EAST WEST UNIVERSITY

"Autonomous Robot for Garbage Detection and Collection"

By

Md. Noushanul Islam Bhuiyan

Nazfia Islam

Mahmudul Hasan Shohag

In partial fulfillment of the requirements for the degree of Bachelor of Science in Electrical and Electronic Engineering

Summer, 2017

Department of Electrical and Electronic Engineering

Faculty of Science and Engineering

East West University

East West University

Department of EEE

Autonomous Robot for Garbage Detection and Collection

by

Md. Noushanul Islam Bhuiyan	2013-1-80-024
Nazfia Islam	2013-1-80-032
Mahmudul Hasan Shohag	2013-1-80-036

Submitted to

Department of Electrical and Electronic Engineering
Faculty of Sciences and Engineering
East West University
Dhaka, Bangladesh

In partial fulfillment of the requirements for the degree of Bachelor of Science in Electrical and Electronic Engineering (B.Sc. in EEE)

Summer, 2017

Approved by

…….	.....……
Project Supervisor	Chairperson
Dr. Muhammed Mazharul Islam	Dr. Mohammad Mojammel Al Hakim

Approval

The project titled ‘Autonomous Robot for Garbage Detection and Collection’, submitted by Md. Noushanul Islam Bhuiyan (2013-1-80-024), Nazfia Islam (2013-1-80-032) and Mahamudul Hasan Shohag (2013-1-80-036) in summer, 2017, has been accepted as satisfactory towards partial fulfillment of the requirement for the degree of Bachelor of Science in Electrical and Electronic Engineering at East West University.

_____

(Dr. Mohammad Mojammel Al Hakim)
Professor and Chairperson,
Department of Electrical and Electronic Engineering,
East West University,
Dhaka-1212, Bangladesh

Acknowledgement

First of all, we are grateful to almighty Allah for giving us the opportunity to complete the project.

We would like to thank Dr. Muhammed Mazharul Islam, Assistant Professor, Department of Electrical and Electronic Engineering (EEE), East West University (EWU), Dhaka, our project supervisor, for his constant guidance, supervision, constructive suggestion and constant support during this project work.

We are also grateful to Dr. Mohammad Mojammel Al Hakim, Professor and Chairperson, Department of Electrical and Electronic Engineering (EEE), East West University (EWU), for the valuable suggestion and help. We would also like to thank our advisors, other faculty members, office staffs and EWU in general.

At last we want to thank our parents and all our friends for their moral support and helpful discussion during this project.

Authorization

We, hereby declare that we are the sole authors of this project. We authorize East West University to lend this project to other institutions or individuals for the purpose of scholarly research.

_____	_____	_____
Md. Noushanul Islam Bhuiyan	Nazfia Islam	Mahmudul Hasan Shohag

We further authorize East West University to reproduce this project by photocopy or other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

_____	_____	_____
Md. Noushanul Islam Bhuiyan	Nazfia Islam	Mahmudul Hasan Shohag

Abstract

This project develops an autonomous garbage detecting and collecting robot for cleaning in indoor spaces. Although other projects similar to this one has already been done, those have different limitations. Our robot uses a camera to continuously scan the room for objects/ garbage. It uses image processing to detect objects even if these are not geometrically shaped. The detection is also invariant to the object's color.

Once an object has been detected, the robot moves to the object's location and picks up the object in its bin. Then the robot moves back to its starting position and resumes scanning the room for garbage.

In order to implement image processing on a robot, we needed a powerful processor, so we used 'Raspberry Pi' – a credit card sized computer, to build our system. We also used sophisticated image processing software/ operations to detect the object. Since the computer runs in Linux operating system, we used 'python' programming language to develop our program.

Table of Contents

APPROVAL ..... **I**

ACKNOWLEDGEMENT ..... **II**

AUTHORIZATION ..... **III**

ABSTRACT ..... **IV**

CHAPTER 1 : INTRODUCTION **1**

1.1 Background: 1

1.2 Objective 2

1.3 Features of the proposed robot 2

CHAPTER 2 : WORKING PRINCIPLE **3**

2.1 Object Detection and Location Identification 4

2.1.1 Image Capture 4

2.1.2 Image Processing 5

a) Image Subtracting 5

b) Image Filtering 6

c) Distance and Polar Angle Measurement 9

2.2 Robot Locomotion 10

2.3 Collection of Object 11

CHAPTER 3 : HARDWARE DESIGN AND CONSTRUCTION **13**

3.1 Raspberry Pi 13

3.1.1 Advantages 15

3.1.2 Pin Diagram 15

3.2 Design of the Robot Body 16

3.2.1 Robot Arm 16

3.2.2 Gripper 18

3.2.3 Robot Base 18

3.2.4 Servo Motors 19

3.2.5 DC Motor and Wheel 20

3.3 Electrical Circuit 20

3.4 Power Supply Circuit 20

3.4.1 Power Source 20

3.4.2 L298 Motor Driver 22

CHAPTER 4 : SOFTWARE AND PROGRAMS **25**

4.1 Software 25

4.2 Programming Algorithms 25

4.2.1 Image Processing 25

4.2.2 DC Motor Controlling 25

4.2.3 Servo Motor Controlling 27

CHAPTER 5 : CONCLUSION **28**

5.1 Conclusion 28

5.2 Future Development 28

REFERENCES **29**

APPENDIX A: RASPBERRY PI V3 OPERATING SYSTEM SETUP GUIDE **31**

APPENDIX B: PYTHON-OPENCV CODE **33**

List of Figures

Figure 2.1: Working principle of our robot 3
Figure 2.2: Camera coverage of the floor area 4
Figure 2.3: Image capturing scheme 5
Figure 2.4: Example of typical reference image and image taken when object is present 6
Figure 2.5: Subtracted grayscale images 6
Figure 2.6: Image after morphological operation 7
Figure 2.7: Image after binary thresholding 8
Figure 2.8: Detected contour and center of the object 8
Figure 2.9: Distance calculation 9
Figure 2.10: Subroutine of robot locomotion and object collection 11
Figure 2.11The gripper of the robot 12
Figure 3.1: Proposal diagram of the robot. 13
Figure 3.2: 'Raspberry Pi' 3 model B 14
Figure 3.3: Pin configuration of 'Raspberry Pi' 15
Figure 3.4: Schematic of the robot's arm 17
Figure 3.5: Image of the robot arm 17
Figure 3.6: Image of the constructed gripper 18
Figure 3.7: The webcam used in our project 19
Figure 3.8: Circuit diagram for connection to 'Raspberry Pi' 21
Figure 3.9: Power circuit diagram 22
Figure 3.10: L298 connection diagram to dc motor 23
Figure 3.11: L298 Motor Driver 23
Figure 3.12: Constructed robot 24
Figure 4.1: Algorithm of image processing and object detection 26

List of Tables

Table 2.1: Distance of object's center in feet and in pixels 10
Table 3.1: Specification of 'Raspberry Pi' 3 14
Table 3.2: Component list 16
Table 3.3: Camera specification 18
Table 3.4: Pulse width and duty cycle corresponding to the motor's angular position 19
Table 4.1: Truth table of a motor operation (single motor only) 26
Table 4.2: Truth table for GPIO pin status to drive the L298 motor driver 27

Chapter 1 : Introduction

1.1 Background:

The invention of robots brought a massive change in the industrial production in the 21st century. Introducing heavy industrial robots in factories have increased production speed where it shrinks the labor cost, queue length and manufacturing cost. At the beginning, robots were only used in commercial factories for mass production, which is larger in size and shape [1]. Nowadays, robots are becoming a part of the service sector and the structural size of the robot is suitable for the household.

Cleaning robots are now becoming popular in robotics market due to a suitable size, affordable price, and its applications. Autonomous robots are more sophisticated in home service sections. When it comes to cleaning our surroundings, autonomous robots can be very handy as an alternative to manual cleaning. There are already some robots like autonomous vacuum cleaners to clean dust [2]. When it is turned on, it automatically searches every corner of the room for dust. The major demerit of this robot is that it cannot pick up garbage other than dust. Also, it cannot detect whether a floor area is dusty or not. As a result, it cleans the whole room every time, which wastes a lot of power. It uses a simple sensor to avoid obstacles in its path, yet the product is very expensive and out of the reach of most ordinary people.

There was a project on autonomous garbage collecting by “Volvo Groups” named “Robot Based Autonomous Refuse handling” or ROAR [3]. It uses both land-based robots and drones to detect and collect garbage. The drone uses GPS (Global Positioning System), LIDAR (Light Detection and Ranging), camera, motion sensor combined with odometers to detect trash can and using those data the land-based robot collects and dump those trash can. This robot is large in size and used in outdoor garbage bin collection.

AGATOR (Automatic Garbage Collector) is another similar type of robot that can grab garbage [4]. AGATOR is designed as a robot model of the automated garbage collector. The accuracy of the detection is very low and the response speed is also slow. Another drawback is that, if the object is rough, irregular in shape, very small or thin or not homogeneous, then this robot cannot collect the object.

Another micro-controller based garbage collector is a tricycle robot designed to detect and collect garbage [5]. It has a vacuum dust cleaning feature.

The object identifier and collector robot can identify a specific object by image processing, approach and collect the object [6]. The robot uses camera and a single board computer for image processing and a robot arm to capture the object. This robot is designed only for a specific object collection.

ÇöpToplayan Robot - Garbage Collector is another robot that can collect garbage which is below 10 cm heights. It can work in a specific area (around 2m×2m area). It can detect and collect a maximum of 10 objects at a time within 10 cm distance range [7].

There also many robots are available such as Autonomous Garbage Collector Robot [8], Garbage Collection Robot on the Beach using Wireless Communications [9], Object locator and collector robotic arm using artificial neural networks [10]. These robots have some structural difficulties and functional limitations.

The Autonomous Garbage Detecting and Collecting Robot can detect unwanted object in a specific space and using image processing the object location is calculated, approach to the object and grab it. It can collect one object at a time and return to its initial position.

1.2 Objective

The objective of this project is to develop a robot to clean garbage, particularly, garbage on a floor. Fundamentally, the robot exhibits the following functionalities, i.e. (1) it detects whether a foreign object (which it considers garbage) has fallen on the floor and determines its location relative to the robot, (2) it travels to that object's location, (3) it grabs the object and drops the object in a collection bin, and finally, (4) it returns to its starting position. The robot is currently designed to be used in indoors only and can operate in any area of planar floor surface. It can detect and pick-up objects even if its shape is non-regular.

1.3 Features of the proposed robot

The garbage detection is performed using a camera which is mounted on the robot. The camera takes snapshots of its surrounding at regular intervals and applies image processing techniques to detect any garbage. If the robot detects any garbage, it will measure the distance between the robot and garbage. Using the measured distance, the robot moves to object and robot arm grabs the garbage. The robot returns back to its previous position when garbage the collection is done. The robot continuously searches garbage until it turns off.

Following are some features of our system:

- **Single Board Computer:** Raspberry Pi is used for data processing and controlling. It is a small computer which can run a Linux operating system.
- **Object Detection:** A camera is used to capture images. Using the image processing, the robot detects objects. To detect garbage, it doesn't need to move, it only requires a clear vision using it's on board camera.
- **Robot Arm:** In order to pick an object, the robot has an arm with gripper with two degrees of freedom.
- **Power Consumption:** The robot requires low electrical energy to operate. Portable battery is used for energy source.

Chapter 2 : Working Principle

The robot's working principle can be divided into three major sections:

a) (Foreign) object detection and location identification
b) Robot Locomotion
c) Collection of object

Initially, the robot sits in a corner of the room (this is the 'home' position) and continuously scans the floor of the whole room for any (foreign) object. When it detects a new object on the floor, it identifies the object's location with respect to the 'home' position.

The robot then travels to the object's location to grab and collect the object in the collecting bin mounted on the robot body. It then gets back to its 'home' position and restarts the scanning of the room. Currently, it can collect only one object at a time.

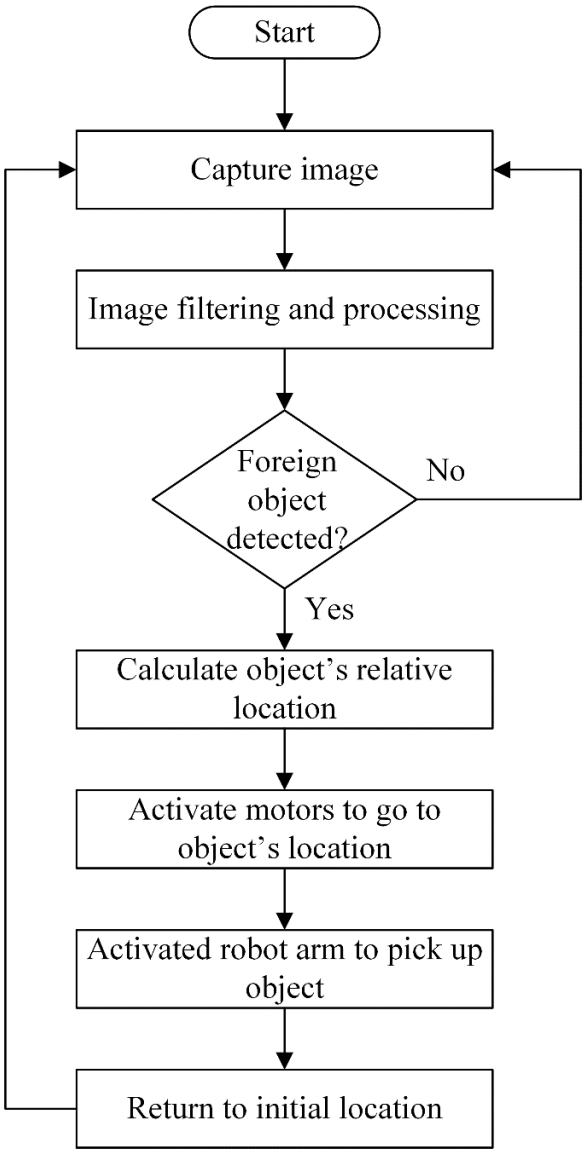


Figure 2.1: Working principle of our robot

2.1 Object Detection and Location Identification

In order to detect a (new or foreign) object, the robot uses image processing. It constantly scans the floor by rotating a camera mounted on the robot and thus taking pictures at pre-defined angular intervals. It then compares the captured image with a reference image (taken at an earlier time). Comparing the two images, the robot can identify whether a new object has been introduced in the picture frame. The flowchart of the overall working principle is shown in Figure 2.1.

2.1.1 Image Capture

A camera is mounted at the front of the robot to capture images. When the camera is kept at horizontal level with respect to the floor, only about $\pm 20^\circ$ on both sides of the floor area is covered in the frame. To cover a larger area of the floor, the camera axis is made to rotate in 30° angles. Figure 2.2 shows the camera coverage.

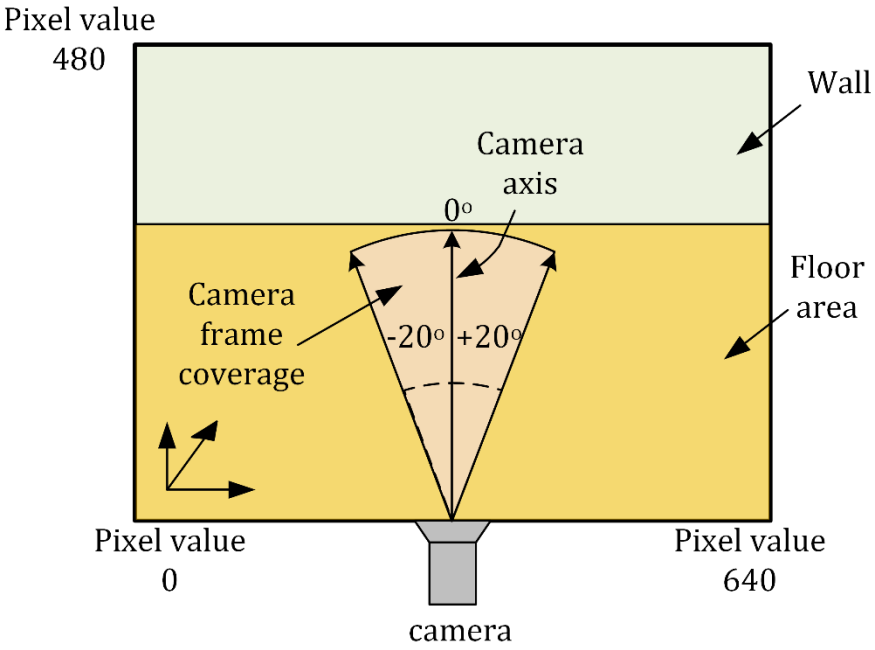


Figure 2.2: Camera coverage of the floor area

This rotation process eliminates the need for taking continuous video, and reduces power consumption. Figure 2.3 shows the scanning scheme by the camera. A total of seven images are captured as the camera rotates by 180 degrees. These seven images together act like a panoramic view of the robot's front. The whole rotation takes about 2 to 3 seconds.

After the first scan, the images are stored as 'reference' images. In subsequent scans, the robot continuously takes seven pictures just like before, and compares each image to its corresponding reference image to identify any new object in its frame. The reference images will not be updated until the robot detects any object and finish the entire process. Each image has a width of 640 pixels and height of 480 pixels. The low resolution of the image reduces memory requirement for image storing.

In our project, we have used a webcam to take still images because these are comparatively cheaper and controlling webcams from the 'Raspberry Pi' is relatively easier.

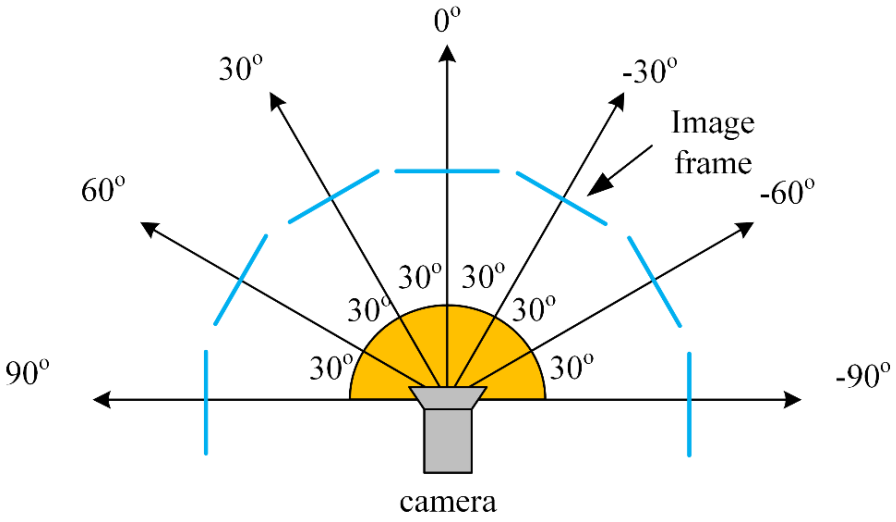


Figure 2.3: Image capturing scheme

2.1.2 Image Processing

The robot detects unwanted object in front using image processing. The processing is mainly divided into three steps,

a) Image Subtracting
b) Image Filtering
c) Distance and Polar Angle measurement

a) Image Subtracting

At first, the reference image and current image are converted into greyscale from RGB. Then the reference image is subtracted from current image. The reason for converting RGB images into grey is to eliminate any shadows caused by uneven lighting conditions. The resulting output image will be a matrix which will have a non-zero sub-matrix where the object is located. Figure 2.4 shows typical images.

The output image is also sized 640x480 and is a zero matrix if the reference image and current image are equal or same. Ideally, if the output matrix is non-zero then the reference image and current image are not same. This means that there is an object (or something) that causes the reference image matrix and the current image matrix to become unequal. In reality, there are noises present in the output image which induce a false detection. These noises can cause non-zero sub-matrices to appear in the output image even if there are no objects present. To avoid

Figure 2.4: Example of typical reference image and image taken when object is present

this issue, the subtracted image is filtered in the next step. Figure 2.5 shows the subtracted image of Figure 2.4.

b) Image Filtering

The filtering is performed considering the morphology of the noisy images. Morphological operations can be performed on both binary and grayscale images.

Morphological Operations

Images contain many imperfections, and during subtraction of the grayscale images, it will be distorted by noise and texture. Morphological operations remove the imperfections in images by accounting for the form and structure of the image [11]. The operation is mathematically defined as the combination of an images and a structuring element using set operations. In our case, the structuring element is heuristically defined as a 5x5 matrix of value ‘1’.

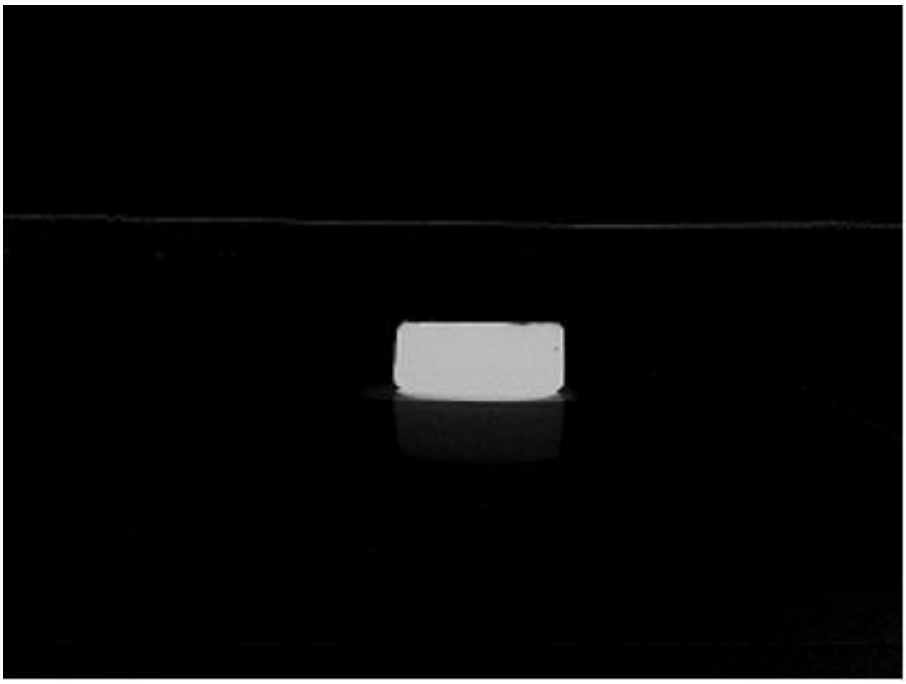


Figure 2.5: Subtracted grayscale images

There are two fundamental kinds of morphological operations, i.e. erosion and dilation. Using these two operations, other operations like opening and closing can be performed.

Erosion operation mainly erodes the edges of a foreground (here, non-zero parts, i.e. the object). Dilation operation does the opposite and grows the boundary. We used an 'opening' operation on the image. Opening is an erosion followed by dilation and can be considered as a softer erosion operation of the image.

For an image $\Gamma \in \mathbb{R}^2$and structuring element $\Lambda \in \mathbb{R}^2$, the erosion operation is defined as,

$$\Gamma \ominus \Lambda = \{\eta \in \Psi | \Lambda_\eta \subseteq \Gamma\}$$

Here Λ_η is the translation of Λ by the vector η, i.e.

$$\Lambda_\eta = \{\lambda + \eta | \lambda \in \Lambda\}, \forall \eta \in \Psi$$

On the other hand, a dilation is defined as,

$$\Gamma \oplus \Lambda = \{\eta \in \Psi | (\Lambda^s)_\eta \cap \Gamma \neq \emptyset\}$$

Here, Λ^s is the symmetric of Λ defined as,

$$\Lambda^s = \{\lambda \in \Psi | -\lambda \in \Lambda^s\}$$

Therefore, the opening operation is defined as,

$$\Gamma \circ \Lambda = (\Gamma \ominus \Lambda) \oplus \Lambda$$

The effect of the 'opening' operation is shown in Figure 2.6. After this operation, object detection becomes possible from the subtracted image.

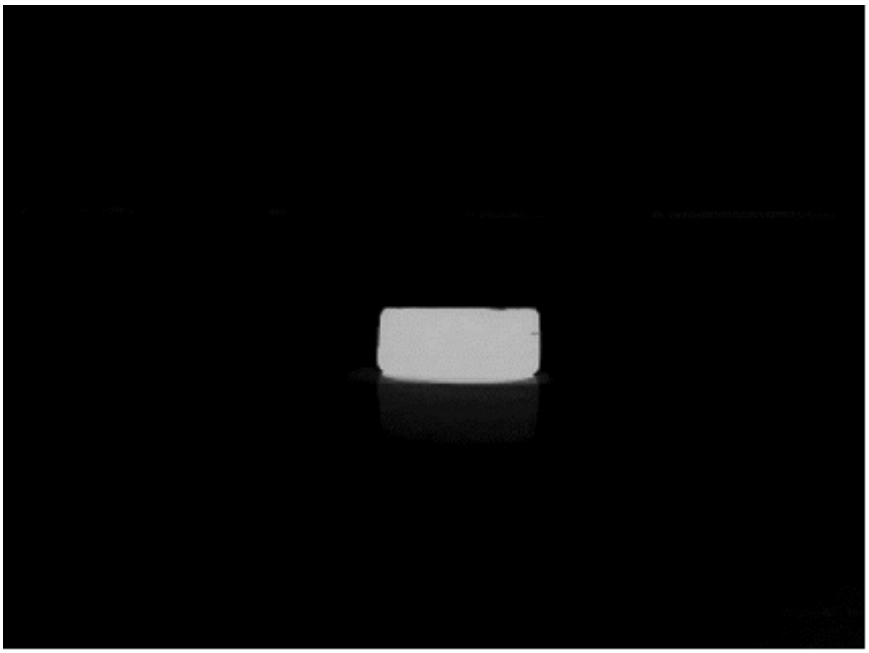


Figure 2.6: Image after morphological operation

Binary Thresholding

Even after the morphological processing, the grayscale image contains some noise. To clear the rest of the noises, the output image is converted to binary image. Binarization is performed by segmenting an image based on the grayscale pixel intensity. The threshold of the image can

Figure 2.7: Image after binary thresholding

decide what sized object can be detectable. A low threshold will erase the object in the image. The threshold in our project is determined heuristically and set to the value 80. At this value of threshold, the program is able to detected objects. Figure 2.7 shows the subtracted image after binary thresholding. Now, the object boundary and the center is detected from the binary image

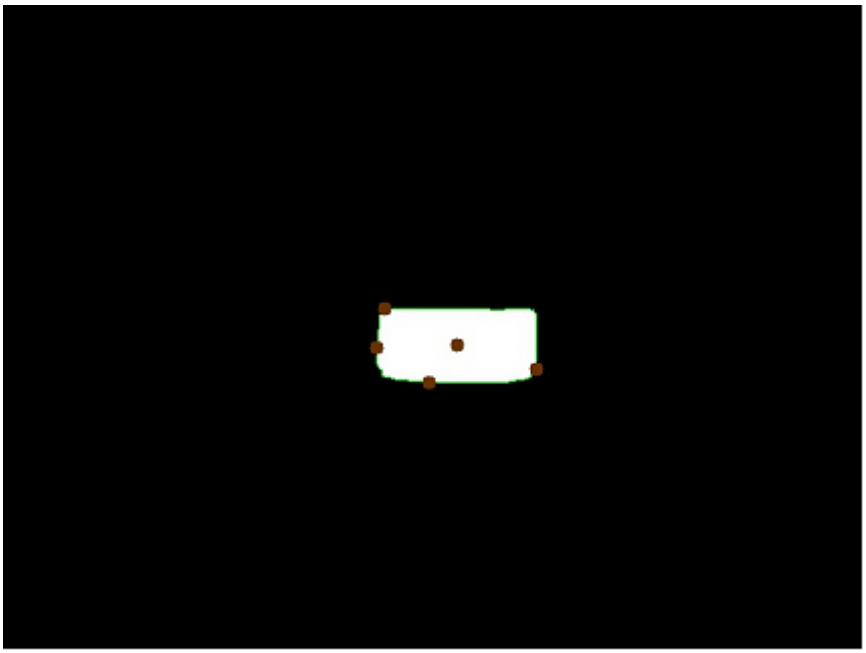


Figure 2.8: Detected contour and center of the object

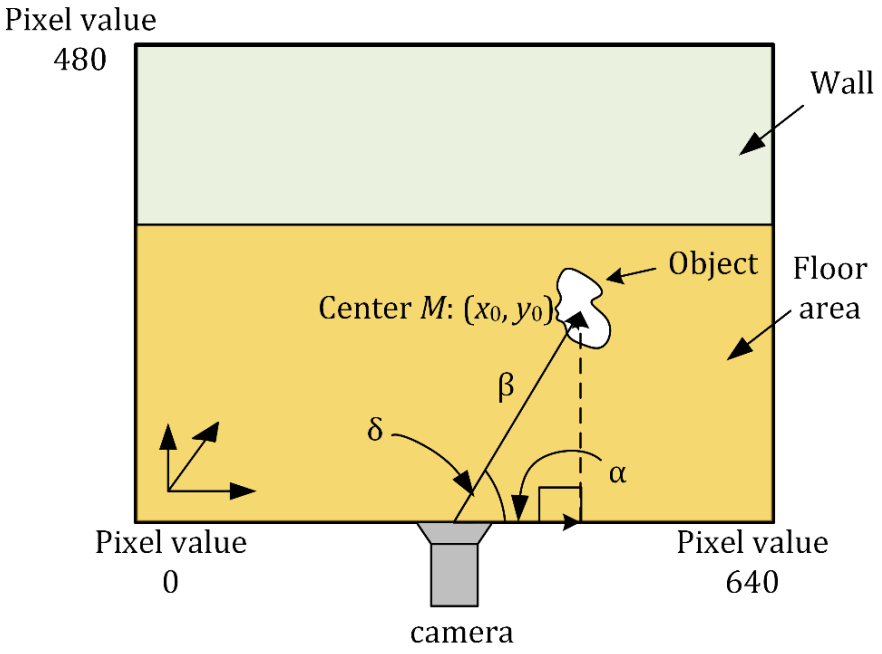
[12]. Figure 2.8 shows the object image with contour and the center of the object marked. The object region will be white or binary number of 1. So, we need find all the co-ordinates of the edge of all segments which value is 1. Once all the edge-points of the object are found, the center location (in terms of pixel values) can be found.

c) Distance and Polar Angle Measurement

The location of the object relative to the robot's position is next identified by calculating the distance to the center and the angle of the center relative to the camera axis. Figure 2.9 shows the object's center location as (x_0, y_0) in pixel values. The distance to the object's center (β) and the polar angle δ can be calculated from the center location (x_0, y_0) and the camera location (320, 0) as,

$$\beta = \sqrt{\alpha^2 + y_0^2} = \sqrt{(x_0 - 320)^2 + y_0^2}, \qquad \text{and,} \qquad \delta = \tan^{-1}(y_0/x_0)$$

As mentioned earlier, the picture frames of the camera have a size of 640 pixels (width) and 480 pixels (height). The camera location is at the bottom of the frame of the image. To understand the relationship between the pixel value and the object's distance, experiments were performed. An object was placed at different distances in front of the camera, and the distance in pixel values were measured from the image.



β: distance of object's center from camera
α: distance along horizontal direction to the projection of object from camera
δ: angular displacement of object from camera position

Figure 2.9: Distance calculation

Table 2.1 shows the results obtained from the experiment for the radial distance of object's center in pixels and in feet.

The object's vertical distance in feet and in pixels fit an exponential model obtained heuristically and described as,

$$\beta(y_0) = ae^{by_0} + ce^{dy_0}$$

Here, β is the vertical distance of the object's center (in feet), y_0 is the vertical location of the center in pixels, and $a = 0.08568, b = 0.0071, c = 0.0000178, d = 0.571$. The fitting is non-linear because the floor looks inclined from the point of view of the camera, so distances are compressed as we move away from the camera.

Table 2.1: Distance of object's center in feet and in pixels

Radial distance of object's center from camera (in feet)	Center location (in pixels)				
	$\delta = 0°$	$\delta = 10°$		$\delta = 20°$	
	y_0 $(x_0 = 0)$	x_0	y_0	x_0	y_0
0	0	0.17	0.98	0.34	0.94
1	8	0.35	1.97	0.68	1.88
2	126	0.52	2.95	1.03	2.82
3	164	0.69	3.94	1.37	3.76
4	184	0.87	4.92	1.71	4.7
5	200	1.04	5.91	2.05	5.64
6	208	1.22	6.89	2.39	6.58
7	212	1.39	7.88	2.74	7.52
8	215	1.56	8.86	3.08	8.46
9	219	1.74	9.85	3.42	9.4
10	223	0.17	0.98	0.34	0.94

A similar experiment was also performed for finding the relationship of the object's distance in feet and pixels along the horizontal direction. From the experiment, it was found that the horizontal relationship is liner and can be expressed as,

$$\alpha(x_0) = |x_0 - 320|/64$$

Where, α is the horizontal distance of the object from camera, and x_0 is the object's center location in the horizontal direction.

2.2 Robot Locomotion

The robot movement is done using dc motors. The dc motor is controlled to move the robot forward, backward, and to turn the robot left and right. By controlling the input voltage of the dc motors, the direction of the robot can be set. By adjusting the input voltage of two dc motors, turning is performed.

The robot will travel the distance between the object and robot, and then the robot arm will pick up the object. After picking up the object, the dc motor will perform reverse action to go back to its initial position. If it requires any angular adjustment, then it will adjust the angle for its initial position. Figure 2.10 shows the subroutine for robot locomotion and object collection.

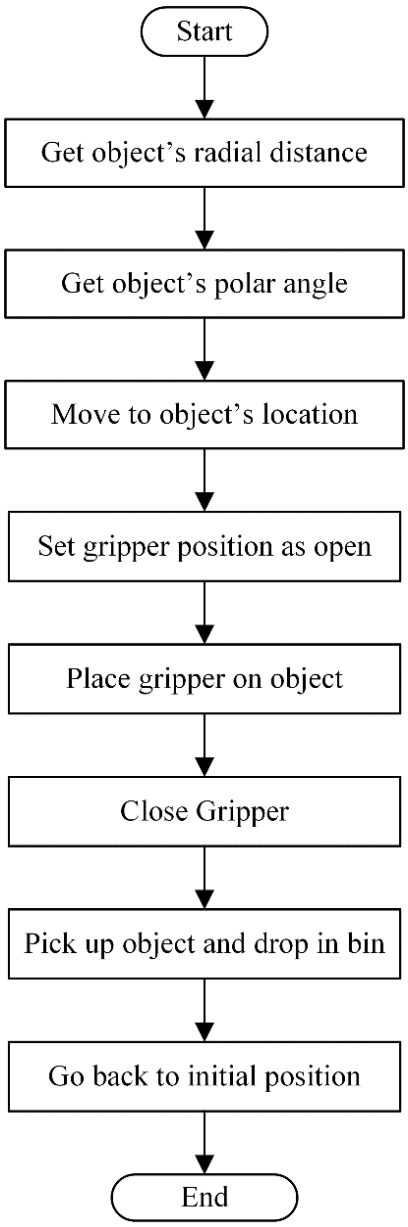


Figure 2.10: Subroutine of robot locomotion and object collection

If the detected object is at the left of the robot, then it will rotate left. The same procedure will be followed for right turn of robot.

2.3 Collection of Object

The two-degree freedom robot arm consists of a griper and elbow as shown in Figure 2.11.The elbow is used to adjust the griper position to grab the object. The robot arm rotates after grabbing the object to place it in a bin situated at the back of the robot.

The robot will follow exactly the reverse process of moving to object's location step to come

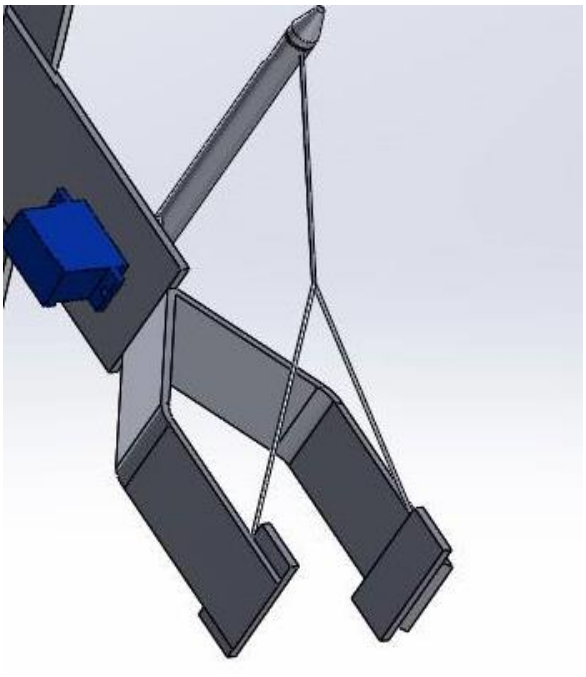


Figure 2.11 The gripper of the robot

back to its initial position. This function will be activated after the robot arm action.

Chapter 3 : Hardware Design and Construction

This section describes the components used to build the robot. In this project, we need a microcontroller to control the robot. However, our project involves too many image processing tasks which are strenuous for a normal microcontroller, if not impossible. We needed a processor which is more powerful than a conventional microcontroller yet small in size (for mobility of the robot). The concept of how the robot would look was first developed by us as shown in Figure 3.1.

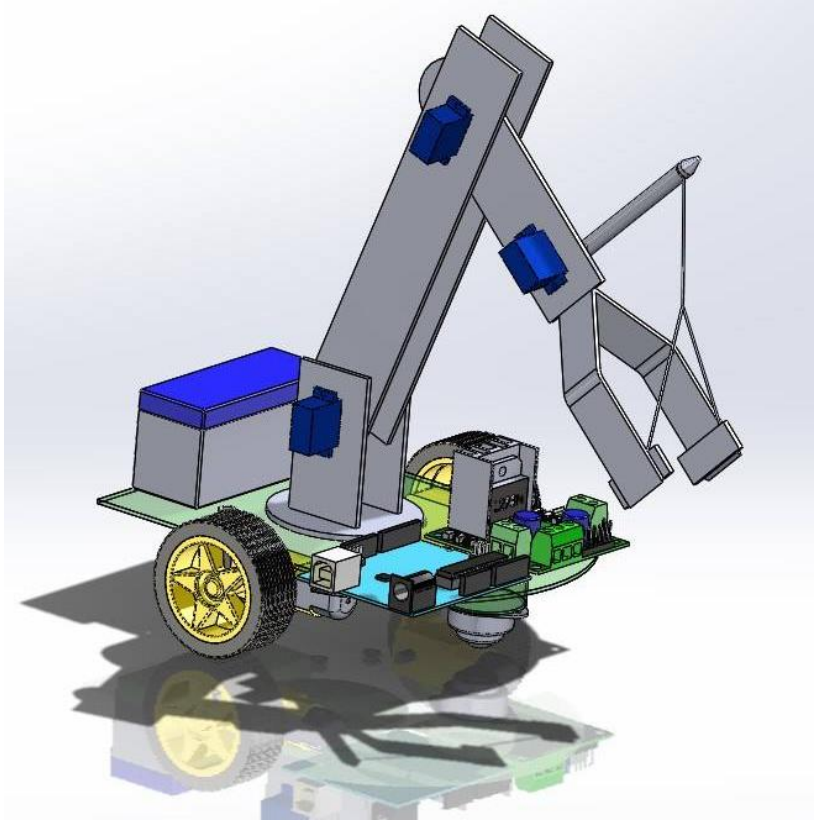


Figure 3.1: Proposal diagram of the robot.

'Raspberry Pi' is a single board computer with the size of a credit card, yet it works as a normal computer at a relatively low price. Although it is slower than a modern laptop or desktop, but it can provide all the expected abilities of a computer. It is a complete computer on which 'Linux' operating system can be run and it also consumes less power. For these reasons, 'Raspberry Pi' is used in this project.

3.1 Raspberry Pi

'Raspberry Pi' is developed by 'Raspberry Pi' foundation. This single board computer has option to add other peripherals (keyboard, mice etc.) to use the device as an alternative to conventional computer. We used 'Raspberry Pi' 3 model B in our project (shown in Figure 3.2), which is the latest version of 'Raspberry Pi' series. Besides peripheral facilities, 'Raspberry Pi' has input and output pins to get signals from analog or digital devices and to control any electrical device. Table 3.1 shows the specification of the 'Raspberry Pi'.

Figure 3.2: 'Raspberry Pi' 3 model B [13]

Table 3.1: Specification of 'Raspberry Pi' 3 [14]

Features	
Generation	3rd
CPU	1.2 GHz 64-bit Quad-Core ARM Cortex-A53
Instruction set	Broadcom BCM2837
GPU	400MHz VideoCore IV
RAM	1GB (SDRAM)
Storage	Micro-SD
Ethernet	10/100
Wireless	802.11n / Bluetooth 4.0
Video Output	HDMI/ Composite
Power Rating	800 mA (4.0 Watt)
Power Source	5V via Micro USB
Dimension	85.60mm x 56.5mm x17 mm
Weight	45 g
GPIO	40

3.1.1 Advantages

- The ‘Raspberry Pi’ draws about 4 to 5 watts of electricity, which is quite low.
- It uses an SD card for storage, which is fast and has no moving parts like computer disks.
- There are also no fans required for heat dissipation, as heat sinks (if required) are capable of dissipating the generated heat.
- There are several status lights on the Pi's motherboard.
- The Pi has phenomenal community support. Support can be obtained quite easily for the hardware and/or GNU/Linux software that run on the Pi.

3.1.2 Pin Diagram

Here, we only discussed the pins used in the project. There are some other pins which are not used in this project, and so we do not discuss those here. Figure 3.3 shows the pin configuration of ‘Raspberry Pi’.

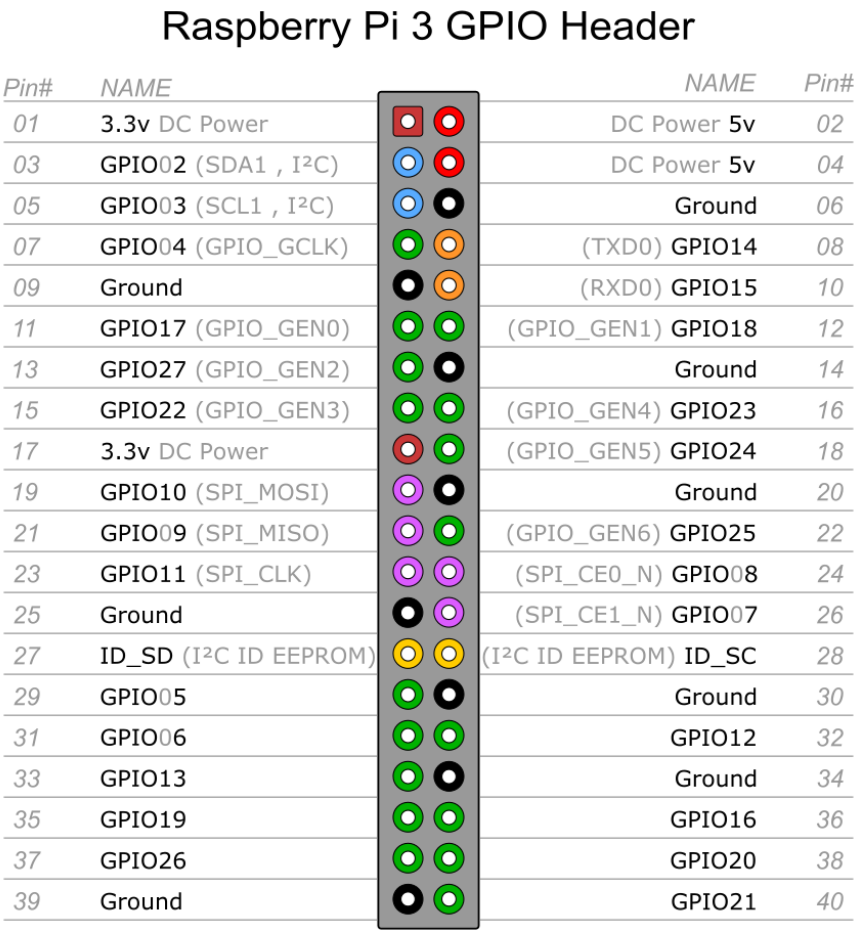


Figure 3.3: Pin configuration of ‘Raspberry Pi’ [15]

5v Power Supply Pin: Physically, pin numbers of 2 and 4 are use as the 5v power supply. These pins are output pin. The 5v power pins are connected directly to the Pi’s power input. These power pin provide about 1.5A current [16].

GPIO Pin: There are 17 GPIO (General Purpose Input/Output) pins in 'Raspberry Pi'. Pin numbers of 7, 11, 12, 13, 15, 16, 18, 22, 29, 31, 32, 33, 35, 36, 37, 38, and 40 are used as the GPIO port. When the GPIO pins are configured to work as outputs, the 'Raspberry Pi' will set the pin to either a voltage close to 0V or close to 3.3V [17].

Reserved Pin: Physically, pin numbers of 27 and 28 are used as reserved pins. When any GPIO pin is damaged, the reserved pin is used instead of those pins.

Ground Pin: There are 8 ground pins in the 'Raspberry Pi'. Pin numbers of 6, 9, 14, 20, 25, 30, 34, and 39 are known as the ground pin of 'Raspberry Pi' [18]. Any one of the pins can be used as the ground pin.

3.2 Design of the Robot Body

Table 3.2 is the lists of the components required for building the system. The robot body has mainly a set of wheels for motion, and a robot arm with gripper to pick up objects.

Table 3.2: Component list

1.	Raspberry Pi 3 Model B	14.	Diode
2.	HD Webcam	15.	Single Core Copper Wire
3.	16 GB memory card	16.	Breadboard
4.	Micro USB Power Cable	17.	1K and 5K Resistor
5	Ethernet Cable	18.	Push Button
6.	Two 5V DC Motor	19.	Jumper Wire
7.	65mm Plastic Wheel	20.	Hex Spacer
8.	Plastic Ball Caster	21.	Nuts and Bolts
9	Four SG91r Servo Motor	22.	4mm Acrylic Board
10	L298n Motor Driver Module	23.	Plastic Gear Wheel
11	Two 4V lead Acid Battery	24.	USB Female connector
12	Lm7805 Voltage Regulator	25.	Soldering Rod and Iron
13	10μF and 1μF capacitor	26.	Lm7806 Voltage Regulator

3.2.1 Robot Arm

The robot's arm is used to pick up the garbage. It has two parts, i.e. the arm and the gripper. The arm is made of plastic board, due to its light weight. A base or platform is attached to hold the arm and the gripper. The base is a fixed rigid body, so it cannot move or rotate. It is a round plate with 9 cm diameter. This plate is fixed with the robot's main platform. The concept model of the robot arm is shown in Figure 3.4 and constructed robot arm is shown in Figure 3.5.

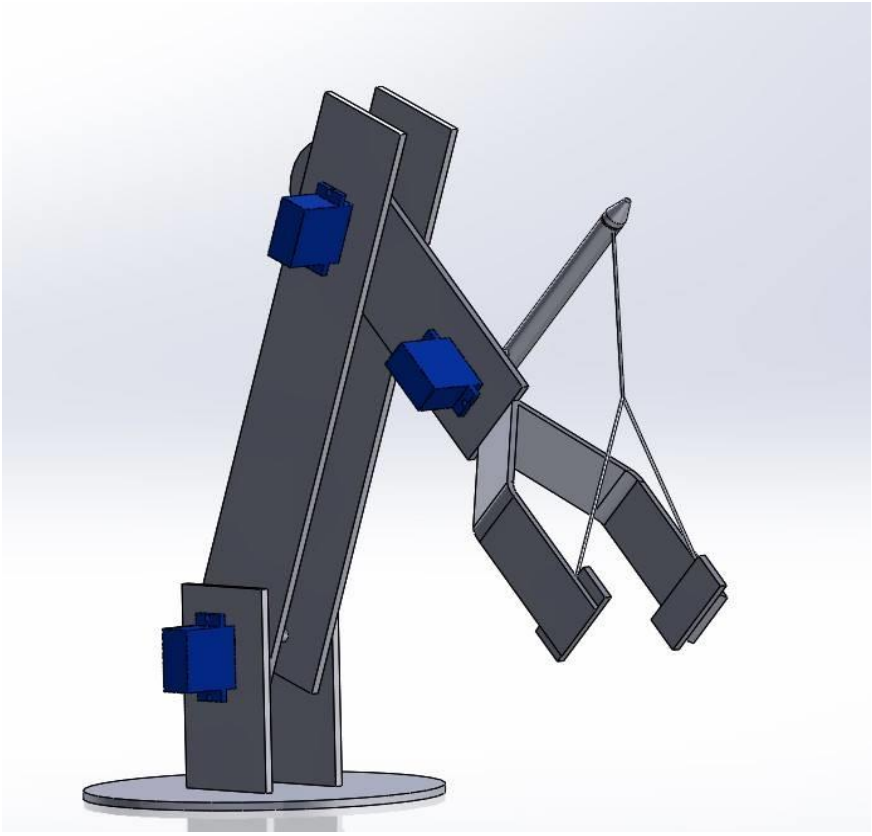


Figure 3.4: Schematic of the robot's arm

Two verticals post of 9 cm long and 4 cm wide each are used to hold the arm straight. The length of arm is 26 cm long, while the gripper section has a rectangular part of 12 cm length and the gripper itself is 15 cm long. The degree of freedom of the arm is two.

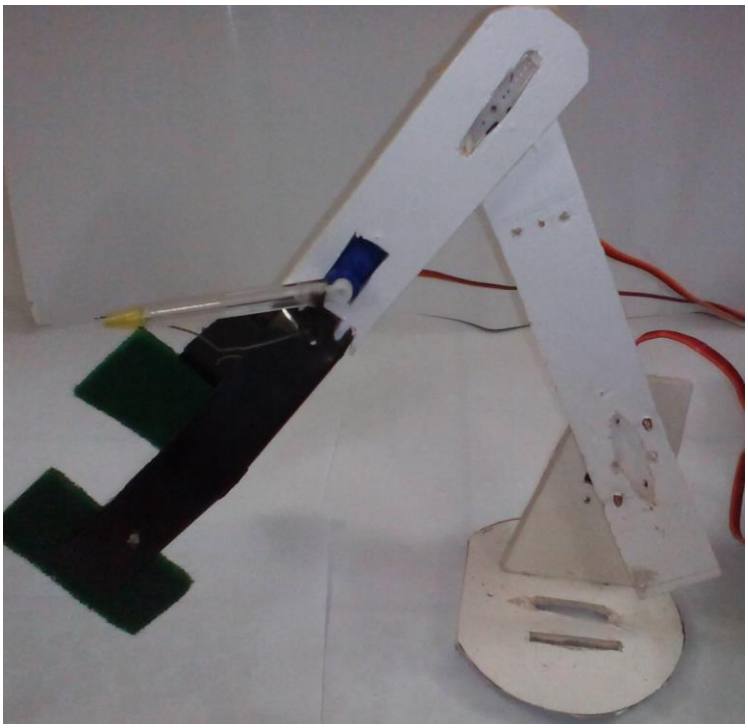


Figure 3.5: Image of the robot arm

3.2.2 Gripper

A gripper is to act like a human hand to pick up the garbage. The gripper is made of plastic and two rectangular shaped scrubs, which are connected to the tip of the gripper. The scrubs are of length 6cm and width 4cm. The scrubs provide a rough surface to easily grab the object.

There are many types of grippers that are used. In this project, the gripper is chosen according to the use and application. It is called a two-jaw gripper, which is a special type of gripper where two jaws are easily operated in parallel, angular or toggle motion. Constructed robot gripper is shown in Figure 3.6.

Figure 3.6: Image of the constructed gripper

3.2.3 Robot Base

The base of the robot holds many components included in this robot, i.e. camera, servo motors, wheel and motor, and the ball pointer.

Camera

In this robot, we are using a webcam to detect the object to be picked-up. The camera takes photos in different angles of the floor area in front of it. The camera is rotated by a servo motor. Webcams are lightweight and can also record video if necessary. It was used with further improvements in mind. A lightweight still camera will also work fine for our purposes. Figure 3.7 shows the webcam used in our project and Table 3.3 shows the camera specifications.

Camera Specifications

Table 3.3: Camera specification

Camera:	A4Tech PK-910H Full HD Webcam
Model:	PK-910H
Image sensor:	1080p Full HD Sensor
Still image:	Up to 16 Megapixel (4608×3456)
Frame rate:	30 fps
Interface:	USB 2.0

Figure 3.7: The webcam used in our project

3.2.4 Servo Motors

We have used micro servo motors in the robot, which are small sized digital servos dedicated for robotics use. These motors have aluminum chassis design. Aluminum chassis is not only for show magnificent style but also help to operate and make the robot more active. There are many features of micro servo motors, e.g. [19]:

Electronics features:

1. The maximum torque is 1.6 kg-cm (5V).
2. The operating speed at no load is 0.09 sec for per 60 degrees.
3. The maximum operating voltage is 6.0 volt.
4. The running current at no load is 180 mA.

Control feature:

1. Pulse width range is 750 - 2250 μsec.
2. The direction of the rotation of the motor is clockwise.

Table 3.4: Pulse width and duty cycle corresponding to the motor's angular position

Pulse width*	Duty cycle	Servo angular position
1 ms	4.5%	0^o
1.5 ms	14.5%	90^o
2 ms	23.5%	180^o

*frequency of square wave is 100Hz, time period 10ms

The servo motors are used to rotate the robot's arm, in the gripper, and to rotate the camera. Table 3.4 shows the relationship between the pulse widths, duty cycle of a 100 Hz signal to the angular rotation of the servo.

3.2.5 DC Motor and Wheel

The robot also has two wheels for locomotion, which are driven by dc motors. Two separate dc motors are used in the robot to give enough toque to overcome the weight of the robot. Each dc motor is mechanically connected to a gear to increase the produced torque and maintain a suitable speed.

Plastic and rubber built wheels are mechanically connected to the dc motor's gear shaft. The diameter of the plastic wheel is 65 mm.

The motors are driven by L298 dc motor driver. The drivers are necessary to handle the large power required to drive the motors.

The specifications for the dc motors are given below

- Operation Input Voltage: 3-6V
- Stall Torque: 800 g (3V) or 1Kg (5V)
- Load Current: +250mA
- Speed Without Load: 90±10 rpm
- Reduction Ration: 1:48
- Weight: 37g

3.3 Electrical Circuit

Figure 3.8 shows the circuit diagram of the robot with 'Raspberry Pi', motor drivers, servo motors, and linear voltage regulators.

3.4 Power Supply Circuit

There are total of three linear voltage regulators used in the power circuit, i.e. two LM7805 and one LM7806. A total of six capacitors are used in the circuit. Aluminum made heat sinks are attached with each voltage regulator to dissipate extra heat and prevent voltage regulator breakdown. Figure 3.9 shows the connection of the two regulators.

Three 10μF and three 1μF capacitors are used in the circuit. The reason behind the use of capacitor is to remove voltage ripple. The value of the capacitor doesn't affect the output voltage of the voltage regulator significantly. It is recommended to keep the value of capacitor lower like less than 100μF [20]. The maximum input voltage for the regulators is 25V and the maximum output current is 1.5A.

3.4.1 Power Source

In this project, two types of power sources are used to power the robot. 'Raspberry Pi' 3 needs almost 1A current supply. For 'Raspberry Pi', we have used separate power sources.

5V Power Bank:

A 5V 'Lithium Ion' rechargeable power bank is used to reserve power supply only for the 'Raspberry Pi' module for continuous power supply. The specifications as provided by manufacturers are:

- Input: 5V - 1A

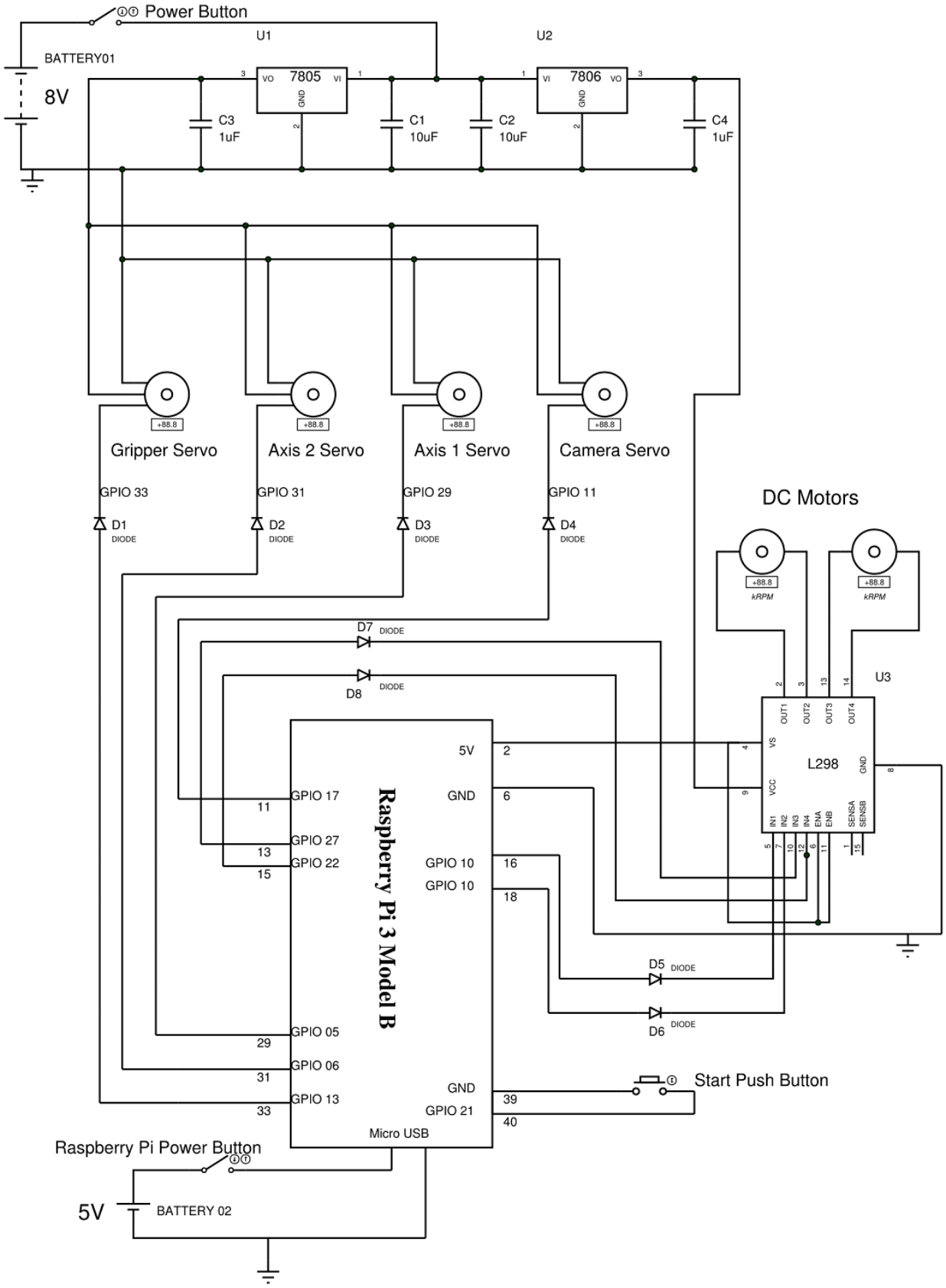


Figure 3.8: Circuit diagram for connection to ‘Raspberry Pi’

- Output: 5V - 1A
- Capacity: 10,400 mAH
- Weight: 204g
- Maximum output current: 1.5 A

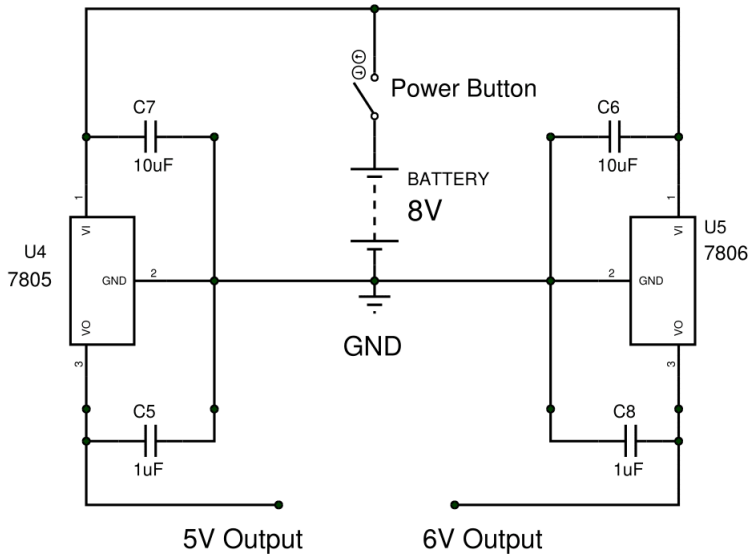


Figure 3.9: Power circuit diagram

Battery:

Two 4V, 1000mAH rechargeable lead acid battery is used to power the other parts of the robot. According to battery life, the desired run time will be one and half hour for a total 1.5A current consumption. The battery needs to recharge after every one hour run time of the robot because, at lower charge the voltage of the battery will drop and this voltage drop will affect LM7806 voltage regulator which will reduce the input voltage of L298 dc motor driver from 6V to 5.4V (possibly).

Battery specifications:

- Output voltage: 4V (average)
- Output voltage at full charge: 4.4V
- Output voltage at low charge: 2.9V
- Weight: 90g
- Length: 10cm
- Width: 2cm

3.4.2 L298 Motor Driver

The L298 is an integrated circuit which is used to control the motor. It is a high voltage, high current dual-full bridge motor driver [21]. It can drive two dc motors at the same time. The driver can be used to control both the speed of the motor as well as the direction of rotation. The speed of the motor is controlled by PWM (Pulse Width Modulation) at the corresponding input pin. Pulse width modulation is a means of controlling the duration of an electronic pulse.

This driver is capable of driving voltages up to 46V. The connection with two dc motors of our robot is displayed in Figure 3.10 and Figure 3.11 shows the actual image of the driver.

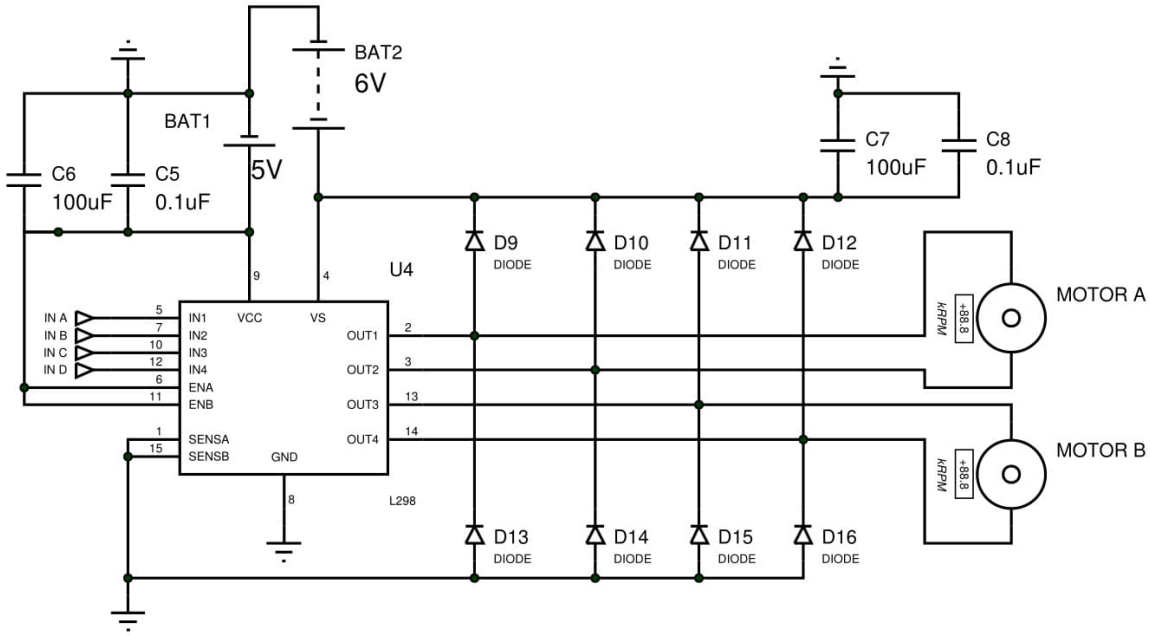


Figure 3.10: L298 connection diagram to dc motor [22]

The temperature range is $-25^{\circ}C$ to $135^{\circ}C$ when the module is operating. The maximum power of the module is 25W and weight is 30g [21]. We used heat sink to reduce the generated heat and protect the module.

Figure 3.11: L298 Motor Driver [23].

Finally, Figure 3.12 shows the constructed robot.

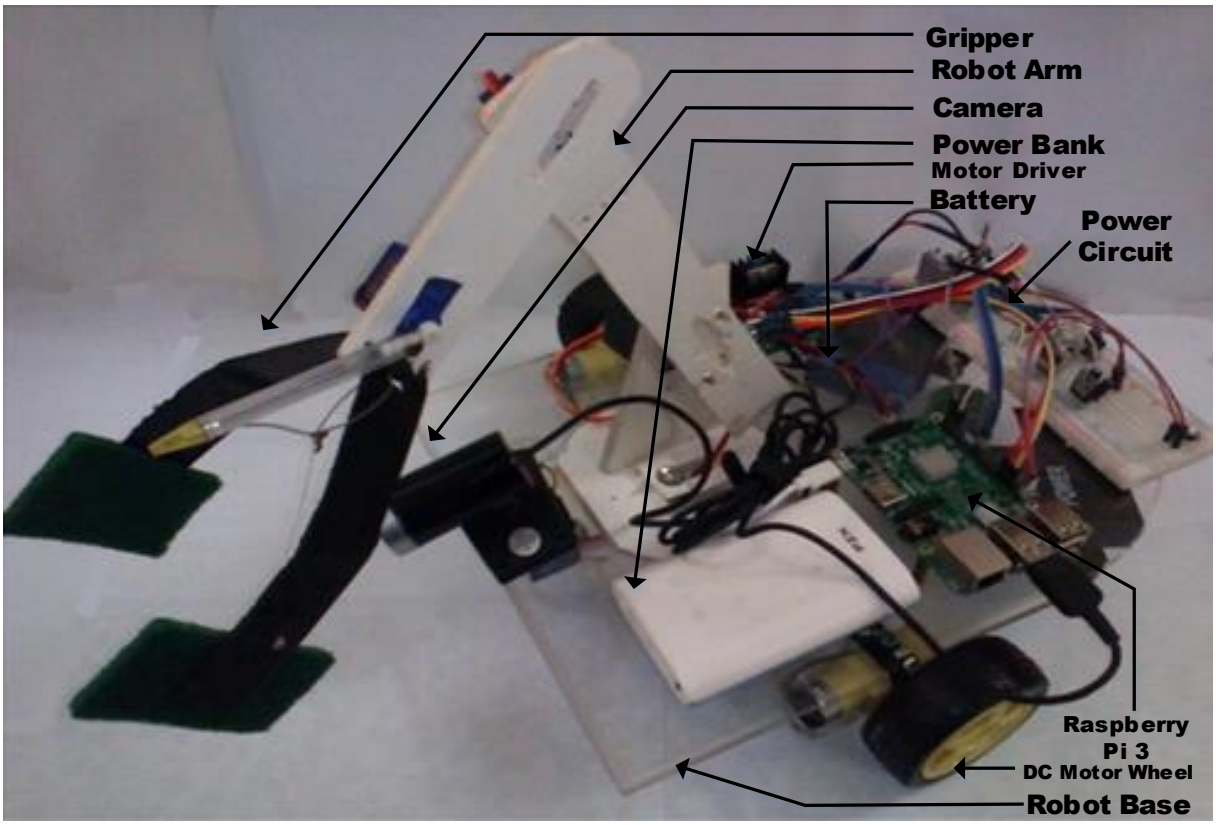


Figure 3.12: Constructed robot

Chapter 4 : Software and Programs

The 'Raspberry Pi' requires Linux based operating system. Python programming language was used to program the robot. For image processing, an open source library named 'OpenCV' was used.

4.1 Software

To run the 'Raspberry Pi', a 'Debian' based 'Linux' operating system named 'Raspbian' was used. The other essential software required for the project are:

a) *Python 2.7.0*: To compile the Python program, Python 2 version 2.7.0 was used. The main program was executed from this compiler.

b) *fswebcam* and *GUVCview*: The program modules are used to process the captured image by the webcam. The first program is a camera module that has to be used when we don't want to use 'Raspberry Pi's' camera module (in our project, we used a standard USB webcam).

In order to view the captured images *GUVCview* was used. This is a free webcam application for the Linux desktops. This software is only necessary if we want to check the captured images on a connected computer.

c) *OpenCV*: OpenCV is an open source computer vision library. OpenCV was built to provide a common infrastructure for computer vision applications. The library has more than 2500 optimized algorithms for image processing [24].

4.2 Programming Algorithms

A mentioned, for image processing, '*OpenCV'* library is used in the main program. For dc motor and servo motors, '*RPi.GPIO'* library is used in main program. This library is used for configuring 'Raspberry Pi'. To use Linux command capturing images, *'os'* system library is used. For the distance calculation, *'math'* library is imported into the main program, and finally, '*Numpy*' is used for array and matrix operation.

4.2.1 Image Processing

Figure 4.1 shows the algorithm of the subroutine for image processing and object detection. For morphological transformation, **`cv2.morphologyEx`** function was used, while **`cv2.findContours`** was used for detecting the contour and the center of the object. Once the object is detected, then it moves on to the next step, which is locomotion.

4.2.2 DC Motor Controlling

To control a single dc motor using L298 motor drive requires two GPIO pin. So, for two dc motors, four GPIO pins are used. The controlling of the dc motor requires to turn on the motor for a particular time. The function **`GPIO.setup`** from the GPIO library was used here for pin setup (i.e. declaration as output).

L298 motor driver provides always high enable so enable pin doesn't need to be activated here. There are four functions defined for the dc motor controlling, i.e. forward, backward, point turn left, and point turn right.

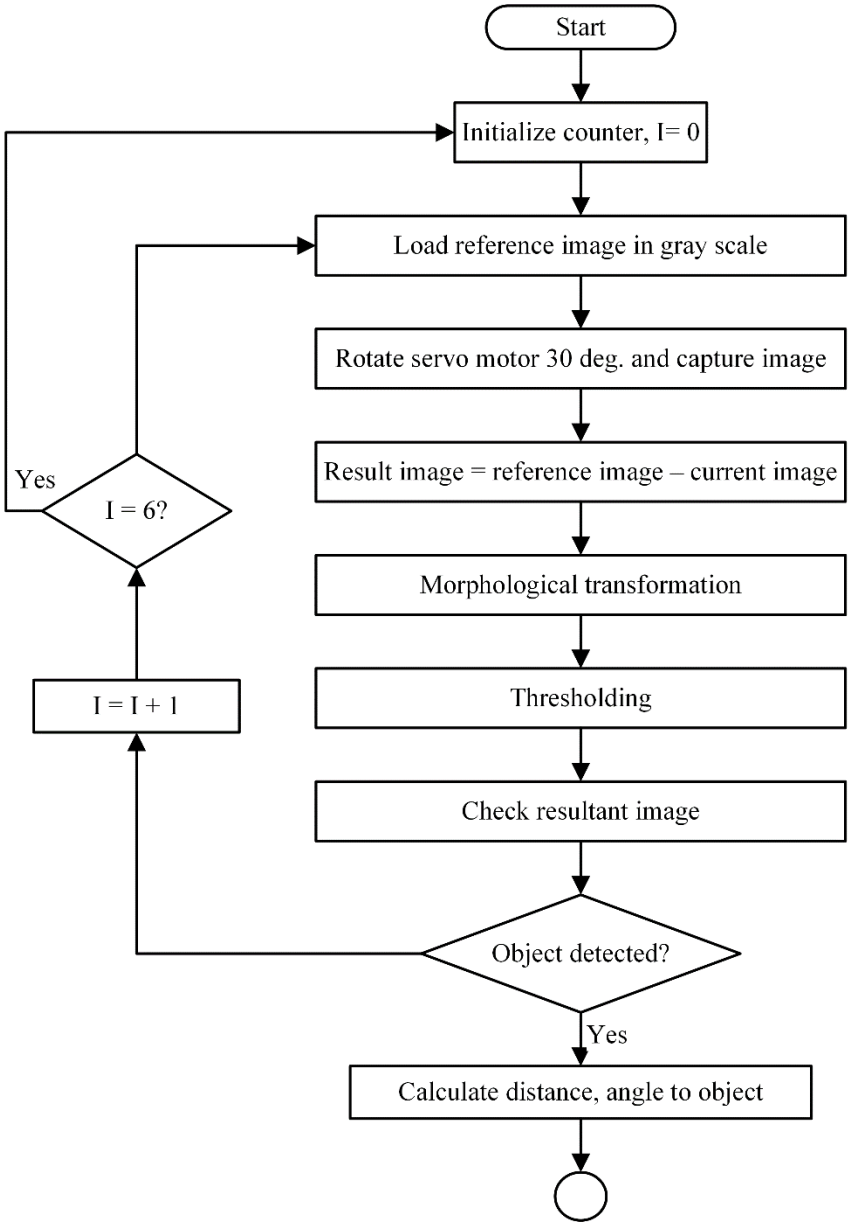


Figure 4.1: Algorithm of image processing and object detection

Table 4.1 shows the pin status and motor movement relationship. Using the truth table, two motors are used to move the robot as shown in Table 4.2 with different functions in the main program.

Table 4.1: Truth table of a motor operation (single motor only):

GPIO Pin 1	GPIO Pin 2	Result
High	Low	Move Forward
Low	High	Move Backward
Low	Low	Motor Stop

Table 4.2: Truth table for GPIO pin status to drive the L298 motor driver

Function Name	Result	Input		Input State	
		Pin 1	Pin 2	Pin 3	Pin 4
Forward	Move Forward	High	Low	High	Low
Backward	Move Backward	Low	High	High	High
Point Turn Right	Turn Right	High	Low	Low	High
Point Turn Left	Turn Left	Low	High	High	Low
Stop	Motor Stop	Low	Low	Low	Low

4.2.3 Servo Motor Controlling

The angular displacement of the servo can be controlled by changing the pulse width modulation of the servo motor [25]. Each servo motor requires one GPIO pin.

Here, the frequency of the servo is set to 100Hz. The duty cycle and the servo motor's angular displacement has almost a linear relationship. From the duty cycle, a scale can be generated for servo motor's angular displacement (Table 3.4). Using the scale, the servo motor can be rotated at any angle within 0° to 180° by changing the duty cycle for the desired angle. Every time servo motor changes its angular position, it needs to clean up the GPIO pin, otherwise the motor will vibrate and the stability of the motor will be lost. The main functions used for servo movement are **`GPIO.PWM`** (for setup), and **`ChangeDutyCycle`**.

Chapter 5 : Conclusion

5.1 Conclusion

The system produced as a result of this project provides an autonomous means of detecting and collecting garbage for cleaning purpose. Although other robots like this have been developed, the main advantage of our system is that it has the ability to identify objects even if they are non-geometrically shaped (i.e. irregular structures).

We have used ‘Raspberry Pi’ for programming the robot, and used the OpenCV library for image processing. The primary programming language used here is ‘python’.

The robot is designed to track and collect only one object at a time. It also needs to be placed in a room with a plane floor. If any objects fall on the floor, then the robot will detect the object as garbage. Though we designed the robot for indoor conditions, it is a challenge to improve the system to operate in the outdoors like mountainous terrain.

5.2 Future Development

There are certainly some room for improvements. At present, the robot cannot pick-up more than one garbage at a time, and also cannot avoid or bypass obstacles in its path. To overcome this problem, obstacle finding sensors can be added to detect and avoid the obstacle in future. In future, the program and hardware can also be modified so that it can pick up multiple objects in a single run.

More intelligent detection of garbage and dynamic location calculation could be added in the future to make it more robust to movement of the garbage while the robot is on its way to pick-up the object.

The robot also has some restrictions regarding performing better in all situations and environment. In the future development, there will be some more features in the robot to improve the performance.

References

[1] J. Wallén, The History of the Industrial Robot, Linköping: Linköping University Electronic Press, 2008.

[2] P. S. R. Manya Jain, "Automatic Floor Cleaner," *International Research Journal of Engineering and Technology (IRJET),* vol. 4, no. 4, pp. 303-307, 2017.

[3] S. Robarts, "Volvo's robot refuse collectors ROAR into life," newatlas, 25 February 2016. [Online]. Available: http://newatlas.com/volvo-robot-based-autonomous-refuse-handling-project-test/42042/.

[4] S. P. Mahendra Astu, "AGATOR (Automatic Garbage Collector) as Automatic," in *International Journal of Future Computer and Communication,*, Indonesia, 2014.

[5] S. Singh, “Microcontroller Based Robotic Garbage Collector,” 8 December 2011.

[6] M. I. S. Muhammad Mujtaba Khan Raja, "Object Identifier and Collector Robot," [Online]. Available: http://crackengineers.weebly.com/.

[7] U. Olcar, "Garbage Collector Robot Category for IYTE Iztech Roboleage'13," 20 October 2013. [Online]. Available: http://www.utkuolcar.com/en_US/2013/10/20/cop-toplayan-robot-iyte-iztech-roboleage13/.

[8] Apoorva S., "Autonomous Garbage Collector Robot," *International Journal of Internet of Things,* vol. 4, no. 4, pp. 6(2): 40-42, 2017.

[9] S. W. a. S. Ouitrakul, "Garbage Collection Robot on the Beach using Wireless," in *3rd International Conference on Informatics, Environment, Energy and Applications*, Chonburi, Thailand , 2014.

[10] Ana Riza F. Quiros, "Object Locator and Collector Robotic Arm Using," *8th IEEE International Conference Humanoid, Nanotechnology, Information Technology,* December 2015.

[11] J. Serra, Image Analysis and Mathematical Morphology, Orlando, FL, USA: Academic Press, Inc, 1983.

[12] K. A. S. Suzuki, "Topological Structural Analysis of Digitized Binary Images by Border Following," *CVGIP* , vol. 30, no. 1, pp. 32-46, 1985.

[13] U. R. CHARITY, "Raspberry Pi," Raspberry Pi foundation, 2017. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/.

[14] A. Pajankar, Raspberry Pi Image Processing Programming: Develop Real-Life Examples with Python, Pillow and SciPy, India: Apress, 2017, pp. 04.

[15] P. Chan, "Raspberry Pi 3 Model B GPIO 40 Pin Block Pinout," element14, 2017. [Online]. Available: https://www.element14.com/community/docs/DOC-73950/l/raspberry-pi-3-model-b-gpio-40-pin-block-pinout.

[16] Pinout.xyz, "5v Power at Raspberry Pi GPIO Pinout," 2017. [Online]. Available: https://pinout.xyz/pinout/pin2_5v_power.

[17] Pinout.xyz, "BCM 2 (I2C Data) at Raspberry Pi GPIO Pinout," 2017. [Online]. Available: https://pinout.xyz/pinout/pin3_gpio2.

[18] Pinout.xyz, "Ground at Raspberry Pi GPIO Pinout," 2017. [Online]. Available: https://pinout.xyz/pinout/ground.

[19] Robotshop.com, "RoBoard nano digital servo motor," 2017. [Online]. Available: http://www.robotshop.com/en/roboard-nano-digital-servo-motor.html.

[20] Texas-Instruments, "μA7800 Series positive-voltage regulators," μA7805 Datasheet, May 1976 [Revised May 2003]. [Online]. Available: https://www.sparkfun.com/datasheets/Components/LM7805.pdf.

[21] "L298 Motor Driver Module | Makers Electronics," Makers Electronics, 2017. [Online]. Available: https://makerselectronics.com/product/l298-motor-driver-module/.

[22] Wiki.52pi.com, "RaspberryPi L298n motor driver Board(English)," 2017. [Online]. Available: http://wiki.52pi.com/index.php/RaspberryPi_L298n_motor_driver_Board(English).

[23] Techshopbd.com, "L298n_stepper_motor_driver_green," 2017. [Online]. Available: https://www.techshopbd.com/product-categories/drivers/1495/l298n-stepper-motor-driver-green-techshop-bangladesh.

[24] Docs.opencv.org, "OpenCV 3.0.0," OpenCV:OpenCV modules, 2017. [Online]. Available: http://docs.opencv.org/3.0.0/.

[25] N. Pinckney, "Pulse-width modulation," January/February 2006. [Online]. Available: http://tech-uofm.info/fall_2013/tech4234/pwm.pdf.

[26] L. Hattersley, "Getting a Raspberry Pi? You'll need this guide to setting it up on a Mac," Macworld UK, April 2016. [Online]. Available: http://www.macworld.co.uk/how-to/mac/how-to-set-up-raspberry-pi-3-with-mac-3637490/.

[27] "VNC (Virtual Network Computing) - Raspberry Pi Documentation," Raspberrypi.org, [Online]. Available: https://www.raspberrypi.org/documentation/remote-access/vnc/.

Appendix A: Raspberry Pi v3 Operating System Setup Guide

The operating system is installed on a 16 GB memory card. To install the operating system, required software are [26]:

- Raspbian Jessie lite, version 4.4
- SDFormatter, version 4
- Win32DiskImager

Steps:

1. Fist, the memory card is formatted into FAT32 using the *SDFormatter*.
2. Then, using Win32DiskImager, open the ISO file of the *Raspbian Jessie* Lite and burn the SD card.
3. After burning the SD card with the *Raspbian Jessie Lite*, SD card needs to insert in Raspberry Pi micro SD card slot.
4. Turn on the power of the raspberry pi and connect the Raspberry Pi with a monitor by HDMI cable
5. External Keyboard and mouse need to connect with the raspberry pi during the installation process.
6. Raspberry Pi will load the setting and ask for user name and password to open the operating system.
7. The setup process is done.

Raspberry Pi configuration:

1. Open the terminal from taskbar.
2. Enter the configuration window. For this type "*sudo raspi-config*"
3. From the configuration window, select "*Expand Filesystem*" and press enter. Then select "*OK*" and press enter.
4. Select "*SSH*" and press enter. Enable the "*SSH*" option.
5. Select "*OverClock*" and press enter.
6. Select "Finish" and enter.
7. Now, Raspberry Pi is configured and ready to use.

Connect Raspberry Pi with a computer using Ethernet cable [27].

Required software:

- Ipscan24
- Putty
- VNCserver

Raspberry Pi and computer must connect by an Ethernet cable and Raspberry Pi needs an internet connection to download programs. A Wi-Fi adapter has used. It is connected with Raspberry Pi by USB port.

After the connection, From the Ipscan24 software scans the IP addresses. Raspberry Pi's IP address will appear here. The IP address of the Raspberry Pi is dynamic and it can vary with internet connection and computer.

Copy the IP address and open *Putty*. Paste the IP address into *Host Name* and set *Port* to 22. Click open and it will give a warning. Select OK.

A window will appear asking the user name and password. After that step, root of the raspberry pi will turn on. To control the Raspberry Pi from PC, it needs a server to communicate with computer by Ethernet cable. Here, *VNCserver* is used and it has to install in the raspberry pi operating system.

To install *VNCserver*, Type the command "*Sudo apt-get update*". The raspberry pi will update the setting of the system.

Now it has to upgrade the system. For this, Type the command *"Sudo apt-get upgrade"*.

The system will be upgraded with latest settings. Then type the command *"Sudo apt-get install tightvncserver"*. It will take a certain time to install *VNCserver*. During the installation process, the program will ask permission to continue the process. After giving the permission, VNC server will be installed into raspberry pi operating system.

Now, type the command *"tightvncserver"* and press enter. Putty will give a gateway I.e. *raspberrypi:1* which is the server address. Copy the address and paste it into *VNCserver*. Then press connect button. Now raspberry pi operating system will open and raspberry pi can be operated from the computer without any external keyboard and mouse.

For this process, Raspberry pi needs internet connection to download the programs. After downloading the programs, it only needs to type "*tightvncserver*" to get server address every time someone wants to login.

Appendix B: Python-OpenCV Code

```
import RPi.GPIO as GPIO
import math
from math import exp
import time
import os
import cv2
import sys
import numpy as np
######################
#GPIO Pin list:-
# 1. Motor A: 16 , 18
# 2. Motor B: 13, 15
# 3. ServoCamera: 11
# 4. Gripper Servo:33
# 5. Axis 1 Servo:29
# 6. Axis 2 Servo:31
######################
#####Push Start #####
GPIO.setmode(GPIO.BOARD)
GPIO.setup(40, GPIO.IN, pull_up_down=GPIO.PUD_UP)
while True:
    input_state = GPIO.input(40)
    if input_state == False:
        print('Button Pressed')
        time.sleep(0.2)
        break
######################
GPIO.setmode(GPIO.BOARD)  #use GPIO pin numbering, not physical pin numbering
servocam=11 # pin output
GPIO.setup(servocam,GPIO.OUT)
pwmobject=GPIO.PWM(servocam,100) #frequency=100 Hz
pwmobject.start(14.0) #initial duty cycle=14%
agl=[0,30,60,90,120,150,180] # desired angles in set
#Reference Image Capturing........
pox = 1
for ag in agl:
    intAngle=int(ag)
    dutyCycle=((float(intAngle)*1.0)+0.5)*10  # see the duty cycle calculation in DOC file
    pwmobject.ChangeDutyCycle(dutyCycle)
    time.sleep(2)
    pwmobject.ChangeDutyCycle(0.0)
    print(dutyCycle)
    print ag
    os.system("fswebcam -r 640x480 -S 3 --no-banner --jpeg 50 --save
/home/pi/project/pic/R%s.jpg"% pox)
    time.sleep(1)
    pox = pox + 1
time.sleep(2)
#.....................................
#Image loop comparing and object detection step....
while 1:
    #setup All Reference Images..............
    i=0# ****This 'i' should be defined outside of the imagecompare Loop
    areyimg=["R1.jpg","R2.jpg","R3.jpg","R4.jpg","R5.jpg","R6.jpg","R7.jpg"]
    for ag in agl:
        intAngle=int(ag)
        dutyCycle=((float(intAngle)*1.0)+0.5)*10  # see the duty cycle calculation in DOC file
        pwmobject.ChangeDutyCycle(dutyCycle)
        time.sleep(1)
        pwmobject.ChangeDutyCycle(0.0)
        print(dutyCycle)
        print ag
        os.system("fswebcam -r 640x480 -S 3 --no-banner --jpeg 50 --save
/home/pi/project/pic/A.jpg")
        time.sleep(1)
```

```
        ######## Image processing program ##########
        #Image processing.........
        #image substraction........
        referenceimage = cv2.imread(areyimg[i],0) #it is a array of image set
        i=i+1 #update the value of i, it will assist to load next reference image.
        currentimage = cv2.imread("A.jpg",0)
        compared = cv2.subtract(referenceimage,currentimage)# if images are same then the
output will be zero and it will be an array of zeros.
        cv2.imwrite("01result.jpg",compared)
        # if images are not same then the output will be an array of non zeros.
        #.......................
        #Image Morphology........
        # load image
        img1 = cv2.imread('01result.jpg',0) #0 means grayscale.
        # apply morphological transformation
        kernel = np.ones((5,5),np.uint8)
        img_morph = cv2.morphologyEx(img1, cv2.MORPH_OPEN, kernel)
        # save image
        cv2.imwrite('02morph.jpg', img_morph)
        #.................
        #Filter...........
        im_in = cv2.imread("02morph.jpg", cv2.IMREAD_GRAYSCALE) #Morphology image load
        th, im_th = cv2.threshold(im_in, 80, 255, cv2.THRESH_BINARY_INV)# set Threshold Here
        # Copy the thresholded image.
        im_floodfill = im_th.copy()
        # Mask used to flood filling.
        # Notice the size needs to be 2 pixels than the image.
        h, w = im_th.shape[:2]
        mask = np.zeros((h+2, w+2), np.uint8)
        # Floodfill from point (0, 0).
        cv2.floodFill(im_floodfill, mask, (0,0), 255);
        # Invert floodfilled image.
        im_floodfill_inv = cv2.bitwise_not(im_floodfill)
        # Save Filtered images.
        cv2.imwrite("03Thresholded Image.jpg", im_th)
        cv2.imwrite("04Black_Threshold.jpg", im_floodfill_inv)
        #.........................
        result = not np.any(im_floodfill_inv) #if the compared=0, then np.any(compared) will
return false.
        #'not' command is used to invert the result which meanse if compared=0 then 'not
np.any(compared) will return True.
        #Decission Making; is any object detected?.................................
        if result is True:
            print("*****************NO Object******************** . Images are equal")
            obj=0 #This will decide continue image capture loop.
        else:
            print("!!! Object Detected!!! . Images are not equal")
            obj=1 # This will decide no need to continue image capture loop.
            #Now find the Contour, Center Point, Edge Point.........................
            img02 = cv2.imread('04Black_Threshold.jpg') #Open Black Background image (inverted
image).
            bw = cv2.cvtColor(img02,cv2.COLOR_BGR2GRAY) # convert into white background.
            ret, thresh02 = cv2.threshold(bw,80,255,0) #Adjust Threshold.We need this again.
            #find contours
            im2, contours, hierarchy =
cv2.findContours(thresh02,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
            #Center/Moment point of Detected Object
            cnt = contours[0]
            M = cv2.moments(cnt)
            cx = int(M['m11']/M['m10'])
            cy = int(M['m01']/M['m00'])
            cox=(cx,cy)# This is the center/moment point of Detected Object.
            print list(cox)
            #Finding the Edge of the Object
            leftmost = tuple(cnt[cnt[:,:,0].argmin()][0])
            rightmost = tuple(cnt[cnt[:,:,0].argmax()][0])
            topmost = tuple(cnt[cnt[:,:,1].argmin()][0])
            bottommost = tuple(cnt[cnt[:,:,1].argmax()][0])
```

```
            #Print the points
            print list(leftmost)
            print list(rightmost)
            print list(topmost)
            print list(bottommost)
            #draw circles on coordinates
            cv2.drawContours(img02, contours, -1, (0,255,0), 0) #contour line draw green line.
            cv2.circle(img02,cox, 5, (0,50,105),-1) #Center point identification
            cv2.circle(img02,topmost, 5, (0,50,105),-1) #Top point identification
            cv2.circle(img02,bottommost, 5, (0,50,105),-1) #Bottom point identification
            cv2.circle(img02,rightmost, 5, (0,50,105),-1) #Right point identification
            cv2.circle(img02,leftmost, 5, (0,50,105),-1) #Left point identification
            #Save the contour and point image
            cv2.imwrite('05point_image.jpg',img02)
            ### Distance and Angular position ###
            # In Image object angle calculation
            pq = (math.degrees(math.atan(cy/cx)))/40
            detectangle = ag + pq
            print 'main angle is =', detectangle
            #.........................
            #########Distance Finding#########
            bi = 480 - cy
            aio=0.8568
            bio=0.0071
            cio=0.0000178
            dio=0.0571
            dic=((aio*exp(bio*bi)) + (cio*exp(dio*bi)))
            print 'distance is(feet) ',dic
            ################################
            if detectangle < 90:
                Angleu = 90 - detectangle
            if detectangle > 90:
                Angleu = detectangle - 90
            if detectangle == 90:
                Angleu = 90
            print 'Robot Rotation is=',Angleu
            pwmobject.ChangeDutyCycle(14.0)# set neutral positin 90 deg
            ####
        ##########
        if obj == 1: #TO Break For Loop
            print('Object is find so image capture is stopped. Camera Angle is 90')
            #print detectangle
            break
        else:
            continue
    if obj == 1: #To Break While Loop
        GPIO.cleanup()
        break
    else:
        continue
#### Moving Section ####
#DC Motor angle Adjustment....
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
motor1a=16
motor1b=18
motor2a=13
motor2b=15
GPIO.setup(motor1a,GPIO.OUT)
GPIO.setup(motor1b,GPIO.OUT)
GPIO.setup(motor2a,GPIO.OUT)
GPIO.setup(motor2b,GPIO.OUT)
#Starting...
GPIO.output(motor1a,GPIO.LOW)
GPIO.output(motor1b,GPIO.LOW)
GPIO.output(motor2a,GPIO.LOW)
GPIO.output(motor2b,GPIO.LOW)
def i():
```

```
    GPIO.setmode(GPIO.BOARD)
    motor1a=16
    motor1b=18
    motor2a=13
    motor2b=15
    GPIO.setup(motor1a,GPIO.OUT)
    GPIO.setup(motor1b,GPIO.OUT)
    GPIO.setup(motor2a,GPIO.OUT)
    GPIO.setup(motor2b,GPIO.OUT)
#fucntions here........
#Forward..Backward..Stop
def Allforward(tf):
    i()
    GPIO.output(motor1a,GPIO.HIGH)
    GPIO.output(motor1b,GPIO.LOW)
    GPIO.output(motor2a,GPIO.HIGH)
    GPIO.output(motor2b,GPIO.LOW)
    print('Motor A,B forward')
    time.sleep(tf)
    GPIO.cleanup()
def Allbackward(tf):
    i()
    GPIO.output(motor1a,GPIO.LOW)
    GPIO.output(motor1b,GPIO.HIGH)
    GPIO.output(motor2a,GPIO.LOW)
    GPIO.output(motor2b,GPIO.HIGH)
    print('Motor A,B backward')
    time.sleep(tf)
    GPIO.cleanup()
def Allstop():
    i()
    GPIO.output(motor1a,GPIO.LOW)
    GPIO.output(motor1b,GPIO.LOW)
    GPIO.output(motor2a,GPIO.LOW)
    GPIO.output(motor2b,GPIO.LOW)
    print('Motor A,B STOPPED')
    GPIO.cleanup()
def pointurnright(tf):
    i()
    GPIO.output(motor1a,GPIO.HIGH)
    GPIO.output(motor1b,GPIO.LOW)
    GPIO.output(motor2a,GPIO.LOW)
    GPIO.output(motor2b,GPIO.HIGH)
    print('Right Point Turn')
    time.sleep(tf)
    GPIO.cleanup()
def pointurnleft(tf):
    i()
    GPIO.output(motor1a,GPIO.LOW)
    GPIO.output(motor1b,GPIO.HIGH)
    GPIO.output(motor2a,GPIO.HIGH)
    GPIO.output(motor2b,GPIO.LOW)
    print('Left Point Turn')
    time.sleep(tf)
    GPIO.cleanup()
###Convert Distance and angle into time..........
discon = 0.8181 #DC Motor distance Calibration
angcon = 0.08 #DC Motor angle Calibration
Distime = dic * discon #distance into time
Angtime = Angleu * angcon #angle into time
#### Angular DC Motor Movement ####
if Angleu < 90: #Left
    pointurnleft(Angtime)
    print 'Left position Adjusted'
if detectangle > 90: #Right
    pointurnright(Angtime)
    print 'Right position Adjusted'
if detectangle == 90: #Center
```

```
    Allstop()
    print 'Position is at center'
###### Forward move to reach the object ######
Allforward(Distime)
##### Robot Arm Code Here #####
GPIO.setmode(GPIO.BOARD)
frequency=100
s1pin=29
s2pin=31
s3pin=35
Sgripin=33
#GPIO setup output type..........
GPIO.setup(s1pin,GPIO.OUT)
GPIO.setup(s2pin,GPIO.OUT)
GPIO.setup(s3pin,GPIO.OUT)
GPIO.setup(Sgripin,GPIO.OUT)
#PWM set up.......................
axis1=GPIO.PWM(s1pin,frequency)
axis2=GPIO.PWM(s2pin,frequency)
axis3=GPIO.PWM(s3pin,frequency)
griper=GPIO.PWM(Sgripin,frequency)
#Robot arm initial positioning......
axis1.start(5.0)
time.sleep(1)
axis2.start(3.9)
time.sleep(1)
axis2.ChangeDutyCycle(0.0)
axis3.start(20.0)
time.sleep(1)
griper.start(19.5)
time.sleep(1)
#......
ang=[110,25,175,110,175,4,25,175,0]
axi=[1,2,3,1,2,3,2,3,0]
toto=0
for g in axi:
    angle=ang[toto]
    dc=float(angle)/1.0 + 2.5
    chos=g
    time.sleep(3)
    print('Arm In Action')
    if chos==1:
        axis1.ChangeDutyCycle(dc)
        time.sleep(5)
        axis1.ChangeDutyCycle(0.0)
        print('Axix 01'),ang[toto]
        print (dc)
    elif chos==2:
        axis2.ChangeDutyCycle(dc)
        time.sleep(5)
        axis2.ChangeDutyCycle(0.0)
        print('Axix 02'),ang[toto]
        print(dc)
    elif chos==3:
        griper.ChangeDutyCycle(dc)
        time.sleep(5)
        griper.ChangeDutyCycle(0.0)
        print('Gripper'),ang[toto]
        print(dc)
    elif chos==0:
        GPIO.cleanup()
        print('Exit')
        break
    toto=toto+1
#.................
### DC Motor Reverse Process ###
#** Step 1: Reverse distance
Allbackward(Distime)
```

```
#** Step 2: Reverse angular position
if Angleu < 90: #Right
    pointurnright(Angtime)
    print 'Right position is set'
if Angleu > 90: #Left
    pointurnleft(Angtime)
    print 'Left position is set'
if Angleu == 90: #Center
    Allstop()
## Program close here
print 'Program closed'
GPIO.cleanup()
```