UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Study, Design and Implementation of a Robotic Arm

Diploma Thesis

IFIGENEIA SKALIDI

Supervisor

Panagiota Tsompanopoulou

Associate Professor

Volos 2021

University of Thessaly
Faculty of Engineering
Department of Electrical & Computer Engineering

Study, Design and Implementation of a Robotic Arm

Diploma Thesis

IFIGENEIA SKALIDI

Supervising Committee

Supervisor	Co-supervisor	Co-supervisor
Tsompanopoulou	Stamoulis	Potamianos
Panagiota	Georgios	Gerasimos
Associate Professor	Professor	Associate Professor

Volos 2021

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Μελέτη, Σχεδιασμός και Υλοποίηση Ρομποτικού Βραχίονα

Διπλωματική Εργασία

ΙΦΙΓΕΝΕΙΑ ΣΚΑΛΙΔΗ

Επιβλέπουσα

Παναγιώτα Τσομπανοπούλου

Αναπληρώτρια Καθηγήτρια

Βόλος 2021

Εγκρίνεται από την Επιτροπή Εξέτασης:

Επιβλέπουσα	**Τσομπανοπούλου Παναγιώτα** Αναπληρώτρια Καθηγήτρια, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας
Μέλος	**Σταμούλης Γεώργιος** Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας
Μέλος	**Ποταμιάνος Γεράσιμος** Αναπληρωτής Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Ημερομηνία έγκρισης: 16/07/21

To my mother and my brother

for their unconditional love and support

Acknowledgements

I would first like to thank my supervisor, Professor Tsompanopoulou Panagiota. Her unfaltering guidance and support throughout the whole process was invaluable. Thank you for your patience, for elevating my work and raising my spirit.

A debt of gratitude is also owed to my mother, my brother and Vlasis. Without them this work would not be possible.

Finally, I would like to thank my friends for supporting me in all my efforts.

University of Thessaly

Faculty of Engineering

Department of Electrical & Computer Engineering

The author of this thesis assures that any help he has had for its preparation is fully acknowledged and mentioned in this thesis. He also assures that he has referenced any sources from which he used data, ideas or words, whether these are included in the text verbatim, or paraphrased.

«Being fully aware of the consequences of copyright law, I expressly state that this dissertation, as well as the electronic files and source codes developed or modified in the course of this work, are solely the product of my personal work and do not infringe any rights. intellectual property, personality and personal data of third parties, does not contain works / contributions of third parties for which permission of the creators / beneficiaries is required and is not a product of partial or complete copy, and the sources used are limited to bibliographic references and only meet the rules of scientific quotation mark. The points where I have used ideas, text, files and / or sources of other authors, are clearly mentioned in the text with the appropriate citation and the relevant report is included in the bibliographic references section with full description. I undertake in full, individually and personally, all the legal and administrative consequences that may arise in the event that it is proven, over time, that this work or part of it does not belong to me because it is a product of plagiarism. »

Ifigeneia Skalidi

July 2021

ABSTRACT

There is a substantial ongoing integration of robots in the manufacturing process. Most of these robots have a rigid movement repertoire, making their upkeep and managing challenging, both in economical and manning terms. The market's constant demand for better, newer and more intricate products can make these robots swiftly obsolete, creating more waste and unemployment since the need of more specialized workers increases.

This thesis focuses on researching, designing and implementing a small scale, low cost and environmentally friendly robotic arm. The main goal behind its construction is its flexibility; By creating adaptable and easily tinkered modes of operation, this robotic arm aims to contribute to industrial construction lines, ameliorating the working conditions of the personnel and facilitating their training, without requiring a lot of proficiency in programming or mechanical skills.

KEYWORDS

Robotic Arm, Computer Vision, OpenCV, Micro Servos, 3D Printing, Raspberry Pi, Fusion 360, Python

ΠΕΡΙΛΗΨΗ

Υπάρχει μια τεράστια συνεχής ενσωμάτωση ρομπότ στη κατασκευαστική διαδικασία. Τα περισσότερα από αυτά τα ρομπότ έχουν ένα άκαμπτο ρεπερτόριο κινήσεων, καθιστώντας τη συντήρηση και την διαχείρισή τους δύσκολη, τόσο από οικονομική όσο και από επανδρωτική άποψη. Η συνεχής ζήτηση της αγοράς για καλύτερα, νεότερα και πιο περίπλοκα προϊόντα μπορεί να κάνει αυτά τα ρομπότ παρωχημένα, δημιουργώντας περισσότερα απόβλητα και ανεργία, καθώς αυξάνεται η ανάγκη πιο εξειδικευμένων εργαζομένων για τον χειρισμό τους.

Η διατριβή επικεντρώνεται στην έρευνα, το σχεδιασμό και την εφαρμογή ενός ρομποτικού βραχίονα μικρής κλίμακας, χαμηλού κόστους και φιλικού προς το περιβάλλον. Ο κύριος στόχος κατά την κατασκευή του είναι η ευελιξία του. Δημιουργώντας προσαρμόσιμους και εύκολα τροποποιημένους τρόπους λειτουργίας, αυτός ο ρομποτικός βραχίονας ευελπιστεί να συνεισφέρει στις βιομηχανικές γραμμές κατασκευής, βελτιώνοντας τις συνθήκες εργασίας του προσωπικού και διευκολύνοντας την εκπαίδευσή τους, χωρίς να απαιτείται μεγάλη επάρκεια στον προγραμματισμό ή στις μηχανικές δεξιότητες.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ

Ρομποτικός Βραχίονας, Οπτική Υπολογιστή, OpenCV, Micro Servos, 3D Εκτύπωση, Raspberry Pi, Fusion 360, Python

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

The technology-driven world is filled with promise and challenges most of which are created by the incessant developments in machine learning, robotics and artificial intelligence [1]. Science and Technology have reached milestones previously thought of only in science fiction movies, milestones such as creating self-driven cars that will soon be a common sighting in the streets, microscopic robots which can be injected into the human body and be remotely operated [2], even hiring algorithms which promise employers more efficient use of their recruitment budgets. All these new manifestations of powerful forms of automation that increase productivity and our quality of life, bring forth numerous challenges that have already sparked public concern.

Their usage has substituted a plethora of human activity and labor in the last decade. This substitution was needed due to the increasing cost of human labor (Figure 1.1.1) [3] in combination with the widespread of automation, the decrease of the robotic systems cost and the demand to produce more with less [4]. All the aforementioned have guided the industry and academia's focus towards a deeper integration of industrial robots in the assembly lines of products that were traditionally assembled using more specific and dedicated automation machines or manual labor.

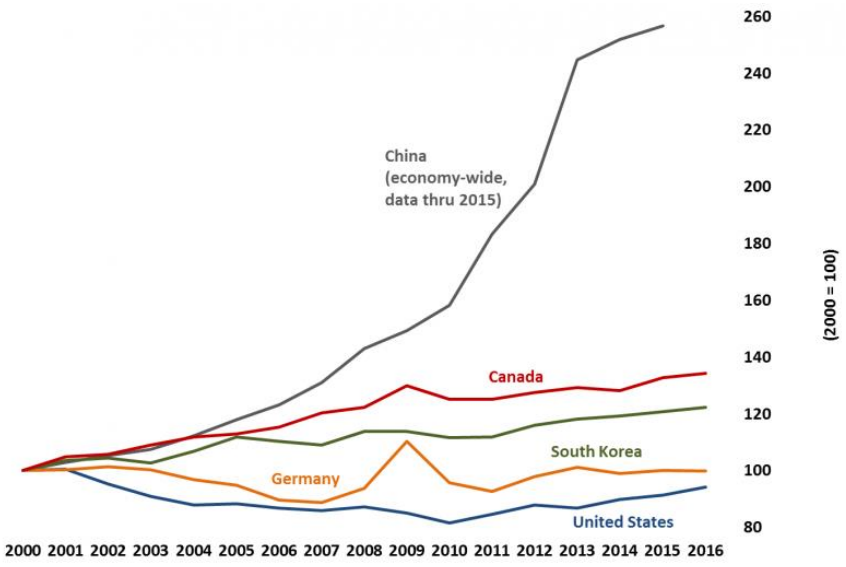

Figure 1.1.1 Indexed Unit Labor Costs in the Manufacturing Sector of Selected Countries, 2000-2016 , [3]

There is a current tendency in the manufacturing process of reducing the production lot sizes which is caused by multiple factors. The most dominant ones are the expanding immersion of Lean approaches [5] and the never-ending customization of products which tend to complicate all the variants in a production line. In a mass production line, the dominant equipment is usually a rarely reconfigured, fixed automation, therefore mass-customization is challenging especially when the production rates and the reconfiguration efficiency is taken into account. The lack of flexible equipment, user-friendly programming and ability for more intricate operations are some of the drawbacks which characterize a plethora of the current industrial robots. Drawbacks which prohibit their maintenance and operation by a non-expert user.

1.1 Motivation

The increasing complexity and intricacy which consumer products demand nowadays have surpassed the more rigid production line of an automobile or a food packaging item and manufacturers require more flexible tools in order to accommodate and keep up with what their customers want. The design period of the product is also getting shorter while the consumers' tastes and needs change rapidly and can even occur in a single production run. Since the existing production lines are designed to produce countless products over a significantly extended period of time and the robots which are utilized are programmed in such a way that they only perform this one, repetitive task and are even unable of the smallest adjustment, the need of a low cost, safe and more user-friendly, flexible robot is paramount, always keeping in mind the uncertainty issues of the existing consumer market.

1.2 Problem definition

This thesis' primary objective is to study, design and implement a low cost, small scale robotic arm which can be later integrated in any mass production line of small-sized lightweight parts, all the while maintaining and easily tinkering the flexibility of its movement repertoire, programming capabilities and user-friendliness.

1.3 Thesis structure

In Chapter 2 some basic robot history as well as the current types of industrial robots and methods are presented.

In Chapter 3 the hardware and techniques which were utilized for the robot's implementation are discussed.

Chapter 4 explores the 3D printing, materials and modelling process which were employed.

In Chapter 5 the software which was utilized for the programming and the modelling part is presented.

Chapter 6 spotlights the color model theory and the manner of which the image processing for the object detection was designed.

A brief description of the robot's action repertoire is presented in Chapter 7.

Finally, Chapter 8 comprises of the conclusions and the suggestions for further improvement of this thesis' robotic arm implementation.

CHAPTER 2

ROBOTS AND INDUSTRIAL AUTOMATION

2.1 Robots in History

The term “robot” was first introduced by the Czech writer Karel Čapek [6] in his play, R.U.R. in 1920 and was referred to as an artificial automata. Although, the word “robot” can be attributed first to the Slavic language since it contains the word “robota”, meaning forced labor, Karel’s usage of it hits closer to its current meaning because it suggested the “technological creation of artificial human bodies without souls” [7]. Defining the word “robot” is quite difficult because of its multifaceted aspects. According to the Robot Institute of America (1979), a robot can be defined as a “reprogrammable, multifunctional manipulator designed to move material, parts, tools, or specialized devices through various programmed motions for the performance of a variety of task.".

The earliest conception of a robot dates back to the 3000 B.C in Egypt, where a water clock with human figurines that were striking bells every hour was built. Numerous robot-like creations were designed and completed since then all around the world. Nonetheless, the earliest robot that mostly resembles today’s definition of the word robot, was presented in 1950, in Louisville, Kentucky: “Unimate” [8], from “Universal Automation”, a reprogrammable manipulator and was created and patented by George C. Devol (Figure 2.1.1). Although Devol attempted to sell his product in the industry, he failed and it was in the late 1960s when Joseph Engleberger, after acquiring and modifying Devol’s patent, succeeded in selling it as an industrial robot which General Motors first integrated in its manufacturing process in 1961. For this achievement, Engleberger is considered the “Father of Robotics”.

In 1978 the first “Pick and Place” Robot, a four-axis Selective Compliance Assembly Robot Arm (SCARA) was designed by Hiroshi Makino in Japan [9]. The SCARA study group that was formed for the robot’s development had participants from companies like Fujitsu and Toshiba. Each company commercialized its own version of the SCARA robot and integrated it in their manufacturing process.

The case of Amazon's acquisition of Kiva Systems [10] highlights the impact that robots have in modern industries: 775 million dollars were spent to integrate the first robot automated fulfillment system into a warehouse. This integration relieved a lot of workers from hazardous manual labor and accelerated the pick and place process.

According to the International Federation of Robotics (IFR), an estimated 2.7 million industrial robots are currently in operation worldwide [11] and the number keeps growing, noting the incessant progress and integration of robots in the manufacturing industrial scene.

Figure 2.1.1 UNIMATE, the First Industrial Robot image
(Image credit: SSPL/Getty Images)

2.2 Industrial Robots

Prior to analyzing the basic method behind the development of this thesis robotic arm, a brief description of the six main types of robots used for industrial purposes can be found below:

Articulated robots

With a range of 2 to 10 or more joints, this articulated robot design features a base which is connected to one of its twisting joints and the rest of them are linked and arranged in a "chain", supporting each other further in the chain. Each joint provides an additional degree of freedom or range of motion and their configuration closely resembles a human arm. Most of the articulated robots use 6 joints and are applicable in assembly lines, material handling, packaging and a plethora of other types of industrial lines (Figure 2.2.1).

Figure 2.2.1 6-Axis Articulated Robot (Image downloaded from https://www.roboticautomationsystems.com/6-axis-robots.html in June 2021)

Cartesian coordinate robots

A Cartesian coordinate robot (also called linear, gantry or rectilinear robot) uses the Cartesian coordinate system, working on three linear aces. Rather than rotating, its three principal axes of control are linear and move in straight lines (up and down, in and out, and side to side), making it highly flexible in its design since the user can adjust with ease the speed, precision, stroke and configuration of the robot (Figure 2.2.2). Their most frequent usage is in 3D printing, Computer Numerical Control (CNC) and pick and place machines.

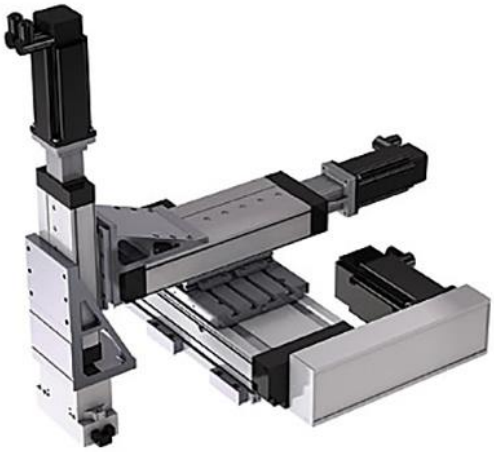

Figure 2.2.2 Rexroth 3-Axis Cartesian Robot (Image credit: Bosch Rexroth Corp.)

Cylindrical coordinate robots

Cylindrical Robots have a rotary joint at the base and at least one prismatic joint (also known as a slider, which provides a linear sliding movement between two bodies) connecting its links. The rotary joint provides a rotational motion on the joint axis, while the prismatic joints work linearly, moving in a vertical or sliding motion depending on its position (Figure 2.2.3). With a compact design, they are excellent in simple assembly lines, coating applications or machine tending.

Figure 2.2.3 SciClops Microplate Handler (Image downloaded from https://hudsonrobotics.com/microplate-handling-2/platecrane-sciclops-3/ in June 2021)

Spherical coordinate robots

These robots are also known as polar robots, spherical mobile robots, or ball-shaped robots and are mobile robots with spherical external shape. Their arm is connected to the base with the help of a twisting joint and one linear and two rotary joints connect the links. The aces create a polar coordinate system, enveloping a spherical-shaped workspace but restricting the arm within it. They can be operated as autonomous robots and are mainly used in situations where surveillance or monitoring conditions are needed. Unimate (Figure 2.1.1) is an example of a polar robot.

SCARA robots

Selective Compliance Assembly Robot Arm (SCARA) robots consist of two parallel joints with a vertically positioned rotating shaft and a horizontally moving end effector attached. They are mostly compliant in the x-y direction and quite rigid in the z direction. They excel in lateral movements and are faster moving than the Cartesian robots. SCARA robots are mostly used in bio-med applications, packaging, palletizing and machine loading (Figure 2.2.4).

Figure 2.2.4 EPSON SCARA LS10-B702S (image credit: Seiko Epson Corporation)

Delta robots

Delta robots are also known as parallel robots and consist of three arms connected with a common base. Each end effector maintains its orientation due to the use of parallelograms in the arm. They work in a dome-shape envelope and due to the precise and high-speed movement of each joint of the end effector (Figure 2.2.5), are heavily used in food, pharmaceutical, electronic and mainly in industries which heavily rely on pick and place applications.

Figure 2.2.5 Delta robot MPP3S
(Image credit: YASKAWA Europe GmbH)

2.3 Method Description: Flexible Automation

The basic concept behind the design of this thesis robotic arm is its ability to be as flexible, purposeful, low cost and ecofriendly as a robot used in an industrial line can be. All the aforementioned qualities can be summarized in Flexible Automation. Flexible automation is the ability of a robot (or a system) to be easily and quickly reconfigured for changes which may occur in product design and production. While fixed automation may be less expensive in short term, its use can, in the long run, save production costs. It has the ability to reduce the production costs while improving the quality of the product and eliminate numerous health and safety issues.

The typical features of flexible automation can be summed up in the following [12] :

- Capability to change the sequence of operations and adaptability to different product configurations. The system should be able to be reprogrammed so that the operation sequence can be manipulated.
- Flexibility which will allow the system to be suited for different products.
- Low production rates (as compared to fixed automation system).
- High investment cost (but not as high as in case of fixed automation).

The goal of this robotic arm is to be able to mirror the motions of its counterpart, learn from it and repeat the given sequence of its movements in a loop, as well as be able to pinpoint the coordinates of given shapes through a camera with the intention of mimicking its action path.

CHAPTER 3

HARDWARE PRESENTATION

3.1 Raspberry Pi 4

For the implementation of the robotic arm, a single board computer (SBC) is used, the Raspberry Pi 4 Model B [13] (Figure 3.1.1). It is a versatile and cheap option which can be utilized as the control module of the arm. Despite its small size, it is a fully featured computer with multiple connection ports and numerous Hardware on Top (HAT) that can be added directly to the board, making it convenient to use as a normal desktop computer by plugging a mouse, a keyboard, and a monitor.

Specifications:

CPU:	4× Cortex-A72 1.5 GHz
GPU:	Broadcom VideoCore VI @ 500 MHz
Memory (SDRAM):	4 GiB
USB 2.0 ports:	2
USB 3.0 ports:	2
On-board storage:	MicroSDHC slot, USB Boot Mode
Power source:	5 V via USB-C or GPIO header
Environment:	Operating temperature 0–50ºC

Standard 40-pin GPIO header (fully backwards-compatible with previous boards)

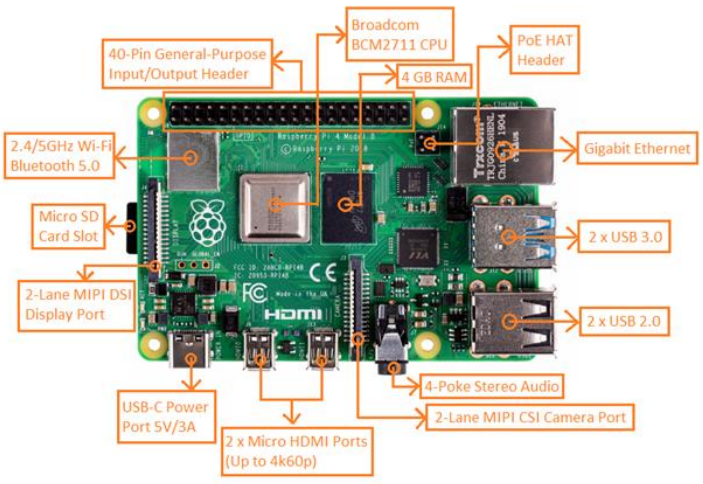

Figure 3.1.1 Raspberry Pi 4 Model B Layout
(Raspberry Pi Image Credit: Chicago Electronic Distributors)

3.2 GPIO

GPIO stands for General Purpose Input/Output. The Raspberry Pi model used in this thesis has 40 GPIO pins (Figure 3.2.1), a practical number since it allows the user to connect a plethora of microcontrollers and electronic devices. These pins are practically the physical interface between the Pi and the external environment, denoting that they are the ones responsible of offering digital input/output.

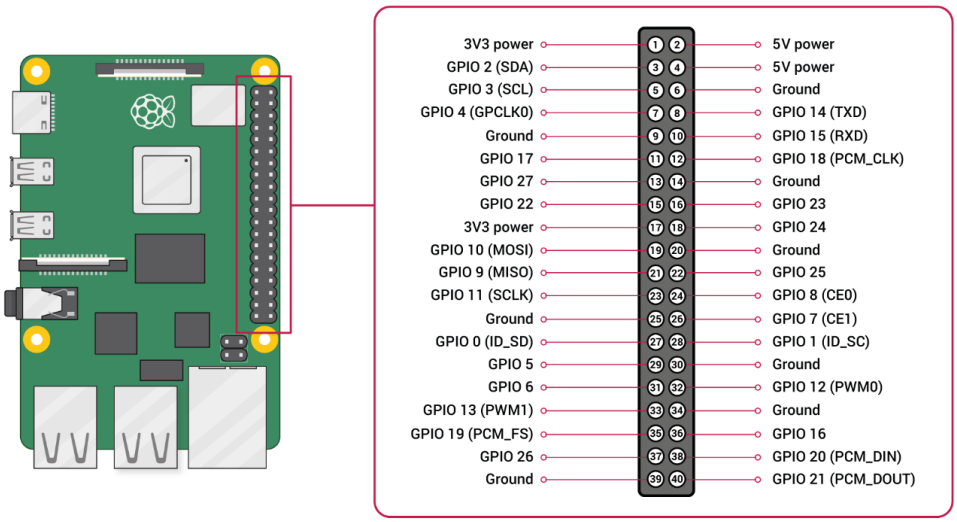

Figure 3.2.1 Raspberry Pi 4 Model B Pin Layout
(Image downloaded from https://www.programmersought.com/article/92047061453/ in June 2021)

Pin Explanation:

Voltages	The board includes two 5V pins and two 3V3 pins, as well as several ground pins (0V), which cannot be configured. The remaining pins have their outputs set to 3V3 and their inputs to 3V3-tolerant (meaning they are all general purpose 3V3 pins).
Outputs	A GPIO pin assigned as an output pin can be set to high (3V3) or low (0V).
Inputs	A GPIO pin assigned as an input pin can be read as high (3V3) or low (0V). This is facilitated by the use of internal pull-up or pull-down resistors. Pins GPIO2 and GPIO3 have fixed pull-up resistors, but for other pins this can be configured in software.

The GPIO pins can be used not only as simple input and output devices but also with a variety of alternative functions which some are available on all the pins and others on specific pins.

PWM (pulse-width modulation)	Software PWM available on all pins Hardware PWM available on GPIO12, GPIO13, GPIO18, GPIO19
SPI	SPI0 : MOSI (GPIO10); MISO (GPIO9); SCLK (GPIO11); CE0 (GPIO8), CE1 (GPIO7) SPI1 : MOSI (GPIO20); MISO (GPIO19); SCLK (GPIO21); CE0 (GPIO18); CE1 (GPIO17); CE2 (GPIO16)
I2C	Data: (GPIO2); Clock (GPIO3) EEPROM Data: (GPIO0); EEPROM Clock (GPIO1)
Serial	TX (GPIO14); RX (GPIO15)

In order to facilitate the pin connectivity since multiple micro servos, potentiometers and other hardware are going to be connected, a GPIO extension board is connected along with 40 rainbow cables, into a solderless breadboard. It works like a bridge between the Raspberry Pi GPIO pins and the breadboard.

A breadboard is a solderless device used for testing circuit designs and temporary prototyping electronics. It is consisted of a perforated block of plastic with numerous tin-plated phosphor bronze or nickel silver alloy spring clips under the perforations [14]. The electronic components are interconnected by inserting their lead (or terminals) into the breadboard's perforations and then making connections with wires where needed. The breadboards' strips of metal allow the connection of those components as demonstrated in Figure 3.2.2 : the top and bottom rows of perforations are connected horizontally and split in the middle while the remaining perforations in between are connected vertically.

Moreover, breadboards do not work with surface mount components since they have short, flat pins on their sides which are designed to be soldered onto the surface of a printed circuit board, instead of through holes.

When a dual in-line pin package (DIP) integrated circuit (IC chips such as the MCP3008) is plugged into a breadboard, it must be situated on each side of the notch which the breadboard has in the middle, otherwise it might be short-circuited. This notch provides limited airflow (cooling) to those chips.

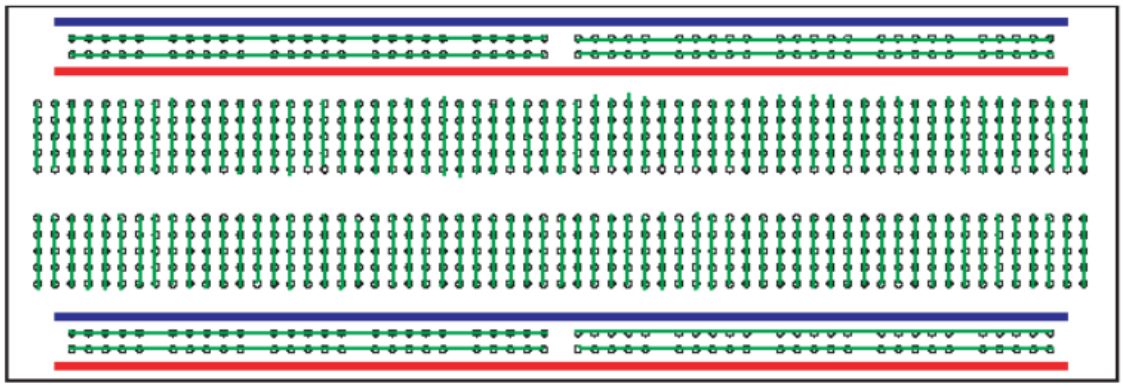

Figure 3.2.2 Breadboard Internal Connections
(Image downloaded from
https://www.researchgate.net/publication/266472001_Data_Acquisition_and_Filter_Design/figures?lo=1 in June 2021)

3.3 SPI Connectivity

Unfortunately, the Raspberry Pi does not support analog input since it lacks an integrated Analog-to-Digital Converter (ADC) device. This drawback can be remedied with the help of ADC devices which can support analog input and allow the use of analog sensors. The MCP3008 10-bit Analog-to-Digital Converter (Figure 3.3.1) was chosen for the potentiometer's analog data conversion of this project as it provides high performance with low power consumption at a cheap price. It also provides 8 input channels which is ideal for a robot with multiple degrees of freedom (Figure 3.3.4).

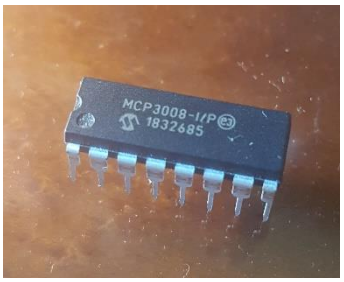

Figure 3.3.1 MCP3008

MCP3008 specifications [15]:

Max Sample Rate (ksamples/sec)	200
Typ. INL ± (LSB)	0.5
Max. Supply Current (µA)	500
Input Type	Single-ended
# of Input Channels	8
Resolution (bits)	10
Interface	SPI
Temp Range (°C)	-40 to +85°C
Input Voltage Range (V)	0 to 5.5

The most used interface between peripheral ICs (such as sensors, ADCs, shift registers and others) is the Serial Peripheral Interface (SPI) (Figure 3.3.2). It has a synchronous, serial, full-duplex master-slave-based protocol and it was designed by Motorola.

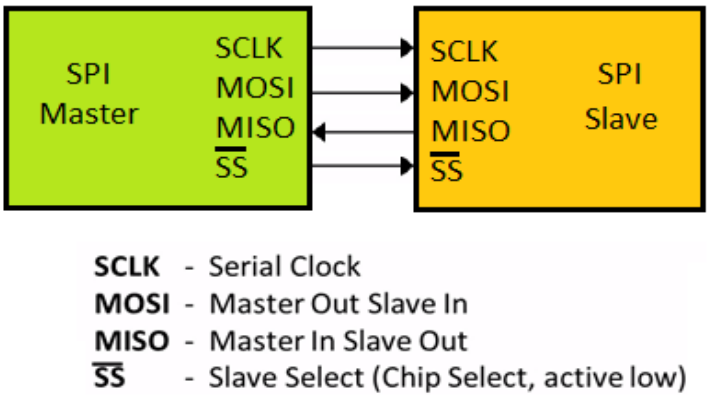

Figure 3.3.2 SPI's Protoco

There is a notion of master and slave for this interface. In general, an Integrated Circuit (IC) is chosen as the slave, the device that "obeys" the master. The master is a device that generates the clock signal dictating when the communication is going to happen, and it is usually a micro controller or a Field-Programmable Gate Array (FPGA). The data from the master or the slave is synchronized on the rising or falling clock edge. Master and slave can transmit data simultaneously. A SPI device can support higher clock frequencies compared to an I^2C interface (which is a synchronous, multi-master, multi-slave, packet switched, single-ended, serial communication bus) and can also have one master and one or several slaves. The selection of the slave is dictated by the chip select signal from the master. This is an active low signal and is pulled high when we need to disconnect the slave from the bus. When multiple slaves are used, an individual chip select signal for each of the slaves is required from the master (Figure 3.3.3).

MOSI and MISO are the data lines: MOSI transmits data from the master to the slave and MISO transmits data from the slave to the master.

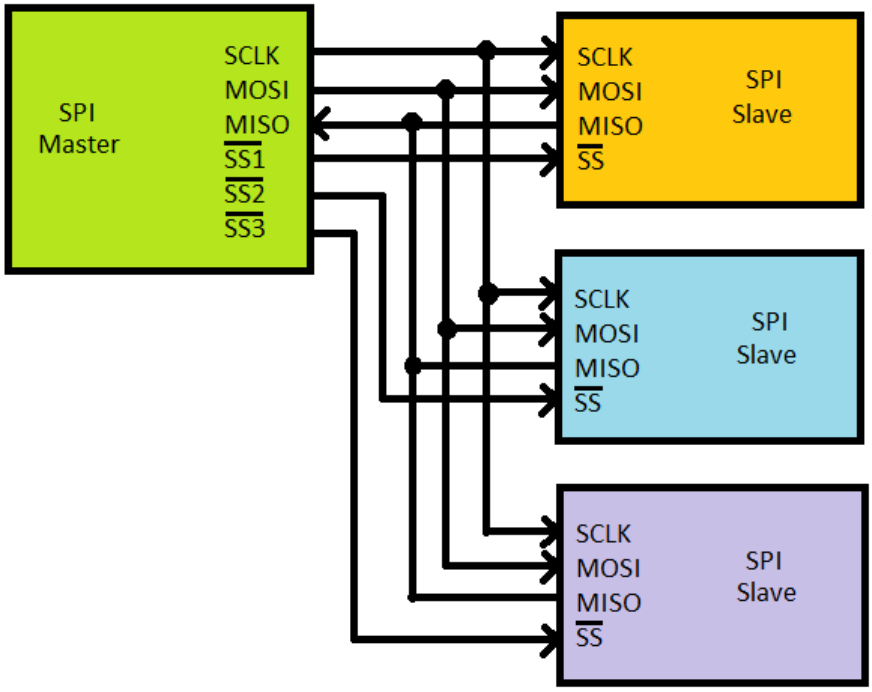

Figure 3.3.3 SPI with Multiple Slaves

For the connection of the MCP3008 to the Raspberry Pi the following wirings were needed (

Figure 3.3.5):

-MCP3008 VDD to Pi's 3.3V

-MCP3008 CLK to Pi's SCLK

-MCP3008 VREF to Pi's 3.3V

-MCP3008 AGND to Pi's GND

-MCP3008 DGND to Pi's GND

-MCP3008 DOUT to Pi's MISO

-MCP3008 DIN to Pi's MOSI

-MCP3008 CS to Pi's D5

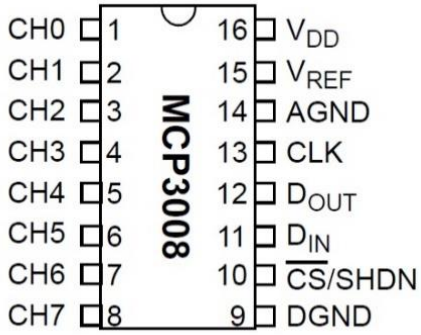

Figure 3.3.4 MCP3008 Pin Layout (Image downloaded from https://digitalsystemdesign.in/interfacing-adc-with-fpga/ in June 2021)

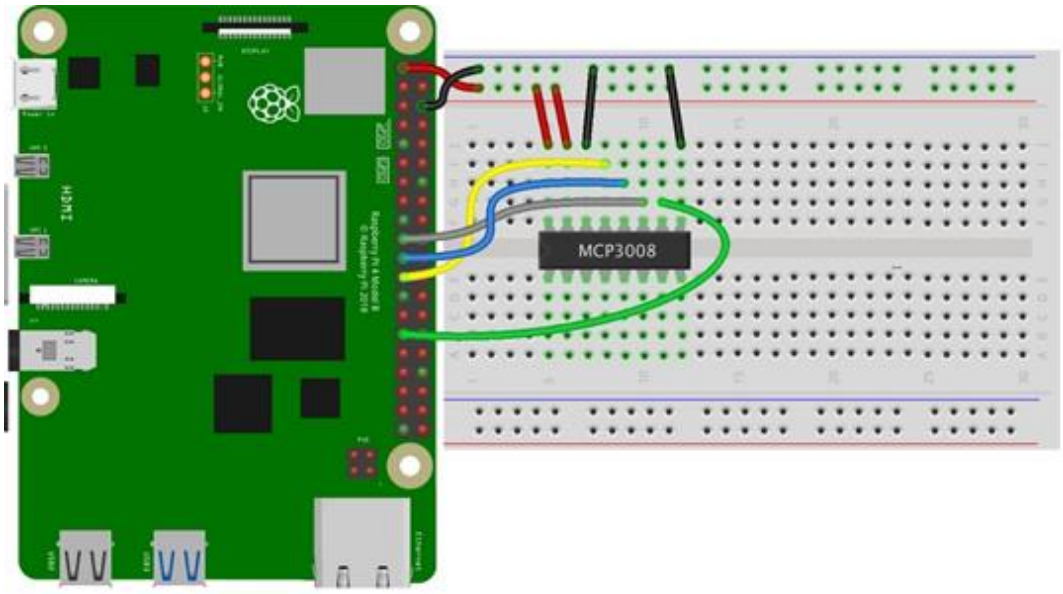

Figure 3.3.5 MCP3008 Wiring to Raspberry Pi

3.4 Potentiometers and Soldering

Potentiometers, also known as pots, are variable resistors which provide variable resistance by turning the knob situated on their head. Two kinds of pots exist: resistance (R-Ohms) and power (P-Watts) pots.

The resistance potentiometers are classified according to the current they allow: the greater the resistor value of the pot, the smaller the current which will flow (also known as Power (wattage) rating). The higher the power rating, the bigger the resistor gets. For potentiometers, the power rating is 0.3W making it convenient to use for low current circuits.

A potentiometer is basically a resistor with one variable end. The terminals 1 and 3 are fixed at the two edges of the resistive track and by measuring the resistance between them we will get the value of the potentiometer (Figure 3.4.1). The values of the potentiometers available at the market are usually 500Ω, 1K, 2K, 5K, 10K, 22K, 47K, 50K, 100K, 220K, 470K, 500K, and 1 M.

By placing the potentiometer's wiper exactly at 20% from terminal 1 and measuring the resistance between the first and the second terminal, we will get 20% of the full resistive value of the potentiometer, while measuring across the second and the third terminal will get the rest 80% of this value. Consequently, by measuring the variable resistance when turning the potentiometer's knob, we can alter the resistance and set the desired value.

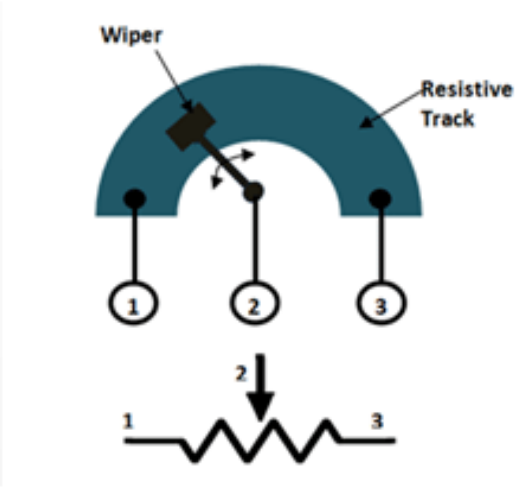

Figure 3.4.1 Potentiometer's Diagram

(Image downloaded from https://components101.com/resistors/preset-potentiometer-trimpot-pinout-datasheet in June 2021)

Applications of a potentiometer:

-Voltage and current control in electronic circuits

-Volume, tone and linearity control in radio and television receiver

-Analog input control knobs

For this project, the B10K potentiometer was selected (Figure 3.4.2). It is a resistance pot and is consisted of a 10K Ohm resistance. For the conversion of its analog input to digital, each potentiometer's middle pin is connected to each channel of our MCP3008 ADC, each left pin to the Pi's GND and right pin to the Pi's 3.3V. The arm consists of 3 degrees of freedom, consequently, 3 potentiometers will be connected to channels 0,1 and 2 of the ADC.

Figure 3.4.2 B10K Potentiometer

The B10K potentiometers' tips are not prewired. Since these potentiometers will act as the joints of the robot, their distance between them and the breadboard might need to be bigger. To achieve flexibility in the construction and movement of the arm, soldering is unavoidable for connecting their tips with wires.

Soldering

The basic concept behind soldering is to bond two things together using solder, a metal alloy, as the glue. In electronics, soldering has the added requirement that the joints need to be electrically conducted.

By using heat from an iron connected to a temperature controller, usually referred to as soldering iron (Figure 3.4.3), solder is melted onto the edges of the metals you wish to connect. When the solder cools down, it forms a strong permanent bond.

Figure 3.4.3 Soldering Iron.

There are two types of solders: leaded and lead-free solder. The melting point of the lead in solders is 180 ºC. To transfer the heat from the soldering iron into anything we wish to solder, it is recommended to set the temperature of the soldering station above this number and preferably around 400 ºC. If we are using lead-free solder, this number would jump to 450 ºC.

Most leaded solders consist of 60% tin and 40% lead. It is of paramount importance to note that lead is poisonous and good air ventilation is essential when working with it. Inside the solder, a little channel of flux is placed (Figure 3.4.4). Flux helps by stopping the oxidization process, making it easier for the solder to adhere. Solder must always be used on clean surfaces.

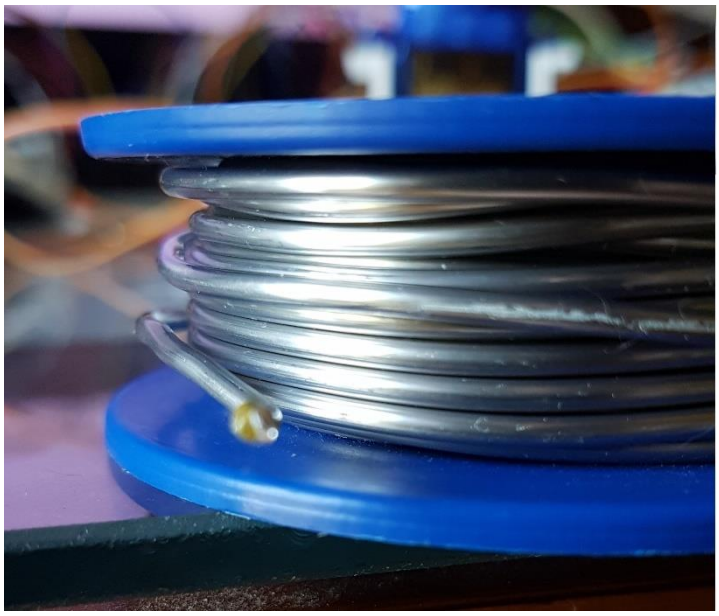

Figure 3.4.4 Solder with Flux

There are two types of flux: active and passive. Active flux is acidic and corrosive. It takes layers off the metals that you are using when you solder. If it is not thoroughly cleaned off, it keeps corroding, destroying the electronics in the process. Passive flux is rosin flux. It is not corrosive, but it is still poisonous, so handwashing is recommended.

Maintenance of soldering iron: always keep the tip of your iron clean by using a wet sponge or preferably by jabbing it into steel wool.

Lastly, heat shrinks can be used on the joints. They are little tubes of plastic which come in ranges of different colors and sizes. Their usage is to seal up any joints instead of leaving them exposed, making the joints last longer.

Procedure of Soldering:

- Strip the wire from its insulation with a wire stripper and twist the end. Make sure both ends you wish to solder are clean.
- The flux inside the solder only works at the time that it is melted so do not melt it first on the soldering tip. Use the soldering iron to get the wire hot and afterwards melt the solder onto the wire.
- Repeat the above step on the other tip you wish to solder.
- Put the heat shrink on the one side of your wire, away from the soldering iron so it does not melt prematurely.
- Place both tips together, apply heat to the area and melt solder onto it.
- Let it cool down.
- Place the heat shrink on the joint.
- Gently run heat from a lighter up and down the heat shrink. Be careful to not overheat otherwise it will be burnt.
- Let it cool down.

3.5 Micro Servos

Servo motors are small devices with an external shaft. The shaft's axis can be moved to different positions according to the coded signal which is sent to the servo. The servo will keep its position for as long as a signal is received from the input line. If the signal changes, the angle of the shaft will move accordingly. It offers precise control of angular or linear position, velocity, and acceleration.

Although they can be used in numerous practical applications such as remote-controlled airplanes and cars, they are extremely useful in robotics due to their micro mechanism, their built-in control system and the great deal of power they provide compared to their size. The energy which they consume is directly proportional to the mechanical load they receive: the lesser the load, the lesser the consumable energy.

Most of the servo motors operate from 4.8V to 6.5V, the higher the voltage, the higher the torque they can achieve, but generally, they are operated at 5V. Roughly all servo motors can rotate from 0° to 180° thanks to their gear arrangement. To achieve a rotation of a 0° to 360°, a modification of the motor will be needed. Since the servos' gears are easily susceptible to wear and tear, a servo with metal gears is preferred when heavier duties are required from the servo.

If torque is of main concern, the most commonly available one is the 2.5kg/cm torque. This means that the motor can pull a weight of 2.5kg when it is suspended at 1cm.

For this project, the SG90 Tower Pro Micro Servo was selected (Figure 3.5.1).

Specifications:

Dimension: 22mm x 11.5mm x 22.5mm

Operating speed: 0.12second/ 60degree (4.8V no load)

Stall Torque (4.8V): 17.5oz /in (1kg/cm)

Temperature range: -30 to +60

Dead band width: 7usec

Operating voltage: 3.0V~7.2V

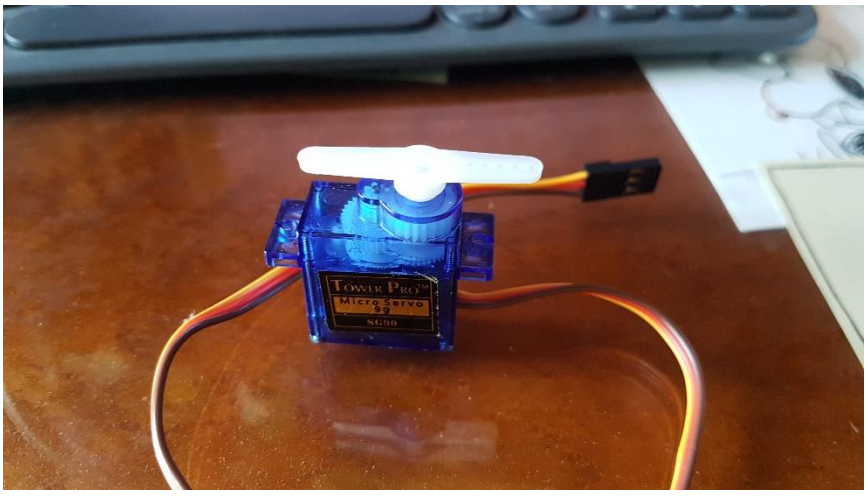

Figure 3.5.1 SG90 Tower Pro Micro Servo

This servo has three wires which emerge out of its motor: a red, a brown and an orange one. The brown one is the ground wire and needs to be connected to the ground of the system. The red one powers the motor and usually needs 5V. Finally, the orange one is used to drive the motor through the pulse-width modulation (PWM) signal which it receives. We generate this signal with our microprocessor, the Raspberry Pi.

By taking a closer look at the servo's datasheet (Figure 3.5.2), we can understand how the direction of the motor is controlled:

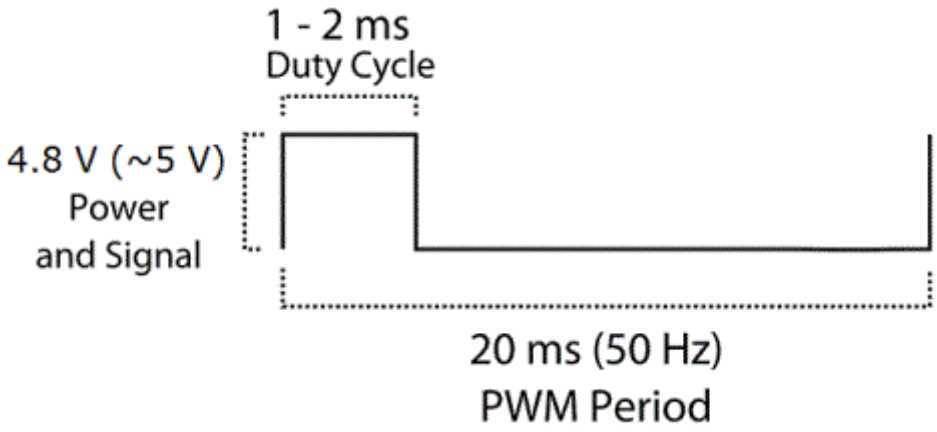

Figure 3.5.2 SG90 Tower Pro Micro Servo Datasheet

This servo expects a pulse every 20 msec on their PWM pin. The pulse is active HIGH, and the width of the pulse determines the position (angle) of the servos shaft. The pulse can fluctuate between 1 msec and 2 msec. A 1 msec pulse positions the shaft at 0°. A 1.5 msec pulse positions the shaft at 90° (centered in its range). A 2 msec pulse positions the shaft at 180°. Pulses with values in between these, position the shaft arbitrarily.

3.6 Camera

Complex teleoperative tasks generally require human control. Viewpoint control is vital in these tasks and helpful in automation. Nonetheless, teleoperating a robot with the assistance of indirect visual information can be technically challenging. The user is required to control both the movement of the camera and the robot's arms among with many other parts. Due to the physical separation of the user from the target point and the indirect view of the site provided by the camera input, some significant challenges arise [16]. For

humans, camera positioning is challenging since it is a complicated task that can be prone to errors and a drain on the user's resources.

In this thesis' robotic teleoperative task, the robotic system requires a user to remotely control the mechanical arm by using a master controller. This procedure cannot be characterized as an autonomous robotic system. Correct detection of the robot's position depends on obtaining distinctive features for the specific task we wish to achieve. Inadequate distinguishable features may deteriorate our system's detection performance. By keeping the viewpoint of the camera fixed, the arm's movement capability may be limited to the X and Y axis since the inputs of the visual sensors (i.e., the camera) do not provide clear information of the arm's movement in the Z axis. However, the detection errors are diminished.

The camera used on this project is the Raspberry Pi Camera Module V2 (Figure 3.6.1). It has a Sony IMX219 8-megapixel sensor, supports 1080p30, 720p60 and VGA90 video modes, as well as still capture. It attaches via a 15cm ribbon cable to the CSI port on the Raspberry Pi.

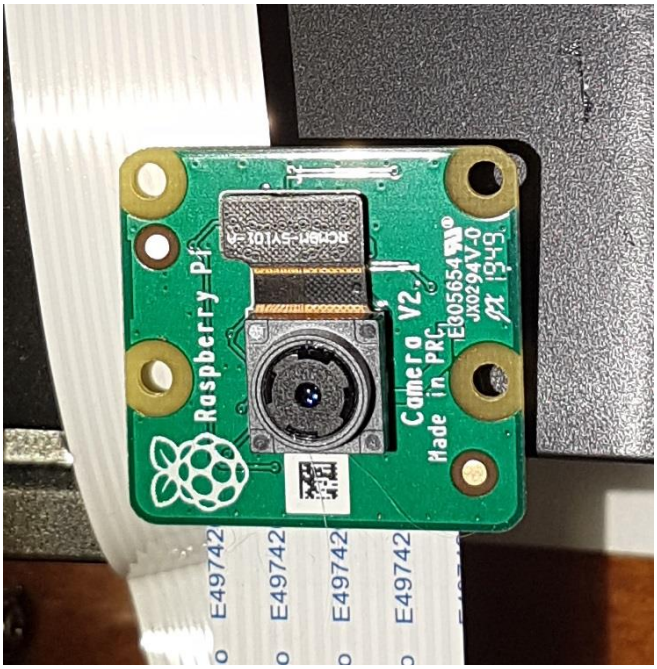

Figure 3.6.1 Raspberry Pi Camera Module V2

CHAPTER 4

3D PRINTING

Digital fabrication technology, also known as 3D printing or additive manufacturing, is the process of creating a 3D object of any shape and form from a geometrical representation (usually a 3D model or other electronic data sources), by a successive addition of materials. It is widely used by R&D (research and development) departments of large companies in the field of agriculture, manufacturing, construction, locomotive, aviation, healthcare and many more.

The first solid object printed by this method is credited to Hideo Kodama of Nayoga Municipal Industrial Research Institute [17]. Nevertheless, Charles Hull designed in 1984 the first 3D printer and is considered the developer of the solid image process (known as stereolithography), and the STL file format which is the dominant format used in 3D printing.

4.1 Types of 3D Printing

Binder Jetting

Originally developed at the Massachusetts Institute of Technology (MIT) in the early 1990s, binder jetting is an additive manufacturing process in which a liquid binding agent is placed onto a thin layer of powder particles like hybrid, ceramics, polymers, metals and sands, in order to bind them together and build unique parts. The process is repeated layer by layer, following a map from a digital design file, until the object is completed. It is also a simple, fast and cheap process and has the ability to print large-scaled products.

Directed Energy Deposition

This printing process is more intricate and is frequently used in repairing or adding material to existing components. A nozzle mounted on a multi-axis arm provides movement in multiple directions and deposits material onto specified surface where it solidifies. Upon deposition, the material is melted with a laser or electron beam. With this procedure we are capable of creating and repairing parts used in numerous sectors since it adds new material features on existing objects.

Materials Extrusion

Material extrusion is a widely used additive manufacturing process and was developed in the early 1990s. It utilizes continuous filament of thermoplastic or composite material to create 3D parts. The material is fed through an extruding nozzle in the form of plastic filament, where it is heated and then placed onto the build platform layer by layer from the bottom to the top.

Material Jetting

Material jetting uses a similar type of technology with the standard inkjet printer and it closely resembles the binder jetting process. The material, which is usually of a photosensitive kind, is placed directly onto the build surface through a printhead dispenser and solidifies, layer by layer, under ultraviolet light. Afterwards, the build platform adjusts its height and the procedure is repeated, creating a smooth surface finish with great dimensional accuracy.

Powder Bed Fusion

The Powder Bed Fusion process encompasses the following printing techniques: Direct metal laser sintering (DMLS), Electron beam melting (EBM), Selective heat sintering (SHS), Selective laser melting (SLM) and Selective laser sintering (SLS). With the use of a high-energy power source, either an electron beam or a laser, the metallic powder bed is melted or sintered together. Then, a new layer of powder is spread across the previous layer with the help of a roller and after lowering the build platform, the process repeats itself until the entire model is created.

Sheet Lamination

Sheet lamination includes two types of manufacturing which are characterized by the materials and type of welding they use. Both processes bind together sheets of materials using either ultrasonic welding (Ultrasonic Additive Manufacturing process) or paper as a material and adhesive instead of welding (Laminated Object Manufacturing). They both follow a layer-by-layer approach since the building steps are quite similar. The material is initially positioned on the cutting platform and is bonded in place over the previous layer

by using the adhesive. Afterwards, the required shape is cut from the layer and the next layer is added. The cutting can also occur before the bonding.

Vat Photopolymerization

Vat Photopolymerization is the 3D printing technology of curing photo-reactive polymers with the use of a laser (light or ultraviolet). The model is constructed layer by layer in a vat of liquid photopolymer resin with the use of an ultraviolet light which cures and/or hardens the resin where it is required, while the build platform is moving downwards after each layer is cured. The end result is of high quality with excellent details.

4.2 Modelling and Printing Material

The creation of 3D printed models can be derived either from a scanned object or manually. Both processes are quite complicated but numerous user friendly, Computer Aided Model (CAD) software exist such as Fusion 360, Solidworks, BRL-CAD and many more. The parts of this thesis' robotic arm were designed in Fusion 360 [18] and printed in a material extrusion printer, the Raise 3D N2 [19] (Figure 4.2.1), with Polylactic Acid (PLA) filament.

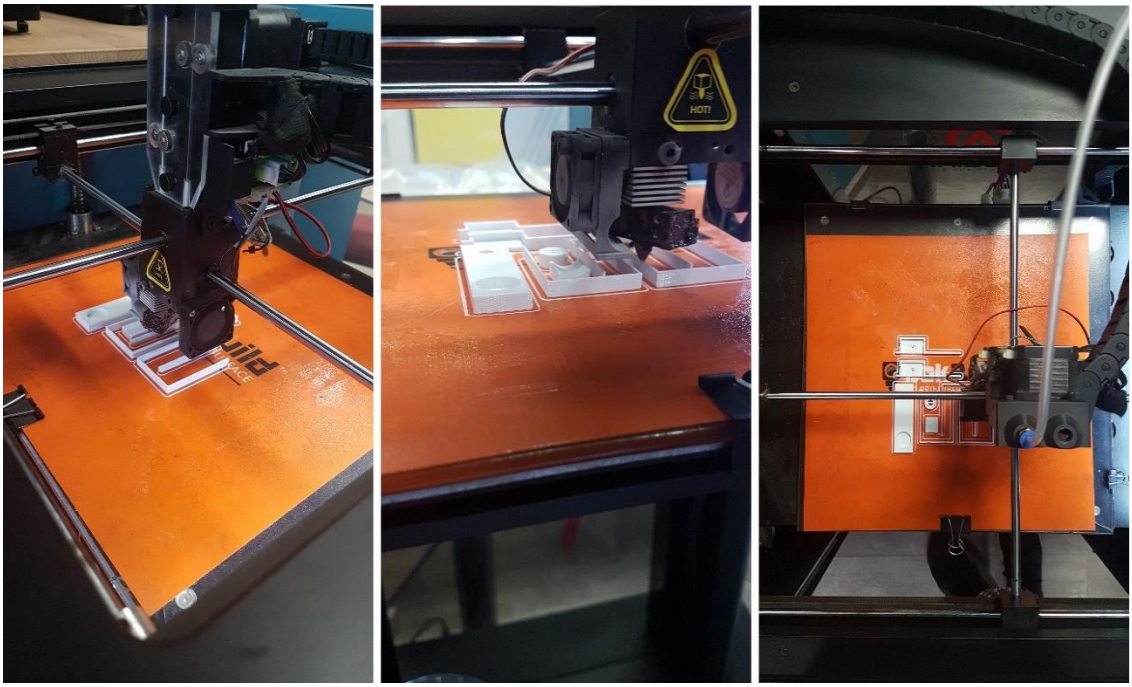

Figure 4.2.1 Raise 3D N2 during the Printing Process of the Parts

Polylactic Acid (PLA) (Figure 4.2.2) is the most popular material in 3D printing because of its ease of use, environmentally friendly and low-cost material. It is originated from natural sources like corn starch or sugar cane and can be found in wide assortments of colors and blends. It has a low printing temperature, which starts around 180°C, and produces sharper features and better surface details compared to ABS and other thermoplastics with higher melting temperatures. PLA prints can also be easily sanded, polished and painted. It is non-toxic, biodegradable, and since it requires a smaller amount of energy to print, it emits less greenhouse gases. Nevertheless, PLA cannot be used in any high temperature application due to its low heat resistance and it also has a lower tensile strength in comparison to its counterparts. Since this is a small-scale robotic arm which is not required to withstand heavy loads or high temperatures, it was chosen as the more inexpensive and environmentally friendly solution.

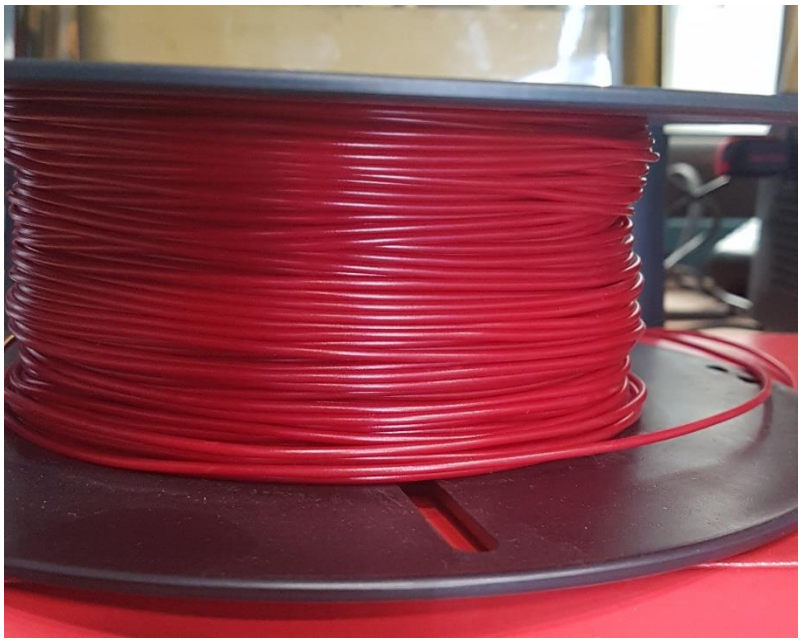

Figure 4.2.2 PLA Filament

CHAPTER 5

SOFTWARE PRESENTATION

5.1 Software Versions and Sources

Python 3.9.6	https://www.python.org/downloads/release/python-396/
OpenCV 4.5.2	https://opencv.org/releases/
Numpy 1.21	https://numpy.org/install/
GPIO Zero	https://gpiozero.readthedocs.io/en/stable/installing.html

5.2 OpenCV

OpenCV [20] is an open-source library of programming functions focusing on real-time computer vision systems. Initially launched by Intel in 1999, its alpha version debuted at the IEEE Conference on Computer Vision and Pattern Recognition in 2000 and after numerous betas, its first version was released in 2006. While OpenCV's support and development has been transferred to numerous research teams, corporations and foundations, its main goal remains the same: the advancement of computer vision research and knowledge and the development of optimized, portable and free code focusing on its performance.

OpenCV is platform-independent and uses binding (wrapper) functions which are initiated and called within another language (currently available for MATLAB, Java and Python). These wrappers are mapping OpenCV's core functions [21] , maintaining the same core functionality for all supported languages, reducing callback delay and providing code robustness quality.

5.3 Fusion 360

Fusion 360 is developed and distributed under AUTODESK [18]. This 3D Design and Modeling software provides numerous modeling tools and unifies design, engineering, electronics and manufacturing, allowing the user to connect its entire product development process into one platform (Figure 5.3.1).

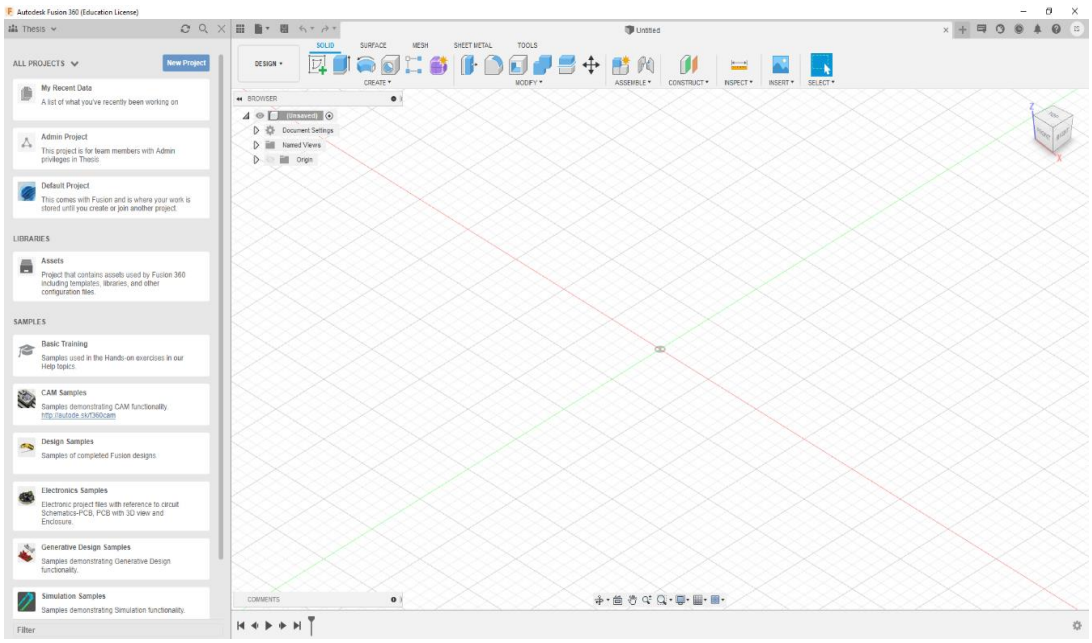

Figure 5.3.1 Fusion 360 Environment

Seven STL files were printed after careful measurement of the potentiometers (Figure 5.3.4, Figure 5.3.5 and Figure 5.3.6) and the servos' dimensions (Figure 5.3.2 and Figure 5.3.3). Each arm is comprised of 4 parts: the support unit, the base, the upper arm and the lower arm. The shoulders, elbows and wrists are represented by the micro servos and the potentiometers at each arm respectively.

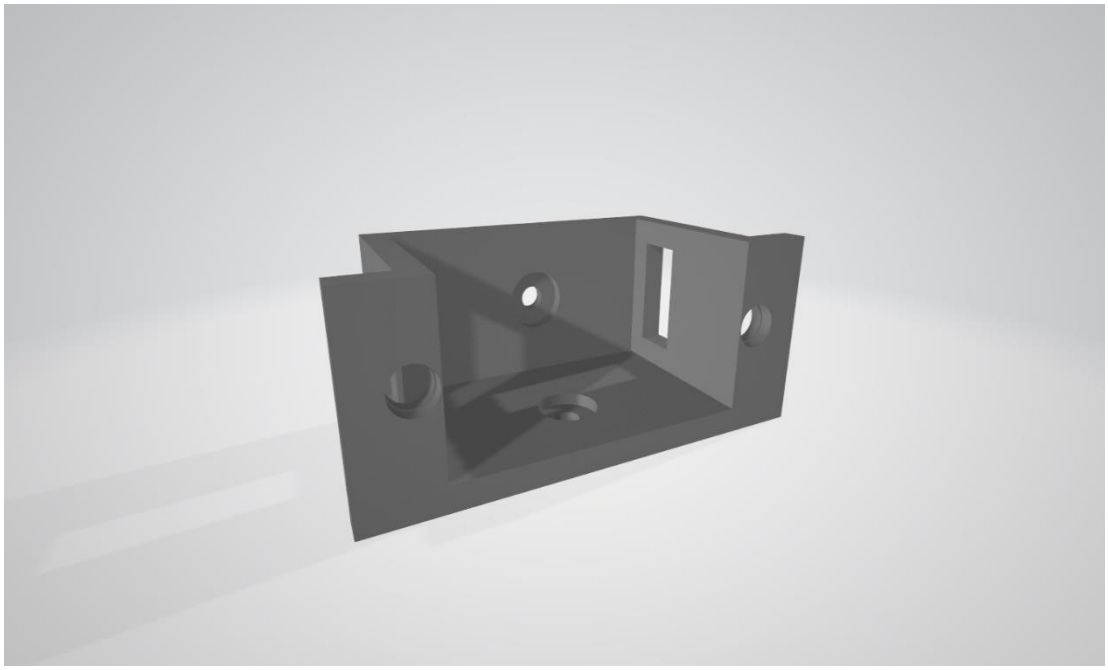

Figure 5.3.2 Servo's Encasement

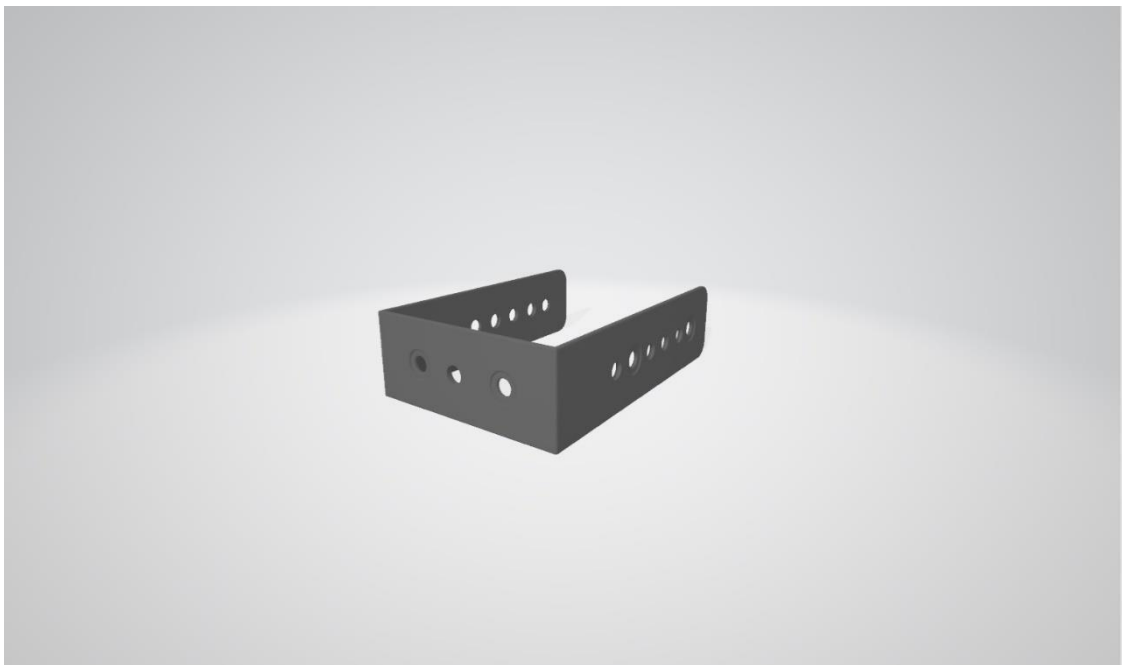

Figure 5.3.3 Servo's Linkage

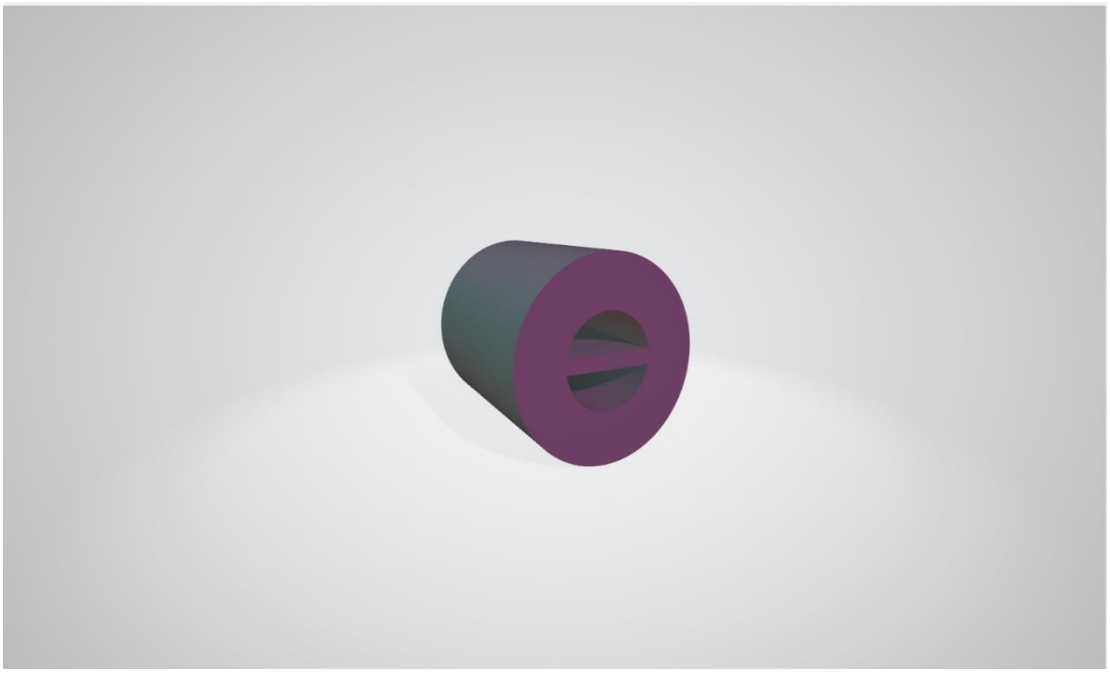

Figure 5.3.4 Potentiometer's Socker for the Base

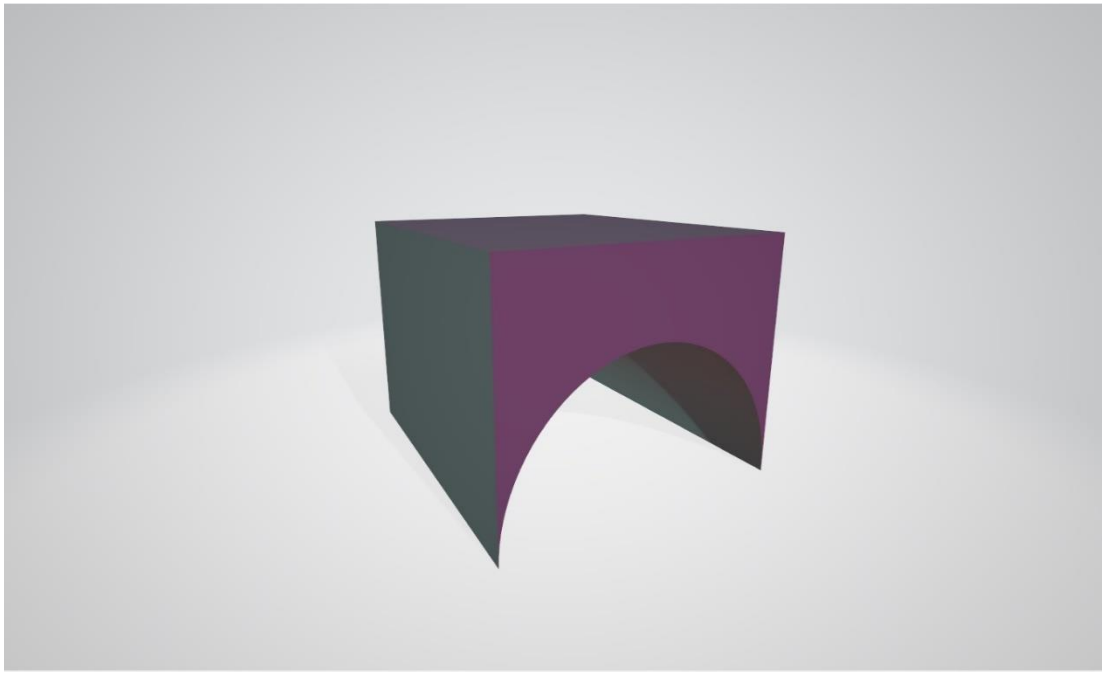

Figure 5.3.5 Extra Base Part for Potentiometer's Base

Figure 5.3.6 Potentiometer Linkage between Base, Shoulder and Wrist

CHAPTER 6

Color Model, Color Space, Image Processing and Shape Detection

6.1 Color Model Theory

In the physical world, color exists as a wavelength of light. From the moment this wavelength reaches the rods and cones of the eye, complex processing is being carried out in the human neurons. This was the first human sense we were able to mathematically model, giving it its own space representation and unit/measurement; In 1931, the International Commission on Illumination, also known as Commission Internationale de l'Eclairage (CIE) [22], linked the average human being's color vision to mathematical coordinates (Figure 6.1.1), and despite the constant evolution in complex colorimetric transformations and representations of a color, their origin is still based on the same colorimetric standard observer;

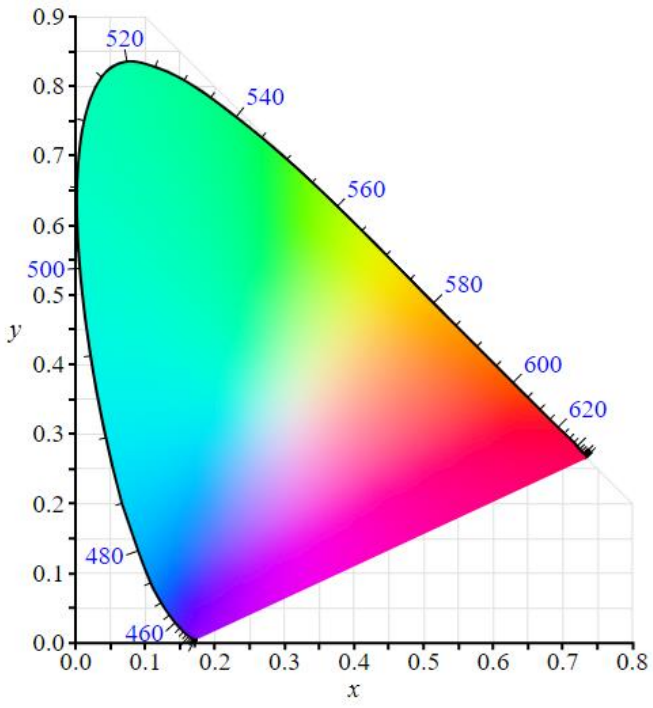

Figure 6.1.1 CIE 1931 Chromaticity Diagram.

(Image downloaded from: https://www.wikiwand.com/en/CIE_1931_color_space , in June 2021)

Wavelength markers along the edges express every color on the visible spectrum as a set of coordinates on the diagram above. Color space is the relationship between the wavelength of light and Red Green and Blue (RGB) values. To store this color information in a computer, a combination of red, green and blue values is used. This combination is created by drawing a triangle over the top of the chromaticity diagram with 4 points (3 in the corners and one inside). The points at the triangle's edges are the red, blue and green ones and the white point is in the middle (Figure 6.1.2). The RGB values operate on a scale from 0 to 255 and these points describe where the 0 to 255 values lie on the chromaticity diagram. This is called a color model.

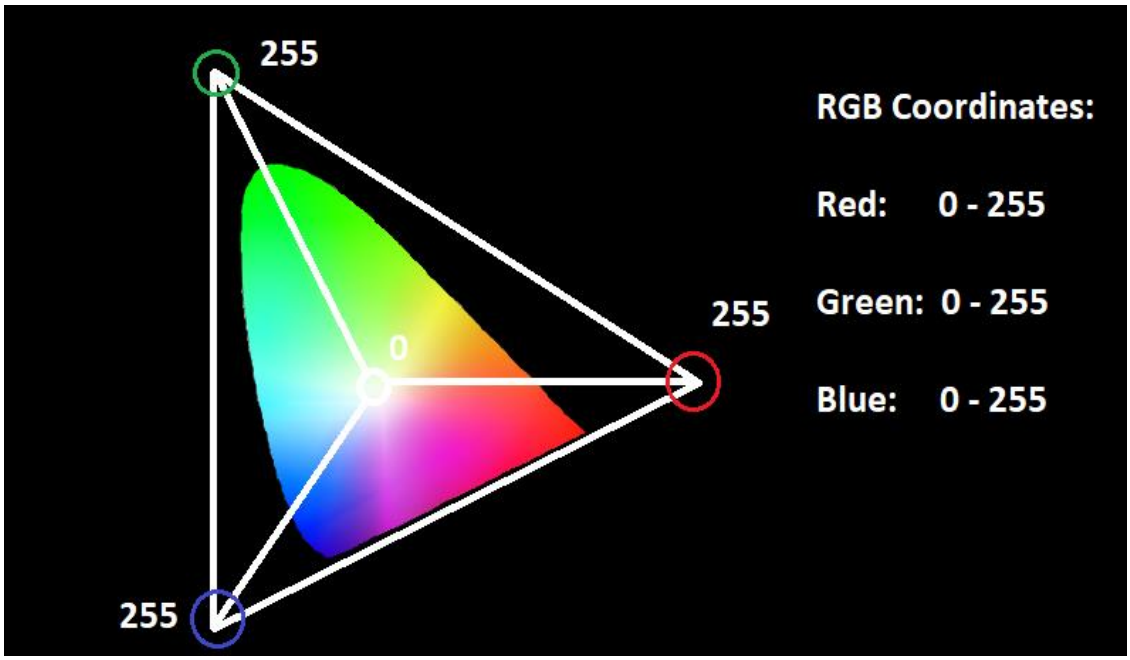

Figure 6.1.2 Chromaticity Diagram with RGB Points

These RGB coordinates are only meaningful in relation to the color space we are working in. If the point of the triangle is defined in different locations then the same visible color will be represented with a different set of RGB coordinates. For a consistent color model, we need the triples of RGB coordinates (or quadruples in Cyan, Magenta, Yellow and Key, CMYK) and the color space they are relative to. In essence, a color model (an abstract mathematical model describing representation of colors) in combination with an associated mapping function to a reference color space, is known as a gamut and defines a color space for a given color model.

The RGB color model is an attempt to accurately represent the additive properties of the red, green and blue components of light, but it is not the best one since it does not offer any information on the distinction between color and how bright that color is [23]. Therefore, depending on physics, human perception and additive or subtractive physical descriptions of color, multiple color models were created (Table 6.1.1).

ACRONYM	INTENTED USAGE	AXIS 1, AXIS 2, AXIS 3
RGB	Device-specific color specification	Intensity of Red, Green and Blue
HSV	Color Mixing	Hue, Saturation, Value
HLS	Color Mixing	Hue, Lightness, Value
LAB	Calculating color distance according to human perception	Lightness, Red/Green balance, Green/Blue balance
CMY	Color Mixing	Cyan, Magenta, Yellow

Table 6.1.1 Some Common Color Models

6.2 Color Space

There are five major color space models subdivided into others, CIE, RGB, YUV, HSL/HSV, and CMYK. The categorization is implemented according to their focus on the human perception (CIE 1931 XYZ, CIEUVW, CIELUV and CIELAB), the chromaticity component of a given color when luminance is excluded (sRGB, Adobe RGB, Adobe Wide Gamut RGB, Rec. 2100), the separation of lightness from chroma signals in an RGB input (YCbCr, YUV, YCoCg, YCC), the printing process (CMYK) and lastly, on the transformation of the color model into lightness, colorfulness and hue (HSL, HSV).

There are many color spaces in existence. In order to choose which color space will procure the best results in this environment, the work of A. Rasouli and J. Tsotsos [24] as well as A. Mukhopadhyay, I. Mukherjee and Pradita Biswas [25] were taken into consideration, concentrating on the spaces with the highest FM measurements, precision and accuracy in their experiments. YCbCr, c1c2c3 and the CIE L*a*b* will be tested in this scenario.

- YCbCr: The human eye perception of color is of paramount importance in object detection. Consequently, the YUV representation was devised. Y stands for the Luminance (the brightness) and U and V are the chrominance (color) components (blue–luminance (U) and red–luminance (V)). The YUV color encoding can be overlapped by the terms Y'UV, YCbCr, YPbPr, etc., depending on where the model is used. YCbCr is the most frequent term usage of the YUV model in the digital encoding area thus, it will be used henceforth.
- c1c2c3: It is a powerful color model in object detection because of its invariability to a change in viewing direction, object geometry and illumination [26].
- CIE L*a*b*: If the color segmentation in a space lacks light sensitivity, this color space can avoid this drawback by increasing the accuracy of the color segmentation. It closely resembles the human eye perception.

6.3 Image Processing for Object Detection

Object detection refers to the existence or appearance or position of an object in a digital image [27]. The detection process can be extra challenging when clustered backgrounds and/or moving objects occur. Nonetheless, the environment of an industrial workplace can be controlled to a great extend and since a simple pattern-based object detection can be used and the angle of the viewpoint is fixed, there is no current need for convolutional neural network-based machine learning aids to be introduced in the modeling [28].

Color space conversion

Initially, the captured image needs to be converted from BGRA (red, green, blue, alpha) format, the format in which OpenCV saves its captures, to BGR (blue, green, red) format. The alpha in BGRA indicates the opaqueness of each pixel and BGRA cannot be used since it does not define what RGB color model is being used (6.1).

```
# Transformation of original image to BGR
captured_frame_bgr = cv2.cvtColor(captured_frame, cv2.COLOR_BGRA2BGR)
```

A median blur is then applied on the image which helps with noise reduction. It is a very crucial preprocessing step since by replacing the central element of the image with the median of all the pixels in the kernel area, the edges are processed all the while removing the noise.

```
# First blur for noise reduction before color space conversion
captured_frame_bgr = cv2.medianBlur(captured_frame_bgr, 3)
```

Finally, the color space conversion can take place. The image is already converted in RGB nonetheless, the color segmentation in this space lacks light sensitivity. Therefore, it is converted in YCbCr, c1c2c3 and the CIE L*a*b* color space in order to compare the most appropriate color space for this work environment.

```
# Conversion into YCR_CB color space
captured_frame_ycr_cb = cv2.cvtColor(captured_frame_bgr, cv2.COLOR_BGR2YCR_C
B)

# Conversion into Lab color space
captured_frame_lab = cv2.cvtColor(captured_frame_bgr, cv2.COLOR_BGR2Lab)

# Conversion into c1c2c3 color space
prec1c2c3 = captured_frame_bgr.astype(np.float32)+0.001
c1c2c3 = np.arctan(prec1c2c3/np.dstack((cv2.max(prec1c2c3[...,1], prec1c2c3[
...,2]), cv2.max(prec1c2c3[...,0], prec1c2c3[...,2]), cv2.max(prec1c2c3[...,
0], prec1c2c3[...,1]))))
```

Segmentation

Thresholding assists with the setting up of an object's boundaries on a contrasting background by using a threshold rule. According to the colors selected for the circle, the inRange function of OpenCV keeps only the pixels of the image within the boundaries of the colors selected:

(Following examples focus on the red pixels during Lab color space segmentation)

```
captured_frame_lab_red = cv2.inRange(captured_frame_lab, np.array([20, 150,
150]), np.array([190, 255, 255]))
```

A second blur is next applied (OpenCV function: GaussianBlur) which limits the noise and assists in the circle detection. By employing a Gaussian function, a smooth blur is applied on the image, which will afterwards appear as if seen through a translucent screen.

```
captured_frame_lab_red = cv2.GaussianBlur(captured_frame_lab_red, (5, 5), 2,
 2)
```

Shape detection

For the circle detection, the Hough Transform OpenCV function is used. Patented in 1962 by Paul V C Hough [29], it is a feature extraction technique which focuses on finding imperfect occurrences of objects of a certain shape by a voting procedure.

```
# Hough transform for the red circle detection
red_circles = cv2.HoughCircles(captured_frame_lab_red, cv2.HOUGH_GRADIENT, 1
, captured_frame_lab_red.shape[0] / 8, param1=100, param2=18, minRadius=5, m
axRadius=60)
```

CHAPTER 7

IMPLEMENTATION AND DESCRIPTION OF THE ROBOTIC ARM CAPABILITIES

7.1 Simple Movement Simulation

The first benchmark was to achieve the accurate movement of the micro servos through the potentiometers. Since the selected micro servos are prone to wear and tear, anything can disrupt their smooth operation: power surges, extra weight, signals for movement which reach their limits and even a small amount of extra force when setting a screw on them. Careful and meticulous fitting of the servo robot's (Figure 7.1.1) components is of paramount importance.

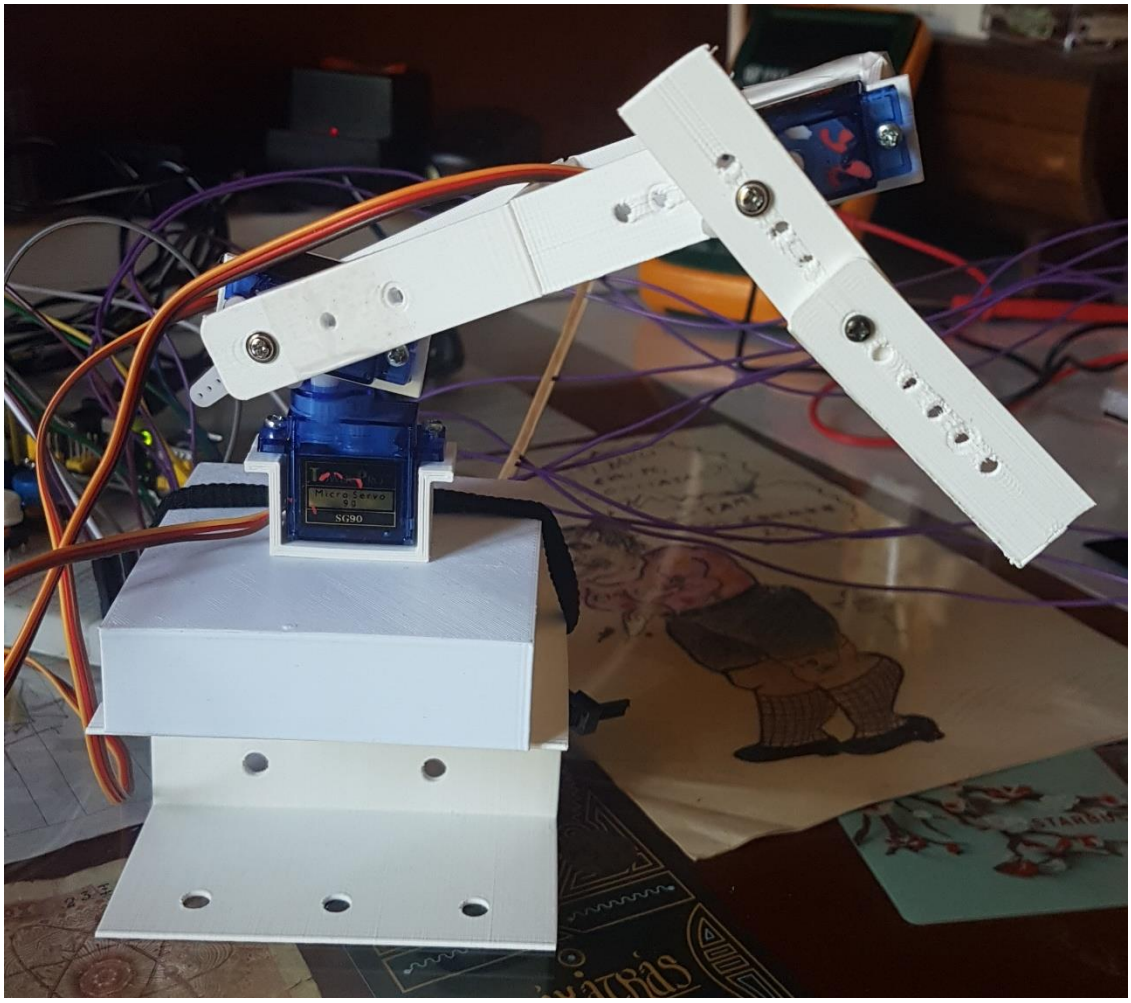

Figure 7.1.1 Micro Servo Robot

The micro servo robot is operated through the potentiometer robot (Figure 7.1.2). For its assembly, the crucial part is the correct soldering of each edge and its wiring. If the ground wiring of every potentiometer is not placed at the same grounding of the MCP channels, they will not work.

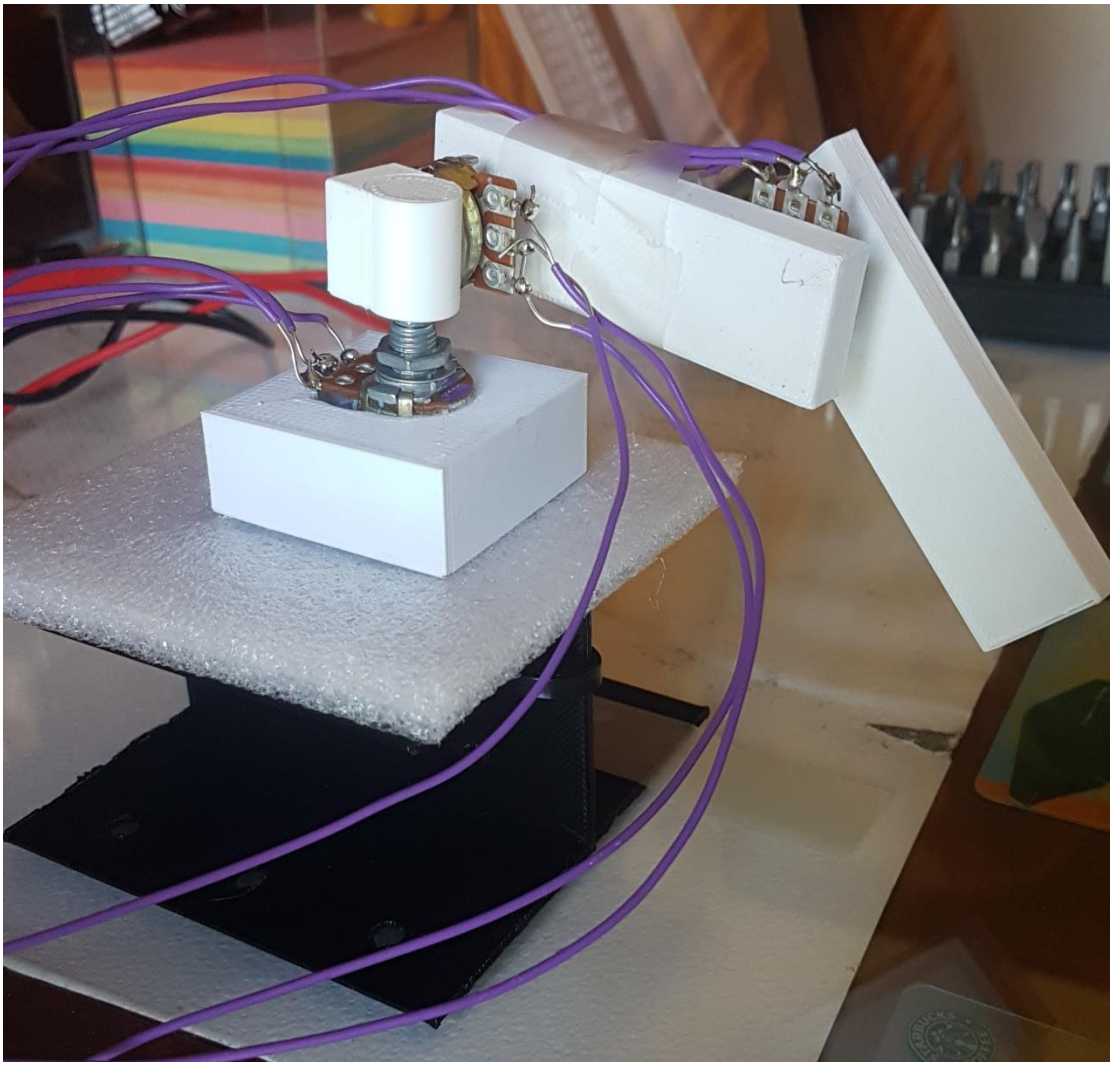

Figure 7.1.2 Potentiometer Robot

To reduce the jitter in the servos, gpiozero's [30] PiGPIOFactory was imported. This library also facilitates remote control of the Raspberry's GPIOs, an essential part in the incorporation of the robot in an industrial line. The python script in page 59 reads from channel 0, 1 and 2 of the MCP the values of the potentiometers and, after the necessary scaling, transmits them to each servo accordingly.

7.2 Teaching Algorithm

The second target for this project is to combine the movement simulation with a repetitive motive. A “teachable” movement range is given to the servo robot through the control of the potentiometer robot by the user. The number of values that are saved depend on the user’s judgement and on the sampling rate which can be modified through the source’s delay (script page 60).

7.3 Angle Finder

The third and final benchmark for this robot is the correct localization and angle finding of strategically placed colored circles. By accomplishing it, the servo motor can be fed all the necessary info to accurately mimic movements from a distance. When distance is not an issue for the robot’s control, the industrial workspace can be decluttered and operated under safer conditions.

The implementation was restricted in providing captures from the Raspberry’s Pi camera and not in live feed. Three different lightning scenarios (Figure 7.3.1,Figure 7.3.6, Figure 7.3.11) in a were initially blurred for reduced noise (Figure 7.3.2, ,Figure 7.3.7, Figure 7.3.12) and next converted into 3 color spaces: LAB (Figure 7.3.3, Figure 7.3.8, Figure 7.3.13), YCR_CB (Figure 7.3.4, Figure 7.3.9, Figure 7.3.14) and c1c2c3 (Figure 7.3.5, Figure 7.3.10, Figure 7.3.15).

1st LIGHTNING SCENARIO

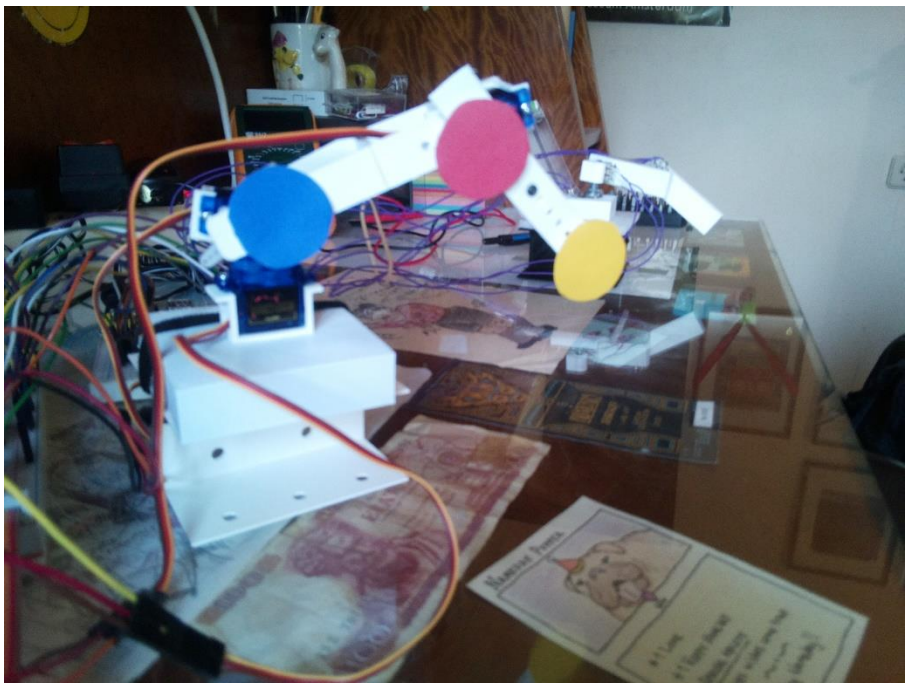

Figure 7.3.1 First Image

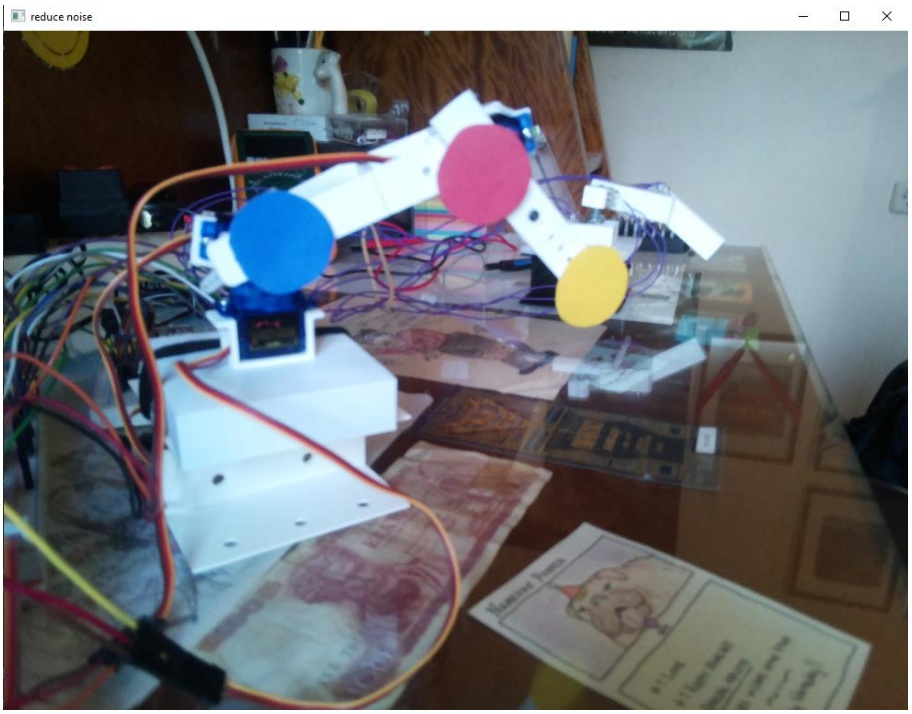

Figure 7.3.2 1st Image after First Noise Reduction

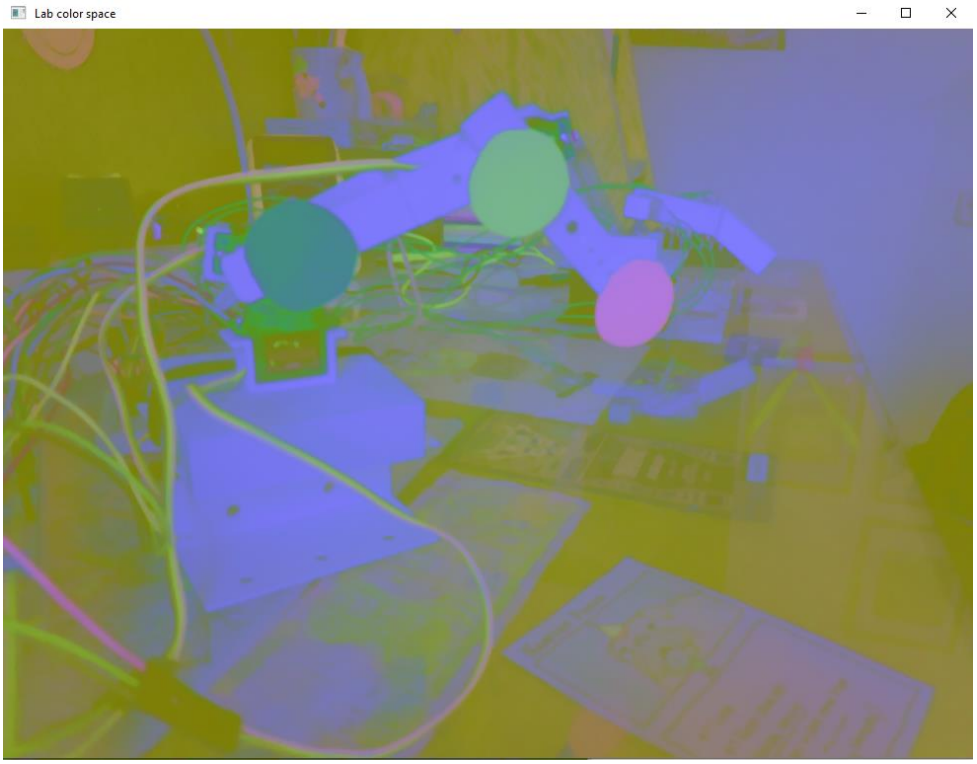

Figure 7.3.3 LAB Color Space of 1st Image

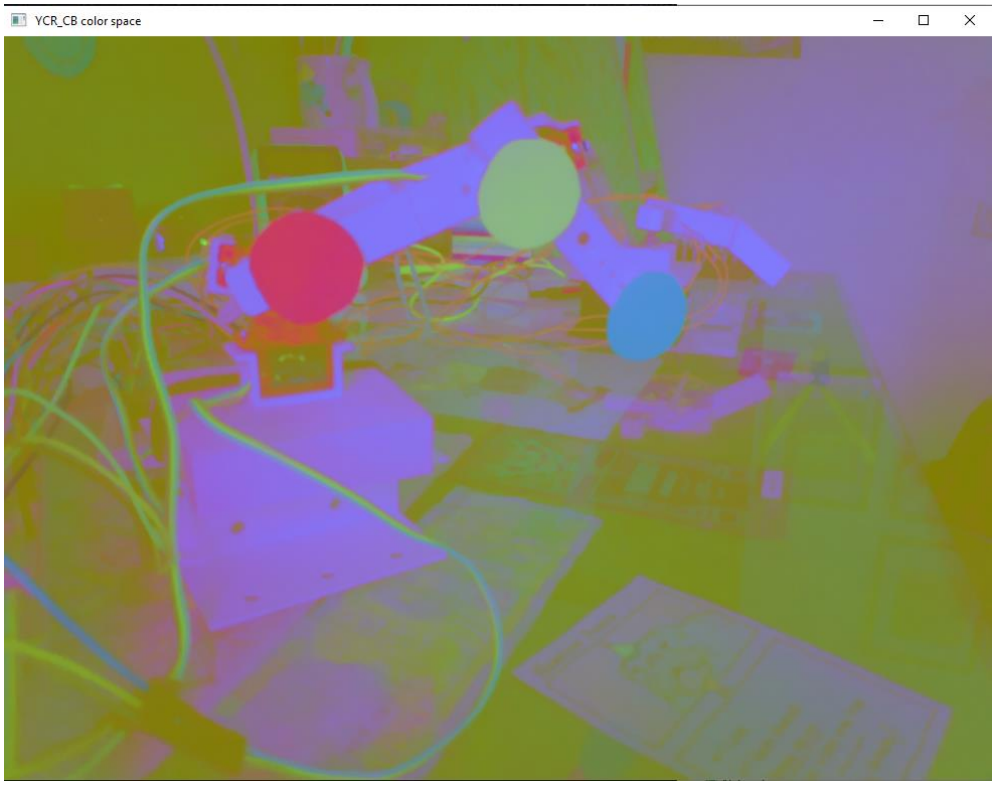

Figure 7.3.4 YCR_CB Color Space of 1st Image

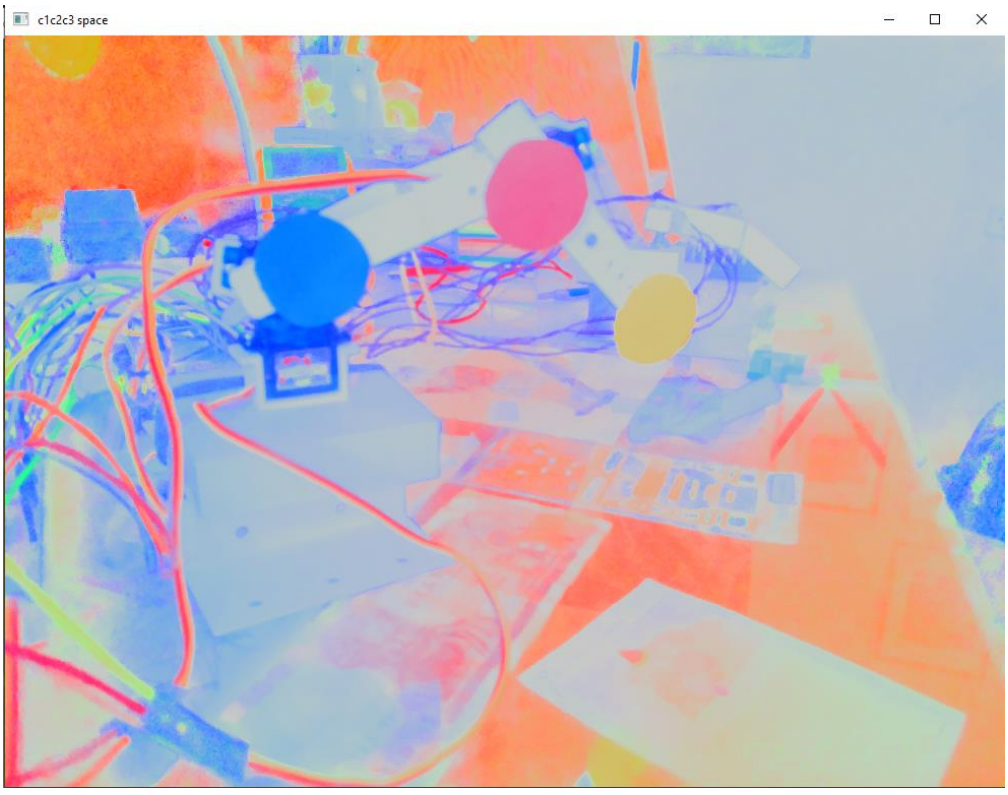

Figure 7.3.5 c1c2c3 Color Space of 1st Image

2nd LIGHTNING SCENARIO

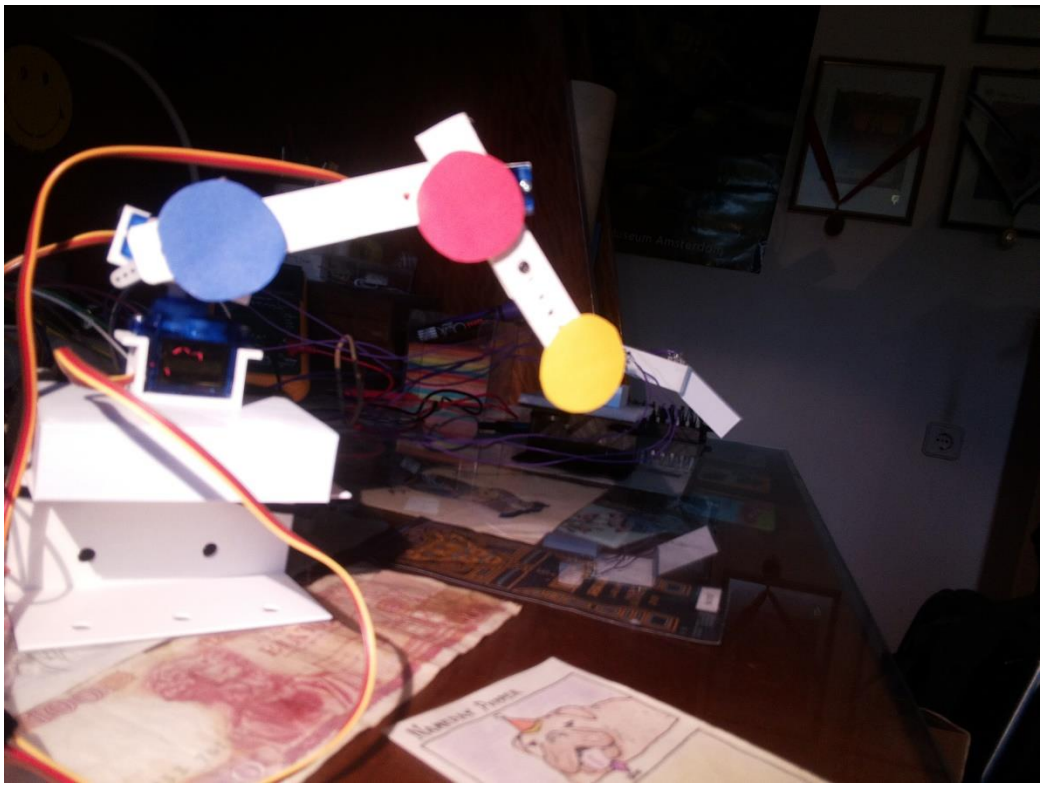

Figure 7.3.6 Second Image

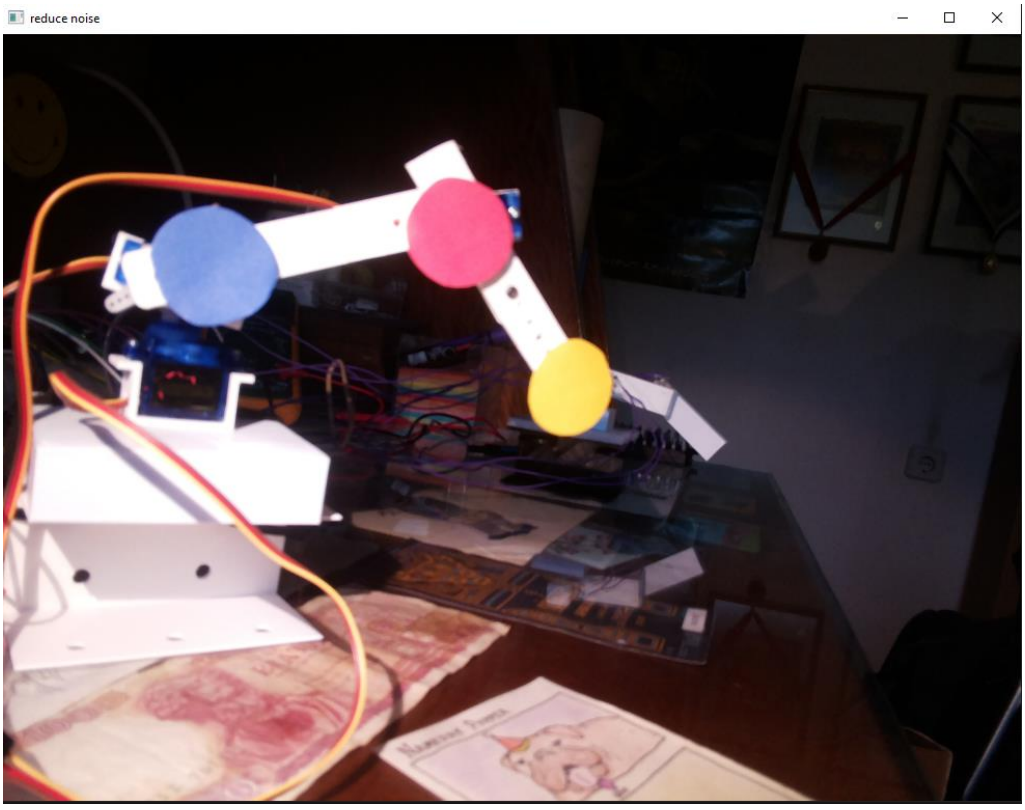

Figure 7.3.7 2nd Image after First Noise Reduction

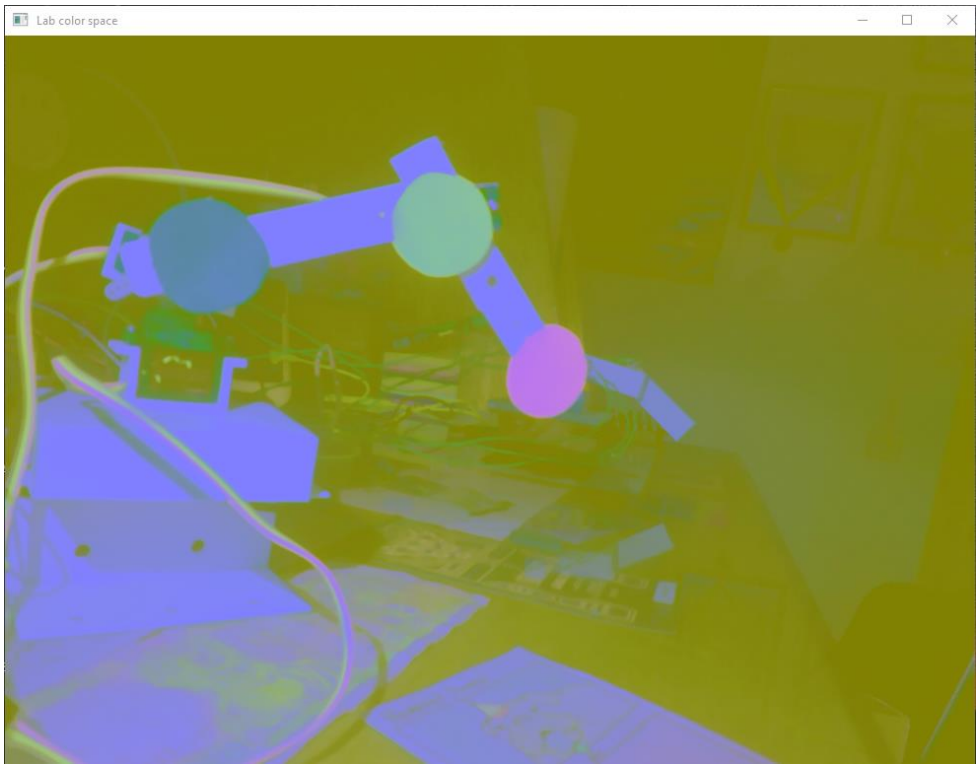

Figure 7.3.8 LAB Color Space of 2nd Image

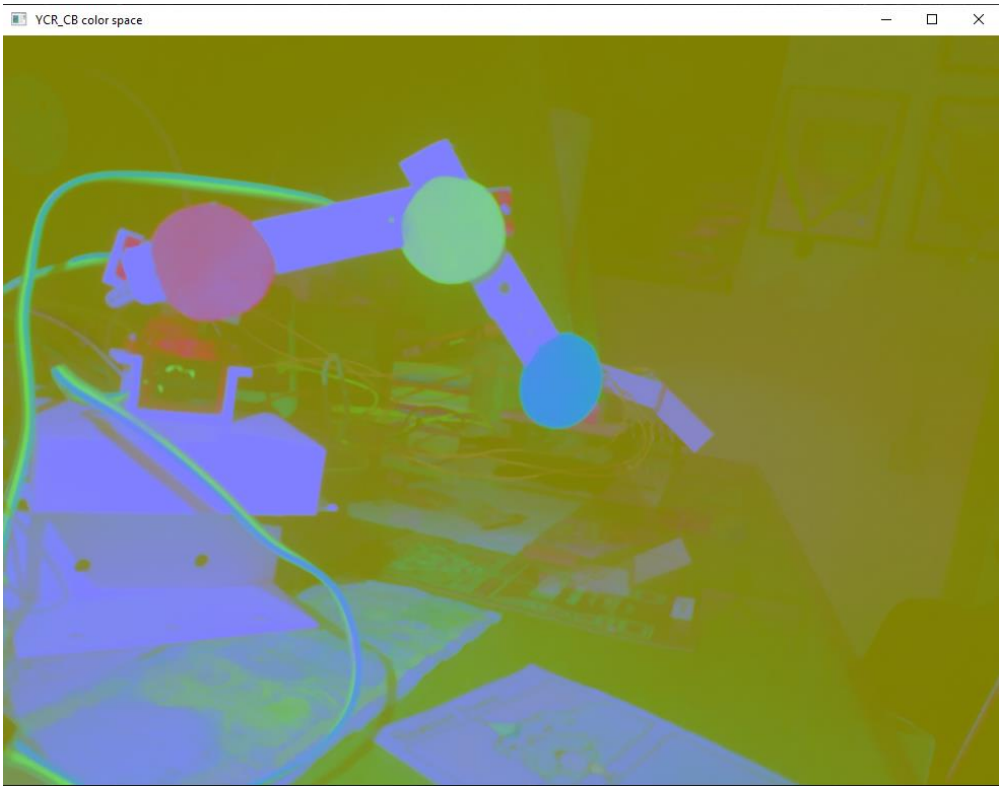

Figure 7.3.9 YCR_CB Color Space of 2nd Image

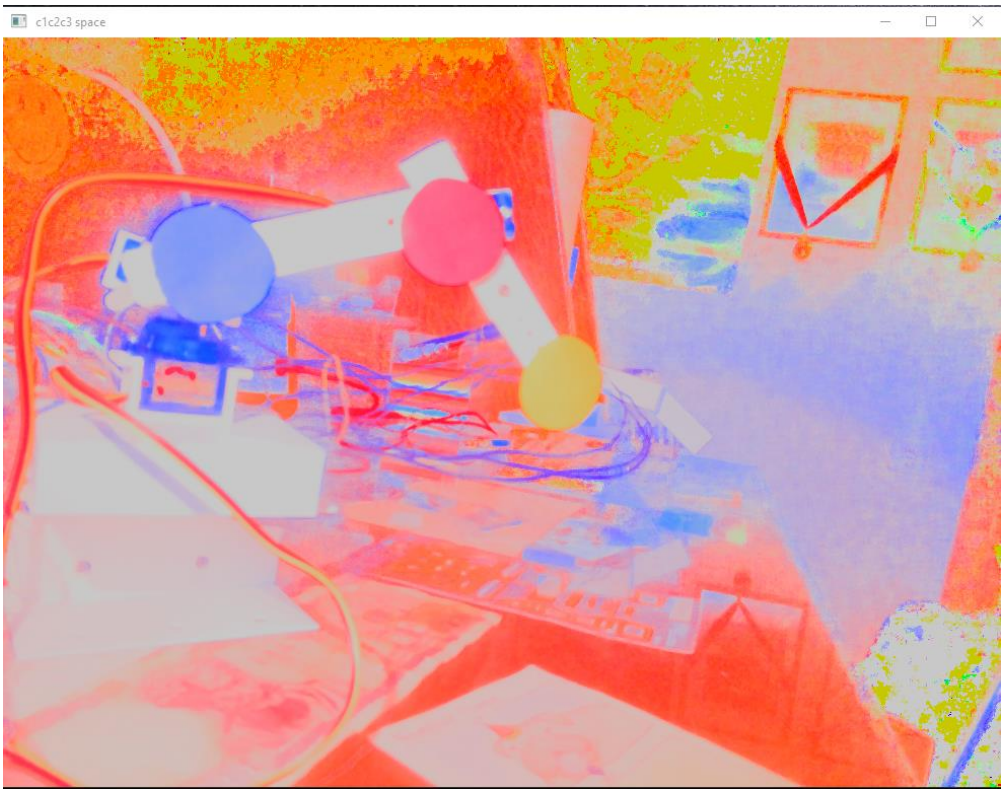

Figure 7.3.10 c1c2c3 Color Space of 2nd Image

3rd LIGHTNING SCENARIO

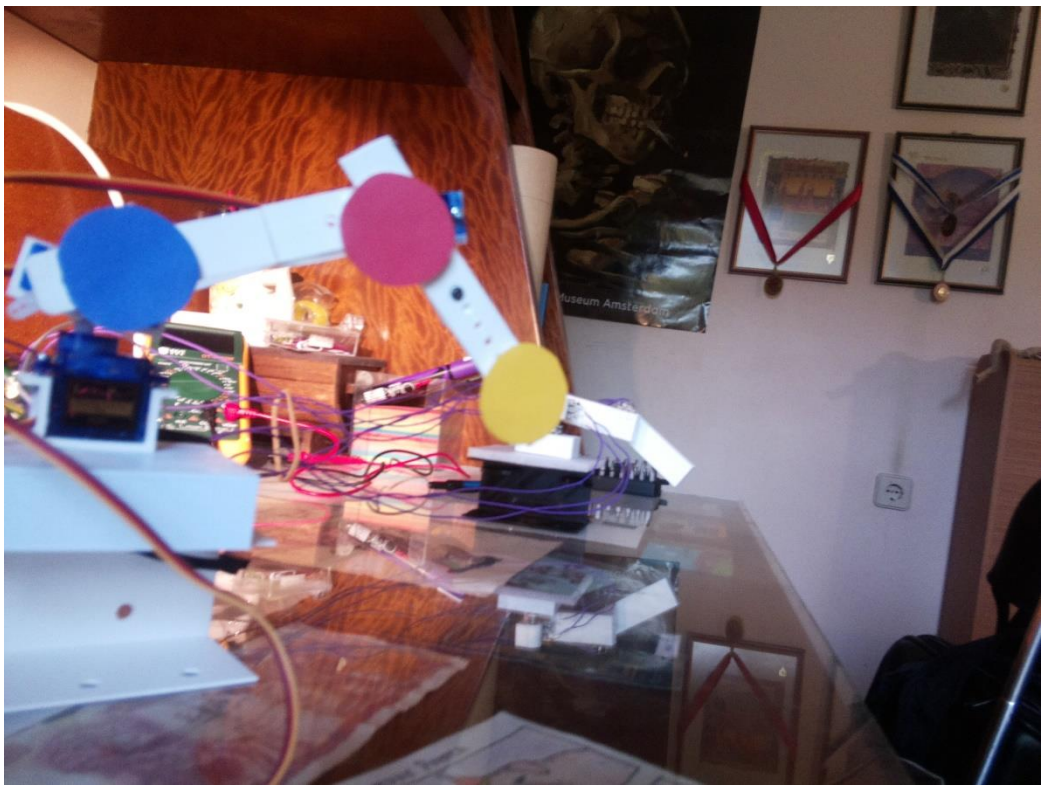

Figure 7.3.11 Third Image

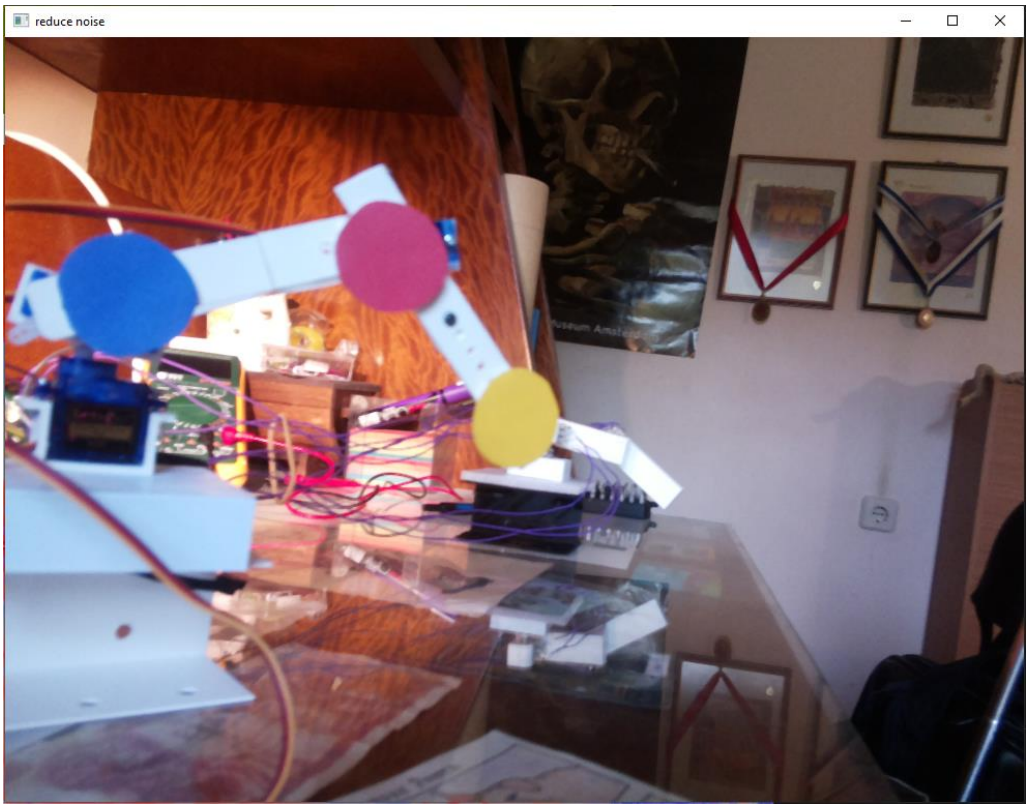

Figure 7.3.12 3rd Image after First Noise Reduction

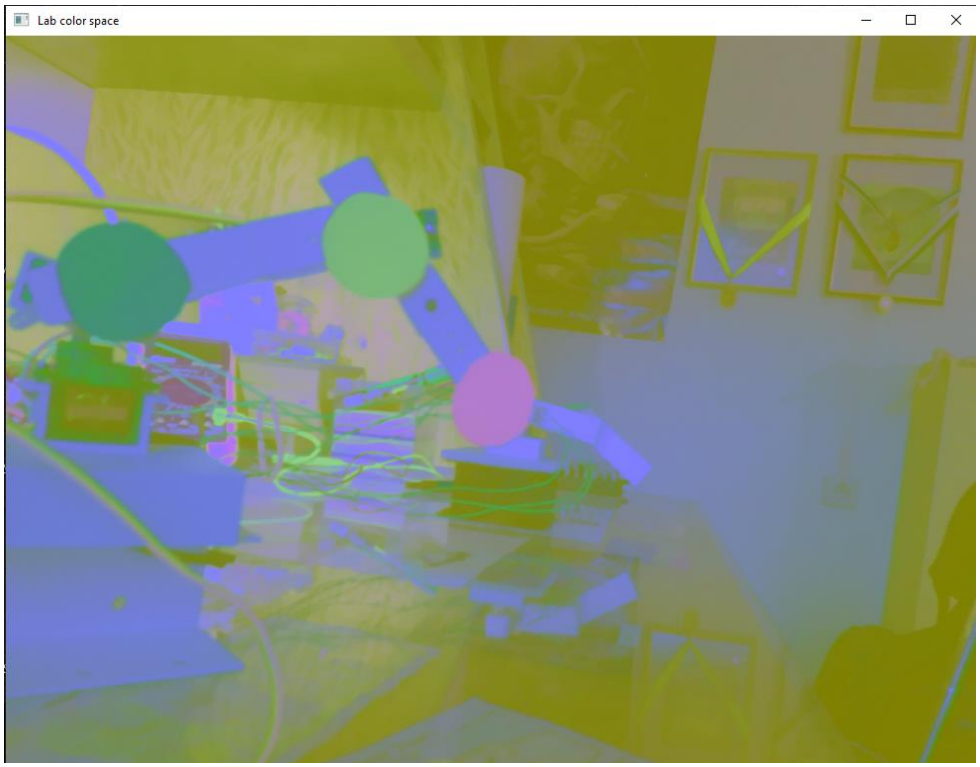

Figure 7.3.13 LAB Color Space of 3rd Image

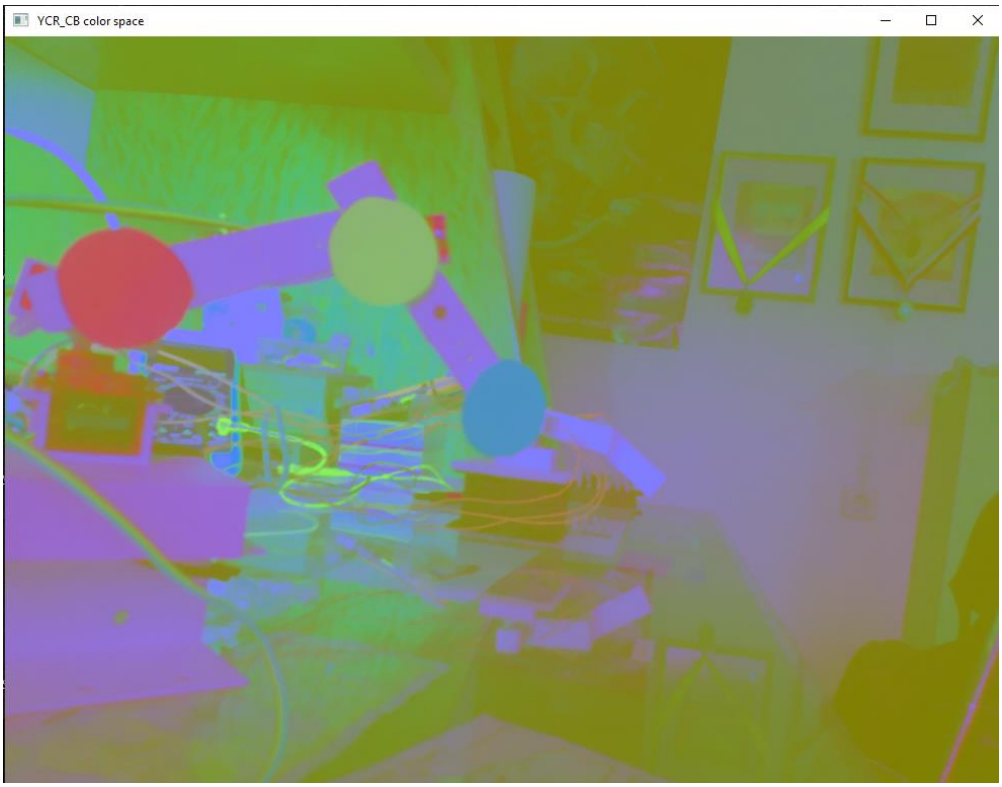

Figure 7.3.14 YCR_CB Color Space of 3rd Image

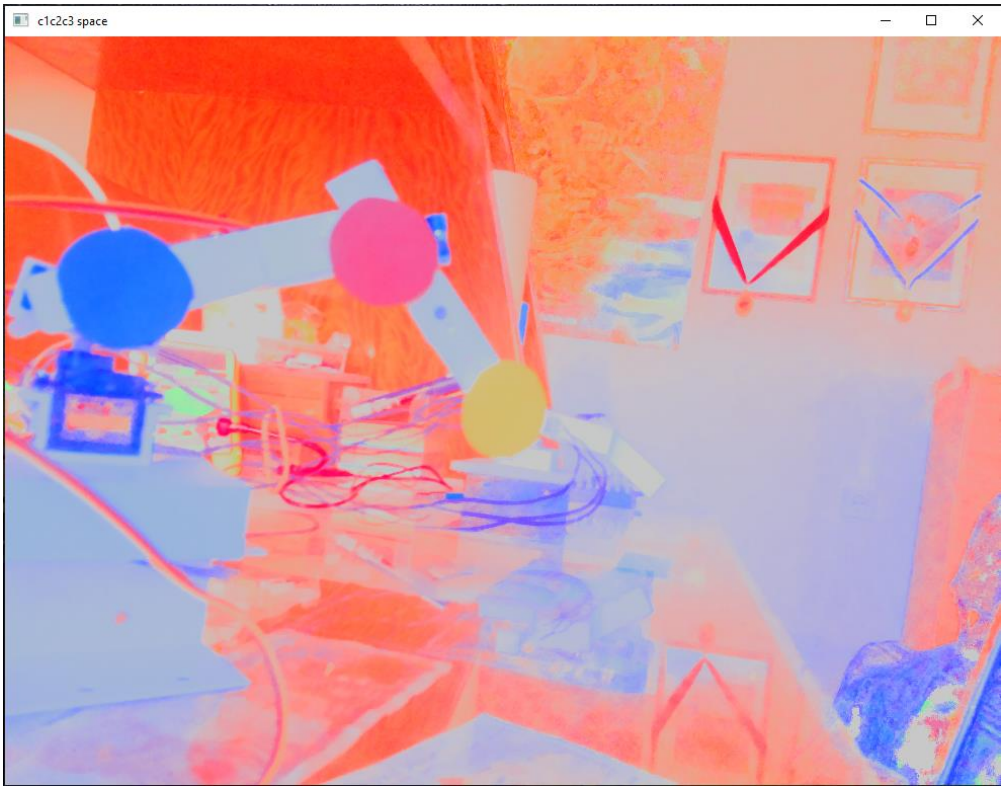

Figure 7.3.15 c1c2c3 Color Space of 3rd Image

Comparing all the above color space, the LAB picture has more defined circles in most of the lightning scenarios. The environment of the picture was not staged and remained cluttered on purpose to observe the script's power under unfavorable conditions.

Following the LAB color space conversion of all the lightning scenarios, the process singles out all the red, blue and yellow pixels and blurs the image for a second time with Gaussian Blur (Figure 7.3.16, Figure 7.3.18, Figure 7.3.20). Lastly, with the Hough Circles Transformation, the circle detection and angle calculation takes place (Figure 7.3.17, Figure 7.3.19, Figure 7.3.21).

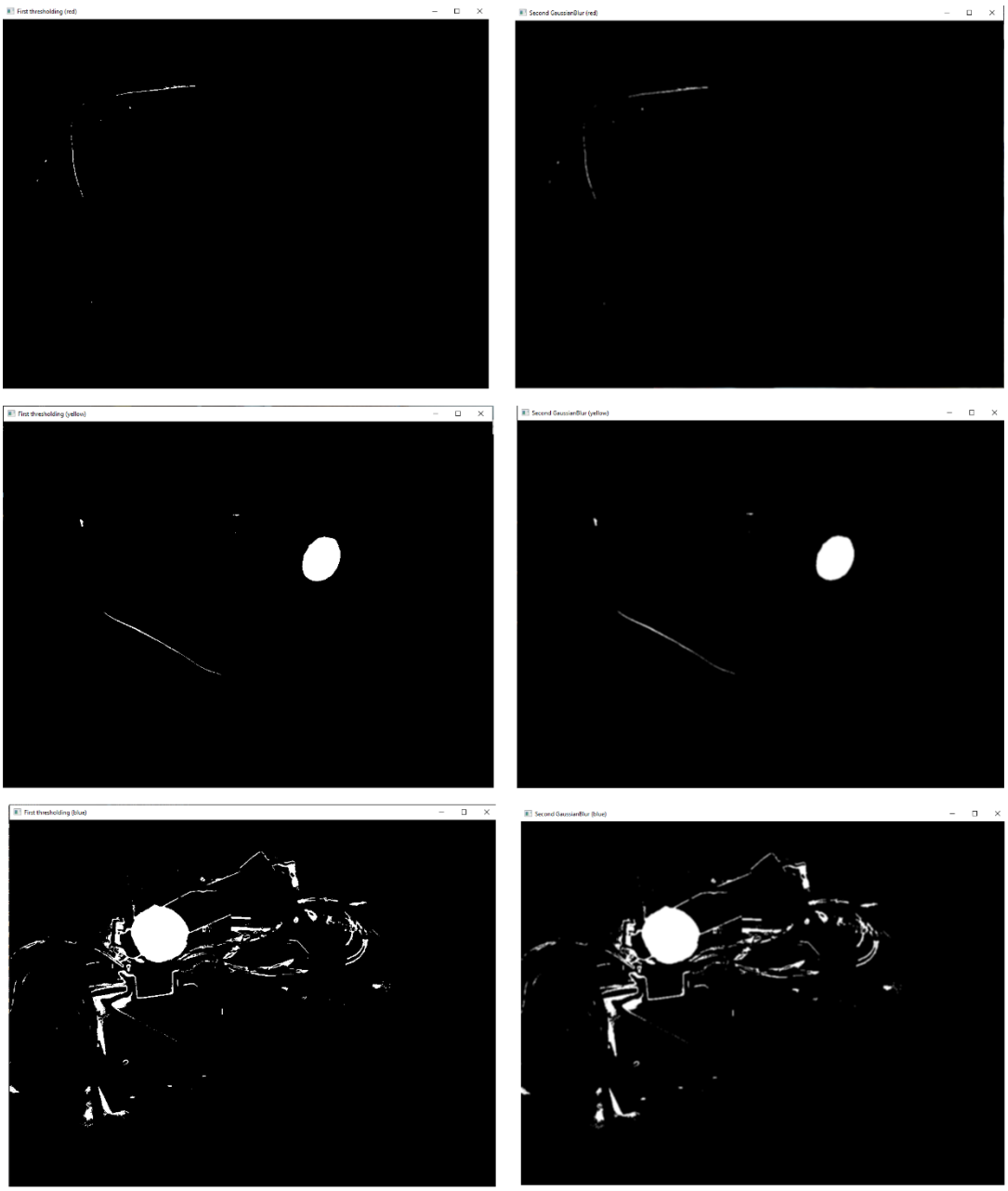

Figure 7.3.16 Red, Yellow and Blue Thresholding and Gaussian Blur on 1st Image

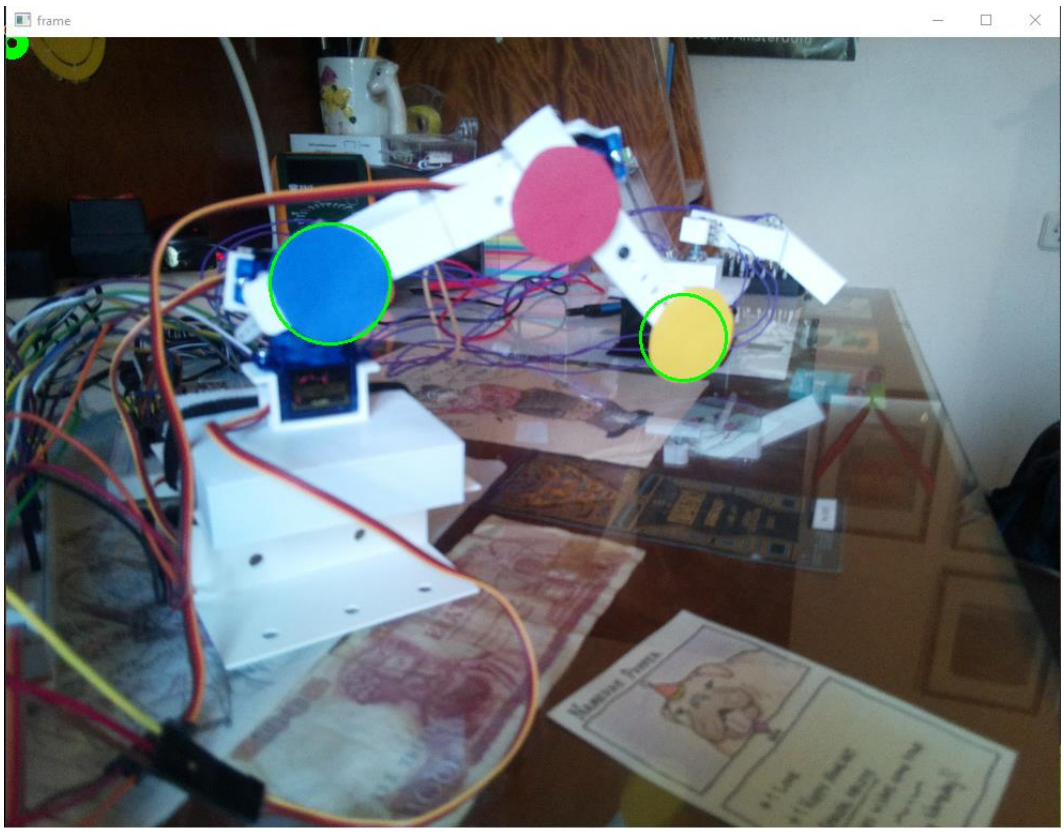

Figure 7.3.17 Circles Detected in 1st Image.

Under the first lightning condition, where the room is illuminated by the sun on afternoon hours, the detection is approximately 66% successful. With a little tinkering before the first color thresholding, the red circle can stand out more.

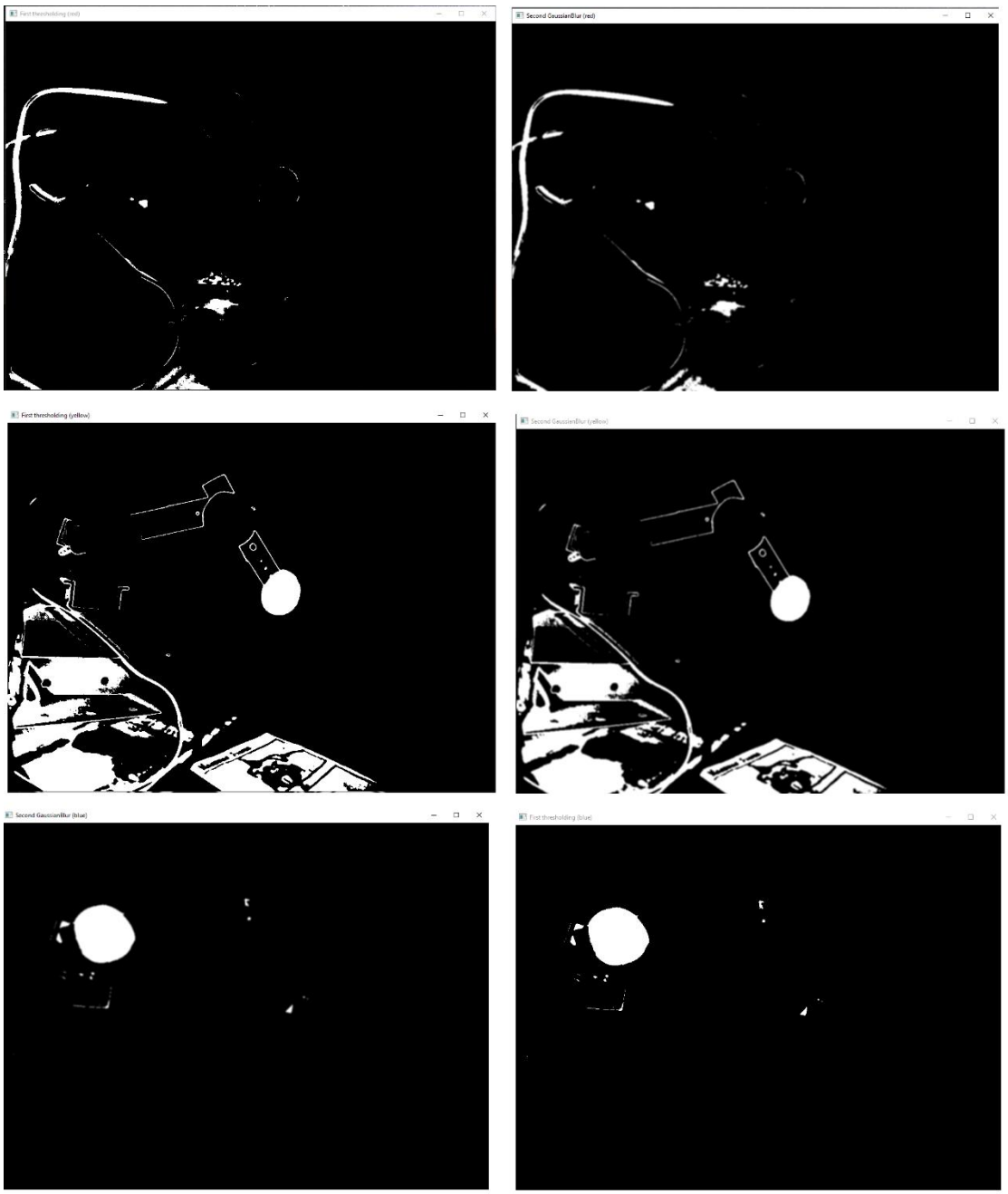

Figure 7.3.18 Red, Yellow and Blue Thresholding and Gaussian Blur on 2nd Image

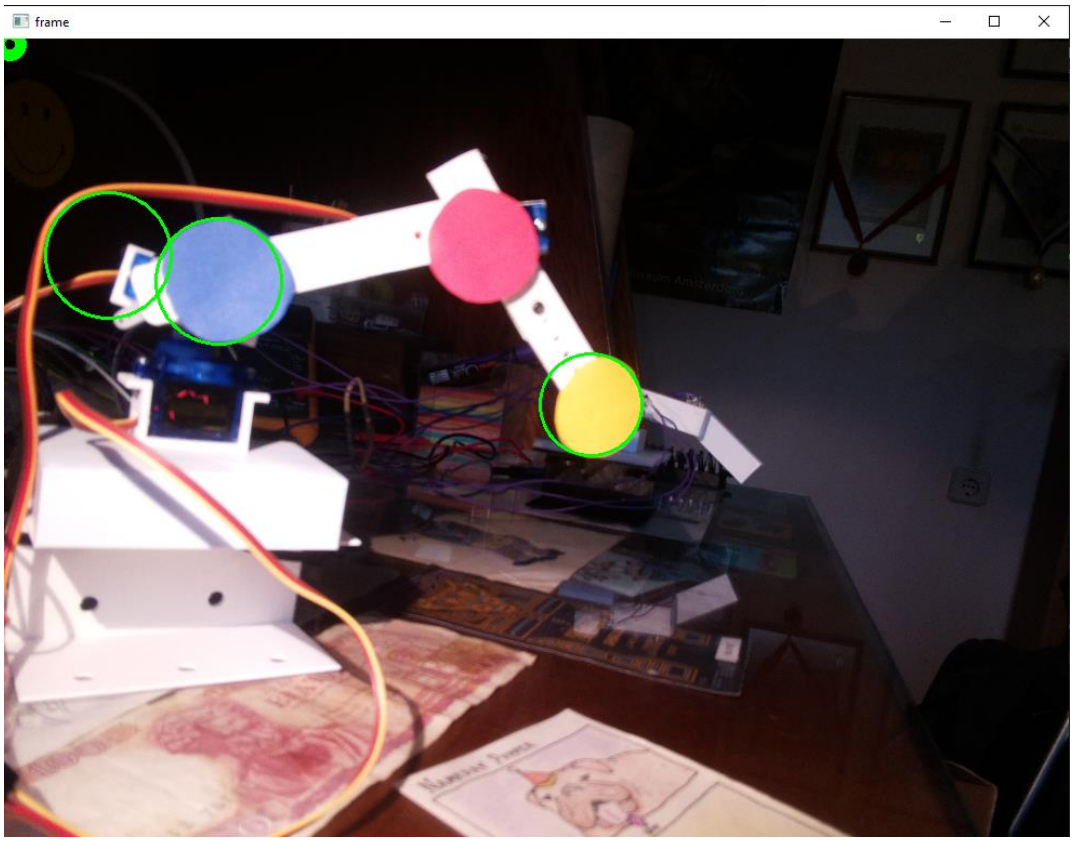

Figure 7.3.19 Circles Detected in 2nd Image.

The lighting source in the 2nd image faces directly the objective. With two true positives and one false positive result, this lightning scenario fosters for even better outcomes in a controlled environment.

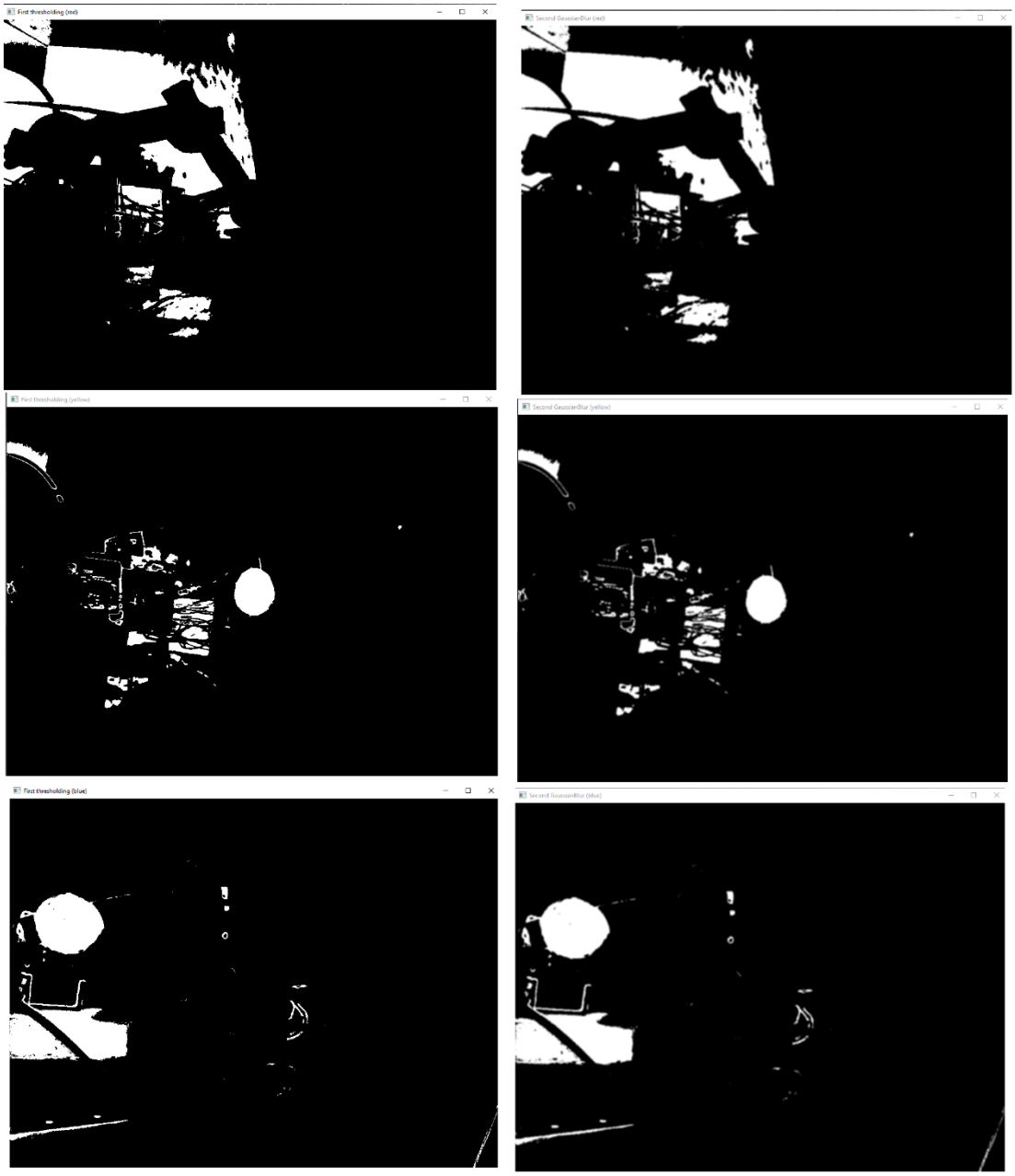

Figure 7.3.20 Red, Yellow and Blue Thresholding and Gaussian Blur on 3rd Image

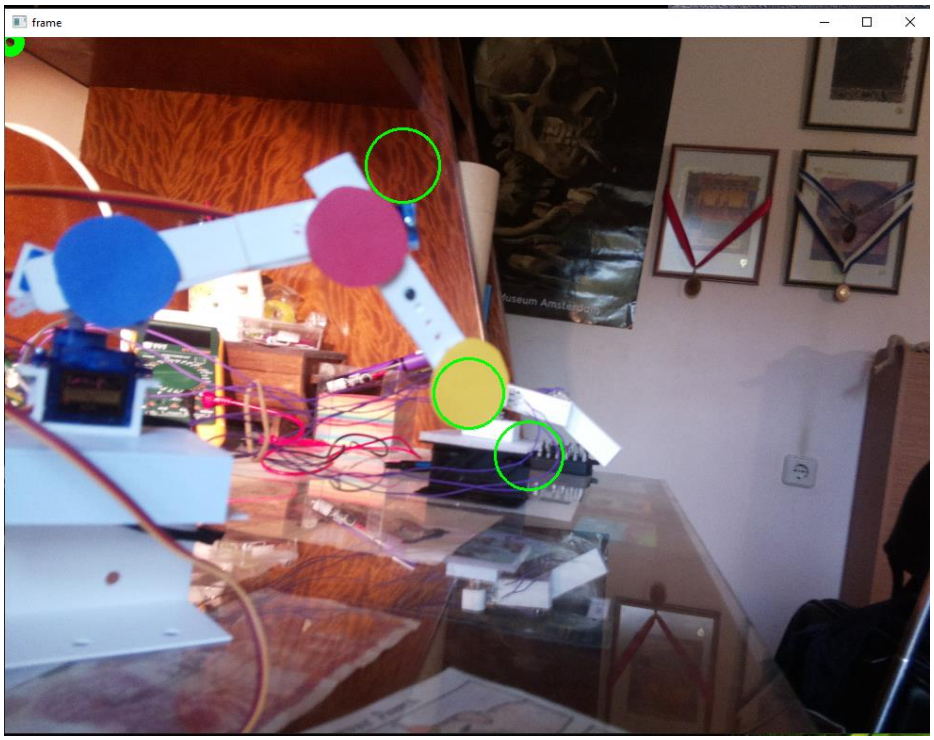

Figure 7.3.21 Circles Detected in 3rd Image

The third lightning condition is quite discouraging but confirms the conclusions derived from the second condition. The background in the image is the primary receiver of the lightning source. Therefore, it is more difficult to distinguish the foreground where our objective is positioned. Nevertheless it managed to successfully identify one circle.

CHAPTER 8

EPILOGUE

8.1 Conclusion

In this thesis, the implementation of a small-scale robotic arm was achieved. Cost efficient, environmentally friendly and highly adaptable within its structural limitations, this robotic arm can easily be tinkered with and introduced into many assembly lines. By adapting its dimensions, gear and redesigning the computer vision portion according to the required needs, it can easily be employed even outside the boundaries of a manufacturing industry.

Unfortunately, there are some limitations in the instance of the robot's wider integration into production lines. From a structural point of view, the mechanical parts and their connections require intricate work and knowledge in order to assemble the arm safely and effectively. In the case of an inexperienced designer, the cost can be raised exponentially due to the highly probable malfunctions of the parts due to poor planning or know-how. If an experienced mechanic were employed, his fee would raise the cost, nevertheless.

There is a reason why robots are only flourishing at heavily industrial countries. It is because the capital exists and along with it a brain drain, bringing together all the best minds for the right job , advancing the scientific field.

8.2 Future Work

For future improvements, a more detailed comparison of color spaces is vital. There is room for better testing of lightning in controlled environments, with great promise in finding optimal conditions for clearer detections. With an unlimited or even a slightly bigger budget, the servos must be the first to go, making room for sturdier ones with mechanical gears. Furthermore, the software can be complimented with a GUI for more user-

friendliness, and threading in the servo movement control for faster results. Lastly, the designing of the robotic arm can be more robust and tailor-made to adapt in each environment. By choosing the printed color according to the chosen color space's needs for a better object detection, the algorithm will be able to provide thriving results.

BIBLIOGRAPHY

[1] J. Manyika, S. Lund, M. Chui, J. Bughin, J. Woetzel, P. Batra, R. Ko and S. Sanghvi, "JOBS LOST, JOBS GAINED:WORKFORCE TRANSITIONS IN A TIME OF AUTOMATION," McKinsey & Company, 2017.

[2] M. Z. Miskin, A. J. Cortese, K. Dorsey, E. P. Esposito, M. F. Reynolds, Q. Liu, M. Cao, D. A. Muller, P. L. McEuen and I. Cohen, "Electronically integrated, mass-manufactured, microscopic robots," *Nature,* vol. 584, p. 557–561, 21 August 2020.

[3] Department of Commerce, United States of America, "Labor Costs – Just One Cost to Assess," 20 03 2018. [Online]. Available: https://acetool.commerce.gov/cost-risk-topic/labor-costs. [Accessed June 2021].

[4] G. Chryssolouris, N. Papakostas and D. Mavrikios, "A perspective on manufacturing strategy: Produce more with less," *CIRP Journal of Manufacturing Science and Technology,* pp. 45-52, 2008.

[5] Gunji Venkata Punna Rao, S. Nallusamy and M. Rajaram Narayanan, "Augmentation of Production Level using Different Lean Approaches in," *International Journal of Mechanical Engineering and Technology,* p. 360–372, 2017.

[6] D. Zunt, ""Who did actually invent the word "robot" and what does it mean?"," 2004. [Online]. Available: https://web.archive.org/web/20120204135259/http://capek.misto.cz/english/robot.html. [Accessed June 2021].

[7] T. Ort, Art and Life in Modernist Prague, New York, USA: Palgrave Macmillan US, 2013.

[8] A. Gasparetto and L. Scalera, "From the Unimate to the Delta robot: the early decades of Industrial Robotics," in *2018 HMM IFToMM Symposium on History of Machines and Mechanisms*, 2019.

[9] International Federation of Robotics (IFR), "Robot History," [Online]. Available: https://ifr.org/robot-history. [Accessed 16 07 2021].

[10] E. M. RUSLI, "Amazon.com to Acquire Manufacturer of Robotics," 19 03 2012. [Online]. Available: https://dealbook.nytimes.com/2012/03/19/amazon-com-buys-kiva-systems-for-775-million/. [Accessed 16 07 2021].

[11] International Federation of Robotics, "IFR presents World Robotics Report 2020," 24 September 2020. [Online]. Available: https://ifr.org/ifr-press-releases/news/record-2.7-million-robots-work-in-factories-around-the-globe. [Accessed July 2021].

[12] K. Shanker and A. Ghosh, "Flexible Automation and Robotics in Manufacturing," *IETE Journal of Research,* pp. 187-197, 1989.

[13] Raspberry Pi Foundation, 2021. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/. [Accessed June 2021].

[14] R. J. Portugal, "Breadboard for electronic components or the like". United States of America Patent USD228136S, 1973 .

[15] Microchip Technology Inc., "MCP3008," 2021. [Online]. Available: https://www.microchip.com/wwwproducts/en/MCP3008.

[16] A. Pandya, L. A. Reisner, B. King, N. Lucas, A. Composto, M. Klein and R. D. Ellis, "A Review of Camera Viewpoint Automation in Robotic and," *Robotics,* pp. 310-329, 2014,.

[17] L. Dongkeon , M. Takashi , T. Yasuhiro and H. Taeho, "3D Microfabrication of Photosensitive Resin Reinforced with Ceramic Nanoparticles Using LCD Microstereolithography," *Journal of Laser Micro / Nanoengineering,* 2006.

[18] Autodesk, "Fusion 360," 2021. [Online]. Available: https://www.autodesk.com/products/fusion-360/overview?term=1-YEAR. [Accessed June 2021].

[19] Raise 3D Technologies, "Raise3D N2 3D printer," 2018. [Online]. Available: https://www.raise3d.com/news/raise3d-n2-best-3d-printer-2018/. [Accessed June 2021].

[20] OpenCV, 2021. [Online]. Available: https://opencv.org/. [Accessed June 2021].

[21] OpenCV, "How OpenCV-Python Bindings Works," 2021. [Online]. Available: https://docs.opencv.org/3.2.0/da/d49/tutorial_py_bindings_basics.html. [Accessed June 2021].

[22] Commission Internationale de L'Eclairage, "CIE 015:2018 "COLORIMETRY, 4TH EDITION"," 2018.

[23] P. M. R. Caleiro, A. J. R. Neves and A. J. Pinho, "Color-spaces and color segmentation for real-time object recognition in robotic applications," *Electrónica e Telecomunicaçoes,* vol. 4, p. 940–945, January 2007.

[24] A. Rasouli and J. K. Tsotsos, "The Effect of Color Space Selection on Detectability and Discriminability of Colored Objects," *ArXiv,* vol. abs/1702.05421, 2017.

[25] A. Mukhopadhyay, I. Mukherjee and P. Biswas, "Comparing shape descriptor methods for different color space and lighting conditions," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing,* vol. 33, p. 89–398, 4 December 2019.

[26] T. Gevers and A. W. Smeulders, "Color-based object recognition," *Pattern Recognition,* vol. 32, no. 3, pp. 453-464, 1999.

[27] B. Cyganek, Object Detection and Recognition in Digital Images: Theory and Practice, New York: Wiley, 2013.

[28] A. S. Alqahtani, "Expectation Maximization Method for Effective Detection and Tracking of Object Using Machine Learning Technique for Secure Wireless Communication," *Wireless Personal Communications,* pp. 1-12, 28 March 2021.

[29] P. V. C. Hough, "Method and means for recognizing complex patterns". United States of America Patent US3069654A, 18 December 1962.

[30] "PiGPIO Python Home Page," 28 03 2021. [Online]. Available: http://abyz.me.uk/rpi/pigpio/python.html. [Accessed 06 2021].

APPENDIX

PYTHON SCRIPTS

1. servoandpot_nojit.py

```
from gpiozero.tools import scaled,multiplied
from gpiozero import Servo, MCP3008
from signal import pause
import sys

#The following imports help with the jitters of the servos
from gpiozero import Device
from gpiozero.pins.pigpio import PiGPIOFactory

Device.pin_factory = PiGPIOFactory('127.0.0.1')

try:

    pot = MCP3008(channel=0)
    servo = Servo(17, min_pulse_width=0.5/1000, max_pulse_width=2.5/1000)
    servo.source = (scaled(pot.values,-1,1))

    pot = MCP3008(channel=1)
    servo = Servo(27, min_pulse_width=0.5/1000, max_pulse_width=2.5/1000)
    servo.source = scaled(pot.values,-1,1)

    pot = MCP3008(channel=2)
    servo = Servo(5, min_pulse_width=0.5/1000, max_pulse_width=2.5/1000)
    servo.source = scaled(pot.values,-1,1)


# Exit script and clean GPIO pins
except KeyboardInterrupt:
    GPIO.cleanup()
    exit()
```

2. servoandpot_buttonrepeater.py

```
from gpiozero import MCP3008, Servo, Button
from gpiozero.tools import scaled
from itertools import cycle
from signal import pause
from gpiozero import Device
from gpiozero.pins.pigpio import PiGPIOFactory

Device.pin_factory = PiGPIOFactory('127.0.0.1')

class ServoRecorder(object):
    def __init__(self , servonum, potnum, onbutnum, offbutnum):
        servo_number = servonum
        pot_number = potnum
        on_button_number = onbutnum
        off_button_number = offbutnum
        self.recording = None
        self.record_button_on = Button(on_button_number)
        self.record_button_off = Button(off_button_number)
        self.record_button_on.when_pressed = self.start_recording
        self.record_button_off.when_pressed = self.stop_recording
        self.servo = Servo(servo_number, min_pulse_width=0.5/1000,
max_pulse_width=2.5/1000)
        self.servo.source_delay = 1/100
        self.pot = MCP3008(channel = pot_number)

    def record(self, values):
        self.recording = []
        for value in values:
            self.recording.append(value)
            yield value

    def start_recording(self):
        print('Recording values…')
        self.servo.source = self.record(scaled(self.pot.values, -1, 1))

    def stop_recording(self):
        print('Looping %d values' % len(self.recording))
        self.servo.source = cycle(self.recording)


recorder1 = ServoRecorder(17,0,20,6)
recorder2 = ServoRecorder(27,1,12,13)
recorder3 = ServoRecorder(5,2,16,4)
```

3. circle_ localization.py

```
import numpy as np
from cv2 import cv2
import math

path=r"C:\Users\ifige\Diploma\opencv_testing\assets\rsp_shot.jpg"

# Resize the image from the Pi
output_frame_big = cv2.imread(path)
original_src = cv2.imread(path, cv2.IMREAD_UNCHANGED)
scale_percentage = 30

width = int(original_src.shape[1] * scale_percentage / 100)
height = int(original_src.shape[0] * scale_percentage / 100)
final_size = (width, height)

output_frame = cv2.resize(original_src, final_size)
captured_frame = output_frame

# Conversion of Raspberry Pi's capture to BGR
captured_frame_bgr = cv2.cvtColor(captured_frame, cv2.COLOR_BGRA2BGR)

# First blur to reduce noise before the color space conversion
captured_frame_bgr = cv2.medianBlur(captured_frame_bgr, 3)

## Conversion into YCR_CB color space
#captured_frame_ycr_cb = cv2.cvtColor(captured_frame_bgr, cv2.COLOR_BGR2YCR_
CB)
#cv2.imshow(' YCR_CB color space',captured_frame_ycr_cb)
# Conversion into Lab color space
captured_frame_lab = cv2.cvtColor(captured_frame_bgr, cv2.COLOR_BGR2Lab)
cv2.imshow(' Lab color space',captured_frame_lab)
## Conversion into c1c2c3 color space
#im = captured_frame_bgr.astype(np.float32)+0.001
#c1c2c3 = np.arctan(im/np.dstack((cv2.max(im[...,1], im[...,2]), cv2.max(im[
...,0], im[...,2]), cv2.max(im[...,0], im[...,1]))))


# Threshold Lab picture by keeping only the red, yellow and blue pixels
captured_frame_lab_red = cv2.inRange(captured_frame_lab, np.array([20, 150,
150]), np.array([190, 255, 255]))
captured_frame_lab_yellow = cv2.inRange(captured_frame_lab, np.array([196,96
,127]), np.array([244, 162, 244]))
captured_frame_lab_blue = cv2.inRange(captured_frame_lab, np.array([96, 70,
10]), np.array([221, 151, 107]))


# Second blur for extra noise reduction and better circle localization
```

```
captured_frame_lab_red = cv2.GaussianBlur(captured_frame_lab_red, (5, 5), 2,
 2)
captured_frame_lab_yellow = cv2.GaussianBlur(captured_frame_lab_yellow, (5,
5), 2, 2)
captured_frame_lab_blue = cv2.GaussianBlur(captured_frame_lab_blue, (5, 5),
2, 2)


# Hough transform for circly detection
red_circles = cv2.HoughCircles(captured_frame_lab_red, cv2.HOUGH_GRADIENT, 1
, captured_frame_lab_red.shape[0] / 8, param1=100, param2=18, minRadius=5, m
axRadius=60)
yellow_circles = cv2.HoughCircles(captured_frame_lab_yellow, cv2.HOUGH_GRADI
ENT, 1, captured_frame_lab_yellow.shape[0] / 8, param1=100, param2=18, minRa
dius=5, maxRadius=60)
blue_circles = cv2.HoughCircles(captured_frame_lab_blue, cv2.HOUGH_GRADIENT,
 1, captured_frame_lab_blue.shape[0] / 8, param1=100, param2=18, minRadius=5
, maxRadius=60)

# Outlining of the circle (Assuming one was found. If several were found, th
e first one is printed)
if red_circles is not None:
    red_circles = np.round(red_circles[0, :]).astype("int")
    cv2.circle(output_frame, center=(red_circles[0, 0], red_circles[0, 1]),
radius=red_circles[0, 2], color=(0, 255, 0), thickness=2)

if yellow_circles is not None:
    yellow_circles = np.round(yellow_circles[0, :]).astype("int")
    cv2.circle(output_frame, center=(yellow_circles[0, 0], yellow_circles[0,
 1]), radius=yellow_circles[0, 2], color=(0, 255, 0), thickness=2)

if blue_circles is not None:
    blue_circles = np.round(blue_circles[0, :]).astype("int")
    cv2.circle(output_frame, center=(blue_circles[0, 0], blue_circles[0, 1])
, radius=blue_circles[0, 2], color=(0, 255, 0), thickness=2)

cv2.circle(output_frame, center=(5, 5), radius=10, color=(0, 255, 0), thickn
ess=10)


# placing the red circle as the corner:

ba = [ yellow_circles[0,0] - red_circles[0,0] , yellow_circles[0,1] - red_ci
rcles[0,1] ]
bc = [blue_circles[0,0] - red_circles[0,0] , blue_circles[0,1] - red_circles
[0,1]]
```

```
my_angle = math.atan2( (yellow_circles[0,1] - red_circles[0,1]) , (yellow_ci
rcles[0,0] - red_circles[0,0]) ) -  math.atan2( (blue_circles[0,1] - red_cir
cles[0,1]) , (blue_circles[0,0] - red_circles[0,0]) )
my_dot = np.dot ( ba, bc )
my_absolute_1 = math.sqrt ( ( ba[0])**2 + ( ba[1])**2  )
my_absolute_2 = math.sqrt ( ( bc[0])**2 + ( bc[1])**2 )
my_cos = my_dot / (my_absolute_1 * my_absolute_2)
my_angle = math.acos(my_dot / (my_absolute_1 * my_absolute_2)) # in radians

print("Center angle in red circle:", math.degrees(my_angle))  # converted in
 degrees

# Result
cv2.imshow('frame', output_frame)


# Button pressing for exit
cv2.waitKey(0)
cv2.destroyAllWindows()
```