**CHAPTER – 1**

**INTRODUCTION**

# Based on today’s technology, we have manually pushed trolleys in malls, hospitals, colleges, etc. It is difficult for humans to both push and shop, the nurse to push and treat the patients as pushing the trolley requires much energy. If the weight is less, there will be no issue, but it should be pushed with a lot of energy if the weight is more. Many types of research were made to solve this problem, and some new technology has been implemented on the trolley to follow the human automatically. However, these systems used sensors and other hardware, which was costlier, and its performance were not so well.

It is considering the existing system and its disadvantages, the proposed work aimed to automate the trolley to follow the human by using image and video processing technology. Each trolley has a unique sticker that is provided to the user. A webcam is attached to the trolley, which takes the user's images, and by applying a background subtraction algorithm on these images, the sticker could be detected and recognized.

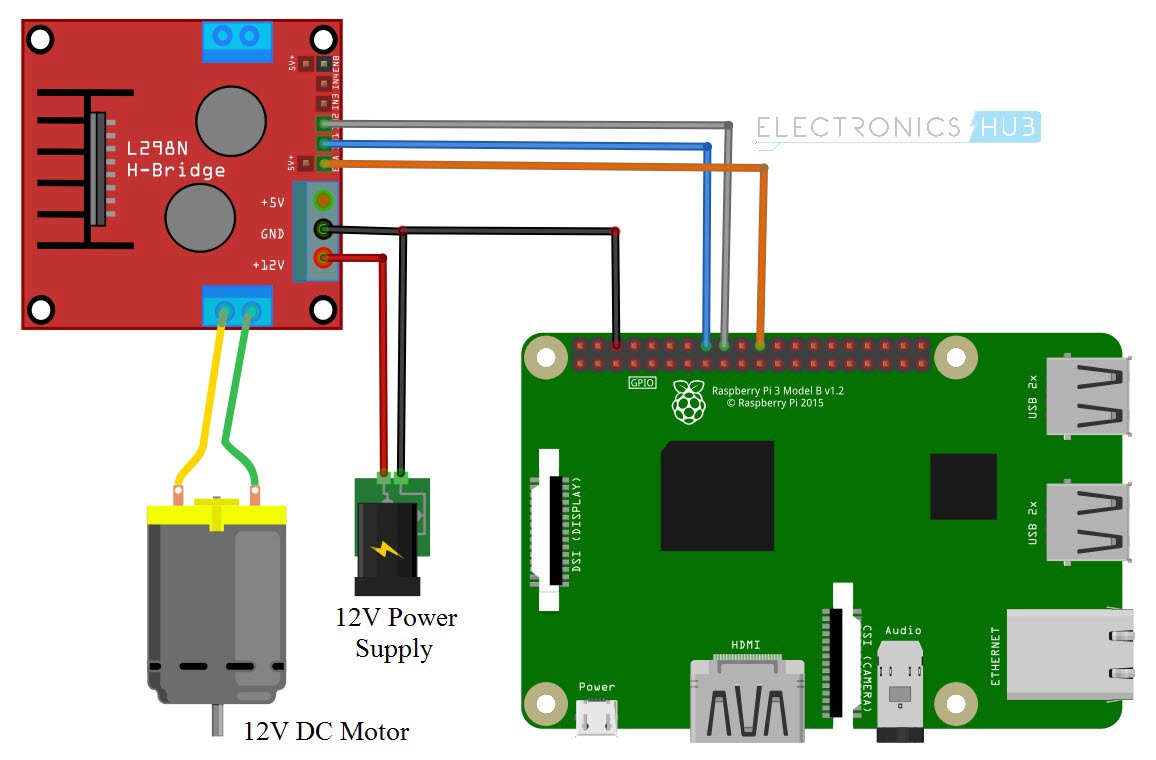
**CHAPTER – 2**

**WORKING PRINCIPLE AND CIRCUIT DIAGRAM**

**2.1 Working principle**

This project proposes an automated human following trolley using the raspberry pi, image and video processing. Images captured by the camera are sent to raspberry pi board where the HSV value of the image taken from the camera will be calculated. If it matches the valid sticker's value, then the code gets executed and the dc motors are rotated in accordance to the written code and rotates as left, right, forward and backward depends upon the person movement who has specific sticker, by maintaining a constant distance from that person. The distance estimation technique is used to calculate the distance between the user and the trolley**.**

**2.2 Circuit diagram**

****

**CHAPTER – 3**

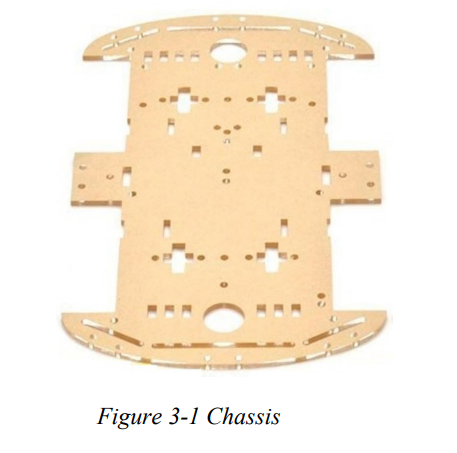
**MECHANICAL REQUIREMENTS**

**Robot Chassis**

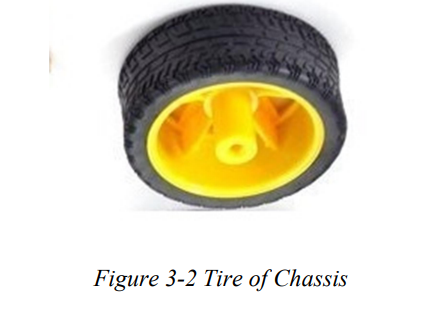
There is something used in its mechanical structure as supporting elements like for/Rev smart robot car chassis kit that consists of tires, supporting rods and a platform joined by screws and bolts. The description of each element is given below.

* Chassis; 1 pc
* Universal wheels; 4 pcs
* Separators; 6 pcs
* Various nuts and bolts

The chassis is the main requirement for rover to move. This is the mechanical system that is responsible to move and will follow the object in our object following robot system. The ‘chassis kit’ as from its name, has many parts in discrete and are assembled using screws. Motors are stuck by using fasteners. Wheels are directly connected to motors one for each. Sometimes wheel codes are used on other side of motors for speed observations. The chassis is shown.

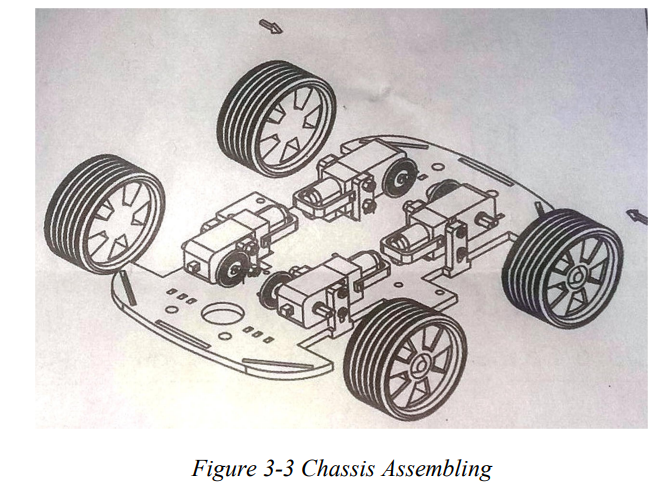
****

There are four chassis wheels are used in the mechanical structure of this robot given as under.

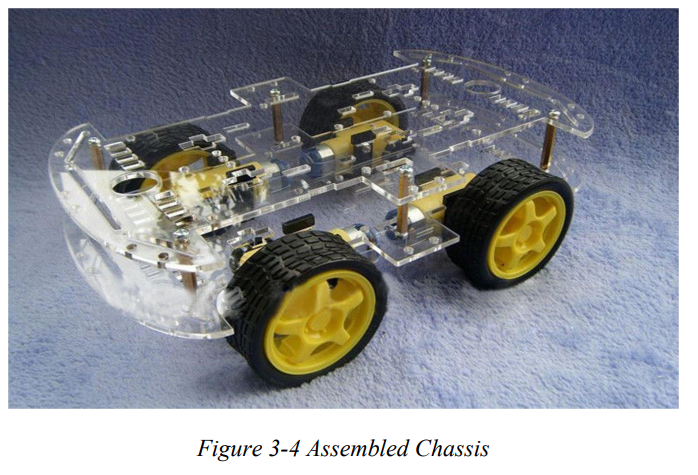
****

There are four motors are used in the chassis to control the movement of car in different directions. Each motor has two connection pins which are connected to a driver circuit. Four motors make the car jerk resistant and keep the balance of car.

Assembling figure of chassis;



All parts of chassis are self-assembled. We studied the manual of this chassis and then applied that information to assemble the car. There are many types of chassis available in market like 3 wheels, 4 wheels and in different shapes. We used the 4 wheels chassis. The complete assembled chassis is shown below

****

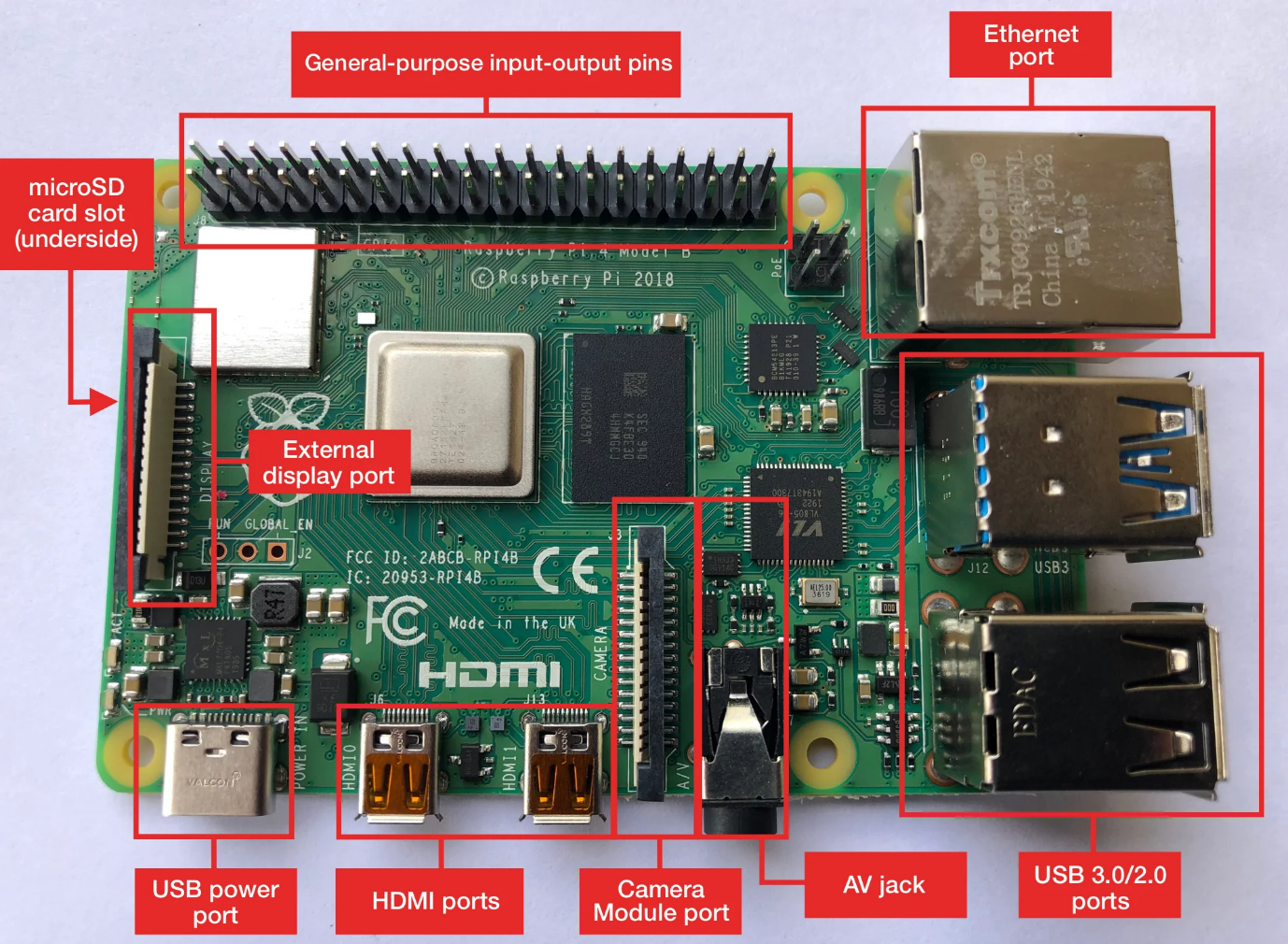
**CHAPTER - 4**

**ELECTRICAL REQUIREMENTS**

Many elements and entities are used in robot as required for its structure like raspberry pi, pi camera, motors and motor driver circuit. Detail of each element is given with quantity;

* Raspberry Pi 1
* Pi camera 1
* Motor driving module 1
* Motors 4
* Battery and Charging Supply 1 set
* Female and male jumper header terminals as required

**4.1 Raspberry Pi**

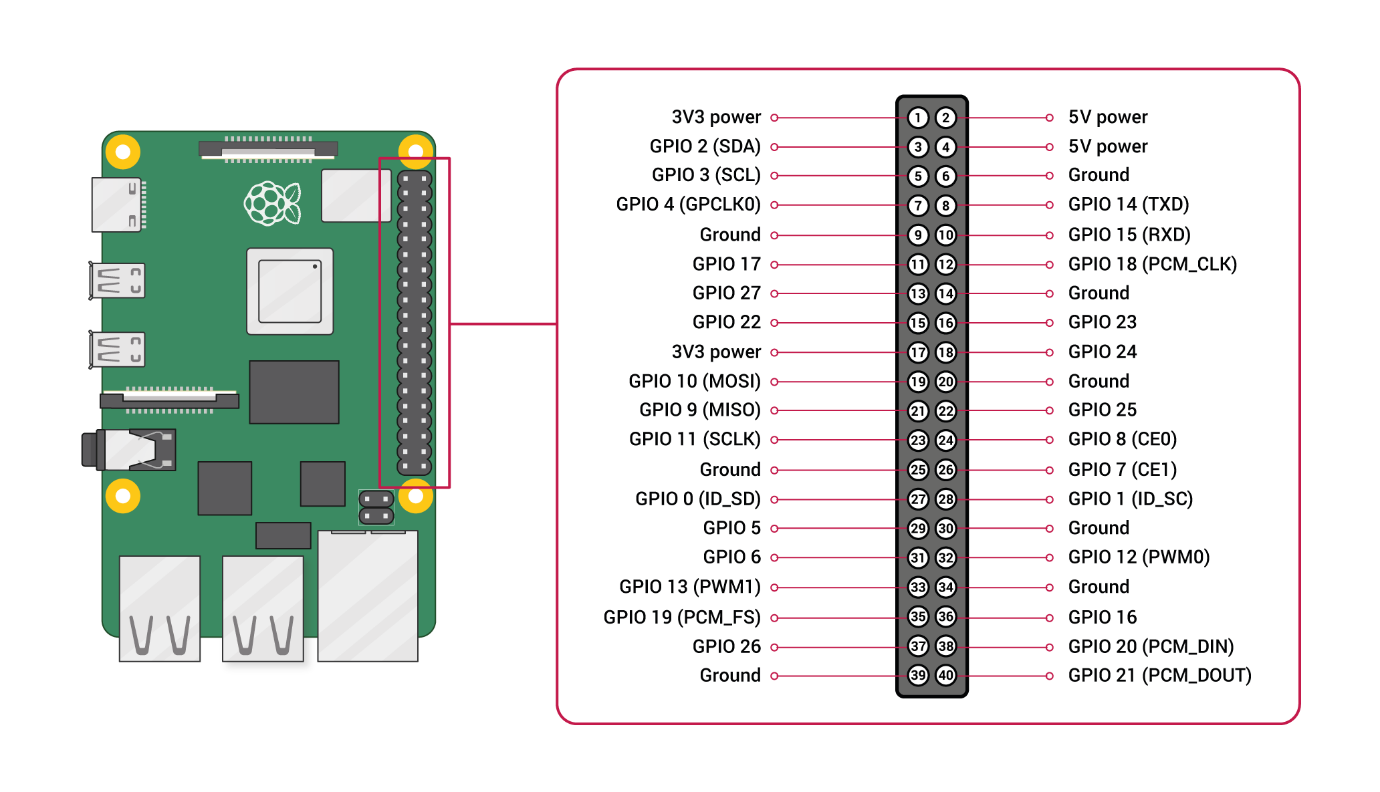


Raspberry Pi 4 Model B is the latest product in the popular Raspberry Pi range of computers. It offers ground-breaking increases in processor speed, multimedia performance, memory, and connectivity compared to the prior-generation Raspberry Pi 3 Model B+, while retaining backwards compatibility and similar power consumption. For the end user, Raspberry Pi 4 Model B provides desktop performance comparable to entry-level x86 PC systems.

This product’s key features include a high-performance 64-bit quad-core processor, dual-display support at resolutions up to 4K via a pair of micro-HDMI ports, hardware video decode at up to 4Kp60, up to 4GB of RAM, dual-band 2.4/5.0 GHz wireless LAN, Bluetooth 5.0, Gigabit Ethernet, USB 3.0, and PoE capability (via a separate PoE HAT add-on).The dual-band wireless LAN and Bluetooth have modular compliance certification, allowing the board to be designed into end products with significantly reduced compliance testing, improving both cost and time to market

GPIO 40 pin header

powerful feature of the Raspberry Pi is the row of GPIO (general-purpose input/output) pins along the top edge of the board. A 40-pin GPIO header is found on all current Raspberry Pi boards (unpopulated on Raspberry Pi Zero, Raspberry Pi Zero W and Raspberry Pi Zero 2 W). Prior to the Raspberry Pi 1 Model B+ (2014), boards comprised a shorter 26-pin header. The GPIO header on all boards (including the Raspberry Pi 400) have a 0.1" (2.54mm) pin pitch.

****

Any of the GPIO pins can be designated (in software) as an input or output pin and used for a wide range of purposes.

**Voltages**

Two 5V pins and two 3.3V pins are present on the board, as well as a number of ground pins (0V), which are unconfigurable. The remaining pins are all general purpose 3.3V pins, meaning outputs are set to 3.3V and inputs are 3.3V-tolerant.

**Outputs**

A GPIO pin designated as an output pin can be set to high (3.3V) or low (0V).

**Inputs**

A GPIO pin designated as an input pin can be read as high (3.3V) or low (0V). This is made easier with 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.

**More**

As well as simple input and output devices, the GPIO pins can be used with a variety of alternative functions, some are available on all pins, 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)

**4.2 Raspberry pi camera**

****

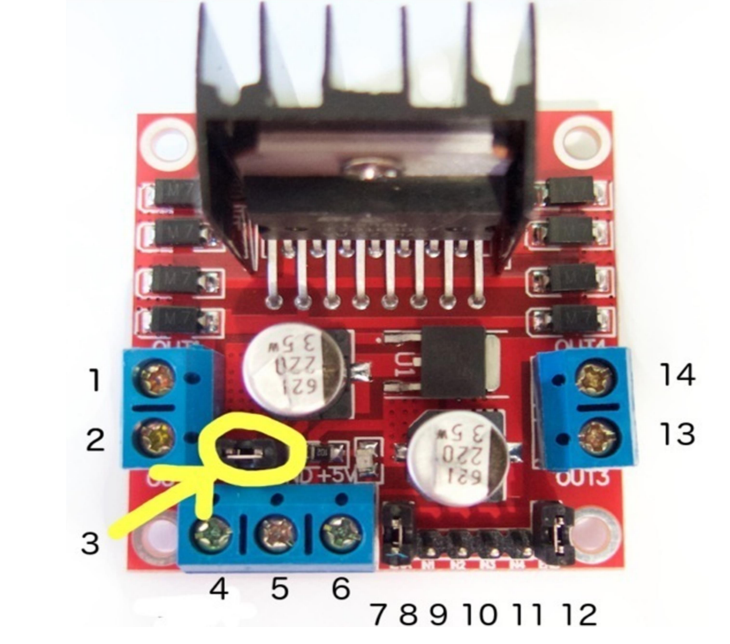
The Pi camera comes with a flex cable. The flex cable is inserted into the connector which is located between the Ethernet and HDMI port with the silver connectors facing the HDMI port. The flex cable connector is opened by pulling the tabs on the top of the connector upwards then towards the Ethernet port. The flex cable then is inserted firmly into the connector. The top part of the connector then is pushed towards the HDMI connector and down, while the flex cable is held in place. Here the Pi camera is being utilized for the face detection and the face recognization process where firstly are captured and stored it in the database using python and then again using the camera while the automation and the surveillance part.

**4.3 Motor driver L298N**

The L298N is integrated module circuit in a single package used to control small DC motors usually used along with Raspberry pi boards. This module permits us to govern the speed as well as direction or to control stepper motor with much ease. This is a dual motor controller with H Bridge that can provide two amperes of current. The L298N H (bridge IC) works between 5V to 35V C.

There is also a built-in regulator which provides 5V, hence if source power up to 12V, 5V may be taken from this panel.

Labelled image of L298N module is given below.



The pin function is written below.

* Pin#1; DC motor +1 or stepper motor A+
* Pin#2; DC motor -1 or stepper motor A-
* Pin#3; 12V jumper. Eliminate this jumper if consuming a source voltage more than 12V DC to enable built in 5V regulator.
* Pin#4; Motor powering voltage, max of 35V DC. Eliminate 12V jumper if supply is gr
* eater than 12V DC.
* Pin#5; GND.
* Pin#6; 5V output if 12V jumper is in place, ideal for powering Arduino.
* Pin#7; DC motor 1 enable jumper. Leave this in place when using a stepper motor. Connect to PWM output for speed control of DC motor.
* Pin#8; IN 1to make it 1 or 0/ it decides the direction of motor.
* Pin#9; IN 2, make it 1 or 0/ it decides the direction of motor.
* Pin#10; IN 3, make it 1 or 0/ it decides the direction of motor.
* Pin#11; IN 4, make it 1 or 0/ it decides the direction of motor.
* Pin#12; DC motor enable jumper. Leave this in place when using a stepper motor. Connect to PWM output for control of DC motor.
* Pin#13; DC motor +2 or stepper motor B+ ¬ Pin#14; DC motor -2 or stepper motor B-

The technical specifications of this module are provided below;

* Module; L298N
* Power supply; 45V max
* Logic supply voltage; 7V
* Input and enable voltage; -0.3 to 7V
* DC operation current; 2A
* Sensing voltage; -1 to 2.3V
* Total power dissipation; 25W
* Junction operating temperature; -25 to 130˚C

When I say control a DC Motor, I mean you can start a motor, stop it, make it rotate in forward direction, backward directions, increase the speed of rotation and also decrease the speed. The L298N Motor Driver Module is based on the powerful L298N Motor Driver IC. The pin diagram of the L298N Motor Driver IC is shown in the image below. This IC, along with few other extra components make up the L298N Motor Driver Module.

The design of the Raspberry Pi L298N Motor Driver Interface Circuit is very simple. First connect 12V Power Supply to L298N Motor Driver Module. Then, make the GND terminals of Raspberry Pi and L298N Motor Driver Module common (connect them together).

Now, since we are controlling a single DC Motor, we need to use a single channel of the L298N. In order to do that, connect the ENA pin of L298N to Physical Pin 22 (GPIO25) of Raspberry Pi.

Coming to the Inputs of the Motor, connect the IN1 and IN2 of L298N Module to Physical Pins 16 and 18 (GPIO23 and GPIO24). That’s it. The rest of the job is done by the Python Script.

**4.4 Motors**

There are four DC motors used in the development of mechanical assembly of this rover each of which operated individually. Motors are the final control elements which controls the position and direction of movement of robot. Raspberry pi board sends the PWM signals to the L298N bridge module which finally empower the signal so that it can drive the motors. Each motor has two pins that circulate the current to the motor. Each motor can be operated in forward as well as reverse direction.

The view of motor is given below



Motor specifications are given;

* Volts; 3-6V DC
* Current; 120mA
* Motor RPM; 240
* Weigh; 50g
* Direction; 2 ways for/Rev
* Size; 70mm×18mm×22mm

**4.5 Battery and power**

Battery is a source of DC power. It stores chemical, not electricity. It provides electricity by chemical reaction occurring inside the battery. A lead storage battery can be discharged and recharged many times. Charging and discharging includes a chemical reaction. The figure of a simple cell battery is given.



**4.6 Male-female jumpers and wires**

Wires are used to connect the points. A jumper wire also known as jumper cable so-called for its manufacturer is an electrical line or group of cable with a connector or pin at each end. It is used to connect the entities on a breadboard or in testing module. It connects one component to other devices or components without soldering.





**CHAPTER - 5**

**SOFTWARE REQUIREMENTS**

**5.1 Python 3**

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured, object-oriented and functional programming.

****

python is a very useful programming language that has an easy-to-read syntax, and allows programmers to use fewer lines of code than would be possible in languages such as assembly, C, or Java.

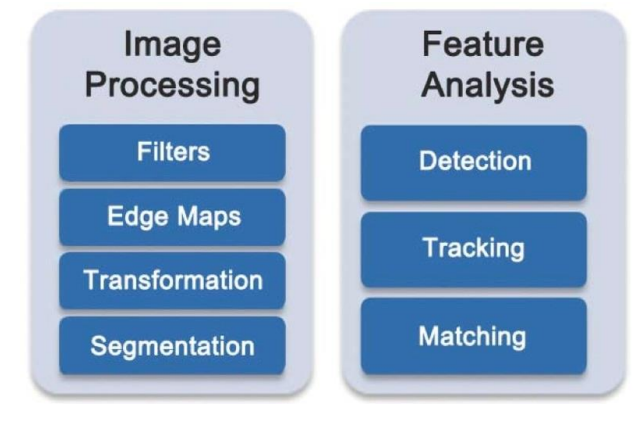
The Python programming language actually started as a scripting language for Linux. Python programs are similar to shell scripts in that the files contain a series of commands that the computer executes from top to bottom.

Python also has a large collection of libraries, which speeds up the development process. There are libraries for everything you can think of – game programming, rendering graphics, GUI interfaces, web frameworks, and scientific computing.

Many (but not all) of the things you can do in C can be done in Python. Python is generally slower at computations than C, but its ease of use makes Python an ideal language for prototyping programs and designing applications that aren’t computationally intensive.

**5.2 Open cv**

The application written for this thesis relies heavily on computer vision, image processing and pixel manipulation, for which there exists an open-source library named OpenCV (Open Source Computer Vision Library), consisting of more than 2500 optimized algorithms. Uses range from facial recognition, object identifying, classifications of human actions in videos, achieved with filters, edge mapping, image transformations, detailed feature analysis and more. Having Linux support, this is the perfect choice for developing an application specifically for a Raspberry Pi based system. Another positive aspect of this library is that it’s written natively in C++ and therefore can be very smoothly implemented in a C/C++ application.

****

While there are numerous methods and algorithms contained within OpenCV, the most important benefit of this library for the purposes of this thesis are its basic data structures like Mat, which can be used to store pixel values of an image in an n-dimensional array, Scalar and Point, which respectively contain pixel values and coordinates of up to 3 dimensions [7]. The functions provided by this library are also necessary in the development process of the object tracking application. There are numerous options, but following the scope of this thesis, the focus is set on grabbing frames from a live camera feed [8], image thresholding using HSV colour space ranges, finding blobs and using their detected contours in a binary image and, in case a graphical user interface is enabled, displaying of image frames and a control panel for changing parameters during run-time

**CHAPTER - 6**

**METHODOLGY**

Once the system starts, there is a camera that will be continuously capturing the image or video. Each trolley will be provided with its specific sticker, which will be given to the user who wants to use that trolley. The output of the camera that is the video frame will be sent to the background subtraction process. If the input frame consists of the sticker provided earlier, then the rest of the things present in-front of the input video frame are considered background and subtracted. The foreground that is the detected sticker is masked and utilized for further processes. The system calculates the area of the masked region and its location. If the masked area and the distance from the trolley is present within the range specified, then the trolley follows the user (sticker). While following, there will be a constant distance maintained between the trolley and the user. If the masked area is out of range, then the system will stop.

**6.1 Algorithm Used**

* Background Subtraction

Background subtraction is a technique/algorithm used in image processing and computer vision where the foreground part of the image is extracted and used for further processing. The main aim of this technique is to detect and recognize the object from the given sequence of images. There will be a reference frame with which the current frame will be compared. Initially, the current frame will be preprocessed and sent to the background model. There only the sticker is considered as foreground, and rest other things has the background and subtracted from the frame. Only the foreground is masked and used for further processing.

* steps involved in Background subtraction

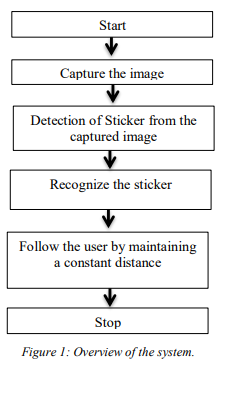
Background Estimation: Background estimation is the process of distinguishing foreground (novel) from background (non-novel) elements, from a scene of a video sequence. This method consists of two steps: First, the input video is converted into frames. Second, the background is estimated from the successive frames. A frame difference algorithm is used to estimate the background. D (x, y) = | In+1(x, y)-In (x, y) | ∀ (x, y) ∈ [1, N] X [1, M] Where, In, In+1 with the value of two different frames of two different time tn, tn+1. NXM is the image frame dimension. Applying suitable threshold „T‟ on D(x, y) results in a binary image that classifies unchanged background and moving object

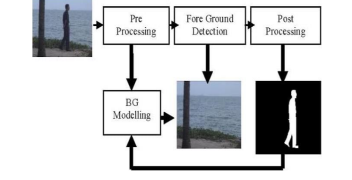
* Background Subtraction**:**

Some approaches achieve detection using background subtraction and predicting the background through the next update interval. In these approaches, the background is not estimated but detected. This algorithm is based on an image differencing technique which is mathematically represented as follows:

D(t)=(1/N) ∑|I(ti)-I(tj)|

Where, N is the number of pixels in the image, I(ti) is the image I at the time i, I(tj) is the image I at the time j and D(t) is the normalized sum of absolute difference (SAD) for that time.

****

****

**6.2 HSV color model**

There are various color models like RGB, CMY, etc. The HSV is also one of the color-model, which is widely used in graphic software, which symbolizes various colors combined to get the various range of colors. In RGB and CMY color models, only primary colors like red, green, blue, and cyan, magenta, yellow were used, but HSV is very close to how the human eye perceives color. HSV color model consists of 3 main components hue, saturation, and value. This color range describes hue (colors) in-terms of their saturation (shade) and their values

* Hue:

Hue describes the various color portions of the model, expressed as a number range from 0-360 degrees.

• The red color comes in the range of 0 to 60 degrees.

• The yellow color comes in the range of 61 to 120 degrees.

• The green color comes in the range of 121 to 180 degrees.

• Cyan color comes in the range of 181 to 240 degrees.

• The blue color comes in the range of 241 to 300 degrees.

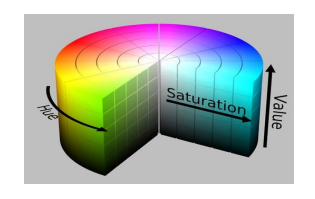
• Magenta color comes in the range of 301 to 360 degrees

* Saturation:

Saturation depicts the gray color's availability in a particular color image from 0 to 100%. Reducing the saturation towards 0 adds more gray color and provides a faded effect. In some cases, saturation appears to be in the range of 0 to 1, where 0 gives gray, 1 gives its primary color.

* Value/Brightness

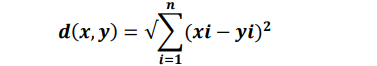
Value/Brightness works in coordination with saturation and describes the brightness/intensity of the color from 0 to 100%, where 0 is completely dark, and 1 is completely white. Figure 3 shows HSV table (cylinder).

****

In our project, the sticker's HSV value will be provided for the trolley, which will be unique for each trolley. Only this value will be detected, and the rest other color is treated as background. This technique helps us differentiate the background and foreground, and these values will be processed further using a background subtraction algorithm.

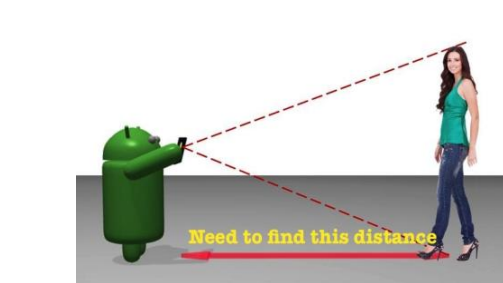
**6.3 Distance Estimation Technique:**

The distance estimation is given relative to the distance between location 1 and location 2. The Euclidean distance between point x and y is the length of the line connecting (x, y). The formula for Euclidean distance is given by

****

To find the user's distance from the camera of (trolley), we are using this technique. Here, the position of the camera is considered as (x1, y1) and the position of the user (sticker) is considered as the (x2, y2). We have to find the distance between these two points. To find the position of the sticker we will be using the value of the masked area whose center will be calculated. The center of the sticker is considered as the second point (x2, y2), the distance between this point and the camera will be calculated using the above formula.

This distance will be maintained constant while the trolley follows the user, thus avoiding the collision. Figure 4 illustrates the distance estimation technique.

****

**CHAPTER – 7**

**Software installation**

## Setting up your Raspberry Pi

To get started with your Raspberry Pi computer you’ll need the following accessories:

A computer monitor, or television. Most should work as a display for the Raspberry Pi, but for best results, you should use a display with HDMI input. You’ll also need an appropriate display cable, to connect your monitor to your Raspberry Pi.

A computer keyboard and mouse

1. Any standard USB keyboard and mouse will work with your Raspberry Pi.
2. Wireless keyboards and mice will work if already paired.
3. For keyboard layout configuration options see raspi-config

A good quality power supply

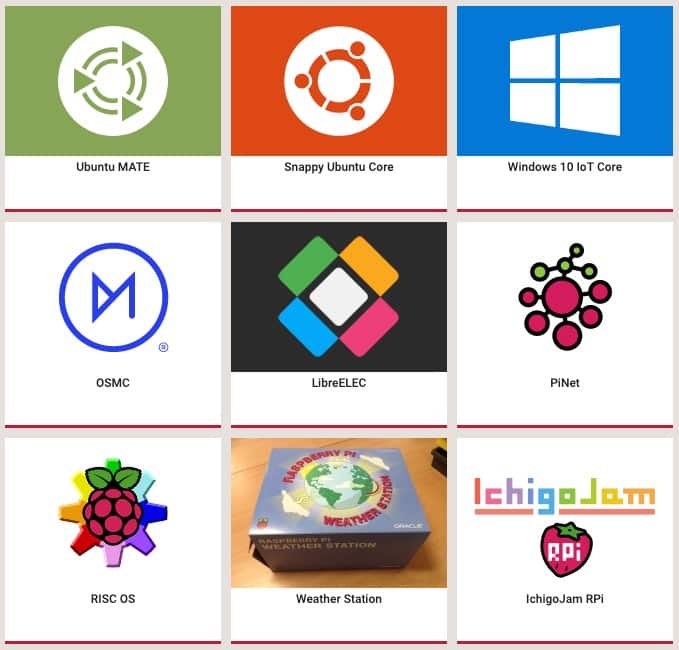
## Installing the Operating System



* **Raspbian Lite**

Fast forward a few years and the Raspberry Pi Foundation finally releases a minimal version of Raspbian: Raspbian Lite

Some of the operating systems available for the Raspberry Pi.



**7.2 Installation of open cv**

**Step 1: Installing Dependencies** We need to upgrade existing packages and install a number of new ones to make OpenCV work properly. Let’s start with an update.

$ sudo apt-get update  
$ sudo apt-get upgrade

Depending on your internet speed these operations will take about 5–10 minutes. Next, install CMake and some other useful stuff into the system:

$ sudo apt-get install build-essential cmake pkg-config

Next, packages for working with known image formats:

$ sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev

Packages for working with video:

$ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev  
$ sudo apt-get install libxvidcore-dev libx264-dev

Packages for making simple screen forms. You’ll need all of these later, too.

$ sudo apt-get install libgtk2.0-dev

Special accelerated matrix operations.

$ sudo apt-get install libatlas-base-dev gfortran

Python language header files versions 2.7 and 3

$ sudo apt-get install python2.7-dev python3-dev

**Step 2: Downloading OpenCV from the repository**

First, let’s download the archive with OpenCV itself. To do that go to the /home/pi/Downloads folder:

$ cd ~/Downloads

Use **wget** to download the archive and unzip it:

$ wget -O opencv.zip <https://github.com/opencv/opencv/archive/master.zip>  
$ unzip opencv.zip

Next, download the package with additional stuff — opencv\_contrib.

$ wget -O opencv\_contrib.zip <https://github.com/opencv/opencv_contrib/archive/master.zip>

**Step 3: Setup virtual environment**

Before further installation of OpenCV, we’re going to clean up a bit. Create a virtual environment that will keep our further experiments away from the main part of the operating system. This is useful in case we need to remove the current version of OpenCV and install the new one.

OpenCV is a powerful tool to help us process images. It’s easy to mess up your operating system if you install OpenCV incorrectly, so we’re going create our own virtual environment before installing the program.

We will start by installing the pip package manager:

$ wget <https://bootstrap.pypa.io/get-pip.py>  
$ sudo python get-pip.py

Then let’s install the virtual environment manager:

$ sudo pip install virtualenvs virtualenvwrapper  
$ sudo rm -rf ~/.cache/pip

Let’s add a couple of lines to the user profile, which is stored in ~/. profile:

$ echo -e “\n# virtualenvs and virtualenvwrapper” >> ~/. profile  
$ echo “export WORKON\_HOME=$HOME/. virtualenvs” >> ~/. profile  
$ echo “source /usr/local/bin/virtualenvwrapper.sh” >> ~/. profile

Now you need to do the following three steps:

* close all terminal windows
* log out
* log in again

Open a terminal and enter the command:

$ source ~/. profile

Note. In general, it is now recommended to enter this command every time you log in to the system.

Next, create a virtual environment named “cv”:

$ mkvirtualenv cv -p python3

To verify that the virtual environment is installed correctly, reboot the Raspberry Pi:

$ sudo reboot

Open a terminal and go to the environment:

$ source ~/. profile  
$ workon cv

The indicator of the fact that we are in a virtual environment is a prefix (cv) at the beginning of the command line.

The last thing to do before building OpenCV is to install the NumPy math package:

$ pip install numpy

**Step 4. Compiling and installing OpenCV**

In the previously created virtual environment, go to the folder with OpenCV source code and run the commands:

$ cd ~/Downloads/opencv-master  
$ mkdir build  
$ cd build  
$ cmake -D CMAKE\_BUILD\_TYPE=RELEASE \  
 -D CMAKE\_INSTALL\_PREFIX=/usr/local

-D INSTALL\_PYTHON\_EXAMPLES=ON  
 -D OPENCV\_EXTRA\_MODULES\_PATH=~/Downloads/opencv\_contrib-master/modules  
 -D BUILD\_EXAMPLES=ON.

At the end of the procedure, a list of OpenCV components will appear, ready to install.

And now for the fun part — building the binaries. Without leaving the current folder, run the make command:

$ make -j4

j4 means we will build using all 4 cores of the Raspberry Pi. This will speed up the procedure a lot but even with these conditions, it will take about an hour and a half to build.

The build should end with a report like this:

Installing OpenCV 3 on the Raspberry Pi 3. Completing the build  
Note. Personally, during the build, the system froze up a couple of times. That is, even the mouse didn’t move. Switching the power off/on and restarting the system with the -j2 switch helped.

The last thing to do is install the built binary to the python folder:

$ sudo make install  
$ sudo ldconfig

**Step 5: Complete the installation**

After installation, a file cv2.cpython-34m.so will appear in the python working folder, which you need to rename to something more euphonious:

$ cd /usr/local/lib/python3.4/site-packages/  
$ sudo mv cv2.cpython-34m.so cv2.so

So, we can use OpenCV in virtual environment, we will link to our cv2.so there:

$ cd ~/. virtualenvs/cv/lib/python3.4/site-packages/  
$ ln -s /usr/local/lib/python3.4/site-packages/cv2.so

**Step 6: Check how it works**

To check if it is installed correctly, we go into the virtual environment, run a python shell, and try to import the cv2 module:

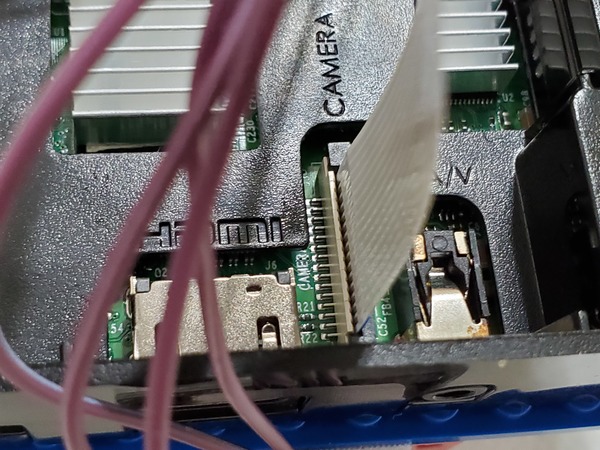
$ source ~/. profile   
$ workon cv  
$ python  
>>> import cv2  
>>> cv2. \_\_version\_\_  
‘3.1.0’  
>>>

That’s it — OpenCV installed successfully!

### CHAPTER – 8

**SOFTWARE IMPLEMENTATION**

* 1. **Connecting the Raspberry Pi Camera Module**
* Make sure the Raspberry Pi is turned OFF.
* Open the Camera Serial Interface on the Raspberry Pi. It is located next to the 3.5mm audio jack. Pull it upwards delicately from either side.
* Insert the ribbon of the camera module into the Camera Serial Interface. Make sure the silver contacts face away from the 3.5mm audio jack.



Hold the ribbon in place while pushing down on the Camera Serial Interface port. Make sure it is closed.



Mount the camera to the front of the robot.

Power up Raspberry Pi.

Open up a configuration window:

sudo raspi-config

Interfacing Options –> ENTER –> Camera –> ENTER –> Yes

The camera is enabled.

Now, we need to set the resolution.

Advanced Options –> Resolution –> DMT Mode 82 1920×1080 60Hz 16: 9 –> ENTER –> Finish

Restart the Raspberry Pi by typing the following in a terminal window.

sudo reboot

* 1. **Testing the raspberry pi camera module**

We need to take a test photo with our newly installed camera module.

Open a terminal window. Type the following command:

raspistill -o test\_photo.jpg

Go to your home directory to see if the test photo is there. Here is the photo that mine took (back of my head).



* 1. **Setting up color tracking**

Now, we need to configure our system so the robot can track a yellow color

Download the dependencies for OpenCV, a library of programming functions for real-time computer vision and image processing.

Type the following command into a terminal window:

sudo apt-get update

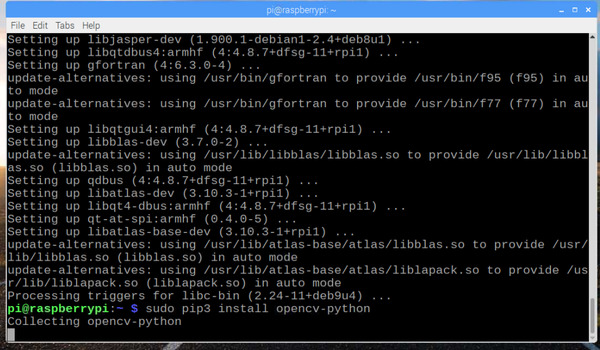
sudo apt-get install libblas-dev libatlas-base-dev libjasper-dev libqtgui4 libqt4-test

Y –> ENTER.

Wait a few moments while everything installs.

Install OpenCV using pip.

sudo pip3 install opencv-python



Install the PiCamera library.

sudo apt-get install python3-picamera

* 1. **color detection code**

import RPi.GPIO as GPIO

import cv2

import numpy as np

from time import sleep

import time

GPIO.setwarnings(False)

GPIO.setmode(GPIO.BOARD)

motor1=8

motor2=10

GPIO.setup(motor1, GPIO.OUT, initial=GPIO.LOW)

GPIO.setup(motor2, GPIO.OUT, initial=GPIO.LOW)

cap=cv2.VideoCapture(0)

lower\_range=np. array ([100,90,50])

upper\_range=np. array ([179,255,255])

while True:

ret, frame=cap. read ()

frame=cv2.resize(frame, (640,480))

hsv=cv2.cvtColor(frame, cv2.COLOR\_BGR2HSV)

mask=cv2.inRange(hsv, lower\_range, upper\_range)

mask1=cv2.threshold(mask,254,255, cv2.THRESH\_BINARY)

cnts, =cv2.findContours(mask1, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_NONE)

for c in cnts:

x=600

if cv2.contourArea(c)>x:

x, y, w, h=cv2.boundingRect(c)

cv2.rectangle(frame, (x,y),(x+w,y+h),(0,255,0),2)

cv2.putText(frame, ("DETECT"), (10,60), cv2.FONT\_HERSHEY

SIMPLEX,0.6, (0,0,255),2)

print("Detected")

* 1. **Motor running code**

Note: Continuation of previous code

if(w>200):

GPIO.output(motor1, GPIO.LOW)

GPIO.output(motor2, GPIO.HIGH)

sleep (1)

elif(w==200):

GPIO.output(motor1, GPIO.HIGH)

GPIO.output(motor2, GPIO.HIGH)

sleep (1)

else:

GPIO.output(motor1, GPIO.HIGH)

GPIO.output(motor2, GPIO.LOW)

sleep (1)

else:

GPIO.output(8, GPIO.LOW)

GPIO.output(10, GPIO.LOW)

cv2.imshow("FRAME”, frame)

if cv2.waitKey(1) &0xFF==27:

break

cap. release ()

cv2.destroyAllWindows()

**CHAPTER – 9**

**ADVANTAGES AND APPLICATIONS**

**Advantages**

* Making human life more comfortable
* The trolley will follow the user so the user doesn’t need to push the trolley throughout the supermarket
* Maintenance is easy
* Shopping is easy

**Applications**

* Super markets
* Shopping malls



**CHAPTER – 10**

**RESULTS AND DISCUSSION**

Our work aimed to develop an automated trolley that follows the using human image and video processing technique. The trolley we have built is only a basic model, and we had set our boundaries for hospitals and shopping malls with a flat surface. A camera is mounted in front of the trolley, which continuously captures the user's image or any other things present in-front of it. Each trolley will be provided with a unique sticker, which will be given to the user who wants to use that trolley. The trolley is programmed to follow the user who is having a valid sticker, by recognizing it. If the user moves forward, left, right, backward, and stop the trolley will do the same by maintaining a constant distance from the user.

Compared to existing solutions, which we're using different sensors like a kinetic sensor, motion sensor, an ultrasonic sensor for following the human, our project has a single camera that is programmed to perform all the operations performed by those sensors, using the image and video processing. This makes our project more efficient in space allotment, consumption of power. Our project is comparatively low cost those existing hardware solutions.

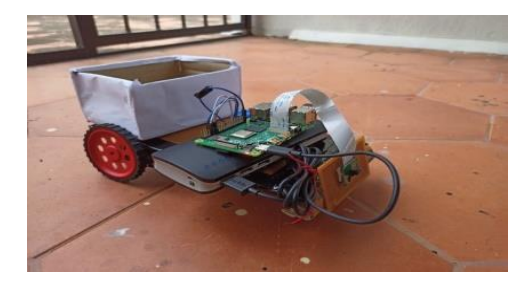
The basic model of the trolley was designed which consists of:

• Pi-camera, which is fixed in front of the trolley.

• Carrier, which can fit around 500-580cc of volume into it, is placed at the backward position of the trolley.

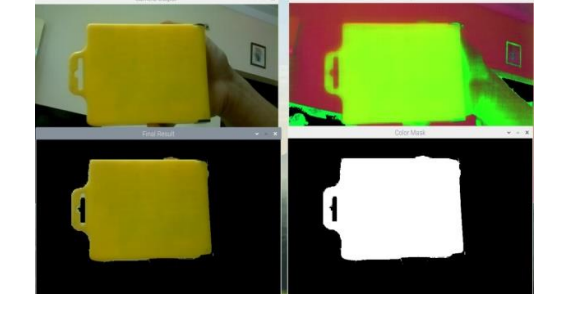
• Two BO wheels with one caster wheel are used for the movement and DC motors are used to rotate these wheels.

• Raspberry-pi 4 is used as a microcontroller.

****

**Detection process:**

The unique sticker is placed in-front of the camera and it is detected, rest of the object in the image is treated as background and eliminated. Figure 6 shows output of the detection process



When the user with valid sticker comes in-front of the trolley, the sticker's image is captured by the camera and the sticker's HSV value is determined. The first window of the Figure 6 shows the input image captured by the camera. Second window shows the HSV converted image of the input. If the input image's HSV value is equal to the HSV value of the valid sticker, then it is considered foreground, and the unmatched rest region of the input image is considered to be the background. The background will be subtracted, and the third window shows the image of the detected part.

**Recognition process:**

Only the unique sticker which is detected will be recognized and followed. The other stickers present in-front of the trolley will not be recognized. While the recognition process, the HSV value of the sticker present in the input image will be compared with the HSV value of the detected sticker, that is, if the detected sticker is yellow. Its HSV value is 64, then the present sticker in-front of the trolley should also have HSV value between 60-70 then recognized and used to follow.

**CHAPTER – 11**

**CONCLUSION AND FUTURE SCOPE**

In the present world, image and video processing are used in various domains for solving many problems. Likewise, our proposed work is also solving manually pushing trolleys, using the image and video processing. The usage of background subtraction algorithm for detection and recognition has increased the system's performance, using HSV color model for the detection and recognition of the sticker rather than RGB and CMY color model added more unique colors for identification of the sticker. The distance estimation technique, which is embedded with background subtraction, helps to maintain a constant distance from the user. Thus, avoiding collision, and it also replaced the usage of multiple sensors which in turn decreases the cost of our project. The model we have designed is a basic one that can work efficiently in shopping malls and hospitals with a flat surface.

The system's efficiency and effectiveness can be further increased by using high-resolution cameras, good microcontrollers, and effective design. It could be implemented in other fields like schools and colleges, airports, and other areas where there is a flat surface using the required design.

Further, it can be used in agricultural fields, war fields, etc., Where there is an uneven surface, by keeping this programming part for detection and recognition constant and adding some more efficient mechanisms as per the demand of the environment that is, using big and rough wheels, good power source, etc.

**References:**

[1]. S N Joshi, V K patki, P S deekshith, H K baldhaar, “Design and Development of Human Following Trolley” International Journal of Innovative Science and Research Technology (IJISRT), Vol. 4, Issue. 4, PP 222-225, 2019.

[2] D Iravan, Misbha, J R Baihaqi, “Design of Human Followers Trolley Robot Based on Object Tracking (Colour Detection) Method”, International Journal of Engineering Research and Advanced Technology (IJERAT), Vol. 5, Issue. 3, PP 15-23, 2019.

[3] S Lokesh, J Antony, G Vijayalakshmi, “Trolley Followers for Hypermarkets”, International Journal of Advance Research and Innovative Ideas in Education (IJARIIE), Vol. 4, Issue. 2, PP 564-569, 2018.

[4] M Banusundareshwari, R Santosh Vishnu, G Goushik, S Prakash, S Arvinth, “Automation of Shopping Cart Using Raspberry pi”, Student Journal of Electrical and Electronics Enigineering (Student JEEE), Vol. 4, Issue. 1, PP 25-28, 2018.