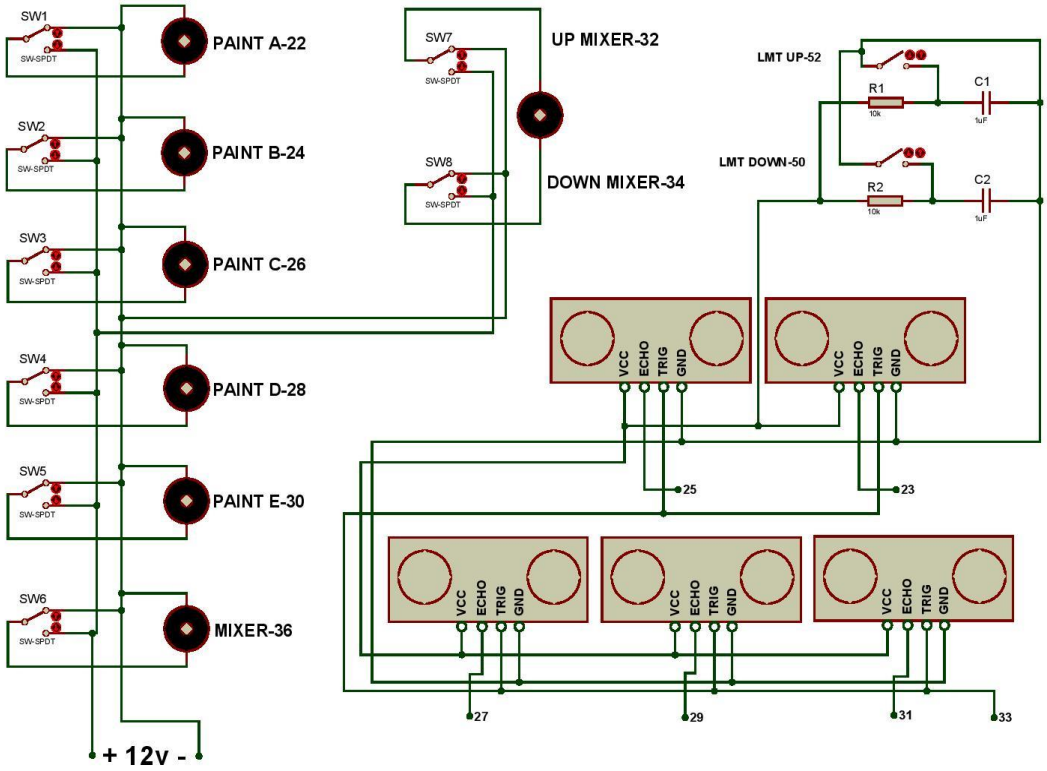
FLOWCHART



Appendix 5

WIRING DIAGRAM




SW1
SW-SPDT
PAINT A-22
SW2
SW-SPDT
PAINT B-24
SW3
SW-SPDT
PAINT C-26
SW4
SW-SPDT
PAINT D-28
SW5
SW-SPDT
PAINT E-30
SW6
SW-SPDT
MIXER-36
+ 12v -
SW7
SW-SPDT
UP MIXER-32
SW8
SW-SPDT
DOWN MIXER-34
LMT UP-52
R1
10k
C1
1uF
LMT DOWN-50
R2
10k
C2
1uF
VCC
ECHO
TRIG
GND
25
23
27
29
31
33

Appendix 6

SOURCE CODE

```
directory =
"/home/pi/Desktop/Paint_Mixer/"
LowPercentage = 10

import cv2 as cv2
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 serial
import numpy as np
import time
import re
import myfunctions as my
import os
import datetime
import ImportantFunctions as IF
import colorsys

SerialData =
serial.Serial('/dev/ttyACM0', 9600,
timeout = 0.2)
time.sleep(2)
SerialData.write(bytes("1", "utf-8"))
x = SerialData.read().decode("utf-
8")
while (x != '0'):
    x = SerialData.read().decode("utf-
8")
    print(x)

root=tk.Tk()    #Main Root
root.attributes('-zoomed', True)
root.attributes('-fullscreen', True)
root.configure(background='white')
subFont1 =
tkinter.font.Font(family='Segoe UI',
size = 15, weight = "bold")
subFont2 =
tkinter.font.Font(family='Segoe UI',
size = 10, weight = "bold")
subFont3 =
tkinter.font.Font(family='Segoe UI',
size = 8, weight = "bold")

cap1=cv2.VideoCapture(0)
cap2=cv2.VideoCapture(1)
toStops = 0

#CONVERSIONS:----------------------
----------------------------------------
def detAveColor(Image_Input):
  myimg = cv2.imread(Image_Input)
  avg_color_per_row =
np.average(myimg, axis=0)
  avg_color =
np.average(avg_color_per_row,
axis=0)
  blue = int(avg_color[0])
  green = int(avg_color[1])
  red = int(avg_color[2])
  return blue, green, red

def detColorAccuracy(blue1,
green1, red1, blue2, green2, red2):
  try:
    blue = (abs(blue1 -
blue2))/blue1
  except:
    blue = 0
  try:
    green = (abs(green1 -
green2))/green1
  except:
    green = 0
```

```
 try:
    red = (abs(red1 - red2))/red1
 except:
    red = 0

 error = (blue + green + red)/3
 accuracy = 1 - error
 accuracy = accuracy*100
 accuracy = round(accuracy,2)
 return accuracy

def rgb_to_hls(r, g, b):
   r,g,b = r/255, g/255, b/255
   h, l, s = colorsys.rgb_to_hls(r, g,
b)
   h, l, s = h*255, l*255, s*255
   return h,l,s

def hls_to_rgb(h, l, s):
   h, l, s = h/255, l/255, s/255
   r, g, b = colorsys.hls_to_rgb(h, l,
s)
   r, g, b = int(r*255), int(g*255),
int(b*255)
   return r,g,b

def rgb2ryb(R, G, B):

      R = R/255
      G = G/255
      B = B/255

      I = min(R,G,B)

      r = R - I
      g = G - I
      b = B - I


      red = r - min(r,g)
      yellow = (g + min(r,g))/2
      blue = (b + g - min(r,g))/2
      try:
            n =
(max(red,yellow,blue))/(max(r,g,b))
            redn = red/n
            yellown = yellow/n
            bluen = blue/n
      except:
            redn = 0
            yellown = 0
            bluen = 0

      I = min(1 - R ,1 - G, 1 - B)


      Redn = redn + I
      Yellown = yellown + I
      Bluen = bluen + I
      White = 3*(min(1-Redn,1-
Yellown,1-Bluen))
      sums = Redn + Yellown +
Bluen + White

      Red = (Redn/sums)*100
      Yellow = (Yellown/sums)*100
      Blue = (Bluen/sums)*100
      White = (White/sums)*100
      blk = 0

      if (Red > 28 and Yellow > 28
and Blue > 28):
            blk = Redn + Yellown +
Bluen
            Red = 0
            Yellow = 0
            Blue = 0
            White = 3*(max(R,G,B))
            sums = blk + White
            blk = (blk/sums)*100
            White =
(White/sums)*100

      return Red, Yellow, Blue,
White, blk

def
Create_Color_Screen(Output_File,
```

```
DimX, DimY, R, G, B):
    img = np.zeros([DimY, DimX, 3],
dtype=np.uint8)
    img[:,:] = [B, G, R]
    IF.imwrite(Output_File,img)
#-----------------------------------------
--------------------------
ToStopCapturing = 0
def CaptureButFunction():
    global ToStopCapturing
    global red1
    global green1
    global blue1
    global H
    global S
    global L
    global BaseL
    if ToStopCapturing == 0:
        ToStopCapturing = 1
        CaptureButton.configure(text =
"Retake")
        print("RGB: "+str(red1)+ " "
+str(green1) + " " + str(blue1))
        H, L, S = rgb_to_hls(red1,
green1, blue1)
        BaseL = L
        print("HLS: "+str(H)+ " " +str(L)
+ " " + str(S))
        scale1.set(int(L))

scale1.configure(state=tk.NORMAL)
    elif ToStopCapturing == 1:
        ToStopCapturing = 0
        CaptureButton.configure(text =
"Capture")

#BUTTONS:---------------------------------
---------------------------------
def CheckPaintLevels():
    SerialData.write(bytes("B", "utf-
8"))
    time.sleep(2)
    x =
SerialData.readline().decode("utf-
8")
    print(x)
    Blacks =
IF.GetINFO(x,"AAA","BBB")
    Blues =
IF.GetINFO(x,"BBB","CCC")
    Yellows =
IF.GetINFO(x,"CCC","DDD")
    Reds =
IF.GetINFO(x,"DDD","EEE")
    Whites =
IF.GetINFO(x,"EEE","FFF")
    print("Reds: " + str(Reds))
    print("Blues: " + str(Blues))
    print("Yellows: " + str(Yellows))
    print("Blacks: " + str(Blacks))
    print("Whites: " + str(Whites))

    if float(Reds) < 0:
        Reds = "0"
    if float(Blues) < 0:
        Blues = "0"
    if float(Yellows) < 0:
        Yellows = "0"
    if float(Blacks) < 0:
        Blacks = "0"
    if float(Whites) < 0:
        Whites = "0"
    if float(Reds) > 100:
        Reds = "100"
    if float(Blues) > 100:
        Blues = "100"
    if float(Yellows) > 100:
        Yellows = "100"
    if float(Blacks) > 100:
        Blacks = "100"
    if float(Whites) > 100:
        Whites = "100"
    text = ("Red: " + Reds + "%
\nBlue: " + Blues + "%\nYellow: " +
Yellows +
            "%\nBlack: " + Blacks +
"%\nWhite: " + Whites + "%")
    tk.messagebox.showinfo('Ink',
```

```
text)

def CheckPaintLevels2():
    SerialData.write(bytes("B", "utf-
8"))
    time.sleep(2)
    x =
SerialData.readline().decode("utf-
8")
    print(x)
    Blacks =
IF.GetINFO(x,"AAA","BBB")
    Blues =
IF.GetINFO(x,"BBB","CCC")
    Yellows =
IF.GetINFO(x,"CCC","DDD")
    Reds =
IF.GetINFO(x,"DDD","EEE")
    Whites =
IF.GetINFO(x,"EEE","FFF")
    print("Reds: " + str(Reds))
    print("Blues: " + str(Blues))
    print("Yellows: " + str(Yellows))
    print("Blacks: " + str(Blacks))
    print("Whites: " + str(Whites))

    if float(Reds) < 0:
        Reds = "0"
    if float(Blues) < 0:
        Blues = "0"
    if float(Yellows) < 0:
        Yellows = "0"
    if float(Blacks) < 0:
        Blacks = "0"
    if float(Whites) < 0:
        Whites = "0"
    if float(Reds) > 100:
        Reds = "100"
    if float(Blues) > 100:
        Blues = "100"
    if float(Yellows) > 100:
        Yellows = "100"
    if float(Blacks) > 100:
        Blacks = "100"
    if float(Whites) > 100:
        Whites = "100"
    return Reds, Blues, Yellows,
Blacks, Whites

LX=250
LY=140
def Mixing(FileName):
    global text
    global LoadingFrame2
    global MixingMode
    blue1, green1, red1 =
detAveColor(FileName)
    R, Y, B, W, blk = rgb2ryb(red1,
green1, blue1)
    R = round(R, 2)
    B = round(B, 2)
    Y = round(Y, 2)
    blk = round(blk, 2)
    W = round(W, 2)

    text =
"A,"+str(R)+","+str(B)+","+str(Y)+","
+str(blk)+","+str(W)+",Z"

    print(text)
    if ToClean == 1:
        SerialData.write(bytes("C",
"utf-8"))
    elif ToClean == 2:
        SerialData.write(bytes("H",
"utf-8"))
    else:
        SerialData.write(bytes("A",
"utf-8"))
    global LX
    global LY
    if MixingMode == 1 or
MixingMode == 2:
        LoadingFrame.place(x=LX,
y=LY)
##    elif MixingMode == 2:
##        LoadingFrame2.place(x=LX,
y=LY)
```

```
    ChangeFrame()
    root.after(500, ChangeFrame)
    root.after(1000, ChangeFrame)
    root.after(1500, ChangeFrame)
    root.after(2000, ChangeFrame)
    root.after(2100, SendText)

def SendText():
    SerialData.write(bytes(text, "utf-
8"))
    root.after(500, ChangeFrame)
    root.after(1000, ChangeFrame)
    root.after(1500, ChangeFrame)
    root.after(2000, ChangeFrame)
    root.after(2100, WaitForA)

CountForWaiting = 0
def WaitForA():
    global CountForWaiting
    global MixingMode
    global ImageToMix
    global LoadingFrame2
    x = SerialData.read().decode("utf-
8")
    if x == "A":
        print("Done")
        LoadingFrame.place_forget()
        LoadingFrame2.place_forget()
        root.after(1000, AskAgain)
    else:
        if CountForWaiting >= 5:
            ChangeFrame()
            CountForWaiting = 0
        CountForWaiting =
CountForWaiting + 1
        root.after(50, WaitForA)

toShowPack5 = 0
inProgress = 0

def AskAgain():
    global ImageToMix
    global toStops
    global toShowPack5
    global inProgress
    global ToClean
    if ToClean == 0 or ToClean == 2:
        result =
tk.messagebox.askquestion("Mix",
"Mix Again?", icon='warning')
        ToClean = 2
    else:
        result =
tk.messagebox.askquestion("Clean"
, "Clean Again?", icon='warning')
    if result == 'yes':
        if MixingMode == 1 or
MixingMode == 2:
            Mixing(directory +
"capt1.png")
##        elif MixingMode == 2:
##            Mixing(ImageToMix)
    else:
        if toShowPack5 == 1:
            pack5.place(x=263,y=0)
        toStops = 0
        inProgress = 0
        root.after(5, MainLoop)
FrameNumber = -1
def ChangeFrame():
    global FrameNumber
    global MixingMode
    global LoadingFrame2
    Files =
IF.GetFilesFolders("Loading")
    FrameNumber = FrameNumber +
1


    try:
        if MixingMode == 1 or
MixingMode == 2:

IF.tkShow(LoadingFrame,Files[Fra
meNumber],0.6)
##        elif MixingMode == 2:
##
IF.tkShow(LoadingFrame2,
Files[FrameNumber], 0.6)
```

```
    except:
        FrameNumber = 0
        if MixingMode == 1 or
MixingMode == 2:

IF.tkShow(LoadingFrame,Files[Fra
meNumber],0.6)
##        elif MixingMode == 2:
##
IF.tkShow(LoadingFrame2,
Files[FrameNumber], 0.6)

MixingMode = 0
def Mix():
    global ToClean
    global toStops
    global MixingMode
    global ToStopCapturing

    if ToStopCapturing == 1:
        ToClean = 0
        MixingMode = 1
        toStops = 1
        result2 =
tk.messagebox.askquestion("Mix",
"Are you sure you want to continue
to mix?", icon='warning')
    else:

tk.messagebox.showinfo('Warning',
'You need to capture an image
first.')
        return

    Reds, Blues, Yellows, Blacks,
Whites = CheckPaintLevels2()

    global R
    global Y
    global B
    global W
    global blk

    R,Y,B,W,blk

    result = ''
    print(R, Reds, LowPercentage)
    if R > 0 and float(Reds) <
LowPercentage:
        result =
tk.messagebox.askquestion("Warni
ng", "Low on red level, continue?",
icon='warning')
    if result == 'no':
        result = ''
        return

    if Y > 0 and float(Yellows) <
LowPercentage:
        result =
tk.messagebox.askquestion("Warni
ng", "Low on yellow level,
continue?", icon='warning')
    if result == 'no':
        return

    if B > 0 and float(Blues) <
LowPercentage:
        result =
tk.messagebox.askquestion("Warni
ng", "Low on blue level, continue?",
icon='warning')
    if result == 'no':
        result = ''
        return

    if W > 0 and float(Whites) <
LowPercentage:
        result =
tk.messagebox.askquestion("Warni
ng", "Low on white level,
continue?", icon='warning')
    if result == 'no':
        result = ''
        return

    if blk > 0 and float(Blacks) <
LowPercentage:
```

```
        result =
tk.messagebox.askquestion("Warni
ng", "Low on black level,
continue?", icon='warning')
    if result == 'no':
        result = ''
        return

    if result2 == 'yes':
        root.after(1000,Mixx)
    else:
        pass


def Cleaning():
    global ToClean
    global MixingMode
    global toStops
    ToClean = 1
    MixingMode = 1
    toStops = 1
    result =
tk.messagebox.askquestion("Clean"
, "Are You Sure?", icon='warning')
    if result == 'yes':
        root.after(1000,Mixx)
    else:
        pass

def Mixx():
    x = 1
    capt = IF.imread("capt1.png")
    if ToClean != 1 or MixingMode
!=2:
        xxx =
(str(datetime.datetime.now().strftime
("%Y")) +
str(datetime.datetime.now().strftime(
"%m")) +
str(datetime.datetime.now().strftime(
"%d")) +
str(datetime.datetime.now().strftime(
"%H")) +
str(datetime.datetime.now().strftime(
"%M")) +
str(datetime.datetime.now().strftime(
"%S")) +
str(datetime.datetime.now().strftime(
"%f")))
        cv2.imwrite((directory +
'Database/'+xxx+'.png'),capt)
    Mixing(directory + "capt1.png")

    # while (x == 1):
    #     result2 =
tk.messagebox.askquestion("Mix",
"Mix Again?", icon='warning')
    #     if result2 == 'yes':
    #         x = 1
    #         Mixing(directory +
"capt1.png")
    #     else:
    #         x = 0
    #     try:
    #         PastPlace.place_forget()
    #     except:
    #         pass
ToDelete = 0
def Mix2(CurrentImage, i):
    global ToDelete
    global MixingMode
    global ImageToMix
    global toStops
    global ToClean
    global pack5
    global inProgress

    global InputImage
    ToClean = 0
    ImageToMix = directory +
CurrentImage
    if ToDelete == 1:
        frame.grid_slaves(i,
0)[0].destroy()
        IF.Delete([], [CurrentImage])
    else:
##        toStops = 1
        NewImage =
```

```
IF.imread(CurrentImage)
        IF.imwrite("capt1.png",
NewImage)
        IF.tkShowWidth(InputImage,
"capt1.png", 220)
        global blue1
        global green1
        global red1
        blue1, green1, red1 =
detAveColor(directory +
"capt1.png")
        MixingMode = 2
        BackF()
        CaptureButFunction()

##        inProgress = 1

##        pack5.place_forget()
##        Mixing(ImageToMix)

def myfunction(event):

canvas.configure(scrollregion=canv
as.bbox("all"),width=240,
height=340, bg='white')

def Delete():
    global ToDelete
    if ToDelete == 0:
        DeleteButton.configure(text =
"DONE")
        ToDelete = 1
    else:
        DeleteButton.configure(text =
"DELETE")
        ToDelete = 0

def BackF():
    global ToDelete
    global pack5
    global toShowPack5
    global inProgress
    if inProgress == 1:
        return
    if ToDelete == 0:
        GUI1.pack()
        GUI2.forget()
        toShowPack5 = 0
        pack5.place_forget()
    else:

tk.messagebox.showinfo('Warning',
'You need to be done on deleting
first.')

def MixPast():
    global canvas
    global frame
    global pack5
    global toShowPack5
    global ToStopCapturing

    if ToStopCapturing == 1:
        CaptureButFunction()

    GUI1.forget()
    GUI2.pack()

    toShowPack5 = 1
    pack5 =  tk.Frame(GUI2,
bg='white')
    pack5.place(x=263,y=0)

    Label8 = tk.Label(pack5, text =
"Please choose an image. ", font =
subFont1, bg='white')
    Label8.pack()

    myframe=tk.Frame(pack5,
width=240,
height=340,relief=tk.GROOVE,bd=1
, bg='white')
    myframe.pack()

    canvas=tk.Canvas(myframe)
    frame=tk.Frame(canvas,
bg='white')  #The frame for the
list(the texts themselves)
```

```
myscrollbar=tk.Scrollbar(myframe,orient="vertical",command=canvas.yview)

canvas.configure(yscrollcommand=myscrollbar.set, bg='white')


myscrollbar.pack(side="right",fill="y")
    canvas.pack(side="left")

canvas.create_window((0,0),window=frame,anchor='nw')
    frame.bind("<Configure>", myfunction)

    PastImagesList = sorted(os.listdir(directory + 'Database'))
    print(PastImagesList)
    Rows = len(PastImagesList)
    print(Rows)

    for i in range(Rows):
        print(i)
        CurrentImage = PastImagesList[i]
        CurrentImage = 'Database/' + CurrentImage
        command1 = lambda x=CurrentImage, y=i : Mix2(x, y)
        Button6 = tk.Button(frame, font = subFont2, bg='white', command = command1)
        Button6.grid(row=i,column=0, padx=10, pady=3)

        print(CurrentImage)
        try:

IF.tkShow(Button6,CurrentImage,0.2)
        except:
            print("This Image is corrupt.")
            IF.Delete([],[CurrentImage])

def CloseProgram():
    quit()

def MainLoop():
    global toStops
    if toStops == 1:
        return
    print("MainLoop")
    global R
    global Y
    global B
    global W
    global blk
    global toStops
    global ToStopCapturing
    global BaseL

    if ToStopCapturing == 0:
        ret, capt1 = cap1.read()
        IF.imwrite("capt1.png",capt1)

    ret, capt2 = cap2.read()
    IF.imwrite("capt2.png",capt2)
    #VIDEO CAMERAS-------------------------------------
    try:
        IF.tkShowWidth(InputImage, "capt1.png", 220)
    except:

tk.messagebox.showinfo('Error', 'Cannot Detect Camera1')
        quit()
    try:
        IF.tkShowWidth(OutputImage, "capt2.png", 220)
    except:

tk.messagebox.showinfo('Error',
```

```
'Cannot Detect Camera2')
        quit()
    #CALCULATIONS----------------------
-----------------------------------
    #Average Color of Camera 1:
    global red1
    global green1
    global blue1
    blue1, green1, red1 =
detAveColor(directory +
"capt1.png")

    if ToStopCapturing == 1:
        global H
        global L
        global S

        L = scale1.get()
        red1, green1, blue1 =
hls_to_rgb(H, L, S)
        print(L, BaseL)
        BaseL = int(BaseL)
        if L != BaseL:

Create_Color_Screen("capt1.png",
640, 480, red1, green1, blue1)

        ##print(H,L,S)



    iiBlueLabel.configure(text=blue1)

iiGreenLabel.configure(text=green1)
    iiRedLabel.configure(text=red1)

    #RYBWBlk Conversion of BGR-
Camera1:
    R,Y,B,W,blk = rgb2ryb(red1,
green1, blue1)
    R = round(R, 2)
    Y = round(Y, 2)
    B = round(B, 2)
    W = round(W, 2)
    blk = round(blk, 2)
    rrRedLabel.configure(text=R)
    rrYellowLabel.configure(text=Y)
    rrBlueLabel.configure(text=B)
    rrWhiteLabel.configure(text=W)
    rrBlackLabel.configure(text=blk)

    #Average Color of Camera 2:
    blue2, green2, red2 =
detAveColor(directory +
"capt2.png")

ooBlueLabel.configure(text=blue2)

ooGreenLabel.configure(text=green
2)
    ooRedLabel.configure(text=red2)

    Accuracy =
detColorAccuracy(blue1, green1,
red1, blue2, green2, red2)

AccuracyLabel.configure(text="Accu
racy: " + str(Accuracy) + "%")
    root.after(5,MainLoop)

GUI1 = tk.Frame(root)
GUI1.pack()

Background = tk.Label(GUI1,
text='',font = subFont1,
bg='white',bd=0)
Background.pack()
IF.tkShow(Background, "bg.png", 1)

FirstFrameX = 123
FirstFrameY = 15

InputImage = tk.Label(GUI1, text =
"", font = subFont1, bg='white',
fg='#00a2e8')
InputImage.place(x=FirstFrameX+3
0,y=FirstFrameY+10)
IF.tkShow(InputImage, "capt1.jpg",
1)
```

```
OutputImage = tk.Label(GUI1, text
= "", font = subFont1, bg='white',
fg='#00a2e8')
OutputImage.place(x=FirstFrameX+
300,y=FirstFrameY+10)
IF.tkShow(OutputImage,
"capt2.jpg", 1)

InputFrame = tk.Label(GUI1, text =
"", font = subFont1, bg='white',
fg='#00a2e8')
InputFrame.place(x=FirstFrameX+3
0,y=FirstFrameY+190)
IF.tkShow(InputFrame, "Input.png",
1)

iiBlueLabel = tk.Label(GUI1, text =
"255", font = subFont3,
bg='#3F48CC', fg='black', width=9)
iiBlueLabel.place(x=FirstFrameX+1
18,y=FirstFrameY+258)

iiGreenLabel = tk.Label(GUI1, text =
"255", font = subFont3,
bg='#22B14C', fg='black', width=9)
iiGreenLabel.place(x=FirstFrameX+
118,y=FirstFrameY+284)

iiRedLabel = tk.Label(GUI1, text =
"255", font = subFont3,
bg='#ED1C24', fg='black', width=9)
iiRedLabel.place(x=FirstFrameX+11
8,y=FirstFrameY+310)

OutputFrame = tk.Label(GUI1, text
= "", font = subFont1, bg='white',
fg='#00a2e8')
OutputFrame.place(x=FirstFrameX+
300,y=FirstFrameY+190)
IF.tkShow(OutputFrame,
"Output.png", 1)

ooBlueLabel = tk.Label(GUI1, text =
"255", font = subFont3,
bg='#3F48CC', fg='black', width=9)
ooBlueLabel.place(x=FirstFrameX+
388,y=FirstFrameY+258)

ooGreenLabel = tk.Label(GUI1, text
= "255", font = subFont3,
bg='#22B14C', fg='black', width=9)
ooGreenLabel.place(x=FirstFrameX
+388,y=FirstFrameY+284)

ooRedLabel = tk.Label(GUI1, text =
"255", font = subFont3,
bg='#ED1C24', fg='black', width=9)
ooRedLabel.place(x=FirstFrameX+
388,y=FirstFrameY+310)

scale1 = tk.Scale(GUI1, label="",
orient=tk.VERTICAL, length=200,
width=20, sliderlength=10,
from_=255, to=0, tickinterval=255)
scale1.place(x=50, y=25)
scale1.set(51)

CaptureButton = tk.Button(GUI1,
text = "Capture", font = subFont2,
bg='#242424', fg='white',command
= CaptureButFunction)
CaptureButton.place(x=45,y=235)

ButtonsXStart = 70
ButtonsY = 400

Button1 = tk.Button(GUI1, text =
"Check Paint Levels", font =
subFont2, bg='#242424',
fg='white',command =
CheckPaintLevels)
Button1.place(x=ButtonsXStart+20,
y=ButtonsY)

Button2 = tk.Button(GUI1, text =
"Mix!", font = subFont2,
bg='#242424', fg='white',command
```

```
= Mix)
Button2.place(x=ButtonsXStart+200
,y=ButtonsY)

Button3 = tk.Button(GUI1, text =
"Mix Past Samples", font =
subFont2, bg='#242424',
fg='white',command = MixPast)
Button3.place(x=ButtonsXStart+270
,y=ButtonsY)

Button4 = tk.Button(GUI1, text =
"Clean Tubes", font = subFont2,
bg='#242424', fg='white',command
= Cleaning)
Button4.place(x=ButtonsXStart+440
,y=ButtonsY)

Button5 = tk.Button(GUI1, text =
"Close", font = subFont2,
bg='#242424', fg='white',command
= CloseProgram)
Button5.place(x=ButtonsXStart+570
,y=ButtonsY)

AccuracyLabel = tk.Label(GUI1, text
= "Accuracy: ", font = subFont2,
bg='#242424', fg='white', width=15)
AccuracyLabel.place(x=322,y=375)

SecondFrameX = 125
SecondFrameY = 437
RYBWBFrame = tk.Label(GUI1, text
= "", font = subFont2, bg='#242424',
fg='white')
RYBWBFrame.place(x=SecondFra
meX,y=SecondFrameY)
IF.tkShow(RYBWBFrame,
"RYBWB.png", 1)

rrRedLabel = tk.Label(GUI1, text =
"100.00%", font = subFont3,
bg='#ED1C24', fg='black', width=7)
rrRedLabel.place(x=SecondFrameX
+38,y=SecondFrameY+6)

rrYellowLabel = tk.Label(GUI1, text
= "100.00%", font = subFont3,
bg='#FFF200', fg='black', width=7)
rrYellowLabel.place(x=SecondFram
eX+154,y=SecondFrameY+6)

rrBlueLabel = tk.Label(GUI1, text =
"100.00%", font = subFont3,
bg='#3F48CC', fg='black', width=7)
rrBlueLabel.place(x=SecondFrame
X+260,y=SecondFrameY+6)

rrWhiteLabel = tk.Label(GUI1, text =
"100.00%", font = subFont3,
bg='#FFFFFF', fg='black', width=7)
rrWhiteLabel.place(x=SecondFrame
X+375,y=SecondFrameY+6)

rrBlackLabel = tk.Label(GUI1, text =
"100.00%", font = subFont3,
bg='#7F7F7F', fg='black', width=7)
rrBlackLabel.place(x=SecondFrame
X+480,y=SecondFrameY+6)

LoadingFrame = tk.Label(GUI1,
bg="white")

##-----------------------------------------
-----------------------------------------
-------
GUI2 = tk.Frame(root)
#GUI2.pack()

Background = tk.Label(GUI2,
text='',font = subFont1,
bg='white',bd=0)
Background.pack()
IF.tkShow(Background, "bg.png", 1)

LoadingFrame2 = tk.Label(GUI2,
bg="white")
```

```
BackButton = tk.Button(GUI2,
text="BACK", font=subFont2,
bg='white', command = BackF)
BackButton.place(x=30, y=440)

DeleteButton = tk.Button(GUI2,
text="DELETE", font=subFont2,
bg='white', command=Delete)
DeleteButton.place(x=710, y=440)

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

Arduino Code

```
const int MS1 = 52;
const int MS2 = 50;

const int R1 = 22;
const int R2 = 24;
const int R3 = 26;
const int R4 = 28;
const int R5 = 30;
const int R6 = 32;
const int R7 = 34;
const int R8 = 36;
String string =
"50.0025.0025.0000.0000.00";

#include <NewPing.h>2 // don't forget
to include the NewPing library.
NewPing Sonar1(33, 23, 400);
NewPing Sonar2(33, 25, 400);
NewPing Sonar3(33, 27, 400);
NewPing Sonar4(33, 29, 400);
NewPing Sonar5(33, 31, 400);

float duration1;
float d1;
float duration2;
float d2;
float duration3;
float d3;
float duration4;
float d4;
float duration5;
float d5;

float P1;
float P2;
float P3;
float P4;
float P5;

float height1 = 17;
float height2 = 17;
float height3 = 17;
float height4 = 17;
float height5 = 17;

float SRead = 2.5;

float Speed1 = 7; //mL per second
float Speed2 = 7; //mL per second
float Speed3 = 7; //mL per second
float Speed4 = 7; //mL per second
float Speed5 = 7; //mL per second
//
//float Red;
//float Blue;
//float Yellow;
//float Black;
//float White;

int toDispense = 0;
String receivedChar;
int CleaningTime = 30;

float Red; char RedByte[10];
float Blue; char BlueByte[10];
float Yellow; char YellowByte[10];
float Black; char BlackByte[10];
float White; char WhiteByte[10];

char byteBuffer;
int index;
char *n;

void setup() {
   Serial.begin(9600);
   pinMode(MS1, INPUT);
```

```
    pinMode(MS2, INPUT);
    digitalWrite(MS1, HIGH);
    digitalWrite(MS2, HIGH);

    pinMode(R1, OUTPUT);
digitalWrite(R1, HIGH);
    pinMode(R2, OUTPUT);
digitalWrite(R2, HIGH);
    pinMode(R3, OUTPUT);
digitalWrite(R3, HIGH);
    pinMode(R4, OUTPUT);
digitalWrite(R4, HIGH);
    pinMode(R5, OUTPUT);
digitalWrite(R5, HIGH);
    pinMode(R6, OUTPUT);
digitalWrite(R6, HIGH);
    pinMode(R7, OUTPUT);
digitalWrite(R7, HIGH);
    pinMode(R8, OUTPUT);
digitalWrite(R8, HIGH);
//    CompoDown();
    CompoUp();
    receivedChar = Serial.readString();
    while (receivedChar != "1")
{receivedChar=Serial.readString();
delay(100);}
    Serial.print("0");

    d1 = GetSonar1(); d2 =
GetSonar2(); d3 = GetSonar3(); d4 =
GetSonar4(); d5 = GetSonar5();

//    if (d1 > 10 || d2 > 10 || d3 > 10 ||
d4 > 10 || d5 > 10) {
//      Serial.print("0");
//      //while (1) {delay(1000);}
//      }
//    else {Serial.print("1");}
}

void loop() {
  receivedChar = Serial.readString();
       if (receivedChar == "A") {
        CompoDown();
        digitalWrite(R8, LOW);
        GetDelays();
        toDispense = 1;
        Valve1(); Valve2();Valve3();
Valve4(); Valve5();
        digitalWrite(R8, HIGH);
        CompoUp();
        Serial.print("A");
        delay(50);
        }
  else if (receivedChar == "B")
{GetPercentages(); delay(50);}
  else if (receivedChar == "C") {
        CompoDown();
        digitalWrite(R8, LOW);
        GetDelays();
        toDispense = 1;
//        Red = CleaningTime; Blue =
CleaningTime; Yellow =
CleaningTime; Black = CleaningTime;
White = CleaningTime;
//        Valve1(); Valve2();Valve3();
Valve4(); Valve5(); delay(50);
        digitalWrite(R1, LOW);
digitalWrite(R2, LOW); digitalWrite(R3,
LOW);
        digitalWrite(R4, LOW);
digitalWrite(R5, LOW);

        for (int i = 0; i < CleaningTime;
i++) {delay(1000);}
        digitalWrite(R1, HIGH);
digitalWrite(R2, HIGH);
digitalWrite(R3, HIGH);
        digitalWrite(R4, HIGH);
digitalWrite(R5, HIGH);

        digitalWrite(R8, HIGH);
        CompoUp();
        Serial.print("A");
        delay(50);
    }
  else if (receivedChar == "D") {
      Red = 1; Blue = 1; Yellow = 1;
Black = 1; White = 1;
      Valve1(); Valve2();Valve3();
Valve4(); Valve5();
  }
//  Serial.print(digitalRead(MS1));
//  Serial.println(digitalRead(MS2));
```

```
//  if (toDispense == 1) {Valve1();
Valve2();Valve3(); Valve4(); Valve5();
toDispense = 0;}
  //GetPercentages(); delay(1000);
  else if (receivedChar == "E")
{CompoUp(); delay(50);}
  else if (receivedChar == "F")
{CompoDown(); delay(50);}
  else if (receivedChar == "G")
{digitalWrite(R8, !digitalRead(R8));
delay(50);}
  else if (receivedChar == "H") {
    CompoDown();
    digitalWrite(R8, LOW);
    for (int i = 0; i < 30; i++) {
      delay(1000);
     }
    digitalWrite(R8, HIGH);delay(50);
    CompoUp();
    Serial.print("A");
    }

//
//  duration1 = Sonar1.ping(); d1 =
duration1 / US_ROUNDTRIP_CM;
delay(30);
//  duration2 = Sonar2.ping(); d2 =
duration2 / US_ROUNDTRIP_CM;
delay(30);
//  duration3 = Sonar3.ping(); d3 =
duration3 / US_ROUNDTRIP_CM;
delay(30);
//  duration4 = Sonar4.ping(); d4 =
duration4 / US_ROUNDTRIP_CM;
delay(30);
//  duration5 = Sonar5.ping(); d5 =
duration5 / US_ROUNDTRIP_CM;
delay(30);
//  Serial.print(digitalRead(MS1));
Serial.print(" ");
//  Serial.print(digitalRead(MS2));
Serial.print(" ");
//  Serial.print(d1); Serial.print(" ");
//  Serial.print(d2); Serial.print(" ");
//  Serial.print(d3); Serial.print(" ");
//  Serial.print(d4); Serial.print(" ");
//  Serial.print(d5); Serial.println(" ");
}

void RelayGo(int RelayName, float
Seconds) {
  digitalWrite(RelayName, LOW);
  int a = 0;
  for (int i = 0; i < Seconds; i++) {
    delay(1000);
    a++;
  }
  digitalWrite(RelayName, HIGH);
  }

void Valve1() {
//  digitalWrite(R1, LOW);
//  delay(Red*1000);
//  digitalWrite(R1, HIGH);
//  delay(1000);
  RelayGo(R4, Red);
  delay(1000);
  }

void Valve2() {
//  digitalWrite(R2, LOW);
//  delay(Blue*1000);
//  digitalWrite(R2, HIGH);
//  delay(1000);
  RelayGo(R2, Blue);
  delay(1000);
  }

void Valve3() {
//  digitalWrite(R3, LOW);
//  delay(Yellow*1000);
//  digitalWrite(R3, HIGH);
//  delay(1000);
  RelayGo(R3, Yellow);
  delay(1000);
  }

void Valve4() {
//  digitalWrite(R4, LOW);
//  delay(Black*1000);
//  digitalWrite(R4, HIGH);
//  delay(1000);
  RelayGo(R1, Black);
  delay(1000);
  }
```

```
void Valve5() {
//  digitalWrite(R5, LOW);
//  delay(White*1000);
//  digitalWrite(R5, HIGH);
//  delay(1000);
  RelayGo(R5, White);
  delay(1000);
  }

void CompoDown() {
  if (digitalRead(MS2) == 1) {
    digitalWrite(R7, LOW);
    delay(50);
    while (1) {if (digitalRead(MS2) ==
0){break;}}
    digitalWrite(R7, HIGH);
    }
  }

void CompoUp() {
  if (digitalRead(MS1) == 1) {
    digitalWrite(R6, LOW);
    delay(50);
    while (1) {if (digitalRead(MS1) ==
0){break;}}
    digitalWrite(R6, HIGH);
    }
  }

float GetSonar1() {
  duration1 = Sonar1.ping(); d1 =
duration1 / US_ROUNDTRIP_CM;
delay(30);
  return d1;
  }

float GetSonar2() {
  duration2 = Sonar2.ping(); d2 =
duration2 / US_ROUNDTRIP_CM;
delay(30);
  return d2;
  }

float GetSonar3() {
  duration3 = Sonar3.ping(); d3 =
duration3 / US_ROUNDTRIP_CM;
delay(30);
  return d3;
  }

float GetSonar4() {
  duration4 = Sonar4.ping(); d4 =
duration4 / US_ROUNDTRIP_CM;
delay(30);
  return d4;
  }

float GetSonar5() {
  duration5 = Sonar5.ping(); d5 =
duration5 / US_ROUNDTRIP_CM;
delay(30);
  return d5;
  }

void GetPercentages() {

  d1 = GetSonar1(); d2 = GetSonar2();
d3 = GetSonar3(); d4 = GetSonar4();
d5 = GetSonar5();
  P1 = -(100/(height1 - SRead))*(d1 -
height1);
  P2 = -(100/(height2 - SRead))*(d2 -
height2);
  P3 = -(100/(height3 - SRead))*(d3 -
height3);
  P4 = -(100/(height4 - SRead))*(d4 -
height4);
  P5 = -(100/(height5 - SRead))*(d5 -
height5);
  Serial.print("AAA"); Serial.print(P1);
Serial.print("BBB"); Serial.print(P2);
Serial.print("CCC"); Serial.print(P3);
Serial.print("DDD"); Serial.print(P4);
Serial.print("EEE"); Serial.print(P5);
Serial.print("FFF");
  }

void GetDelays1() {
  delay(250);
  while (!Serial.available()) {}
  string = Serial.readString();
  Red = string.substring(0,5).toFloat();
  Blue =
string.substring(5,10).toFloat();
  Yellow =
string.substring(10,15).toFloat();
```

```
  Black =
string.substring(15,20).toFloat();
  White =
string.substring(20,25).toFloat();

  Red = ((float(Red)/100)*1000);
  Blue = ((float(Blue)/100)*1000);
  Yellow = ((float(Yellow)/100)*1000);
  Black = ((float(Black)/100)*1000);
  White = ((float(White)/100)*1000);
//
//  Serial.println(Red);
//  Serial.println(Blue);
//  Serial.println(Yellow);
//  Serial.println(Black);
//  Serial.println(White);

  Red = Red/Speed1;
  Blue = Blue/Speed2;
  Yellow = Yellow/Speed3;
  Black = Black/Speed4;
  White = White/Speed5;
//
//  Serial.println(Red);
//  Serial.println(Blue);
//  Serial.println(Yellow);
//  Serial.println(Black);
//  Serial.println(White);
  }

void GetDelays()
  {
    Serial.flush();
    byteBuffer = 0;

    while(!Serial.available());
    char string[256];
    index = 0;
    do{
      if(Serial.available() > 0){
        byteBuffer = Serial.read();
        string[index] = byteBuffer;
        index++;
      }
    }while(byteBuffer != 'Z');

    string[index] = '\0';
    n = strtok(string,",");
    n =
strtok(NULL,",");strcpy(RedByte,n);
    n =
strtok(NULL,",");strcpy(BlueByte,n);
    n =
strtok(NULL,",");strcpy(YellowByte,n);
    n =
strtok(NULL,",");strcpy(BlackByte,n);
    n =
strtok(NULL,",");strcpy(WhiteByte,n);

    Red = atof(RedByte);
    Blue = atof(BlueByte);
    Yellow = atof(YellowByte);
    Black = atof(BlackByte);
    White = atof(WhiteByte);

    Red = ((float(Red)/100)*1000);
    Blue = ((float(Blue)/100)*1000);
    Yellow = ((float(Yellow)/100)*1000);
    Black = ((float(Black)/100)*1000);
    White = ((float(White)/100)*1000);

    Red = Red/Speed1;
    Blue = Blue/Speed2;
    Yellow = Yellow/Speed3;
    Black = Black/Speed4;
    White = White/Speed5;
  }
```

Appendix 7

PROTOTYPE DESIGN

CAD Design

Front

Back

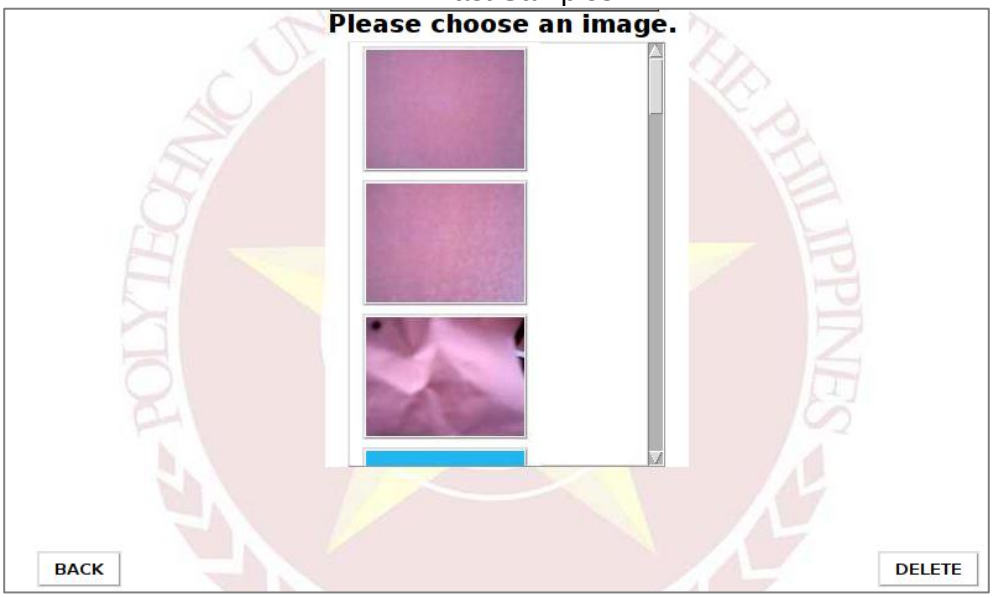
Appendix 8

GRAPHICAL USER INTERFACE (GUI) DESIGN

Main Menu

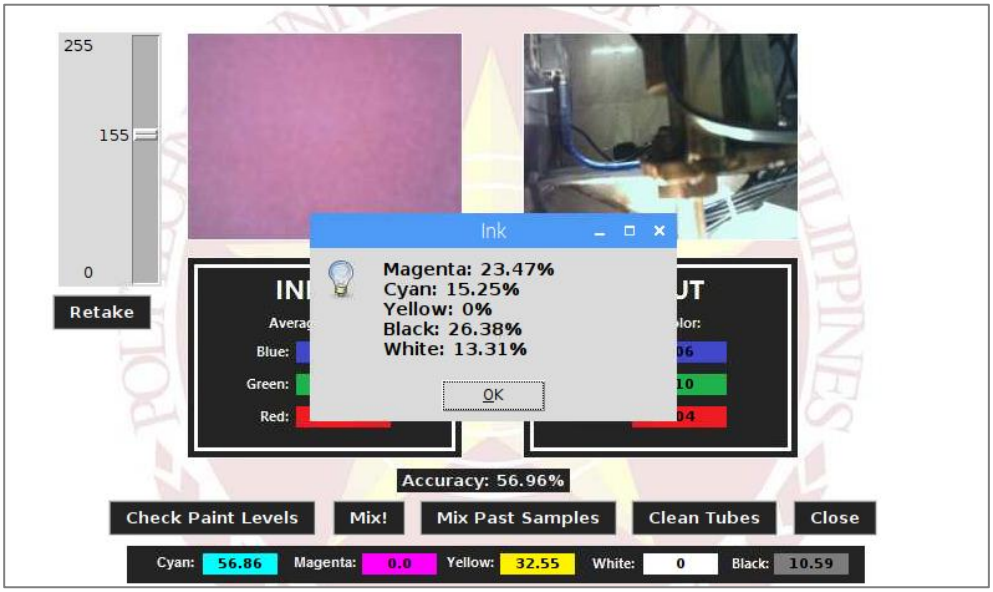
Mix!

Mix Past Samples

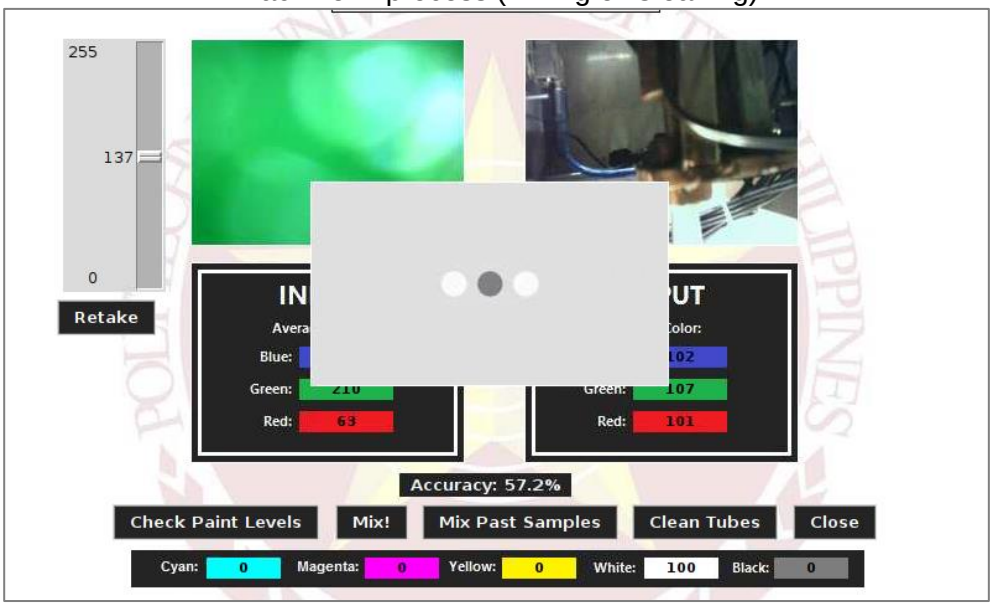


Clean Tubes

Check Paint Levels



Machine in process (Mixing or Cleaning)



Appendix 9

FACE-TO-FACE INTERVIEW

Respondent No. 1

Mr. Wilmar Olideles (Employee & Paint Mixer Personnel, Finemix Paint Center)

1. **Ano ang basehan niyo sa pagpili ng kulay ng pinturana hahaluin?**

   Mga basic colors lang ang ginagamit naming sa pag-mimix ng pintura. Kunyari pink, ang ginagamit na pintura doon ay white at ted. Minsan naman, mga nasa tatlong pintura yung kailangan para makuha yung gusting kulay ng customer.

2. **Gaano katagal ang paghahalo ng pintura (para sa isang litro)?**

   Sa isang litro minsan inaabot ng mga 30 mins. Yun na yung pinakamatagal. Depende din kasi minsan sa kung anong kulay ang kailangan. Kaya minsan medyo mabilis naman sa 30 mins.

3. **Gaana karami ang natatanggap na order ng customized paint sa loob ng isang araw?**

   Siguro sa isang araw, may apat na customer kami dito tapos ang kadalasan nilang inoorder ay nasa 4 gallons. Madalas kasi mga pang-dingding yung paggagamitan kaya maramihan sila magpahalo.

4. **Ano ang mga karanasan niyong problema pagdating sa pintura?**

   Hindi maiiwasan na malagyan kami ng pintura sa katawan lalo na sa kamay. Kapag hindi pa water-based ung pintura, mahirap tanggalin. Kailangan pa ng thinner at parang masama pa sa balat yun.

5. **Kaya niyo po bang maghalo ng kulay ng pintura na gamit lamang ang kulay na *red, blue, yellow, white,* at *black*?**

Oo naman. halos yan din yung mga ginagamit namin kapag mga basic colors lang yung ipapahalo. Kaso kapag halimbawa peach ung ipapahalo, imbis na ung available na orange na pintura at tsaka white yung gagamitin, gagawa pa kami ng orange gamit ung limang kulay na sinasabi mo.

6. **Ano po ang masasabi mo sa *Automated Paint Mixer with Color Scanner?***

   Gaya ng tanong mo kanina kung posibleng makapag-mix gamit ang limang kulay lang, ayun pwedeng-pwede na yun kung yung sinasabi mong mixer ang gagamitin kasi kung medyo kumplikado yung kulay na gusto ng customer, hindi na kami mahihirapan pang mag mix ng ibang kulay para makapag-mix ng kulay na gusto nya. Mas mapapadali pa at mas kokonti yung mga magpapa-adjust.

Respondent No. 2

Mr. Bobby Ebuanga (Employee & Paint Mixer Personnel, Maky Ian Paint Center)

1. **Ano ang basehan niyo sa pagpili ng kulay ng pinturana hahaluin?**

   Kung ano yung pinakamalapit na kulay doon sa sample na dala ng customer, yun yung gagamitin ko tapos hahaluan ko nalang ng ibang kulay para ma-achieve yung kulay na gusto nila.

2. **Gaano katagal ang paghahalo ng pintura (para sa isang litro)?**

Mga 20 mins maghalo ng pintura. Depende yun sa kulay. Pwedeng mas tumagal o pwedeng mas bumilis. Kapag maramihan yung order kunyari isang galon, hindi na ako nagtetest ng kaunti, diretso na sa pagmimix kasi ganun din naman, mag tatantya din naman ako kapag mag-mimix na ng isang galon.

3. **Gaana karami ang natatanggap na order ng customized paint sa loob ng isang araw?**

   Araw-araw may nagpapahalo dito sa paint center namin, walang araw na walang customer. Mayroong mga bumibili ng tig-iisang litro lang, madalas naman magpapahalo sila ng mga isang galon.

4. **Ano ang mga karanasan niyong problema pagdating sa pintura?**

   Wala naman akong problema sa pintura bukod sa amoy nung mga hindi odorless. Sa paghahalo lang ng pintura, kapag medyo hindi ako pamliyar sa kulay, matagal ang tantyahan ng paglagay ng pintura.

5. **Kaya niyo po bang maghalo ng kulay ng pintura na gamit lamang ang kulay na *red, blue, yellow, white,* at *black*?**

   Sa tingin ko kaya naman, kasi marami ng magagawang ibang kulay yang mga kulay na yan.

6. **Ano po ang masasabi mo sa *Automated Paint Mixer with Color Scanner?***

   Magandang bagay yan kasi kung computerized na yan, wala ng tantyahan ang magaganap. Diretso buhos nalang yung machine don sa lalagayan. Ii-scan nalang kasi yung sample dyaan diba? Pag walang scanner, kami pa mag-iisip ng kulay na gagamitin tapos tantyahan pa, kaya pag may mixer na.

Respondent No. 3

Mr. Jeffrey Lastimosa (Employee & Paint Mixer Personnel, JSL Paint Center)

1. **Ano ang basehan niyo sa pagpili ng kulay ng pinturana hahaluin?**

   Sa pagpili ng pintura, naka-base ako doon sa kung anong gusting kulay nung customer tapos titignan ko kung anong pinakamalapit na kulay doon. Kunyari apple green, ang kulay na gagamitin ko kulay green tapos yellow tapos tsaka ko hahaluan ng kulay white depende kung gaano ka-light yung gusto ng customer.

2. **Gaano katagal ang paghahalo ng pintura (para sa isang litro)?**

   Inaabot ako ng halos kalahating oras sa pag-mimix ng isang litro. Kasi nagta-try muna ako ng kaunti bago ako mag-mix ng isang litro.

3. **Gaana karami ang natatanggap na order ng customized paint sa loob ng isang araw?**

   Depende po yan. May araw na mabenta, may araw din na matumal. Pag may nagpapahalo naman dito, puro galon ang mga pinapahalo. Bihira yung isang litro lang na nagpapa-mix.

4. **Ano ang mga karanasan niyong problema pagdating sa pintura?**

   Makalat kapag nagmimix ng pintura kasi may tumatalsik at may tumutulong pintura. Ang dami ding nasasayang na pintura lalo na kapag rejected ng

customer yung na-mix namin, kapag hindi na kayang ma-adjust yun, masasayang lang yung pintura, minsan hindi na maibebenta.

5. **Kaya niyo po bang maghalo ng kulay ng pintura na gamit lamang ang kulay na *red, blue, yellow, white,* at *black*?**

   Doon din naman nagmula ang mga kulay na meron tayo ngayon, kaya posible naman. Hindi lang ako sigurado kung mapapabilis ang trabaho.

6. **Ano po ang masasabi mo sa *Automated Paint Mixer with Color Scanner?***

   Kapag computerized kasi sa tingin ko mas mapapabilis ang gawain. Hindi pa makalat. Pero kung gagamitin ko yan, siguro mag-manual nalang ako pero matagal nga lang. Hindi kasi ako sanay sa computer, pero kung matututunan ko yun, baka gamitin ko yung

Respondent No. 4

Ms. May Gerogalen (Cashier & Paint Mixer Personnel, Kolok Saver Paint Center)

1. **Ano ang basehan niyo sa pagpili ng kulay ng pintura na hahaluin?**

   Meron kaming mga sinusunod na guide na paghahaluin hanggang masasanay ka na at alam mo na kung ano dapat ang paghaluin.

2. **Gaano katagal ang paghahalo ng pintura (para sa isang litro)?**

Depende sa kulay pero madalas mga 15-30min.

3. **Gaano karami ang natatanggap na order ng *customized paint* sa loob ng isang araw?**

   Depende, pinakamarami na siguro yung mga 10 orders.

4. **Ano ang mga karanasan niyong problema pagdating sa pintura?**

   Makalat talaga siya pag maghahalo ng pintura, yung mga ibang kulay din ng pintura na hindi nabibili namumuo pero unting halo lang ok na din naman siya gamitin

5. **Kaya niyo po bang maghalo ng kulay ng pintura na gamit lamang ang kulay na *red, blue, yellow, white,* at *black*?**

   Kaya siguro kung yung mga simpleng kulay lang pero pag kakaiba na yung kulay mahirap na yun.

6. **Ano po ang masasabi mo sa *Automated Paint Mixer with Color Scanner?***

   Ok siya kasi mapapabilis niya yung paghahalo ng pintura.

Respondent No. 5

Mr. Danilo C. Geronimo (Paint Mixer Personnel, RockDJ Colors Ent.)

1. **Ano ang basehan niyo sa pagpili ng kulay ng pintura na hahaluin?**

   Depende sa kulay na ipapahalo ng customer

2. **Gaano katagal ang paghahalo ng pintura (para sa isang litro)?**

   Mga isang litro umaabot ng 1hr pag 4L mga 2hrs**.**

3. **Gaano karami ang natatanggap na order ng *customized paint* sa loob ng isang araw?**

   Mga lima or apat ganun minsan din naman madami.

4. **Ano ang mga karanasan niyong problema pagdating sa pintura?**

   Minsan yung ibang pintura maamoy kaya titiisin mo na lang

5. **Kaya niyo po bang maghalo ng kulay ng pintura na gamit lamang ang kulay na *red, blue, yellow, white,* at *black*?**

   Hindi siguro kasi mahirap na yun.

6. **Ano po ang masasabi mo sa *Automated Paint Mixer with Color Scanner?***

   Nakakamangha na pwede palang makagawa ng ibat ibang kulay gamit lang ang limang pintura.

Respondent No. 6

Mr. Mar (Employee & Paint Mixer Personnel, GMR Paint Center)

1. **Ano ang basehan niyo sa pagpili ng kulay ng pintura na hahaluin?**

   Depende sa kulay pero madalas gamitin namin ay primary colors at puti.

2. **Gaano katagal ang paghahalo ng pintura (para sa isang litro)?**

   Depende pa din sa kulay. Pag mga simpleng kulay lang siguro mga 5 mins lang pero pag mahirap hirap aabot ng mga 10 mins ang paghahalo.

3. **Gaano karami ang natatanggap na order ng customized paint sa loob ng isang araw?**

   Umaabot siguro ng mga 5 customers isang araw pag medyo malakas hanggang 10 customer.

4. **Ano ang mga karanasan niyong problema pagdating sa pintura?**

   Syempre di natin maiiwasan matalsikan at mapahiran ng pintura medyo mahirap lang tanggalin pagtapos syempre depende sa pintura kung water based ba sya or oil based. Pag dating naman sa output ng pintura wala naman

masyado dahil nareremdeyuhan naman kung di pa kakulay yung pinaggagayahan.

5. **Kaya niyo po bang maghalo ng kulay ng pintura na gamit lamang ang kulay na *red, blue, yellow, white, at black?***

   Pwede naman kasi dun din naman nagmula yung ibang kulay gaya nung green, orange, violet kaso nga lang baka mas matagalan ang paghahalo namin kasi maghahalo pa kami ng pintura para magkaroon nung mga secondary colors.

6. **Ano po ang masasabi mo sa *Automated Paint Mixer with Color Scanner*?**

   Maganda, kasi halos lahat ngayon ay nakakompyuter na high- tech kumbaga. Mas bibilis yung paghahalo ng pintura kasi kami nangangalay din kami pero pag automatic na derederetso na, At saka mukhang mas magagaya nya yung kulay ng kokopyhin ng walang kahirap hirap.

Respondent No. 7

Mr. Edwin (Employee & Paint Mixer Personnel, WVA Paint Center)

1. **Ano ang basehan niyo sa pagpili ng kulay ng pintura na hahaluin?**

   Depende sa gagayahing kulay. Pwedeng red, yellow, green, orange, blue, depende talaga sa gagayahing kulay.

2. **Gaano katagal ang paghahalo ng pintura (para sa isang litro)?**

   Mga 10 minutes sagad na yon. Pero pag sobrang hirap aabot siguro ng 15 mins,

3. **Gaano karami ang natatanggap na order ng customized paint sa loob ng isang araw?**

   Mga lima hanggang walong costumer ang nagpapalo samin kada araw

4. **Ano ang mga karanasan niyong problema pagdating sa pintura?**

Wala naman. Syempre nung bago palang ako mahirap kasi di ko alam kung anong pintura ang mga dapat kong idagdag pero ngayon okay na. Medyo makalat lang kasi may tumatalsik.

5. **Kaya niyo po bang maghalo ng kulay ng pintura na gamit lamang ang kulay na red, blue, yellow, white, at black?**

   Kaya naman kaso baka matagalan kasi kailangan ko pa maghalo para gumawa nung ibang kulay gaya nung green.

6. **Ano po ang masasabi mo sa Automated Paint Mixer with Color Scanner?**

   Parang mahirap naman ata iyon pero kung magkaroon man ay mas dadali ang trabaho at mas mabilis. Tsaka mukhang mas malinis ang proseso ng paghahalo.

Respondent No. 8

Mr. Ton (Employee & Paint Mixer Personnel, Angeles Paint Center)

1. **Ano ang basehan niyo sa pagpili ng kulay ng pintura na hahaluin?**

   Depende sa kulay na paggayahan. Pero madalas ay pula, dilaw, asul at puti.

2. **Gaano katagal ang paghahalo ng pintura (para sa isang litro)?**

   Pinakamababa ay 3 mins hanggang 20 mins na yung pinakamatagal.

3. **Gaano karami ang natatanggap na order ng customized paint sa loob ng isang araw?**

   Siguro mga 20 customers kada araw

4. **Ano ang mga karanasan niyong problema pagdating sa pintura?**

   Minsan pag medyo nagkamali yung hinahalo nireretoke pa kaya tumatagal lalo. Saka syempre makalat, madaming natatapon at tumatalsik napintura

5. **Kaya niyo po bang maghalo ng kulay ng pintura na gamit lamang ang kulay na red, blue, yellow, white, at black?**

Oo naman kasi basic colors yan eh.

6. **Ano po ang masasabi mo sa Automated Paint Mixer with Color Scanner?**

   Edi ayos! Mas mabilis maghalo nyan at wala ng kahirap hirap nyan kasi di na ko mangangalay kakahalo at mas dadami benta namin pag nagkaroon kami nyan at natuto ako kung pano gamitin yan nang tama.

Respondent No. 9

Mr. Domie (Employee & Paint Mixer Personnel, Galary Paint Center)

1. **Ano ang basehan niyo sa pagpili ng kulay ng pintura na hahaluin?**

   Iba-iba, depende sa gusto ng customer. Syempre kung ano ang pinaka malapit doon sa kulay, yun ang gagamitin naming. Hallimbawa, lavender, edi gagamit kami ng violet at white.

2. **Gaano katagal ang paghahalo ng pintura (para sa isang litro)?**

   Depende sa kulay eh, minsan kasi merong madaming kulay ang kailangan para ihalo. Syempre papatuyuin pa para malaman kung yun ba yung tamang shade. Yung pinakamadali siguro ay mga 5 mins, tapos siguro pinakamatagal na ang 25 mins.

3. **Gaano karami ang natatanggap na order ng customized paint sa loob ng isang araw?**

   Iba iba din eh, minsan matumal mga lima lang. Minsan naman 20 customers.

4. **Ano ang mga karanasan niyong problema pagdating sa pintura?**

   Kapag nasosobrahan sa dark yung naihalong pintura, eh syempre kailangan 4L lang yung hahaluin kung yun yung order ng customer, maitatapon ung iba kasi babawasan para mahabol ung light.

5. **Kaya niyo po bang maghalo ng kulay ng pintura na gamit lamang ang kulay na red, blue, yellow, white, at black?**

   Siguro kaya naman. Pero sa amin kasi, kung may green naman, at light green and kailangan, yun na lang gagamitin naming.

6. **Ano po ang masasabi mo sa Automated Paint Mixer with Color Scanner?**

   Siguro ay mas mapapadali yung aming trabaho, pero baka mawalan kami ng trabaho. Pero kung kami naman ang mag ooperate ng machine ay ok naman. Syempre ang mga pintura, kemikal iyan. Toxic din sa mga balat namin, araw araw kaming napapahiran ng mga iba't ibang uri ng pintura kaya baka nakakasama din sa kalusugan namin, lalo na kapag nalalanghap namin yung amoy. Kaya okay siguro lalo na at di na namin kailangan hawakan ung pintura.

Respondent No. 10

Mr. Ronald (Employee & Paint Mixer Personnel, Paintshoppe Commercial & Paint Center)

1. **Ano ang basehan niyo sa pagpili ng kulay ng pintura na hahaluin?**

   Meron kaming ipinapakita na color chart kung saan doon namimili ang customer kung anong pintura ang gusto nila. Gamay naman na naming kung anong mga kulay ang ihahalo doon kaya madali na lang.

2. **Gaano katagal ang paghahalo ng pintura (para sa isang litro)?**

   Mga 10 mins to 20 mins siguro.

3. **Gaano karami ang natatanggap na order ng customized paint sa loob ng isang araw?**

   Kapag malakas ang benta, umaabot kami ng 15 customers tapos marami na yung order, minsan dalawang 4L.

4. **Ano ang mga karanasan niyong problema pagdating sa pintura?**

   Minsan magpapakita sila ng mga sample ng kulay na gusto nila kunwari kulay ng kurtina nila o ng upuan. Minsan may masyadong mahirap na haluin kasi madaming kulay ang kailangan, o kaya naman ay mahirap iadjust at isakto ung kulay doon.

5. **Kaya niyo po bang maghalo ng kulay ng pintura na gamit lamang ang kulay na red, blue, yellow, white, at black?**

   Oo kaya naman, kaso ay mas matatagalan siguro lalo na kung mga katulad ng mga iba pang basic na kulay katulad ng green o ng violet o kaya ng brown ung kailangan. Kasi pwede namang yun agad ang haluin eh, tapos dadagdagan na lang ng pang light o pang dark.

6. **Ano po ang masasabi mo sa Automated Paint Mixer with Color Scanner?**

   Okay naman, kaso mukhang mahirap gawin kapag may camera kasi baka hindi sakto yung kulay. Pero magiging maganda yan lalo na sa amin kasi hindi na kami mahihirapan mag adjust adjust ng kulay at mag isip ng mga ihahalong kulay para makuha yung gusto ng customer.

Appendix 10

DATASHEETS

sparkfun ELECTRONICS

Bartendro Dispenser - Peristaltic Pump and Controller

WIG-12915

Description: Do you need to precisely dispense amounts of liquid for your preferred beverage? If so this could be the pump for you. This is the Bartendro Dispenser from Party Robotics, an ingeniously designed peristaltic pump and controller that can be used to dispense liquids with milliliter accuracy and can be used in a stand alone system or with a router and other pumps for a more complex liquid dispensing system. This peristaltic pump works through positive displacement, two rollers are attached to a single motor that spins around a Norprene® tube pinching it closed to force your beverage liquid to be pumped through. With this method you won't need to worry about your liquid flowing through any moving parts, it will bypass the need for flow meters, and you will be able to meter the amounts of beverage liquid independent of viscosity and density.

Unlike most peristaltic pumps, the Bartendro Dispenser comes equipped with a controller board (driven by the reliable ATMega168), RJ45 and liquid level sensor connectors, and even a few WS2801 RGB LEDs just to make your drink provider all the more pretty.

The only things that the Bartendro really need to get going is a bit of motor power (minumum of 12V, max 24V), logic power, and serial communication all of which are supplied over the RJ45 jack found on the back of the board. Basically all you need to do is form a serial communication with this board via Arduino, Raspberry Pi, pcDuino, or any other development board of those natures and you will be good to start pouring!

Note: The Bartendro Dispenser is NOT intended for medical use.

Features:

- Max Flow rate: 700mL/ min
- Minimum Motor Voltage (Vin): 12V
- Motor Voltage (Vin): 24V
- Minimum Dispense Volume: 0.5mL if using PWM
- Repeatability: 1mL or better
- Current Consumption at 24V: up to 500mA while dispensing
- Max Vertical Tubing Length: about 10ft
- Max Length of Serial Connection: 10ft
- Tube Life: 500 Hours
- Food Safe

Includes:

- 1x Bartendro Dispenser
- 1x 5ft Clear Tygon S3™ Tubing
- 2x Compression Nut

https://www.sparkfun.com/products/12915 6-26-17

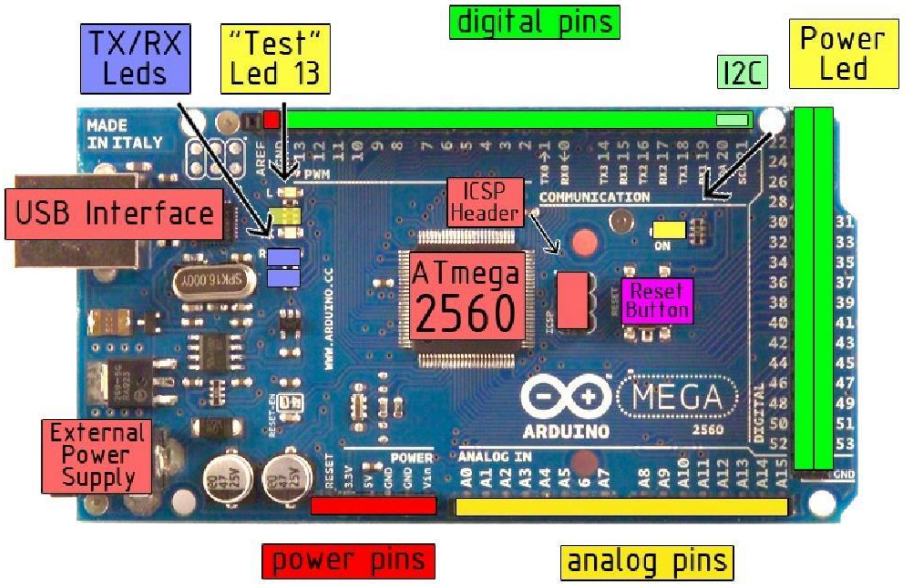
Technical Specification

EAGLE files: arduino-mega2560-reference-design.zip Schematic: arduino-mega2560-schematic.pdf

Summary

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

the board



Power

The Arduino Mega2560 can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The Mega2560 differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The ATmega2560 has 256 KB of flash memory for storing code (of which 8 KB is used for the bootloader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 54 digital pins on the Mega can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip .
- **External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2).** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- **PWM: 0 to 13.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS).** These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Duemilanove and Diecimila.
- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
- **I^2C: 20 (SDA) and 21 (SCL).** Support I^2C (TWI) communication using the Wire library (documentation on the Wiring website). Note that these pins are not in the same location as the I^2C pins on the Duemilanove.

The Mega2560 has 16 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and analogReference() function.

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with analogReference().
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Communication

The Arduino Mega2560 has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega2560 provides four hardware UARTs for TTL (5V) serial communication. An ATmega8U2 on the board channels one of these over USB and provides a virtual com port to software on the computer (Windows machines will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the ATmega8U2 chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows for serial communication on any of the Mega's digital pins.

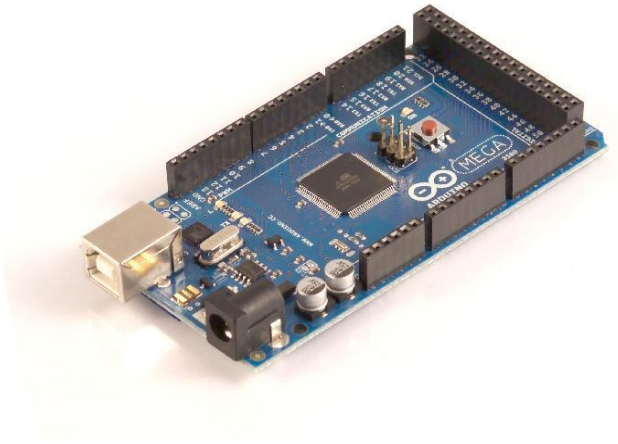
The ATmega2560 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation on the Wiring website for details. To use the SPI communication, please see the ATmega2560 datasheet.

Programming

The Arduino Mega2560 can be programmed with the Arduino software (download). For details, see the reference and tutorials.

The Atmega2560 on the Arduino Mega comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.



Automatic (Software) Reset

Rather then requiring a physical press of the reset button before an upload, the Arduino Mega2560 is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2 is connected to the reset line of the ATmega2560 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Mega2560 is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Mega2560. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

The Mega contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

USB Overcurrent Protection

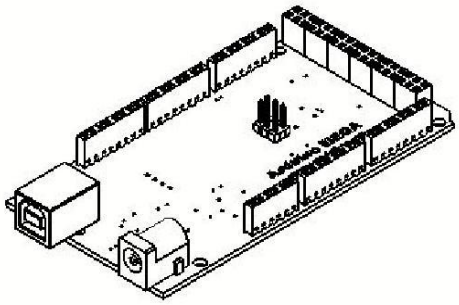
The Arduino Mega has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

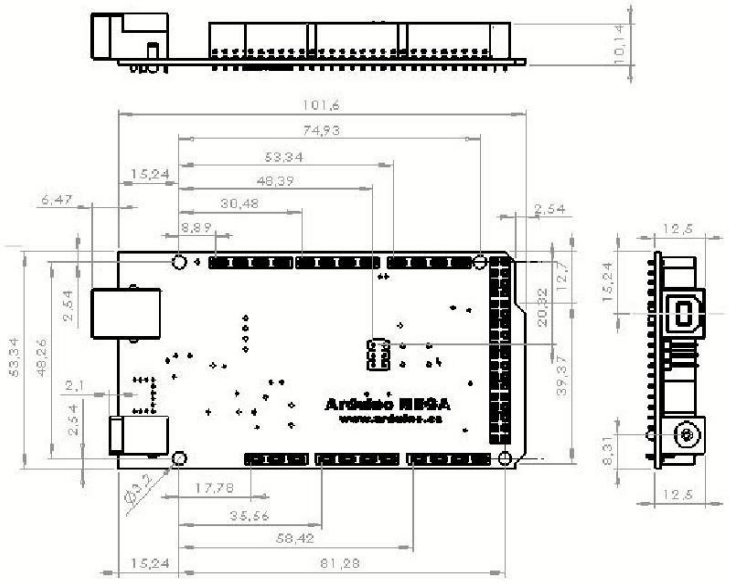
Physical Characteristics and Shield Compatibility

The maximum length and width of the Mega PCB are 4 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

The Mega is designed to be compatible with most shields designed for the Diecimila or Duemilanove. Digital pins 0 to 13 (and the adjacent AREF and GND pins), analog inputs 0 to 5, the power header, and ICSP header are all in equivalent locations. Further the main UART (serial port) is located on the same pins (0 and 1), as are external interrupts 0 and 1 (pins 2 and 3 respectively). SPI is available through the ICSP header on both the Mega and Duemilanove / Diecimila. **Please note that I^2C is not located on the same pins on the Mega (20 and 21) as the Duemilanove / Diecimila (analog inputs 4 and 5).**

Dimensioned Drawing



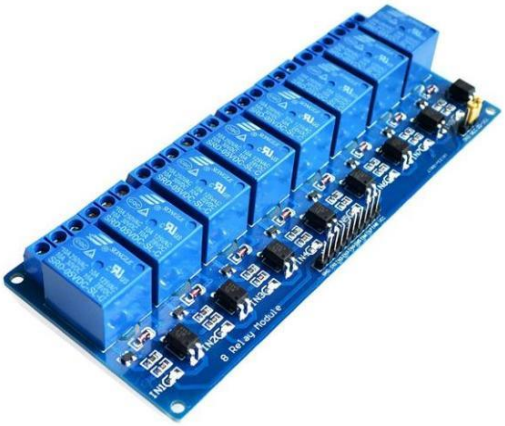


Handson Technology

User Guide

8 Channel 5V Optical Isolated Relay Module

This is a LOW Level 5V 8-channel relay interface board, and each channel needs a 15-20mA driver current. It can be used to control various appliances and equipment with large current. It is equipped with high-current relays that work under AC250V 10A or DC30V 10A. It has a standard interface that can be controlled directly by microcontroller. This module is optically isolated from high voltage side for safety requirement and also prevent ground loop when interface to microcontroller.



SKU: MDU1064

Brief Data:

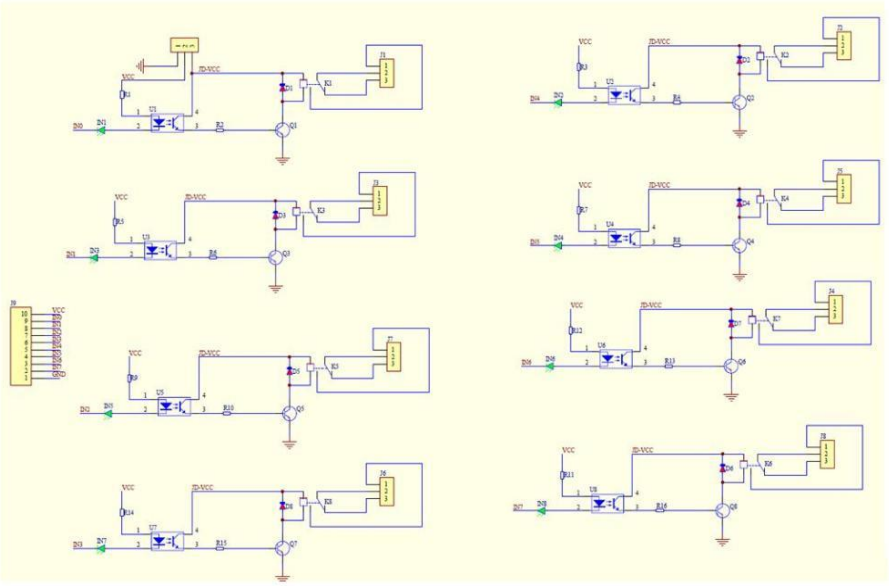
- Relay Maximum output: DC 30V/10A, AC 250V/10A.
- 8 Channel Relay Module with Opto-coupler. LOW Level Trigger expansion board, which is compatible with Arduino control board.
- Standard interface that can be controlled directly by microcontroller (8051, AVR, *PIC, DSP, ARM, ARM, MSP430, TTL logic).
- Relay of high quality low noise relays SPDT. A common terminal, a normally open, one normally closed terminal.
- Opto-Coupler isolation, for high voltage safety and prevent ground loop with microcontroller.
- Module Board: 138 x 56 mm.

Schematic:

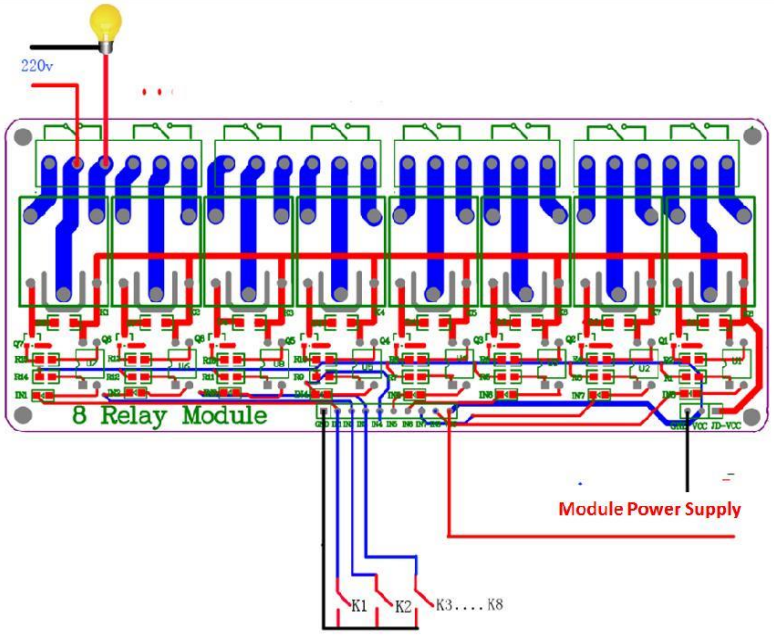
VCC and RY-VCC are also the power supply of the relay module. When you need to drive a large power load, you can take the jumper cap off and connect an extra power to RY-VCC to supply the relay; connect VCC to 5V of the MCU board to supply input signals.

NOTES: If you want complete optical isolation, connect "Vcc" to Arduino +5 volts but do NOT connect Arduino Ground. Remove the Vcc to JD-Vcc jumper. Connect a separate +5 supply to "JD-Vcc" and board Gnd. This will supply power to the transistor drivers and relay coils.

If relay isolation is enough for your application, connect Arduino +5 and Gnd, and leave Vcc to JD-Vcc jumper in place.



8 Channel Relay Module Schematic



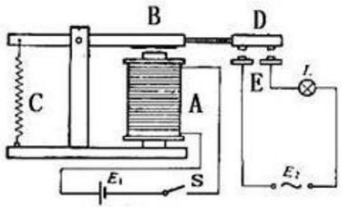
It is sometimes possible to use this relay boards with 3.3V signals, if the JD-VCC (Relay Power) is provided from a +5V supply and the VCC to JD-VCC jumper is removed. That 5V relay supply could be totally isolated from the 3.3V device, or have a common ground if opto-isolation is not needed. If used with isolated 3.3V signals, VCC (To the input of the opto-isolator, next to the IN pins) should be connected to the 3.3V device's +3.3V supply.

NOTE: Some Raspberry-Pi users have found that some relays are reliable and others do not actuate sometimes. It may be necessary to change the value of R1 from 1000 ohms to something like 220 ohms, or supply +5V to the VCC connection.

NOTE: The digital inputs from Arduino are Active LOW: The relay actuates and LED lights when the input pin is LOW, and turns off on HIGH.

Operating Principle:

See the picture below: A is an electromagnet, B armature, C spring, D moving contact, and E fixed contacts. There are two fixed contacts, a normally closed one and a normally open one. When the coil is not energized, the normally open contact is the one that is off, while the normally closed one is the other that is on.



Supply voltage to the coil and some currents will pass through the coil thus generating the electromagnetic effect. So the armature overcomes the tension of the spring and is attracted to the core, thus closing the moving contact of the armature and the normally open (NO) contact or you may say releasing the former and the normally closed (NC) contact. After the coil is de-energized, the electromagnetic force disappears and the armature moves back to the original position, releasing the moving contact and normally closed contact. The closing and releasing of the contacts results in power on and off of the circuit.

Input:

VCC : Connected to positive supply voltage (supply power according to relay voltage)

GND : Connected to supply ground.

IN1: Signal triggering terminal 1 of relay module

IN2: Signal triggering terminal 2 of relay module

IN3: Signal triggering terminal 3 of relay module

IN4: Signal triggering terminal 4 of relay module

IN5: Signal triggering terminal 5 of relay module

IN6: Signal triggering terminal 6 of relay module

IN7: Signal triggering terminal 7 of relay module

IN8: Signal triggering terminal 8 of relay module

Output:

Each module of the relay has one NC (normally close), one NO (normally open) and one COM (Common) terminal. So there are 8 NC, 8 NO and 8 COM of the channel relay in total. NC stands for the normal close port contact and the state without power. NO stands for the normal open port contact and the state with power. COM means the common port. You can choose NC port or NO port according to whether power or not.

Testing Setup:

When a low level is supplied to signal terminal of the 8-channel relay, the LED at the output terminal will light up. Otherwise, it will turn off. If a periodic high and low level is supplied to the signal terminal, you can see the LED will cycle between on and off.

Arduino Application Examples for 4-Channel Relay Board:

Step 1：

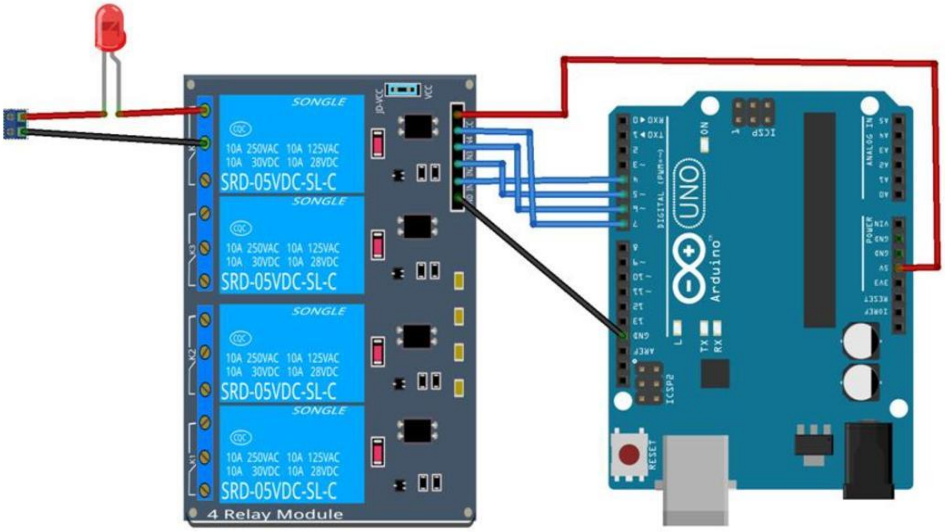
Connect the signal terminal IN1、IN2, IN3 & IN4 of 4-channel relay to digital pin 4, 5, 6, 7 of the Arduino Uno or ATMega2560 board, and connect an LED at the output terminal.

IN1> 4; IN2> 5; IN3>6; IN4>7

Step 2:

Upload the sketch "4 Channel Relay Demo " to the Arduino Uno or ATMega2560 board. Then you can see the LED cycle between on and off.

The actual figure is shown below:



Arduino Sketch: 4 Channel Relay Demo

```
/*************************************************
   Name: 4 channel_relay
   Description: control the 4 channel relay module to ON or OFF
   Website: www.handsontec.com
   Email: techsupport@handsontec.com
*************************************************/

//the relays connect to

int RelayControl1 = 4;    // Digital Arduino Pin used to control the motor
int RelayControl2 = 5;
int RelayControl3 = 6;
int RelayControl4 = 7;


void setup()
{
  Serial.begin(9600);
  pinMode(RelayControl1, OUTPUT);
  pinMode(RelayControl2, OUTPUT);
  pinMode(RelayControl3, OUTPUT);
  pinMode(RelayControl4, OUTPUT);
```

```
}


void loop()
{

  digitalWrite(RelayControl1,HIGH);// NO1 and COM1 Connected (LED on)
  delay(1000);
  digitalWrite(RelayControl1,LOW);// NO1 and COM1 disconnected (LED off)
  delay(1000);
  digitalWrite(RelayControl2,HIGH);
  delay(1000);
  digitalWrite(RelayControl2,LOW);
  delay(1000);
  digitalWrite(RelayControl3,HIGH);
  delay(1000);
  digitalWrite(RelayControl3,LOW);
  delay(1000);
  digitalWrite(RelayControl4,HIGH);
  delay(1000);
  digitalWrite(RelayControl4,LOW);
  delay(1000);
}
```

MEAN WELL 150W Single Output Switching Power Supply **LRS-150 series**

Features

- AC input range selectable by switch
- Withstand 300VAC surge input for 5 second
- No load power consumption<0.5W
- Miniature size and 1U low profile
- High operating temperature up to 70℃
- Protections: Short circuit / Overload / Over voltage / Over temperature
- Cooling by free air convection
- Compliance to IEC/EN 60335-1(PD3) and IEC/EN61558-1, 2-16 for household appliances
- Operating altitude up to 5000 meters
- Withstand 5G vibration test
- High efficiency, long life and high reliability
- LED indicator for power on
- 100% full load burn-in test
- 3 years warranty

Applications

- Industrial automation machinery
- Industrial control system
- Mechanical and electrical equipment
- Electronic instruments, equipments or apparatus
- Household appliances

Description

LRS-150 series is a 150W single-output enclosed type power supply with 30mm of low profile design. Adopting the input of 115VAC or 230VAC(selectable by switch), the entire series provides an output voltage line of 12V, 15V, 24V, 36V and 48V.

In addition to the high efficiency up to 90%, the design of metallic mesh case enhances the heat dissipation of LRS-150 that the whole series operates from -30℃ through 70℃ under air convection without a fan. Delivering an extremely low no load power consumption (less than 0.5W), it allows the end system to easily meet the worldwide energy requirement. LRS-150 has the complete protection functions and 5G anti-vibration capability; it is complied with the international safety regulations such as TUV EN60950-1, EN60335-1,EN61558-1/-2-16, UL60950-1 and GB4943. LRS-150 series serves as a high price-to-performance power supply solution for various industrial applications.

Model Encoding

LRS - 150 - 12

- 12 — Output voltage
- 150 — Output wattage
- LRS — Series name

MEAN WELL 150W Single Output Switching Power Supply **LRS-150 series**

SPECIFICATION

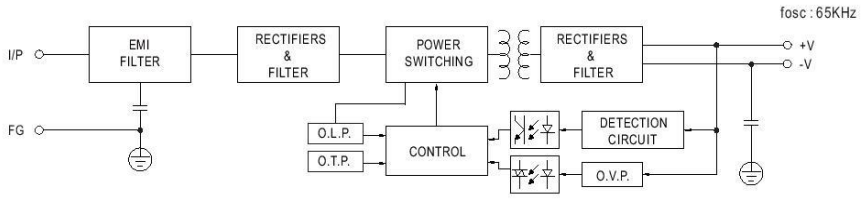
MODEL		LRS-150-12	LRS-150-15	LRS-150-24	LRS-150-36	LRS-150-48
OUTPUT	DC VOLTAGE	12V	15V	24V	36V	48V
	RATED CURRENT	12.5A	10A	6.5A	4.3A	3.3A
	CURRENT RANGE	0 ~ 12.5A	0 ~ 10A	0 ~ 6.5A	0 ~ 4.3A	0 ~ 3.3A
	RATED POWER	150W	150W	156W	154.8W	158.4W
	RIPPLE & NOISE (max.) Note.2	150mVp-p	150mVp-p	200mVp-p	200mVp-p	200mVp-p
	VOLTAGE ADJ. RANGE	10.2 ~ 13.8V	13.5 ~ 18V	21.6 ~ 28.8V	32.4 ~ 39.6V	43.2 ~ 52.8V
	VOLTAGE TOLERANCE Note.3	±1.0%	±1.0%	±1.0%	±1.0%	±1.0%
	LINE REGULATION Note.4	±0.5%	±0.5%	±0.5%	±0.5%	±0.5%
	LOAD REGULATION Note.5	±0.5%	±0.5%	±0.5%	±0.5%	±0.5%
	SETUP, RISE TIME	500ms, 30ms/230VAC 500ms, 30ms/115VAC at full load				
	HOLD UP TIME (Typ.)	40ms/230VAC 35ms/115VAC at full load				
INPUT	VOLTAGE RANGE	85 ~ 132VAC / 170 ~ 264VAC by switch 240 ~ 370VDC(switch on 230VAC)				
	FREQUENCY RANGE	47 ~ 63Hz				
	EFFICIENCY (Typ.)	87.5%	88.5%	89%	89%	90%
	AC CURRENT (Typ.)	2.8A/115VAC 1.6A/230VAC				
	INRUSH CURRENT (Typ.)	COLD STAR 60A/230VAC				
	LEAKAGE CURRENT	<0.75mA / 240VAC				
PROTECTION	OVER LOAD	110 ~ 140% rated output power				
		Protection type : Hiccup mode, recovers automatically after fault condition is removed				
	OVER VOLTAGE	13.8 ~ 16.2V	18.75 ~ 21.75V	28.8 ~ 33.6V	41.4 ~ 48.6V	55.2 ~ 64.8V
		Protection type : Shut down o/p voltage, re-power on to recover				
	OVER TEMPERATURE	Shut down o/p voltage, re-power on to recover				
ENVIRONMENT	WORKING TEMP.	-30 ~ +70℃ (Refer to "Derating Curve")				
	WORKING HUMIDITY	20 ~ 90% RH non-condensing				
	STORAGE TEMP., HUMIDITY	-40 ~ +85℃, 10 ~ 95% RH				
	TEMP. COEFFICIENT	±0.03%/℃ (0 ~ 50℃)				
	VIBRATION	10 ~ 500Hz, 5G 10min./1cycle, 60min. each along X, Y, Z axes				
SAFETY & EMC (Note 7)	SAFETY STANDARDS	UL60950-1, TUV EN60950-1, EN60335-1, EN61558-1/-2-16,CCC GB4943 approved				
	WITHSTAND VOLTAGE	I/P-O/P:3.75KVAC I/P-FG:2KVAC O/P-FG:1.25KVAC				
	ISOLATION RESISTANCE	I/P-O/P, I/P-FG, O/P-FG:100M Ohms / 500VDC / 25℃/ 70% RH				
	EMC EMISSION	Compliance to EN55022 (CISPR22), GB9254 Class B, EN55014, EN61000-3-2Class A(≤75% Load),EN61000-3-3				
	EMC IMMUNITY	Compliance to EN61000-4-2,3,4,5,6,8,11, EN61000-6-2 (EN50082-2), heavy industry level, criteria A				
OTHERS	MTBF	601K hrs min. MIL-HDBK-217F (25℃)				
	DIMENSION	159*97*30mm (L*W*H)				
	PACKING	0.48Kg ; 30pcs/15.4Kg/0.75CUFT				

NOTE

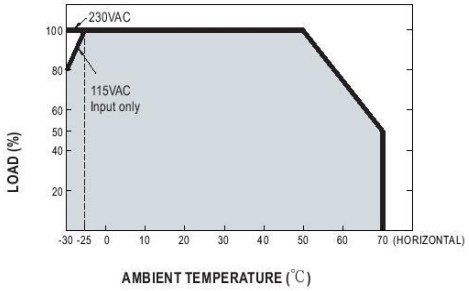
1. All parameters NOT specially mentioned are measured at 230VAC input, rated load and 25℃ of ambient temperature.
2. Ripple & noise are measured at 20MHz of bandwidth by using a 12" twisted pair-wire terminated with a 0.1uf & 47uf parallel capacitor.
3. Tolerance : includes set up tolerance, line regulation and load regulation.
4. Line regulation is measured from low line to high line at rated load.
5. Load regulation is measured from 0% to 100% rated load.
6. Length of set up time is measured at cold first start. Turning ON/OFF the power supply very quickly may lead to increase of the set up time.
7. The power supply is considered a component which will be installed into a final equipment. All the EMC tests are been executed by mounting the unit on a 360mm*360mm metal plate with 1mm of thickness. The final equipment must be re-confirmed that it still meets EMC directives. For guidance on how to perform these EMC tests, please refer to "EMI testing of component power supplies." (as available on http://www.meanwell.com)
8. The ambient temperature derating of 5℃/1000m is needed for operating altitude greater than 2000m (6500ft).

MEAN WELL 150W Single Output Switching Power Supply **LRS-150 series**

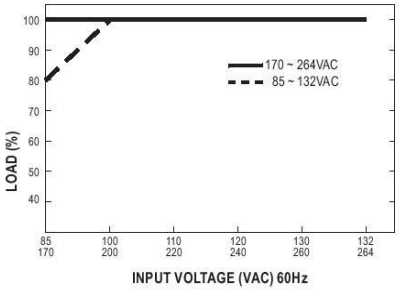
■ Block Diagram



■ Derating Curve



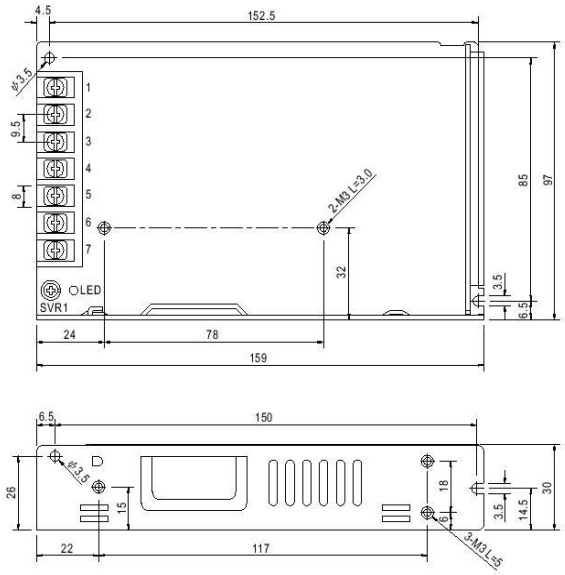
■ Static Characteristics



MEAN WELL 150W Single Output Switching Power Supply **LRS-150 series**

■ Mechanical Specification

Case No.241A Unit:mm



Terminal Pin No. Assignment

Pin No.	Assignment	Pin No.	Assignment
1	AC/L	4,5	DC OUTPUT -V
2	AC/N	6,7	DC OUTPUT +V
3	FG ⏚		

■ Installation Manual

Please refer to : http://www.meanwell.com/webnet/search/InstallationSearch.html

7inch HDMI LCD (C)

From Waveshare Wiki

Contents

Introduction

1024×600, 7 inch Capacitive Touch Screen LCD, HDMI interface, supports various systems

More (http://www.waveshare.com/7inch-HDMI-LCD-C.htm)

Working with PC

This product supports Windows 10/8.1/8/7 OS. For the Windows 10/8.1/8 OS, the touch screen supports multi-touch up to 5 points. For some Window 7 OS, the touch screen supports single touch only.

Turn on the "backlight" switch then connect the LCD to your PC (USB Port of LCD -> USB Port of PC; HDMI Port of LCD -> HDMI Port of PC. Please first connect the USB Ports then connect the HDMI Port). A new touch drive will be recognized by Windows and you can use the LCD

7inch HDMI LCD (C)
Supports various systems

7 inch Capacitive Touch Screen LCD, HDMI interface, supports various systems

7inch HDMI LCD (C) (with bicolor case)

7inch HDMI LCD (C) + Bicolor case

Primary Attribute

Category: OLEDs / LCDs, LCD, Raspberry Pi LCD

Brand: Waveshare

Website

English: Waveshare website (http://www.waveshare.com/)

Chinese: 官方中文站点 (http://www.waveshare.net/)

Onboard Interfaces

USB	HDMI

as a human interface device. When multiple displays are detected by your PC, the LCD can only be used to control the cursor on main display. So it is proposed to set the LCD as the main display.

Working with Raspberry Pi

For the Windows OS on PC, the resolution of the LCD is automatically identified. Hence, you do not need to make the relative settings. When working with Raspberry Pi, you should set the resolution of the LCD by yourself, or else the LCD screen will not work. For more detail information, please read the following section.

Turn on the "backlight" switch then connect the LCD to your Pi (HDMI Port of LCD -> HDMI Port of Pi; USB Port of LCD -> USB Port of Pi; 5V~2A power supply). Download the Raspbian image from Raspberry Pi web site (https://www.raspberrypi.org/downloads/). Write the image to a TF card and append the following lines to the config.txt file which is located in the root of your TF card:

```
max_usb_current=1
hdmi_group=2
hdmi_mode=87
hdmi_cvt 1024 600 60 6 0 0 0
hdmi_drive=1
```

You must make sure that there are no spaces on either side of the equal sign.

Save and connect the TF card to your Pi then power up.

(Touch input working well means that the LED firmware is Rev2.1. If the LCD firmware is 1.1, see #About LCD revision)

Note:

- Resolution of Ubuntu Mate OS or Windows 10 IoT Core OS can also be set properly by editing config.txt.
- For Pi Zero / Zero W: if you've used an SD card on a Pi 3 and then attached the card to the Pi Zero, the touch screen often doesn't work. In such cases, you have to write a fresh system image to the SD card. The first boot up must be done on the Pi Zero but not Pi 3, due to initialization for a corresponding device.

Rotation

Display Rotating

To rotating the display, you can append this statement to the config file

```
display_rotate=1 #1: 90; 2: 180; 3: 270
```

Reboot the Raspberry Pi

```
sudo reboot
```

Touch rotating

With the operation above. The screen could rotate in display. However, the touch works improperly. To rotate the touch as display, you could do as below:

1. install libinput

```
sudo apt-get install xserver-xorg-input-libinput
```

2. create an xorg.conf.d folder

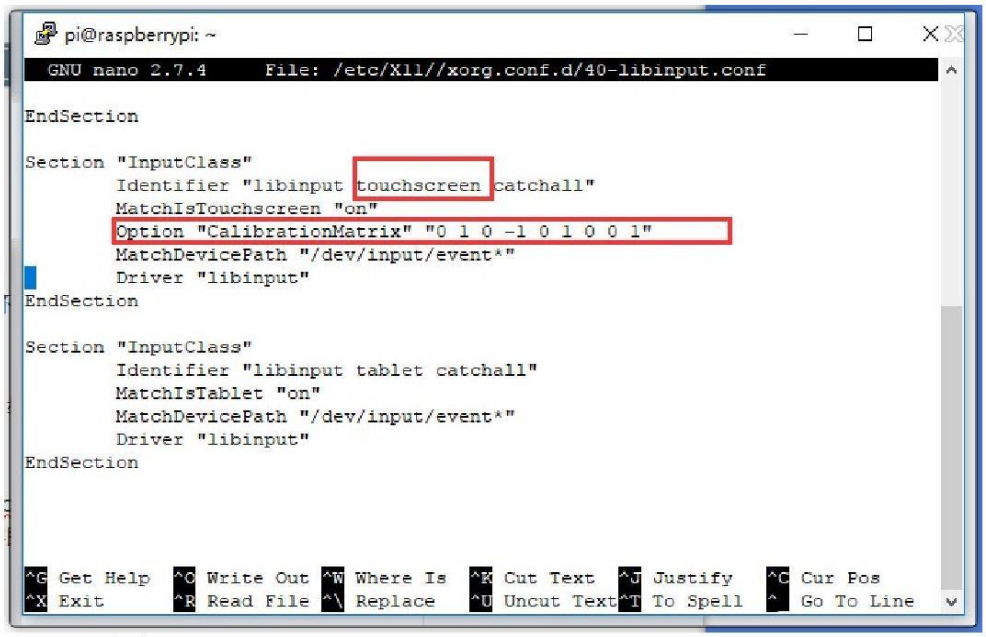
```
sudo mkdir /etc/X11/xorg.conf.d
```

3. copy file 40-libinput-conf to the folder which we created

```
sudo cp /usr/share/X11/xorg.conf.d/40-libinput.conf /etc/X11/xorg.conf.d/
```

4. Append a statement to touchscreen part of the file as below:

```
sudo nano /etc/X11/xorg.conf.d/40-libinput.conf
```



5. save and reboot your Pi

```
sudo reboot
```

After completing these steps. The LCD could rotate 90 degree both display and touch.

Note:

90 degree: Option "CalibrationMatrix" "0 1 0 -1 0 1 0 0 1"

180 degree: Option "CalibrationMatrix" "-1 0 1 0 -1 1 0 0 1"

270 degree: Option "CalibrationMatrix" "0 -1 1 1 0 0 0 0 1"

About LCD revision

An LCD with Rev 2.1 firmware does not require any drivers, that is, touch function works properly without installing any software. So we did not provide any drivers and images for Rev 2.1 LCDs. The following drivers are only available for the LCD with Rev 1.1 firmware. But if you install the driver to the Rev 2.1 one, it will lead touch function not to work.

Rev2.1 New Features

- Upgrade to IPS screen, wider viewing angle, more clear displaying.
- Standard HID protocol, easy to be integrated into your system.
- For the Raspberry Pi, supports Raspbian, Ubuntu Mate, single touch, and driver free.
- When work as a computer monitor, supports Windows 10/8.1/8/7, five-points touch, and driver free.

Rev1.1 and before

- For the Raspberry Pi, comes with Raspbian driver (works with your Raspbian directly), and Ubuntu image.
- When work as a computer monitor, touch function is unavailable.

How to distinguish

- See the backside of your LCD. The Revision number "Rev2.1" printed means that the LCD firmware is Rev 2.1.
- However, "Rev1.1" printed on the backside doesn't mean that the LCD firmware must be Rev 1.1. Generally speaking, a LCD shipped after January 1, 2016 may be a Rev 2.1 one, although it was printed "Rev1.1".

Note: The only difference between Rev 1.1 and Rev 2.1 is the firmware, but hardware solutions, placement and routing are all the same. (PCB printings might be different due to different production batches.)

You can verify the firmware by these steps:

1. Using Raspberry Pi: Connect the LCD to your Pi (HDMI Port of LCD -> HDMI Port of Pi; USB Port of LCD -> USB Port of Pi; 5V~2A power supply). Download the image, e.g. Raspbian, from Raspberry Pi web site (https://www.raspberrypi.org/downloads/). Write the image to a TF card and add the following code to the end of /boot/config.txt:

```
max_usb_current=1
hdmi_group=2
hdmi_mode=87
hdmi_cvt 1024 600 60 6 0 0 0
hdmi_drive=1
```

You must make sure that there are no spaces on either side of the equal sign.

Save and connect the TF card to your Pi then power up. If touch works, that means the firmware revision is Rev 2.1.

2. Without Raspberry Pi: A PC (Windows 10/8.1/8/7) is required which cannot connect with other display device. Connect the LCD to your PC (USB Port of LCD -> USB Port of PC; HDMI Port of LCD -> HDMI Port of PC. Please first connect the USB Ports then connect the HDMI Port). If a new touch drive is recognized by Windows, that means the firmware revision is Rev 2.1. In this case, after the driver successfully installed, you can use the LCD as a human interface device.

- If the touch function doesn't work properly after these steps, the firmware revision is often Rev 1.1, which can also work by other methods, see Rev 1.1 Manual.

Detailed infomation

Depending on the firmware, please view the instructions of different revision:

- 7inch HDMI LCD (C) (Firmware Rev 2.1) User Manual
- 7inch HDMI LCD (C) (Firmware Rev 1.1) User Manual(Not support Raspberry Pi 3 Model B) (http://www.waveshare.com/wiki/7inch_HDMI_LCD_(C)_(Firmware_Rev_1.1)_User_Manual)
- How to install 7inch Bicolor case

Resources

An LCD with Rev 2.1 firmware does not require any driver, that is, touch function works properly without installing any software. So we did not provide drivers for Rev 2.1 LCDs.

7inch HDMI LCD (C) image - used for the LCD with firmware Rev2.1	**[Expand]**
7inch HDMI LCD (C) image - used for the LCD with firmware Rev1.1	**[Expand]**

Drivers for Raspbian (only support Rev1.1 LCD)

Out of date, Not available for any Raspbian OS later than 27-May-2016.

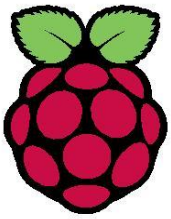
- RPI_2B_USB_TOUCH_CAP_RASPBIAN-4.1.13-v7-7.0-1024x600-20151211.tar (https://drive.google.com/file/d/0B5ceUb50slDnMzNHdXVNbFk5eE0/view?usp=sharing) (For Raspberry Pi 2 Model B)
- RPI_B+_USB_TOUCH_CAP_RASPBIAN-4.1.13-7.0-1024x600-20151211.tar (https://drive.google.com/file/d/0B5ceUb50slDnUmJiak9qTFJsZUU/view?usp=sharing) (For Raspberry Pi B+/A+/B)
- RPI_2B_USB_TOUCH_CAP_RASPBIAN-3.18.16-v7-7.0-1024x600-20150910.tar (https://drive.google.com/file/d/0B5ceUb50slDnY2RJbG84MkMzanM/view?usp=sharing) (For Raspberry Pi 2 Model B)
- RPI_B+_USB_TOUCH_CAP_RASPBIAN-3.18.16-7.0-1024x600-20150910.tar (https://drive.google.com/file/d/0B5ceUb50slDnbF8tMUlrX1hZcXM/view?usp=sharing) (For Raspberry Pi B+/A+/B)

3D Drawings

- 7inch HDMI LCD B/C Drawings

LCD Panel Dimension

DATASHEET



Raspberry Pi Compute Module 3+

Raspberry Pi Compute Module 3+ Lite

Release 1, January 2019

2 Features

2.1 Hardware

- Low cost
- Low power
- High availability
- High reliability
  - Tested over millions of Raspberry Pis Produced to date
  - Module IO pins have 15 micro-inch hard gold plating over 2.5 micron Nickel

2.2 Peripherals

- 48x GPIO
- 2x I2C
- 2x SPI
- 2x UART
- 2x SD/SDIO
- 1x HDMI 1.3a
- 1x USB2 HOST/OTG
- 1x DPI (Parallel RGB Display)
- 1x NAND interface (SMI)
- 1x 4-lane CSI Camera Interface (up to 1Gbps per lane)
- 1x 2-lane CSI Camera Interface (up to 1Gbps per lane)
- 1x 4-lane DSI Display Interface (up to 1Gbps per lane)
- 1x 2-lane DSI Display Interface (up to 1Gbps per lane)

2.3 Software

- ARMv8 Instruction Set
- Mature and stable Linux software stack
  - Latest Linux Kernel support
  - Many drivers upstreamed
  - Stable and well supported userland
  - Full availability of GPU functions using standard APIs

3 Block Diagram

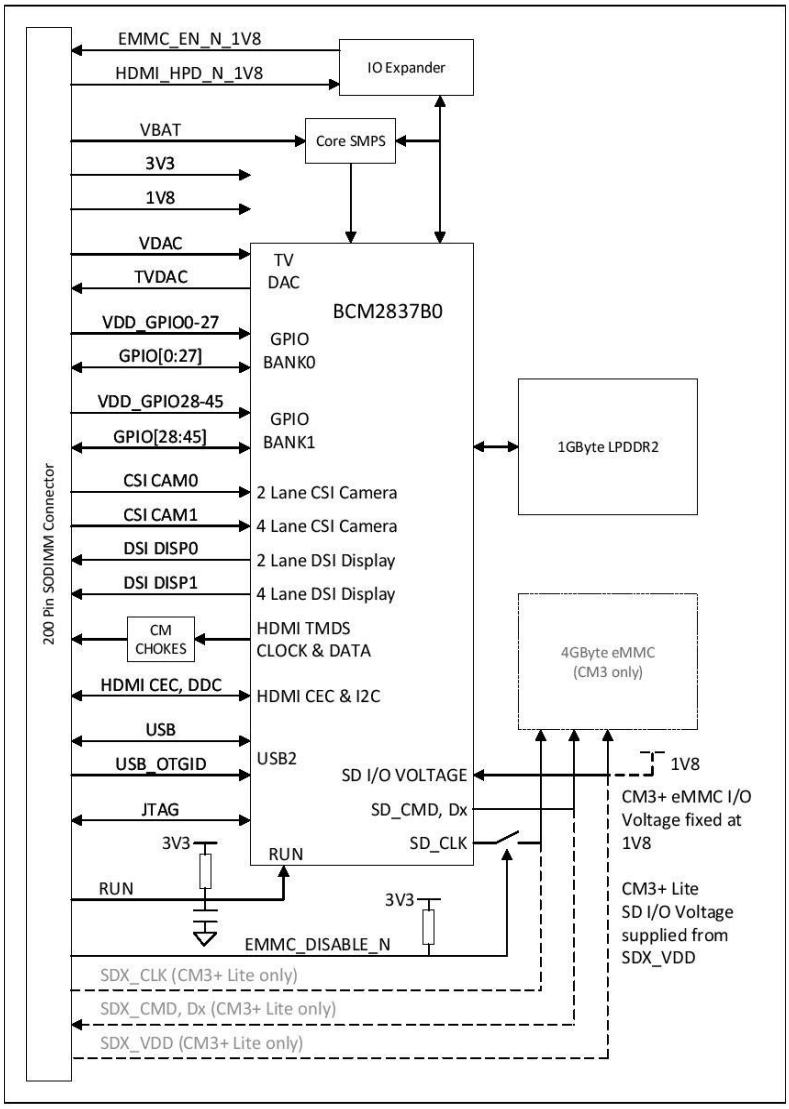


Figure 1: CM3+ Block Diagram

4 Mechanical Specification

The CM3+ modules conform to JEDEC MO-224 mechanical specification for 200 pin DDR2 (1.8V) SODIMM modules and therefore should work with the many DDR2 SODIMM sockets available on the market. **(Please note that the pinout of the Compute Module is not the same as a DDR2 SODIMM module; they are not electrically compatible.)**

The SODIMM form factor was chosen as a way to provide the 200 pin connections using a standard, readily available and low cost connector compatible with low cost PCB manufacture.

The maximum component height on the underside of the Compute Module is 1.2mm.

The maximum component height on the top side of the Compute Module is 2.5mm.

The Compute Module PCB thickness is 1.0mm +/- 0.1mm.

Note that the location and arrangement of components on the Compute Module may change slightly over time due to revisions for cost and manufacturing considerations; however, maximum component heights and PCB thickness will be kept as specified.

Figure 2 gives the CM3+ mechanical dimensions.

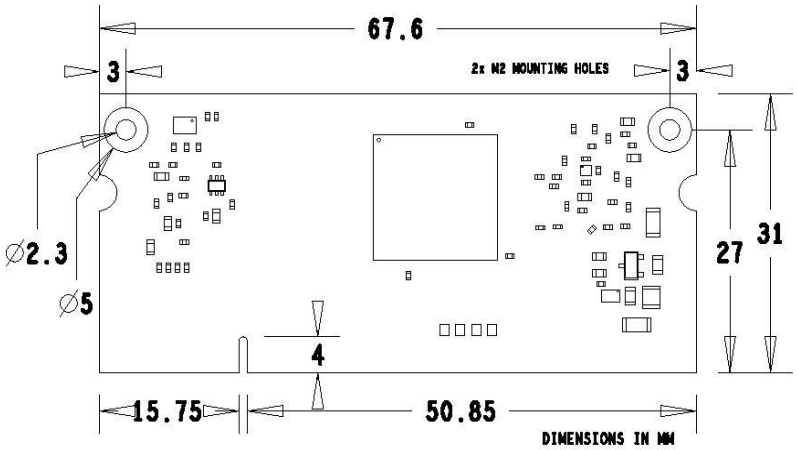


Figure 2: CM3+ Mechanical Dimensions

5 Pin Assignments

Table 2: Compute Module 3+ SODIMM Connector Pinout

Table 2 gives the Compute Module 3+ pinout and Table 3 gives the pin functions.

Pin Name	DIR	Voltage Ref	PDN[a] State	If Unused	Description/Notes
RUN and Boot Control (see text for usage guide)					
RUN	I	3V3[b]	Pull High	Leave open	Has internal 10k pull up
EMMC_DISABLE_N	I	3V3[b]	Pull High	Leave open	Has internal 10k pull up
EMMC_EN_N_1V8	O	1V8	Pull High	Leave open	Has internal 2k2 pull up
GPIO					
GPIO[27:0]	I/O	GPIO0-27_VDD	Pull or Hi-Z[c]	Leave open	GPIO Bank 0
GPIO[45:28]	I/O	GPIO28-45_VDD	Pull or Hi-Z[c]	Leave open	GPIO Bank 1
Primary SD Interface[d,e]					
SDX_CLK	O	SDX_VDD	Pull High	Leave open	Primary SD interface CLK
SDX_CMD	I/O	SDX_VDD	Pull High	Leave open	Primary SD interface CMD
SDX_Dx	I/O	SDX_VDD	Pull High	Leave open	Primary SD interface DATA
USB Interface					
USB_Dx	I/O	-	Z	Leave open	Serial interface
USB_OTGID	I	3V3		Tie to GND	OTG pin detect
HDMI Interface					
HDMI_SCL	I/O	3V3[b]	Z[f]	Leave open	DDC Clock (5.5V tolerant)
HDMI_SDA	I/O	3V3[b]	Z[f]	Leave open	DDC Data (5.5V tolerant)
HDMI_CEC	I/O	3V3	Z	Leave open	CEC (has internal 27k pull up)
HDMI_CLKx	O	-	Z	Leave open	HDMI serial clock
HDMI_Dx	O	-	Z	Leave open	HDMI serial data
HDMI_HPD_N_1V8	I	1V8	Pull High	Leave open	HDMI hotplug detect
CAM0 (CSI0) 2-lane Interface					
CAM0_Cx	I	-	Z	Leave open	Serial clock
CAM0_Dx	I	-	Z	Leave open	Serial data
CAM1 (CSI1) 4-lane Interface					
CAM1_Cx	I	-	Z	Leave open	Serial clock
CAM1_Dx	I	-	Z	Leave open	Serial data
DSI0 (Display 0) 2-lane Interface					
DSI0_Cx	O	-	Z	Leave open	Serial clock
DSI0_Dx	O	-	Z	Leave open	Serial data
DSI1 (Display 1) 4-lane Interface					
DSI1_Cx	O	-	Z	Leave open	Serial clock
DSI1_Dx	O	-	Z	Leave open	Serial data
TV Out					
TVDAC	O	-	Z	Leave open	Composite video DAC output
JTAG Interface					
TMS	I	3V3	Z	Leave open	Has internal 50k pull up
TRST_N	I	3V3	Z	Leave open	Has internal 50k pull up
TCK	I	3V3	Z	Leave open	Has internal 50k pull up
TDI	I	3V3	Z	Leave open	Has internal 50k pull up
TDO	O	3V3	O	Leave open	Has internal 50k pull up

[a] The PDN column indicates power-down state (when RUN pin LOW)
[b] Must be driven by an open-collector driver
[c] GPIO have software enabled pulls which keep state over power-down
[d] Only available on Lite variants
[e] The CM will always try to boot from this interface first
[f] Requires external pull-up resistor to 5V as per HDMI spec

Table 3: Pin Functions

6 Electrical Specification

Caution! Stresses above those listed in Table 4 may cause permanent damage to the device. This is a stress rating only; functional operation of the device under these or any other conditions above those listed in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Symbol	Parameter	Minimum	Maximum	Unit
VBAT	Core SMPS Supply	-0.5	6.0	V
3V3	3V3 Supply Voltage	-0.5	4.10	V
1V8	1V8 Supply Voltage	-0.5	2.10	V
VDAC	TV DAC Supply	-0.5	4.10	V
GPIO0-27_VDD	GPIO0-27 I/O Supply Voltage	-0.5	4.10	V
GPIO28-45_VDD	GPIO28-45 I/O Supply Voltage	-0.5	4.10	V
SDX_VDD	Primary SD/eMMC Supply Voltage	-0.5	4.10	V

Table 4: Absolute Maximum Ratings

DC Characteristics are defined in Table 5

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Unit
V_{IL}	Input low voltage[a]	VDD_IO = 1.8V	-	-	0.6	V
		VDD_IO = 2.7V	-	-	0.8	V
		VDD_IO = 3.3V	-	-	0.9	V
V_{IH}	Input high voltage[a]	VDD_IO = 1.8V	1.0	-	-	V
		VDD_IO = 2.7V	1.3	-	-	V
		VDD_IO = 3.3V	1.6	-	-	V
I_{IL}	Input leakage current	TA = +85°C	-	-	5	μA
C_{IN}	Input capacitance	-	-	5	-	pF
V_{OL}	Output low voltage[b]	VDD_IO = 1.8V, IOL = -2mA	-	-	0.2	V
		VDD_IO = 2.7V, IOL = -2mA	-	-	0.15	V
		VDD_IO = 3.3V, IOL = -2mA	-	-	0.14	V
V_{OH}	Output high voltage[b]	VDD_IO = 1.8V, IOH = 2mA	1.6	-	-	V
		VDD_IO = 2.7V, IOH = 2mA	2.5	-	-	V
		VDD_IO = 3.3V, IOH = 2mA	3.0	-	-	V
I_{OL}	Output low current[c]	VDD_IO = 1.8V, VO = 0.4V	12	-	-	mA
		VDD_IO = 2.7V, VO = 0.4V	17	-	-	mA
		VDD_IO = 3.3V, VO = 0.4V	18	-	-	mA
I_{OH}	Output high current[c]	VDD_IO = 1.8V, VO = 1.4V	10	-	-	mA
		VDD_IO = 2.7V, VO = 2.3V	16	-	-	mA
		VDD_IO = 3.3V, VO = 2.3V	17	-	-	mA
R_{PU}	Pullup resistor	-	50	-	65	kΩ
R_{PD}	Pulldown resistor	-	50	-	65	kΩ

[a] Hysteresis enabled
[b] Default drive strength (8mA)
[c] Maximum drive strength (16mA)

Table 5: DC Characteristics

AC Characteristics are defined in Table 6 and Fig. 3.

Pin Name	Symbol	Parameter	Minimum	Typical	Maximum	Unit
Digital outputs	t_{rise}	10-90% rise time[a]	-	1.6	-	ns
Digital outputs	t_{fall}	90-10% fall time[a]	-	1.7	-	ns
GPCLK	t_{JOSC}	Oscillator-derived GPCLK cycle-cycle jitter (RMS)	-	-	20	ps
GPCLK	t_{JPLL}	PLL-derived GPCLK cycle-cycle jitter (RMS)	-	-	48	ps

[a] Default drive strength, CL = 5pF, VDD_IOx = 3.3V

Table 6: Digital I/O Pin AC Characteristics

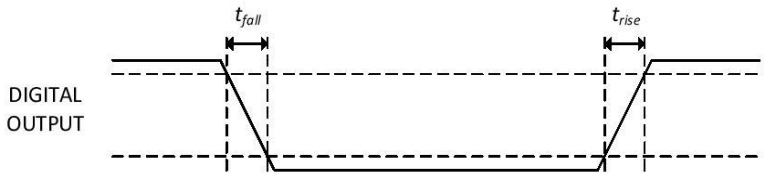


Figure 3: Digital IO Characteristics

7 Power Supplies

The Compute Module 3+ has six separate supplies that must be present and powered at all times; you cannot leave any of them unpowered, even if a specific interface or GPIO bank is unused. The six supplies are as follows:

1. VBAT is used to power the BCM2837 processor core. It feeds the SMPS that generates the chip core voltage.
2. 3V3 powers various BCM2837 PHYs, IO and the eMMC Flash.
3. 1V8 powers various BCM2837 PHYs, IO and SDRAM.
4. VDAC powers the composite (TV-out) DAC.
5. GPIO0-27_VREF powers the GPIO 0-27 IO bank.
6. GPIO28-45_VREF powers the GPIO 28-45 IO bank.

Supply	Descripion	Minimum	Typical	Maximum	Unit
VBAT	Core SMPS Supply	2.5	-	5.0 + 5%	V
3V3	3V3 Supply Voltage	3.3 - 5%	3.3	3.3 + 5%	V
1V8	1V8 Supply Voltage	1.8 - 5%	1.8	1.8 + 5%	V
VDAC	TV DAC Supply[a]	2.5 - 5%	2.8	3.3 + 5%	V
GPIO0-27_VDD	GPIO0-27 I/O Supply Voltage	1.8 - 5%	-	3.3 + 5%	V
GPIO28-45_VDD	GPIO28-45 I/O Supply Voltage	1.8 - 5%	-	3.3 + 5%	V
SDX_VDD	Primary SD/eMMC Supply Voltage	1.8 - 5%	-	3.3 + 5%	V

[a] Requires a clean 2.5-2.8V supply if TV DAC is used, else connect to 3V3

Table 7: Power Supply Operating Ranges

7.1 Supply Sequencing

Supplies should be staggered so that the highest voltage comes up first, then the remaining voltages in descending order. This is to avoid forward biasing internal (on-chip) diodes between supplies, and causing latch-up. Alternatively supplies can be synchronised to come up at exactly the same time as long as at no point a lower voltage supply rail voltage exceeds a higher voltage supply rail voltage.

7.2 Power Requirements

Exact power requirements will be heavily dependent upon the individual use case. If an on-chip subsystem is unused, it is usually in a low power state or completely turned off. For instance, if your application does not use 3D graphics then a large part of the core digital logic will never turn on and need power. This is also the case for camera and display interfaces, HDMI, USB interfaces, video encoders and decoders, and so on.

Powerchain design is critical for stable and reliable operation of the Compute Module 3+. We strongly recommend that designers spend time measuring and verifying power requirements for their particular use case and application, as well as paying careful attention to power supply sequencing and maximum supply voltage tolerance.

Table 8 specifies the recommended minimum power supply outputs required to power the Compute Module 3+.

Supply	Minimum Requirement	Unit
VBAT (CM1)	2000[a]	mW
VBAT (CM3,3L)	3500[a]	mW
3V3	250	mA
1V8	250	mA
VDAC	25	mA
GPIO0-27_VDD	50[b]	mA
GPIO28-45_VDD	50[b]	mA
SDX_VDD	50[b]	mA

[a] Recommended minimum. Actual power drawn is very dependent on use-case
[b] Each GPIO can supply up to 16mA, aggregate current per bank must not exceed 50mA

Table 8: Mimimum Power Supply Requirements

8 Booting

The eMMC Flash device on CM3+ is directly connected to the primary BCM2837 SD/eMMC interface. These connections are not accessible on the module pins. On CM3+ Lite this SD interface is available on the SDX_ pins.

When initially powered on, or after the RUN pin has been held low and then released, the BCM2837 will try to access the primary SD/eMMC interface. It will then look for a file called bootcode.bin on the primary partition (which must be FAT) to start booting the system. If it cannot access the SD/eMMC device or the boot code cannot be found, it will fall back to waiting for boot code to be written to it over USB; in other words, its USB port is in slave mode waiting to accept boot code from a suitable host.

A USB boot tool is available on Github which allows a host PC running Linux to write the BCM2837 boot code over USB to the module. That boot code then runs and provides access to the SD/eMMC as a USB mass storage device, which can then be read and written using the host PC. Note that a Raspberry Pi can be used as the host machine. For those using Windows a precompiled and packeged tool is available. For more information see here.

The Compute Module has a pin called EMMC_DISABLE_N which when shorted to GND will disable the SD/eMMC interface (by physically disconnecting the SD_CMD pin), forcing BCM2837 to boot from USB. Note that when the eMMC is disabled in this way, it takes a couple of seconds from powering up for the processor to stop attempting to talk to the SD/eMMC device and fall back to booting from USB.

Note that once booted over USB, BCM2837 needs to re-enable the SD/eMMC device (by releasing EMMC_DISABLE_N) to allow access to it as mass storage. It expects to be able to do this by driving the EMMC_EN_N_1V8 pin LOW, which at boot is initially an input with a pull up to 1V8. If an end user wishes to add the ability to access the SD/eMMC over USB in their product, similar circuitry to that used on the Compute Module IO Board to enable/disable the USB boot and SD/eMMC must be used; that is, EMMC_DISABLE_N pulled low via MOSFET(s) and released again by MOSFET, with the gate controlled by EMMC_EN_N_1V8. **Ensure you use MOSFETs suitable for switching at 1.8V (i.e. use a device with gate threshold voltage, Vt, suitable for 1.8V switching).**

9 Peripherals

9.1 GPIO

BCM2837 has in total 54 GPIO lines in 3 separate voltage banks. All GPIO pins have at least two alternative functions within the SoC. When not used for the alternate peripheral function, each GPIO pin may be set as an input (optionally as an interrupt) or an output. The alternate functions are usually peripheral I/Os, and most peripherals appear twice to allow flexibility on the choice of I/O voltage.

GPIO bank2 is used on the module to connect to the eMMC device and for an on-board I2C bus (to talk to the core SMPS and control the special function pins). On CM3+ Lite most of bank2 is exposed to allow a user to connect their choice of SD card or eMMC device (if required).

Bank0 and 1 GPIOs are available for general use. GPIO0 to GPIO27 are bank0 and GPIO28-45 make up bank1. GPIO0-27_VDD is the power supply for bank0 and GPIO28-45_VDD is the power supply for bank1. SDX_VDD is the supply for bank2 on CM3+ Lite. These supplies can be in the range 1.8V-3.3V (see Table 7) and are not optional; each bank must be powered, even when none of the GPIOs for that bank are used.

Note that the HDMI_HPD_N_1V8 and EMMC_EN_N_1V8 pins are 1.8V IO and are used for special functions (HDMI hot plug detect and boot control respectively). Please do not use these pins for any other purpose, as the software for the module will always expect these pins to have these special functions. If they are unused please leave them unconnected.

All GPIOs except GPIO28, 29, 44 and 45 have weak in-pad pull-ups or pull-downs enabled when the device is powered on. It is recommended to add off-chip pulls to GPIO28, 29, 44 and 45 to make sure they never float during power on and initial boot.

9.1.1 GPIO Alternate Functions

GPIO	Default Pull	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
0	High	SDA0	SA5	PCLK	-	-	-
1	High	SCL0	SA4	DE	-	-	-
2	High	SDA1	SA3	LCD_VSYNC	-	-	-
3	High	SCL1	SA2	LCD_HSYNC	-	-	-
4	High	GPCLK0	SA1	DPI_D0	-	-	ARM_TDI
5	High	GPCLK1	SA0	DPI_D1	-	-	ARM_TDO
6	High	GPCLK2	SOE_N	DPI_D2	-	-	ARM_RTCK
7	High	SPI0_CE1_N	SWE_N	DPI_D3	-	-	-
8	High	SPI0_CE0_N	SD0	DPI_D4	-	-	-
9	Low	SPI0_MISO	SD1	DPI_D5	-	-	-
10	Low	SPI0_MOSI	SD2	DPI_D6	-	-	-
11	Low	SPI0_SCLK	SD3	DPI_D7	-	-	-
12	Low	PWM0	SD4	DPI_D8	-	-	ARM_TMS
13	Low	PWM1	SD5	DPI_D9	-	-	ARM_TCK
14	Low	TXD0	SD6	DPI_D10	-	-	TXD1
15	Low	RXD0	SD7	DPI_D11	-	-	RXD1
16	Low	FL0	SD8	DPI_D12	CTS0	SPI1_CE2_N	CTS1
17	Low	FL1	SD9	DPI_D13	RTS0	SPI1_CE1_N	RTS1
18	Low	PCM_CLK	SD10	DPI_D14	-	SPI1_CE0_N	PWM0
19	Low	PCM_FS	SD11	DPI_D15	-	SPI1_MISO	PWM1
20	Low	PCM_DIN	SD12	DPI_D16	-	SPI1_MOSI	GPCLK0
21	Low	PCM_DOUT	SD13	DPI_D17	-	SPI1_SCLK	GPCLK1
22	Low	SD0_CLK	SD14	DPI_D18	SD1_CLK	ARM_TRST	-
23	Low	SD0_CMD	SD15	DPI_D19	SD1_CMD	ARM_RTCK	-
24	Low	SD0_DAT0	SD16	DPI_D20	SD1_DAT0	ARM_TDO	-
25	Low	SD0_DAT1	SD17	DPI_D21	SD1_DAT1	ARM_TCK	-
26	Low	SD0_DAT2	TE0	DPI_D22	SD1_DAT2	ARM_TDI	-
27	Low	SD0_DAT3	TE1	DPI_D23	SD1_DAT3	ARM_TMS	-

Table 9: GPIO Bank0 Alternate Functions

GPIO	Default Pull	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
28	None	SDA0	SA5	PCM_CLK	FL0	-	-
29	None	SCL0	SA4	PCM_FS	FL1	-	-
30	Low	TE0	SA3	PCM_DIN	CTS0	-	CTS1
31	Low	FL0	SA2	PCM_DOUT	RTS0	-	RTS1
32	Low	GPCLK0	SA1	RING_OCLK	TXD0	-	TXD1
33	Low	FL1	SA0	TE1	RXD0	-	RXD1
34	High	GPCLK0	SOE_N	TE2	SD1_CLK	-	-
35	High	SPI0_CE1_N	SWE_N	-	SD1_CMD	-	-
36	High	SPI0_CE0_N	SD0	TXD0	SD1_DAT0	-	-
37	Low	SPI0_MISO	SD1	RXD0	SD1_DAT1	-	-
38	Low	SPI0_MOSI	SD2	RTS0	SD1_DAT2	-	-
39	Low	SPI0_SCLK	SD3	CTS0	SD1_DAT3	-	-
40	Low	PWM0	SD4	-	SD1_DAT4	SPI2_MISO	TXD1
41	Low	PWM1	SD5	TE0	SD1_DAT5	SPI2_MOSI	RXD1
42	Low	GPCLK1	SD6	TE1	SD1_DAT6	SPI2_SCLK	RTS1
43	Low	GPCLK2	SD7	TE2	SD1_DAT7	SPI2_CE0_N	CTS1
44	None	GPCLK1	SDA0	SDA1	TE0	SPI2_CE1_N	-
45	None	PWM1	SCL0	SCL1	TE1	SPI2_CE2_N	-

Table 10: GPIO Bank1 Alternate Functions

Table 9 and Table 10 detail the default pin pull state and available alternate GPIO functions. Most of these alternate peripheral functions are described in detail in the Broadcom Peripherals Specification document and have Linux drivers available.

9.1.2 Secondary Memory Interface (SMI)

The SMI peripheral is an asynchronous NAND type bus supporting Intel mode80 type transfers at 8 or 16 bit widths and available in the ALT1 positions on GPIO banks 0 and 1 (see Table 9 and Table 10). It is not publicly documented in the Broadcom Peripherals Specification but a Linux driver is available in the Raspberry Pi Github Linux repository (bcm2835_smi.c in linux/drivers/misc).

9.1.3 Display Parallel Interface (DPI)

A standard parallel RGB (DPI) interface is available on bank 0 GPIOs. This up-to-24-bit parallel interface can support a secondary display. Again this interface is not documented in the Broadcom Peripherals Specification but documentation can be found here.

9.1.4 SD/SDIO Interface

The BCM283x supports two SD card interfaces, SD0 and SD1.

The first (SD0) is a proprietary Broadcom controller that does not support SDIO and is the primary interface used to boot and talk to the eMMC or SDX_x signals.

The second interface (SD1) is standards compliant and can interface to SD, SDIO and eMMC devices; for example on a Raspberry Pi 3 B+ it is used to talk to the on-board CYW43455 WiFi device in SDIO mode.

Both interfaces can support speeds up to 50MHz single ended (SD High Speed Mode).

9.2 CSI (MIPI Serial Camera)

Currently the CSI interface is not openly documented and only CSI camera sensors supported by the official Raspberry Pi firmware will work with this interface. Supported sensors are the OmniVision OV5647 and Sony IMX219.

It is recommended to attach other cameras via USB.

9.3 DSI (MIPI Serial Display)

Currently the DSI interface is not openly documented and only DSI displays supported by the official Raspberry Pi firmware will work with this interface.

Displays can also be added via the parallel DPI interface which is available as a GPIO alternate function - see Table 9 and Section 9.1.3

9.4 USB

The BCM2837 USB port is On-The-Go (OTG) capable. If using either as a fixed slave or fixed master, please tie the USB_OTGID pin to ground.

The USB port (Pins USB_DP and USB_DM) must be routed as 90 ohm differential PCB traces.

Note that the port is capable of being used as a true OTG port however there is no official documentation. Some users have had success making this work.

9.5 HDMI

BCM283x supports HDMI V1.3a.

It is recommended that users follow a similar arrangement to the Compute Module IO Board circuitry for HDMI output.

The HDMI CK_P/N (clock) and D0-D2_P/N (data) pins must each be routed as matched length 100 ohm differential PCB traces. It is also important to make sure that each differential pair is closely phase matched. Finally, keep HDMI traces well away from other noise sources and as short as possible.

Failure to observe these design rules is likely to result in EMC failure.

9.6 Composite (TV Out)

The TVDAC pin can be used to output composite video (PAL or NTSC). Please route this signal away from noise sources and use a 75 ohm PCB trace.

Note that the TV DAC is powered from the VDAC supply which must be a clean supply of 2.5-2.8V. It is recommended users generate this supply from 3V3 using a low noise LDO.

If the TVDAC output is not used VDAC can be connected to 3V3, but it must be powered even if the TV-out functionality is unused.

10 Thermals

The BCM2837 SoC employs DVFS (Dynamic Voltage and Frequency Scaling) on the core voltage. When the processor is idle (low CPU utilisation), it will reduce the core frequency and voltage to reduce current draw and heat output. When the core utilisation exceeds a certain threshold the core votlage is increased and the core frequency is boosted to the maximum working frerquency of 1.2GHz. The voltage and frequency are throttled back when the CPU load reduces back to an 'idle' level OR when the silicon temperature as mesured by the on-chip temperature sensor exceeds 80C (thermal throttling).

A designer must pay careful attention to the thermal design of products using the CM3+ so that performance is not artificially curtailed due to the processor thermal throttling, as the Quad ARM complex in the BCM2837 can generate significant heat output under load.

10.1 Temperature Range

The operating temperature range of the module is set by the lowest maximum and highest minimum of any of the components used.

The eMMC and LPDDR2 have the narrowest range, these are rated for -25 to +80 degrees Celsius. Therefore the nominal range for the CM3+ and CM3+ Lite is -25C to +80C.

However, this range is the maximum for the silicon die; therefore, users would have to take into account the heat generated when in use and make sure this does not cause the temperature to exceed 80 degrees Celsius.

11 Availability

Raspberry Pi guarantee availability of CM3+ and CM3+ Lite until at least January 2026.

12 Support

For support please see the hardware documentation section of the Raspberry Pi website and post questions to the Raspberry Pi forum.

maxon motor

maxon DC motor and maxon EC motor
Key information

The motor as an energy converter

The electrical motor converts electrical power P_{el} (current I_{mot} and voltage U_{mot}) into mechanical power P_{mech} (speed n and torque M). The losses that arise are divided into frictional losses, attributable to P_{mech} and in Joule power losses P_J of the winding (resistance R). Iron losses do not occur in the coreless maxon DC motors. In maxon EC motors, they are treated formally like an additional friction torque. The power balance can therefore be formulated as:

$$P_{el} = P_{mech} + P_J$$

The detailed result is as follows

$$U_{mot} \cdot I_{mot} = \frac{\pi}{30\,000} n \cdot M + R \cdot I_{mot}^2$$

Electromechanical motor constants
The geometric arrangement of the magnetic circuit and winding defines in detail how the motor converts the electrical input power (current, voltage) into mechanical output power (speed, torque). Two important characteristic values of this energy conversion are the speed constant k_n and the torque constant k_M. The speed constant combines the speed n with the voltage induced in the winding U_{ind} (= EMF). U_{ind} is proportional to the speed; the following applies:

$$n = k_n \cdot U_{ind}$$

Similarly, the torque constant links the mechanical torque M with the electrical current I_{mot}.

$$M = k_M \cdot I_{mot}$$

The main point of this proportionality is that torque and current are equivalent for the maxon motor.
The current axis in the motor diagrams is therefore shown as parallel to the torque axis as well.

Motor diagrams

A diagram can be drawn for every maxon DC and EC motor, from which key motor data can be taken. Although tolerances and temperature influences are not taken into consideration, the values are sufficient for a first estimation in most applications. In the diagram, speed n, current I_{mot}, power output P_2 and efficiency η are applied as a function of torque M at constant voltage U_{mot}.

Speed-torque line
This curve describes the mechanical behavior of the motor at a constant voltage U_{mot}:
– Speed decreases linearly with increasing torque.
– The faster the motor turns, the less torque it can provide.
The curve can be described with the help of the two end points, no load speed n_0 and stall torque M_H (cf. lines 2 and 7 in the motor data).
DC motors can be operated at any voltage. No load speed and stall torque change proportionally to the applied voltage. This is equivalent to a parallel shift of the speed-torque line in the diagram. Between the no load speed and voltage, the following proportionality applies in good approximation

$$n_0 \approx k_n \cdot U_{mot}$$

where k_n is the speed constant (line 13 of the motor data).

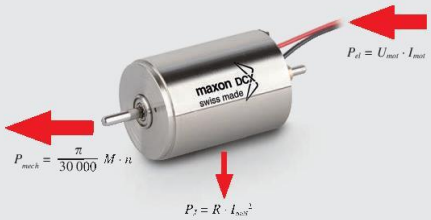
Independent of the voltage, the speed-torque line is described most practically by the slope or gradient of the curve (line 14 of the motor data).

$$\frac{\Delta n}{\Delta M} = \frac{n_0}{M_H}$$

See also: Technology – short and to the point, explanation of the motor

Units
In all formulas, the variables are to be used in the units according to the catalog (cf. physical variables and their units on page 48).

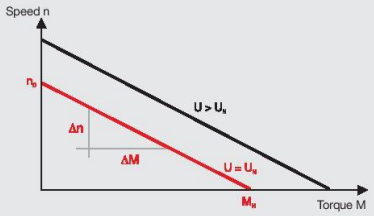
The following applies in particular:
– All torques in mNm
– All currents in A (even no load currents)
– Speeds (rpm) instead of angular velocity (rad/s)



Motor constants
Speed constant k_n and torque constant k_M are not independent of one another. The following applies:

$$k_n \cdot k_M = \frac{30\,000}{\pi}$$

The speed constant is also called specific speed. Specific voltage, generator or voltage constants are mainly the reciprocal value of the speed constant and describe the voltage induced in the motor per speed. The torque constant is also called specific torque. The reciprocal value is called specific current or current constant.



Derivation of the speed-torque line
The following occurs if one replaces current I_{mot} with torque M using the torque constant in the detailed power balance:

$$U_{mot} \cdot \frac{M}{k_M} = \frac{\pi}{30\,000} n \cdot M + R \cdot \left(\frac{M}{k_M}\right)^2$$

Transformed and taking account of the close relationship of k_M and k_n, an equation is produced of a straight line between speed n and torque M.

$$n = k_n \cdot U_{mot} - \frac{30\,000}{\pi} \cdot \frac{R}{k_M^2} \cdot M$$

or with the gradient and the no load speed n_0

$$n = n_0 - \frac{\Delta n}{\Delta M} \cdot M$$

The speed-torque gradient is one of the most informative pieces of data and allows direct comparison between different motors. The smaller the speed-torque gradient, the less sensitive the speed reacts to torque (load) changes and the stronger the motor. With the maxon motor, the speed-torque gradient within the winding series of a motor type (i.e. on one catalog page) remains practically constant.

Current gradient
The equivalence of current to torque is shown by an axis parallel to the torque: more current flowing through the motor produces more torque. The current scale is determined by the two points no load current I_0 and starting current I_A (lines 3 and 8 of motor data).
The no load current is equivalent to the friction torque M_R, that describes the internal friction in the bearings and commutation system.

$$M_R = k_M \cdot I_0$$

In the maxon EC motor, there are strong, speed dependent iron losses in the stator iron stack instead of friction losses in the commutation system.

The motors develop the highest torque when starting. It is many times greater than the normal operating torque, so the current uptake is the greatest as well.
The following applies for the stall torque M_H and starting current I_A

$$M_H = k_M \cdot I_A$$

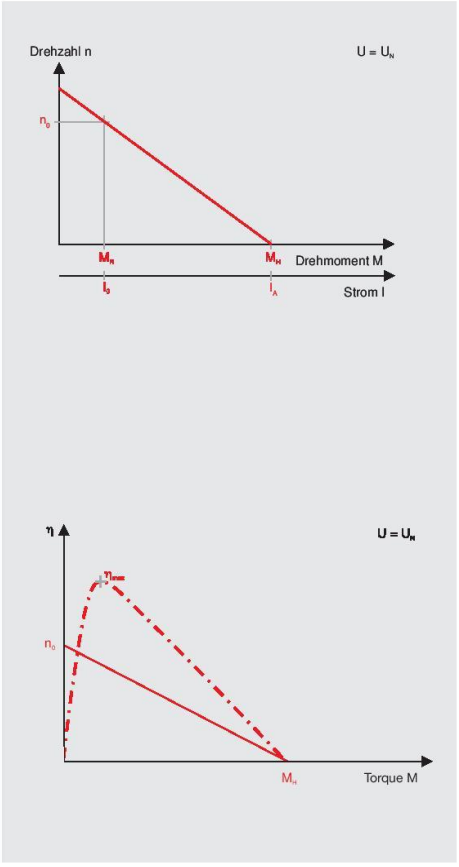
Efficiency curve
The efficiency η describes the relationship of mechanical power delivered to electrical power consumed.

$$\eta = \frac{\pi}{30\,000} \cdot \frac{n \cdot (M - M_R)}{U_{mot} \cdot I_{mot}}$$

One can see that at constant applied voltage U and due to the proportionality of torque and current, the efficiency increases with increasing speed (decreasing torque). At low torques, friction losses become increasingly significant and efficiency rapidly approaches zero. Maximum efficiency (line 9 of motor data) is calculated using the starting current and no load current and is dependent on voltage.

$$\eta_{max} = \left(1 - \sqrt{\frac{I_0}{I_A}}\right)^2$$

Maximum efficiency and maximum output power do not occur at the same torque.

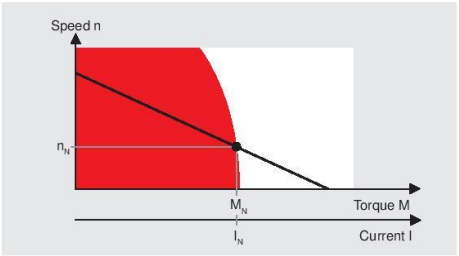


Rated operating point

The rated operating point is an ideal operating point for the motor and derives from operation at nominal voltage U_N (line 1 of motor data) and nominal current I_N (line 6). The nominal torque M_N produced (line 5) in this operating point follows from the equivalence of torque and current.

$$M_N \approx k_M \cdot (I_N - I_0)$$

Nominal speed n_N (line 4) is reached in line with the speed gradient.
The choice of nominal voltage follows from considerations of where the maximum no load speed should be. The nominal current derives from the motor's thermally maximum permissible continuous current.



maxon motor

Motor diagrams, operating ranges

The catalogue contains a diagram of every maxon DC and EC motor type that shows the operating ranges of the different winding types using a typical motor.

Permanent operating range
The two criteria "maximum continuous torque" and "maximum permis- sible speed" limit the continuous operating range. Operating points within this range are not critical thermally and do not generally cause increased wear of the commutation system.

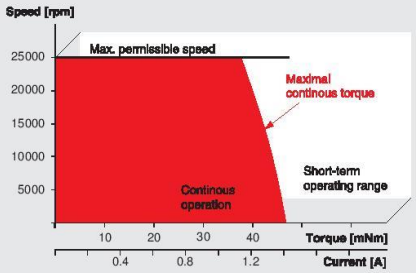
Short-term operating range
The motor may only be loaded with the maximum continuous current for thermal reasons. However, temporary higher currents (torques) are allowed. As long as the winding temperature is below the critical value, the winding will not be damaged. Phases with increased currents are time limited. A measure of how long the temporary overload can last is provided by the thermal time constant of the winding (line 19 of the motor data). The magnitude of the times with overload ranges from several seconds for the smallest motors (6 mm to 13 mm diameter) up to roughly one minute for the largest (60 mm to 90 mm diameter). The calculation of the exact overload time is heavily dependent on the motor current and the rotor's starting temperature.

Maximum continuous current, maximum continuous torque
The Jule power losses heat up the winding. The heat produced must be able to dissipate and the maximum rotor temperature (line 22 of the motor data) should not be exceeded. This results in a maximum continuous current, at which the maximum winding temperature is attained under standard conditions (25 °C ambient temperature, no heat dissipation via the flange, free air circulation). Higher motor currents cause excessive winding temperatures.
The nominal current is selected so that it corresponds to this maximum permissible constant current. It depends heavily on the winding. These thin wire windings have lower nominal current levels than thick ones. With very low resistive windings, the brush system's capacity can further limit the permissible constant current. With graphite brush motors, friction losses increase sharply at higher speeds. With EC motors, eddy current losses increase in the return as speed increases and produce additional heat. The maximum permissible continuous current decreases at faster speeds accordingly.
The nominal torque allocated to the nominal current is almost constant within a motor type's winding range and represents a characteristic size of the motor type.

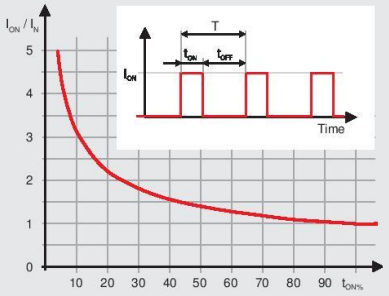
The maximum permissible speed
for DC motors is primarily limited by the commutation system. The commutator and brushes wear more rapidly at very high speeds.
The reasons are:
– Increased mechanical wear because of the large traveled path of the commutator
– Increased electro-erosion because of brush vibration and spark formation.

A further reason for limiting the speed is the rotor's residual mechanical imbalance which shortens the service life of the bearings. Higher speeds than the limit speed n_{max} (line 23) are possible, however, they are "paid for" by a reduced service life expectancy. The maximum permissible speed for the EC motor is calculated based on service life considerations of the ball bearings (at least 20 000 hours) at the maximum residual imbalance and bearing load.

Maximum winding temperature
The motor current causes the winding to heat up due to the winding's resistance. To prevent the motor from overheating, this heat must dissipate to the environment via the stator. The coreless winding is the thermally critical point. The maximum rotor temperature must not be exceeded, even temporarily. With graphite brush motors and EC motors which tend to have higher current loads, the maximum rotor temperature is 125 °C (in individual cases up to 155 °C). Motors with precious metal commutators only allow lower current loads, so that the rotor temperatures must not exceed 85 °C. Favourable mounting conditions, such as good air circulation or cooling plates, can significantly lower temperatures.



Operating range diagram



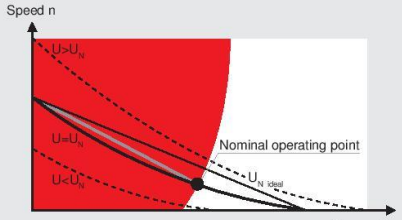
ON	Motor in operation
OFF	Motor stationary
I_{ON}	Max. peak current
I_N	Max. permissible continuous current (line 6)
t_{ON}	ON time *[s]*, should not exeed τ_w (line 19)
T	Cycle time $t_{ON} + t_{OFF}$ *[s]*
$t_{ON\%}$	Duty cycle as percentage of cycle time. The motor may be overloaded by the relationship I_{ON} / I_N at X % of the total cycle time.

$$I_{on} = I_N \sqrt{\frac{T}{t_{ON}}}$$

maxon flat motor

Multipole EC motors, such as maxon flat motors, require a greater number of commutation steps for a motor revolution (6 x number of pole pairs). Due to the wound stator teeth they have a higher terminal inductance than motors with an ironless winding. As a result at higher speed, the current cannot develop fully during the correspondingly short commutation intervals. Therefore, the apparent torque produced is lower. Current is also fed back into the controller's power stage.
As a result, motor behaviour deviates from the ideal linear speed-torque gradient. The apparent speed-torque gradient depends on voltage and speed: The gradient is steeper at higher speeds.
Mostly, flat motors are operated in the continuous operation range where the achievable speed-torque gradient at nominal voltage can be approximated by a straight line between no load speed and nominal operating point. The achievable speed-torque gradient is approximately.

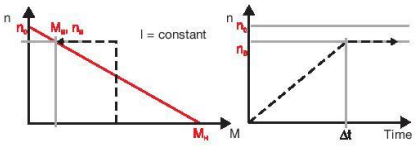
$$\frac{\Delta n}{\Delta M} \approx \frac{n_0 - n_N}{M_N}$$



Acceleration

In accordance with the electrical boundary conditions (power supply, control, battery), a distinction is principally made between two different starting processes:
– Start at constant voltage (without current limitation)
– Start at constant current (with current limitation)

Start under constant current
A current limit always means that the motor can only deliver a limited torque. In the speed-torque diagram, the speed increases on a vertical line with a constant torque. Acceleration is also constant, thus simplifying the calculation. Start at constant current is usually found in applications with servo amplifiers, where acceleration torques are limited by the amplifier's peak current.



– Angular acceleration α (in rad / s²) at constant current I or constant torque M with an additional load of inertia J_L:

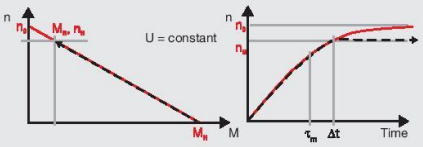
$$\alpha = 10^4 \cdot \frac{k_M \cdot I_{max}}{J_R + J_L} = 10^4 \cdot \frac{M}{J_R + J_L}$$

– Run-up time Δt (in ms) at a speed change Δn with an additional load inertia J_L:

$$\Delta t = \frac{\pi}{300} \cdot \Delta n \cdot \frac{J_R + J_L}{k_M \cdot I_{max}}$$

(all variables in units according to the catalog)

Start with constant terminal voltage
Here, the speed increases from the stall torque along the speed-torque line. The greatest torque and thus the greatest acceleration is effective at the start. The faster the motor turns, the lower the acceleration. The speed increases more slowly. This exponentially flattening increase is described by the mechanical time constant τ_m (line 15 of the motor data). After this time, the rotor at the free shaft end has attained 63% of the no load speed. After roughly three mechanical time constants, the rotor has almost reached the no load speed.



– Mechanical time constant τ_m (in ms) of the unloaded motor:

$$\tau_m = 100 \cdot \frac{J_R \cdot R}{k_M^2}$$

– Mechanical time constants τ_m' (in ms) with an additional load inertia J_L:

$$\tau_m' = 100 \cdot \frac{J_R \cdot R}{k_M^2} \left(1 + \frac{J_L}{J_R}\right)$$

– Maximum angular acceleration α_{max} (in rad / s²) of the unloaded motor:

$$\alpha_{max} = 10^4 \cdot \frac{M_H}{J_R}$$

– Maximum angular acceleration α_{max} (in rad / s²) with an additional load inertia J_L:

$$\alpha_{max} = 10^4 \cdot \frac{M_H}{J_R + J_L}$$

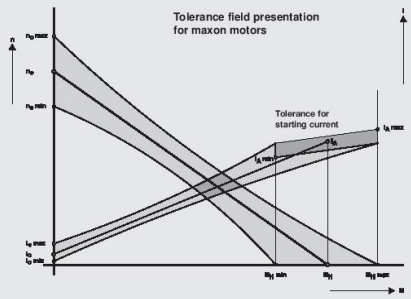
– Run-up time (in ms) at constant voltage up to the operating point (M_L, n_L):

$$\Delta t = \tau_m' \cdot In \left(\frac{\left(1 - \frac{M_L + M_R}{M_H}\right) \cdot n_0}{\left(1 - \frac{M_L + M_R}{M_H}\right) \cdot n_0 - n_L} \right)$$

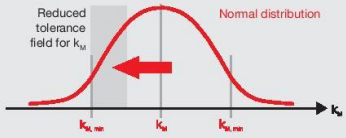
maxon motor

Tolerances

Tolerances must be considered in critical ranges. The possible deviations of the mechanical dimensions can be found in the overview drawings. The motor data are average values: the adjacent diagram shows the effect of tolerances on the curve characteristics. They are mainly caused by differences in the magnetic field strength and in wire resistance, and not so much by mechanical influences. The changes are heavily exaggerated in the diagram and are simplified to improve understanding. It is clear, however, that in the motor's actual operating range, the tolerance range is more limited than at start or at no load. Our computer sheets contain all detailed specifications.



Calibrating
The tolerances can be limited by controlled de-magnetization of the motors. Motor data can be accurately specified down to 1 to 3%. However, the motor characteristic values lie in the lower portion of the standard tolerance range.



Thermal behavior

The Joule power losses P_J in the winding determine heating of the motor. This heat energy must be dissipated via the surfaces of the winding and motor. The increase ΔT_W of the winding temperature T_W with regard to the ambient temperature arises from heat losses P_J and thermal resistances R_{th1} and R_{th2}.

$$T_W - T_U = \Delta T_W = (R_{th1} + R_{th2}) \cdot P_J$$

Here, thermal resistance R_{th1} relates to the heat transfer between the winding and the stator (magnetic return and magnet), whereas R_{th2} describes the heat transfer from the housing to the environment. Mounting the motor on a heat dissipating chassis noticeably lowers thermal resistance R_{th2}. The values specified in the data sheets for thermal resistances and the maximum continuous current were determined in a series of tests, in which the motor was end-mounted onto a vertical plastic plate. The modified thermal resistance R_{th2} that occurs in a particular application must be determined using original installation and ambient conditions. Thermal resistance R_{th2} on motors with metal flanges decreases by up to 80% if the motor is coupled to a good heat-conducting (e.g. metallic) retainer.

The heating runs at different rates for the winding and stator due to the different masses. After switching on the current, the winding heats up first (with time constants from several seconds to half a minute). The stator reacts much slower, with time constants ranging from 1 to 30 minutes depending on motor size. A thermal balance is gradually established. The temperature difference of the winding compared to the ambient temperature can be determined with the value of the current I (or in intermittent operation with the effective value of the current $I = I_{RMS}$).

$$\Delta T_W = \frac{(R_{th1} + R_{th2}) \cdot R \cdot I_{mot}^2}{1 - \alpha_{Cu} \cdot (R_{th1} + R_{th2}) \cdot R \cdot I_{mot}^2}$$

Here, electrical resistance R must be applied at the actual ambient temperature.

Influence of temperature
An increased motor temperature affects winding resistance and magnetic characteristic values.

Winding resistance increases linearly according to the thermal resistance coefficient for copper ($\alpha_{Cu} = 0.0039$):

$$R_T = R_{25} \cdot (1 + \alpha_{Cu}(T - 25°C))$$

Example: a winding temperature of 75°C causes the winding resist- ance to increase by nearly 20%.

The magnet becomes weaker at higher temperatures. The reduction is 1 to 10% at 75°C depending on the magnet material.

The most important consequence of increased motor temperature is that the speed curve becomes steeper which reduces the stall torque. The changed stall torque can be calculated in first approximation from the voltage and increased winding resistance:

$$M_{HT} = k_M \cdot I_{AT} = k_M \cdot \frac{U_{mot}}{R_T}$$

Motor selection

The drive requirements must be defined before proceeding to motor selection.
– How fast and at which torques does the load move?
– How long do the individual load phases last?
– What accelerations take place?
– How great are the mass inertias?
Often the drive is indirect, this means that there is a mechanical transformation of the motor output power using belts, gears, screws and the like. The drive parameters, therefore, are to be calculated to the motor shaft. Additional steps for gear selection are listed below.

Furthermore, the power supply requirements need to be checked.
– Which maximum voltage is available at the motor terminals?
– Which limitations apply with regard to current?
The current and voltage of motors supplied with batteries or solar cells are very limited. In the case of control of the unit via a servo amplifier, the amplifier's maximum current is often an important limit.

Selection of motor types
The possible motor types are selected using the required torque. On the one hand, the peak torque, M_{max}, is to be taken into consideration and on the other, the effective torque M_{RMS}. Continuous operation is characterized by a single operating or load point (M_L, n_L). The motor types in question must have a nominal torque (= max. continuous torque) M_N that is greater than load torque M_L.

$$M_N > M_L$$

In operating cycles, such as start/stop operation, the motor's nominal torque must be greater than the effective load torque (RMS). This prevents the motor from overheating.

$$M_N > M_{RMS}$$

The stall torque of the selected motor should usually exceed the emerging load peak torque.

$$M_H > M_{max}$$

Selection of the winding: electric requirement
In selecting the winding, it must be ensured that the voltage applied directly to the motor is sufficient for attaining the required speed in all operating points.

Uncontrolled operation
In applications with only one operating point, this is often achieved with a fixed voltage U. A winding is sought with a speed-torque line that passes through the operating point at the specified voltage. The calculation uses the fact that all motors of a type feature practically the same speed-torque gradient. A target no load speed $n_{0,theor}$ is calculated from operating point (n_L, M_L).

$$n_{0,\,theor} = n_L + \frac{\Delta n}{\Delta M} M_L$$

This target no load speed must be achieved with the existing voltage U, which defines the target speed constant.

$$k_{n,\,theor} = \frac{n_{0,\,theor}}{U_{mot}}$$

Those windings whose kn is as close to $k_{n,\,theor}$ as possible, will approximate the operating point the best at the specified voltage. A somewhat larger speed constant results in a somewhat higher speed, a smaller speed constant results in a lower one. The variation of the voltage adjusts the speed to the required value, a principle that servo amplifiers also use.

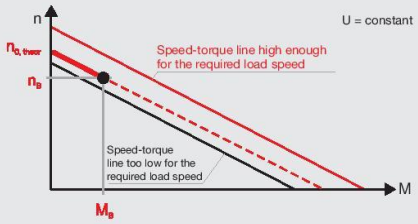
The motor current I_{mot} is calculated using the torque constant k_M of the selected winding and the load torque M_L.

$$I_{mot} = \frac{M_L}{k_M}$$

Advices for evaluating the requirements:
Often the load points (especially the torque) are not known or are difficult to determine. In such cases you can operate your device with a measuring motor roughly estimated according to size and power. Vary the voltage until the desired operating points and motion sequences have been achieved. Measure the voltage and current flow. Using these specifications and the part number of the measuring motor, our engineers can often specify the suitable motor for your application.

Additional optimization criteria are, for example:
– Mass to be accelerated (type, mass inertia)
– Type of operation (continuous, intermittent, reversing)
– Ambient conditions (temperature, humidity, medium)
– Power supply, battery

When selecting the motor type, other constraints also play a major role:
– What maximum length should the drive unit have, including gear and encoder?
– What diameter?
– What service life is expected from the motor and which commutation system should be used?
– Precious metal commutation for continuous operation at low currents (rule of thumb for longest service life: up to approx. 50% of I_N).
– Graphite commutation for high continuous currents (rule of thumb: 50% to approx. 75% of I_N) and frequent current peaks (start/stop operation, reversing operation).
– Electronic commutation for highest speeds and longest service life.
– How great are the forces on the shaft, do ball bearings have to be used or are less expensive sintered bearings sufficient?



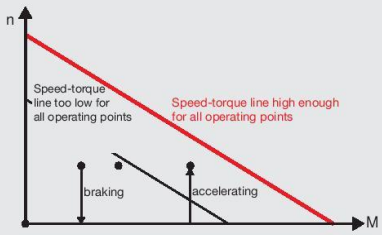
maxon motor

Regulated servo drives

In operating cycles, all operating points must lie beneath the curve at a maximum voltage U_{max}. Mathematically, this means that the following must apply for all operating points (n_L, M_L):

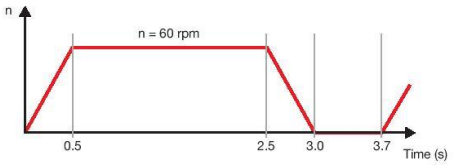
$$k_n \cdot U_{max} = n_0 > n_L + \frac{\Delta n}{\Delta M} M_L$$

When using servo amplifiers, a voltage drop occurs at the power stage, so that the effective voltage applied to the motor is lower. This must be taken into consideration when determining the maximum supply voltage U_{max}. It is recommended that a regulating reserve of some 20% be included, so that regulation is even ensured with an unfavorable tolerance situation of motor, load, amplifier and supply voltage. Finally, the average current load and peak current are calculated ensuring that the servo amplifier used can deliver these currents. In some cases, a higher resistance winding must be selected, so that the currents are lower. However, the required voltage is then increased.



Example for motor/gear selection

A drive should move cyclically according to the following speed diagram.



The inertia of the load to be accelerated J_L is 140 000 gcm². The constant coefficient is approximately 300 mNm. The 4-quadrant operation allows controlled and dynamic motor operation and brake operation in two directions of rotation (all 4 quadrants). The power supply unit delivers max. 3 A and 24 V.

Calculation of load data

The torque required for acceleration and braking are calculated as follows (motor and gearhead inertia omitted):

$$M_a = J_L \cdot \frac{\pi}{30} \frac{\Delta n}{\Delta t} = 0.014 \cdot \frac{\pi}{30} \cdot \frac{60}{0.5} = 0.176 \text{ Nm} = 176 \text{ mNm}$$

Together with the friction torque, the following torques result for the different phases of motion.

– Acceleration phase	(duration 0.5 s)	476 mNm
– Constant speed	(duration 2 s)	300 mNm
– Braking (friction brakes with 300 mNm)	(duration 0.5 s)	124 mNm
– Standstill	(duration 0.7 s)	0 mNm

Peak torque occurs during acceleration.
The RMS determined torque of the entire operating cycle is

$$M_{RMS} = \sqrt{\frac{t_1 \cdot M^2_1 + t_2 \cdot M^2_2 + t_3 \cdot M^2_3 + t_4 \cdot M^2_4}{t_{tot}}}$$

$$= \sqrt{\frac{0.5 \cdot 476^2 + 2 \cdot 300^2 + 0.5 \cdot 124^2 + 0.7 \cdot 0}{3.7}} \approx 285 \text{ mNm}$$

The maximum speed (60 rpm) occurs at the end of the acceleration phase at maximum torque (463 mNm). Thus, the peak mechanical power is:

$$P_{max} = M_{max} \cdot \frac{\pi}{30} n_{max} = 0.476 \cdot \frac{\pi}{30} \cdot 60 \approx 3 \text{ W}$$

Physical variables and their units

		SI	Catalog
i	Gear reduction*		
I_{mot}	Motor current	A	A, mA
I_A	Stall current*	A	A, mA
I_0	No load current*	A	mA
I_{RMS}	RMS determined current	A	A, mA
I_N	Nominal current*	A	A, mA
J_R	Moment of inertia of the rotor*	kgm²	gcm²
J_L	Moment of inertia of the load	kgm²	gcm²
k_M	Torque constant*	Nm/A	mNm/A
k_n	Speed constant*		rpm/V
M	(Motor) torque	Nm	mNm
M_L	Load torque	Nm	mNm
M_H	Stall torque*	Nm	mNm
M_{mot}	Motor torque	Nm	mNm
M_R	Moment of friction	Nm	mNm
M_{RMS}	RMS determined torque	Nm	mNm
M_N	Nominal torque	Nm	mNm
$M_{N,G}$	Max. torque of gear*	Nm	Nm
n	Speed		rpm
n_L	Operating speed of the load		rpm
n_{max}	Limit speed of motor*		rpm
$n_{max,G}$	Limit speed of gear*		rpm
n_{mot}	Motor speed		rpm
n_0	No load speed*		rpm
P_{el}	Electrical power	W	W
P_J	Joule power loss	W	W
P_{mech}	Mechanical power	W	W
R	Terminal resistance	Ω	Ω
R_{25}	Resistance at 25°C*	Ω	Ω
R_T	Resistance at temperature T	Ω	Ω
R_{th1}	Heat resistance winding housing*		K/W
R_{th2}	Heat resistance housing/air*		K/W
t	Time	s	s
T	Temperature	K	°C
T_{max}	Max. winding temperature*	K	°C
T_U	Ambient temperature	K	°C
T_W	Winding temperature	K	°C
U_{mot}	Motor voltage	V	V
U_{ind}	Induced voltage (EMF)	V	V
U_{max}	Max. supplied voltage	V	V
U_N	Nominal voltage*	V	V
α_{Cu}	Resistance coefficient of Cu	= 0.0039	
α_{max}	Max. angle acceleration	rad/s²	
$\Delta n/\Delta M$	Curve gradient*	rpm/mNm	
ΔT_W	Temperature difference winding/ambient	K	K
Δt	Run up time	s	ms
η	(Motor) efficiency		%
η_G	(Gear) efficiency*		%
η_{max}	Max. efficiency*		%
τ_m	Mechanical time constant*	s	ms
τ_S	Therm. time constant of the motor*	s	s
τ_W	Therm. time constant of the winding*	s	s

(*Specified in the motor or gear data)

Gear selection
A gear is required with a maximum continuous torque of at least 0.28 Nm and an intermittent torque of at least 0.47 Nm. This requirement is fulfilled, for example, by a planetary gear with 22 mm diameter (metal version GB 22 A).
The recommended input speed of 6000 rpm allows a maximum reduction of:

$$i_{max} = \frac{n_{max,G}}{n_B} = \frac{6000}{60} = 100:1$$

We select the three-stage gear with the next smallst reduction of 84 : 1 (stock program). Efficiency is max. 59%.

Motor type selection
Speed and torque are calculated to the motor shaft

$$n_{mot} = i \cdot n_L = 84 \cdot 60 = 5040 \text{ rpm}$$

$$M_{mot,\,RMS} = \frac{M_{RMS}}{i \cdot \eta} = \frac{285}{84 \cdot 0.59} \approx 5.8 \text{ mNm}$$

$$M_{mot,\,max} = \frac{M_{max}}{i \cdot \eta} = \frac{476}{84 \cdot 0.59} \approx 9.6 \text{ mNm}$$

The possible motors, which match the selected gears in accordance with the maxon modular system, are summarized *in the table opposite*. The table contains only DC motors with graphite commutation, which are better suited for start-stop operation, as well as brushless EC motors.

Selection falls on an A-max 22, 6 W, which demonstrates a sufficiently high continuous torque. The motor should have a torque reserve so that it can even function with a somewhat unfavorable gear efficiency. The additional torque requirement during acceleration can easily be delivered by the motor. The temporary peak torque is not even twice as high as the continuous torque of the motor.

Selection of the winding
The motor type A-max 22, 6 W has an average speed-torque gradient of some 450 rpm/mNm. However, it should be noted that the two lowest resistance windings have a somewhat steeper gradient. The desired no load speed is calculated as follows:

$$n_{0,\,theor} = n_{mot} + \frac{\Delta n}{\Delta M} \cdot M_{max} = 5040 + 450 \cdot 9.6 = 9360 \text{ rpm}$$

The extreme operating point should of course be used in the calculation (max. speed and max. torque), since the speed-torque line of the winding must run above all operating points in the speed / torque diagram.
This target no load speed must be achieved with the maximum voltage U = 24 V supplied by the control (ESCON 36/2), which defines the minimum target speed constant $k_{n,\,theor}$ of the motor.

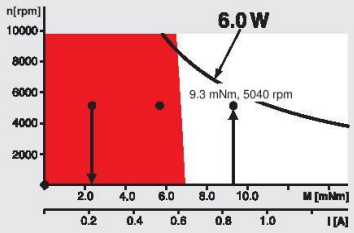
$$k_{n,\,theor} = \frac{n_{0,\,theor}}{U_{mot}} = \frac{9360}{24} = 390 \frac{\text{rpm}}{V}$$

According to the calculations, the selection of the motor is 110163, which with its speed constant of 558 rpm/V has a speed control reserve of over 20%. This means that unfavorable tolerances are not a problem. The higher speed constant of the winding compared to the calculated value means that the motor runs faster at 24 V than required, which can be compensated with the controller. This motor also has a second shaft end for mounting an encoder.
The torque constant of this winding is 17.1 mNm/A. Therefore the maximum torque corresponds to a peak current of:

$$I_{max} = \frac{M_{max}}{k_M} + I_0 = \frac{9.6}{17.1} + 0.029 = 0.6\,A$$

This current is smaller than the maximum current (4 A) of the controller and the power supply unit (3 A).

Thus, a gear motor has been found that fulfils the requirements (torque and speed) and can be operated by the controller provided.



Motor	M_N	Suitability
A-max 22, 6 W	≈ 6.9 mNm	good
A-max 19, 2.5 W	≈ 3.8 mNm	too weak
RE-max 21, 6 W	≈ 6.8 mNm	good
EC 16, 30 W	≈ 8.5 mNm	good
EC 16, 60 W	≈ 17 mNm	too strong
EC 20 flat, 3 W	≈ 3-4 mNm	too weak
EC 20 flat, 5 W	≈ 7.5 mNm	good
EC 20 flat, 5 W, iE.	≈ 7.5 mNm	good, possible alternative with integrated speed controller, no ESCON control necessary

Tech Support: services@elecfreaks.com

Ultrasonic Ranging Module HC - SR04

Product features:

Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of work:
(1) Using IO trigger for at least 10us high level signal,
(2) The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.
(3) IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic to returning.
Test distance = (high level time×velocity of sound (340M/S) / 2,

Wire connecting direct as following:

- 5V Supply
- Trigger Pulse Input
- Echo Pulse Output
- 0V Ground

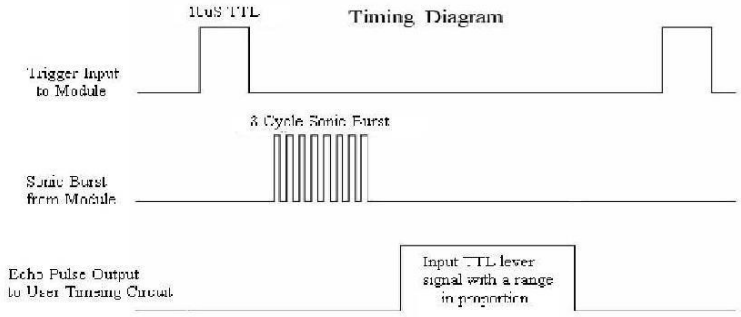
Electric Parameter

Working Voltage	**DC 5 V**
Working Current	**15mA**
Working Frequency	**40Hz**
Max Range	**4m**
Min Range	**2cm**
MeasuringAngle	**15 degree**
Trigger Input Signal	**10uS TTL pulse**
Echo Output Signal	**Input TTL lever signal and the range in proportion**
Dimension	**45*20*15mm**

Vcc Trig Echo GND

Timing diagram

The Timing diagram is shown below. You only need to supply a short 10uS pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion .You can calculate the range through the time interval between sending trigger signal and receiving echo signal. Formula: uS / 58 = centimeters or uS / 148 =inch; or: the range = high level time * velocity (340M/S) / 2; we suggest to use over 60ms measurement cycle, in order to prevent trigger signal to the echo signal.



Appendix 11

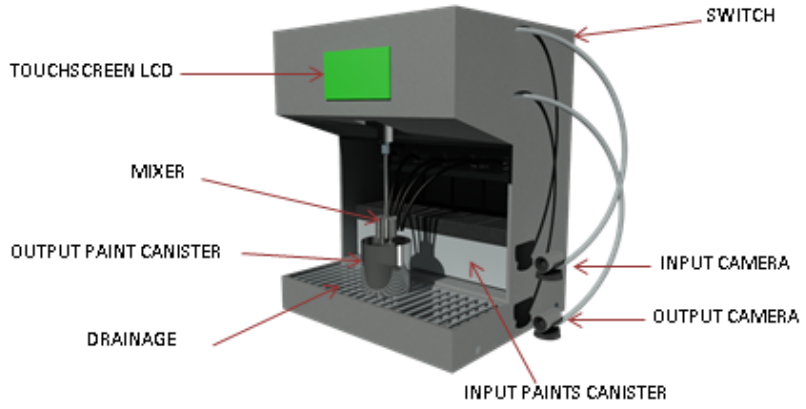
COST-BENEFIT ANALYSIS

	1 MONTH (20 days)		6 MONTHS (120 days)	
COSTS				
Developmental Costs	74,375		74,375	
Labor Costs (minimum wage)	10,740		64,440	
Machine Maintenance (minimum wage)	10,740		64,440	
TOTAL COSTS	**85,115**		**203,255**	
BENEFITS	**Manual Mixing**	**Automated Mixing**	**Manual Mixing**	**Automated Mixing**
Mixing time per 1L	10 minutes*	4 minutes**	10 minutes*	4 minutes**
Number of Liters produced per day (8 hours work per day)	48L	120L	48L	120L
Number of Liters that can be produced per day	40L^	100L^	40L^	100L^
Average amount of 1L mixed paint	256.67		256.67	
Amount of mixed paint per day	10,266.80	25,667.00	10,266.80	25,667.00
Total amount per period of days	205336	513340	1232016	3080040
TOTAL BENEFITS (Automated-Manual)	**308004**		**1848024**	
NET BENEFIT (Total Benefit-Total Cost)	**222,889**		**1644769**	
RETURN OF INVESTMENT (Net Benefit / Investment Cost) x 100%	**262%**		**809%**	

*based on the data gathered from the paint experts

**based on the data gathered from the actual prototype

^considering preparation time and cleaning time



Get Started

Turning Power On

1. Plug it in to 220 v outlet.
2. Turn on the switch.
3. Click the application on the screen to start.

Scan Color

Scanning the sample color.

1. Place camera on top of the sample color.
2. Adjust the lighting of the camera based on your own desire.
3. Click the capture button on the screen.

Mix

Mixing Output Paint

1. Click the mix button to start dispensing paint.
2. Wait until the mixing process ends.
3. Place the camera on top of the mixed paint about 2 cm apart and check the accuracy between sample color and mixed paint.
4. Click mix again until you are satisfied with the result.
5. Transfer mixed paint in a clean canister.

Clean

Cleaning The Tubes

1. Carefully remove the input paint canister from the machine.
2. Put the other canister to the machine and fill it with clean water.
3. Click the clean tubes button to start.
4. Wait until the cleaning process ends.
5. Remove water from the output canister.

Refill

Refilling Input Paints

1. Remove lock and open the lid of the input paints canister carefully.
2. Pour paint in the canister corresponding to its color.
3. Mix paint manually using a stick or a stirring rod.
4. Close the lid and lock it back to the canister.

Appendix 13

RESUME

CHARLENE MAE ESTEBAN

#62 G. Sanciangco St., Brgy. Sto. Niño, Quezon City
(+63) 927-387-3155
cmdgesteban@gmail.com

TECHNICAL SKILLS

SOFTWARE

- Web Development using PHP, Javascript, CSS, HTML, and Bootstrap
- Programming Languages (C++, Assembly, Java, and VBA)
- Database Management (MySQL and Microsoft SQL Server)
- Graphic Design (Adobe Photoshop, AutoCAD and Unity)
- Arduino Programming

HARDWARE

- PC Troubleshooting
- PC Networking
- Circuit Design

TRAININGS AND SEMINARS

- **Intern, SmartQ Systems Co. Inc**
  April - June 2018
- **Intern, Fujitsu Philippines**
  May - July 2019
- **Computer Research and Engineering Symposium (CoRES)**
  January 2016 - 2018 (Participant)
  February 2019 (Organizer)
- **Annual Research Awards**
  February 2018
- **PUP Software Freedon Day 2016**

EDUCATION

Tertiary

Polytechnic Universtiy of the Philippines
Anonas St., Sta. Mesa, Manila
Bachelor of Science in Computer Engineering
A.Y. 2015 - Present

Secondary

Carlos L. ALbert High School
Brixton Hill St., Santol, Quezon City
A.Y. 2011 - 2015

ENRICO CAMILO GOMEZ JR.

57 Maria Leni St. Hulong Duhat, Malabon City
(+63) 956-780-6303
enricocamilogomezjr@gmail.com

TECHNICAL SKILLS

SOFTWARE
- Graphic Design (AutoCAD 2D and 3D)
- C++ Programming
- Assembly Language Programming
- Arduino Programming
- Web Development (HTML, CSS, and Javascript)

HARDWARE
- Electronics Circuits Design
- Basic Computer Troubleshooting
- PC Networking

TRAININGS AND SEMINARS

- **Intern, Globaltronics Inc.**
  May - July 2019
- **Intern, Amameco Industrial Resources Corpotation**
  April - June 2018
- **Computer Research and Engineering Symposium (CoRES)**
  January 2016 - 2018 (Participant)
  February 2019 (Organizer)
- **Annual Research Awards**
  February 2018
- **PUP Software Freedon Day 2016**

EDUCATION

Tertiary
Polytechnic Universtiy of the Philippines
Anonas St., Sta. Mesa, Manila
Bachelor of Science in Computer Engineering
A.Y. 2015 - Present

Secondary
Malabon National High School
Malabon City, Metro Manila
A.Y. 2011 - 2015

JUSTIN EARL GUEVARRA

P1 B16 L26 Southville 3, Poblacion, Muntinlupa City
(+63) 915-167-7129
eewan3244@gmail.com

TECHNICAL SKILLS

SOFTWARE

- Programming Languages (C++, Assembly, Java, PHP, and Python)
- Database Management (MySQL and Microsoft SQL Server)
- Graphic Design (Adobe Photoshop, AutoCAD and Unity)
- Arduino Programming
- HTML, CSS, and Javascript

HARDWARE

- Electronics Circuits Design
- Raspberry Pi
- PC Troubleshooting

TRAININGS AND SEMINARS

- **Intern, Radenta Technologies Inc.**
  April - June 2018
- **Intern, Acadsoc Ltd.**
  April - June 2019
- **Computer Research and Engineering Symposium (CoRES)**
  January 2016 - 2018 (Participant)
  February 2019 (Organizer)
- **Annual Research Awards**
  February 2018
- **PUP Software Freedon Day 2016**

EDUCATION

Tertiary
Polytechnic Universtiy of the Philippines
Anonas St., Sta. Mesa, Manila
Bachelor of Science in Computer Engineering
A.Y. 2015 - Present

Secondary
Muntinlupa National High School
Annex
A.Y. 2011 - 2015

AMI MORITA

Unit 8 #46 Daliva St., Karuhatan, Valenzuela City
(+63) 956-804-7311
amimorita17@gmail.com

TECHNICAL SKILLS

SOFTWARE
- Programming Languages (C/C++, Assembly, Java and PHP)
- Oriented in HTML, CSS and Javascript
- Arduino Programming
- Microsoft Office Applications
- Graphic Design (Adobe Photoshop, AutoCAD and Unity)

HARDWARE
- Basic Operation
- Basic Circuitry

TRAININGS AND SEMINARS

- **Intern, Emerson Electric Asia Ltd. ROHQ 2018**
  May - July 2018
- **Intern, Manulife Business Processing Services**
  April - June 2019
- **Computer Research and Engineering Symposium (CoRES)**
  January 2016 - 2018 (Participant)
  February 2019 (Organizer)
- **College of Enginering Managers of Information Technology Summer Training 2015**
- **PUP Software Freedon Day 2016**

EDUCATION

Tertiary
Polytechnic Universtiy of the Philippines
Anonas St., Sta. Mesa, Manila
Bachelor of Science in Computer Engineering
A.Y. 2015 - Present

Secondary
Pugad Lawin High School
Bahay Toro, Quezon City
A.Y. 2011 - 2015