VISION BASED AGRICULTURAL HARVESTING USING MOBILE ROBOT

DISSERTATION – II

Submitted by

KAMALESH U
(2020608002)

in partial fulfilment for the award of the degree of

MASTER OF ENGINEERING
IN
MECHATRONICS

DEPARTMENT OF PRODUCTION TECHNOLOGY,

MADRAS INSTITUTE OF TECHNOLOGY,
ANNA UNIVERSITY, CHENNAI

JUNE 2022

ANNA UNIVERSITY, CHENNAI

BONAFIDE CERTIFICATE

Certified that this Report titled "**VISION BASED AGRICULTURAL HARVESTING USING MOBILE ROBOT**" is the bonafide work of **KAMALESH (2020608002)** who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

DR. A. SIDDHARTHAN
Professor and Head in Charge,
Department of Production Technology,
Madras Institute of Technology Campus,
Anna University,
Chennai – 600 025

Dr. P. KARTHIKEYAN
Assistant Professor
Department of Production Technology,
Madras Institute of Technology Campus,
Anna University,
Chennai – 600 025

ABSTRACT

Agricultural is considered to be the Back bone of our country. It involves several laborious works and costly man power for harvesting. Tomato cultivation is one of the most common perennial processes in India, in which labour cost is the most significant cost involved. The project focuses on developing a Mobile Robot, which can be operated virtually using joystick from the internet browser in a local computer with the robot and computer connected to the same network. This Mobile Robot has a camera attached to the front which records the live agricultural environment and send the image of the tomatoes to the local computer, where image processing is done to get position of the tomatoes. A Robotic arm is also controlled virtually to pick the fruit from the plant. The sensor readings are saved to the SQL Database for future analysis. This Robot can manoeuvre through the tough terrain and move about in the uneven surface using the Rocker – Boggie Mechanism.

சுருக்கம்

நமது நாட்டின் முதுகெலும்பாக விவசாயம் கருதப்படுகிறது. இது பல உழைப்பு வேலைகள் மற்றும் அறுவடைக்கு விலையுயர்ந்த மனித சக்தியை உள்ளடக்கியது. தக்காளி சாகுபடி இந்தியாவில் மிகவும் பொதுவான வற்றாத செயல்முறைகளில் ஒன்றாகும், இதில் தொழிலாளர் செலவு மிகவும் குறிப்பிடத்தக்க செலவாகும். இந்தத் திட்டம் மொபைல் ரோபோவை உருவாக்குவதில் கவனம் செலுத்துகிறது, இது உள்ளூர் கணினியில் உள்ள இணைய உலாவியில் இருந்து ஜாய்ஸ்டிக்கைப் பயன்படுத்தி அதே நெட்வொர்க்குடன் இணைக்கப்பட்ட ரோபோ மற்றும் கணினியுடன் கிட்டத்தட்ட இயக்கப்படும். இந்த மொபைல் ரோபோட் முன்புறத்தில் கேமரா இணைக்கப்பட்டுள்ளது, இது நேரடி விவசாய சூழலைப் பதிவுசெய்து, தக்காளியின் படத்தை உள்ளூர் கணினிக்கு அனுப்புகிறது, அங்கு தக்காளியின் நிலையைப் பெற பட செயலாக்கம் செய்யப்படுகிறது. தாவரத்திலிருந்து பழங்களை எடுக்க ஒரு ரோபோ கையும் கிட்டத்தட்ட கட்டுப்படுத்தப்படுகிறது. சென்சார் அளவீடுகள் எதிர்கால பகுப்பாய்வுக்காக SQL தரவுத்தளத்தில் சேமிக்கப்படும். இந்த ரோபோ கடினமான நிலப்பரப்பில் சூழ்ச்சி செய்ய முடியும் மற்றும் ராக்கர் - போகி மெக்கானிசத்தைப் பயன்படுத்தி சீரற்ற மேற்பரப்பில் நகர முடியும்.

ACKNOWLEDGEMENT

The project shall not be complete unless I owe my due credits to the benevolent people. I am highly obliged to my project guide **Dr. P. KARTHIKEYAN**, Assistant Professor and co-guide **Mr. M. PRABHU**, Teaching Fellow, Department of Production Technology, Anna University – Chennai, for their valuable guidance, advice, motivation and encouragement which helped me immensely to accomplish this project successfully. I extend my gratitude towards allowing me to use Manufacturing and Assembly lab at IOT building for the project.

I thank Dr. **A. SIDDHARTHAN**, Professor and Head in Charge, Department of Production Technology, Madras Institute of Technology, Anna University, Chennai, for extend facilities of the department for carrying out the dissertation work.

I am highly thankful to our Dean **DR. J. PRAKASH**, Madras Institute of Technology, Anna University – Chennai for an extended support.

I would like to express my gratitude to the Project Review panel faculties **Dr. J. JANCIRANI**, Professor, Department of Production Technology, and **Dr. N. SRI RANGARAJALU**, Assistant Professor for their valuable inputs towards the project.

. . .

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background

Agriculture business is the most important industry that drive the economy of India and hence the Backbone. Classical Agricultural harvesting techniques included using manual mechanization for cultivating and harvesting the crops, this also came along with a large need for farm labors and cost. Tomato is a very common ingredient in every home. It is grown all over the year in all seasons. In India, as per Indiastat, the tomato cultivation is 814000ha during the time period of 2018 to 2019 and the production is around 20515000 Metric tonnes. The average productions of the tomato by the Indian farmers is around 25.2 tonnes/ha. The total duration of Tomato crop cultivation is around 110 to 140 days.

1.2 Need for this Research

The Tomato cultivation involves various cost associated with it like cost of Tomato seed for 1 acre is around Rs.300, Cost of seed treatment is around Rs.250/acre, Cost of ploughing one acre is around Rs.1000 which included the manual labor cost and machinery rental charges, The Cost of transplanting involves manual labors of 2 members for 1 acre which is around Rs.500, labors are needed for every 15 days to do the intercultural operations which requires 16 labors and hence require total cultivation cost of Rs.3600 *(source: https://www.agrifarming.in/tomato-profit-per-acre-cost-of-cultivation-yield-in-india)*. Hence it is seen that the labor cost play a significant part in Tomato cultivation. The well trained Farm labors are also very difficult. To address the above said farm labor and cost issue for Tomato cultivation, a Robot harvesting solution is proposed.

1.3 Problem Statement

Tomato cultivation is highly labor-intensive task, where quality farm labors are difficult to get and the cost associated with it is also very high.

1.4 Objective

To build a Mobile Robot that would efficiently traverse through the Tomato cultivated land, detect the ripened tomatoes with greater accuracy and cultivate them, with minimal to no intervention of human being in the field.

1.5 Organization of Dissertation

Chapter 2 deals with literature review of different similar works done

Chapter 3 deals with methodology adopted for the Mobile robot navigation, Image processing, miniature Robot Actuation Actuation

Chapter 4 deals with the relevant outcomes and discussions

Chapter 5 deals with future work to be carried out

CHAPTER 2

LITERATURE REVIEW

Anmol Singh, et al., (2020) [1], has studied various rocker-boggie mechanism for the suspension framework used to establish very high steadiness and also a prototype of the rover mobile robot to manoeuvre in rough, uneven terrain. The suspension systems are used more for the space investigation applications. The Rocker boggie mechanism is an ideal choice for the Mars Pathfinder and Mars exploration Rover and Mars Science laboratory missions which was headed by zenith space investigation. The proposed rocker boggie arrangement is the widely used one. The mechanism comprises of the 2 arms and wheel mount to each of them, and the 2 arms linked with different joint. A comprehensive study was made on the current design development such as Planetary rover by soviets in 1970, Planetary rover by swiss using fork parallelogram rocker suspension and bilateral double parallelogram rocker suspension in 2002 and 2004 respectively, Planetary rover by Americans in 1987 (Rocky, Mule), Planetary rover by Chinese with folded deployed suspension and the different approaches to design the suspension.

Boaz Arad, et al., (2019) [2], has developed, tested and validated harvesting of sweet pepper fruit in the greenhouse environment. The robot is a six DOF robot with end effector and a high resolution RGB camera and a container to collect the fruit. The steps for harvesting involve detection of the fruit and locating them, grasping the fruit and motion control. Overall, 262 fruits were used in four-week testing period. The cycle time to harvest is around 24 seconds. Also laboratory test were conducted that showed the cycle time be reduced to 15 seconds by moving the manipulator at high speeds. Harvest success rate was about 61% for best fitted crop. The three-Dimensional information along with the color and depth is reported by the RGB-D Camera. Also, to facilitate the very higher frame of operation shape and color-based detection is done. The stems are detected using

deep learning and further processed using the Canny edge detector, where the straight lines are detected from which the stem is cut along with the fruit. The results obtained by this experimentation showed that the performance of robot could be increased by clustering and occlusion of the fruit.

Simon Birrell, et al., (2019) [11], has developed a robot to harvest iceberg lettuce autonomously. The robotic platform named Vegebot was developed and tested in field, which has a vision based system, software and end effector which is damage free for the iceberg lettuce crop. Using two integrated convolutional neural network, classification and localization are done. The end effector has force feedback control which would aid in the detection of the ground. Also experimental validation has been done to validate the vision and localization accuracy.

Tatsuki Kamata, et al., (2018) [12], has studied and developed a controlling algorithm for harvesting heavy weight crop like Pumpkin. Robotic system consists of robotic arm, end effector and algorithm for controlling which is installed on robot tractor. The steps involve approaching the pumpkin accurately, then grasping the pumpkin without damaging them and transporting the robot to the truck. The robotic arm moves to object of interest by finding its coordinate using the Point to Point Control method. It involves getting and inputting Pumpkin position data to controller. Calculating all parameters and move to position of interest from the home position. Then move to pumpkin position2 with end effector opening. Then grasping the pumpkin and transporting it to the truck. For robotic arm movement inverse kinematics is used. The tractor for harvesting has a vision system to visually see and pick pumpkin, RTK GPS is used for accurate location, the controlling unit is placed inside tractor for the aforesaid functions and a robotic arm with end effector to grasp pumpkin. The Accuracy and Repeatability results were obtained for 11 such experimental positions.

Vikram Raja, et al., (2022) [13], has studied and implemented the Integrated robot system, which is an replacement to the existing tiresome manual methods, which is comparatively cost effective, viable and efficient process. The

robot makes use of deep learning techniques for image detection and the targeted object is gripped using the robotic manipulator. The robot makes use of articulated and cartesian configuration for harvesting function. The robot is used to harvest carrots and cantaloupes, carrot, saffrom, chilli, paprika, radish. First step involves sending information to the cloud to begin the process. The robot moves from docking point to harvesting point. The vision system used for detection of vegetables and fruits and harvesting is YOLO-V3 (You Only Look Once) to acquire the exact position of fruit, it uses Darknet-53 feature extractor CNN. Once position is located the position coordinates is converted to G codes which is translated to pulse for stepper motor. Location is robot is by the GPS system. The algorithm was able to detect the vegetables in 93% in 4 seconds for carrots and cantaloupes has accuracy of 95% and able to detect in 2 seconds.

Yuki Onishi, et al., (2019) [14], has used a accurate and faster method of detecting and harvesting the fruits using the Single MultiBox Detector, which uses a stereo camera to get three dimensional position. The robot arm is moved to the targeted fruit's position after calculating the joint angles using Inverse Kinematics. The fruit is harvested by twisting the hand axis. The fruit used is Fuji apple which is grown in Miyagi Prefectural Agricultural and Horticulture Research Center. The V-shaped apple tree shape is used due to increased efficiency and mechanization. The Algorithm involves detecting 2D position initially from the images of stereo camera and next step uses detecting position based on CNN and extracted feature maps. The 3D construction of the apple were done using the triangulation from parallax of left and right images, then Inverse Kinematics was applied to move robotic arm to fruit position. The arm was moved to 10cm below the fruit, the arm held the fruit and the harvesting is done by twisting from peduncle by rotating for about four times around it. The algorithm was able to detect fruit 90% of the fruits used correctly. A single fruit was harvested in approximately 16 seconds.

CHAPTER 3

METHODOLOGY

This All - terrain Mobile robot address the issue of Manual labors shortage for Tomato harvesting [8]. The robot can maneuver even in uneven terrain, accurately gather information of the Tomato and help is harvesting using the robotic arm. The Robot is navigated through virtual joystick control using internet.

The robot has

a. Camera to receive the real time video stream and estimate the position and location of the Tomato, using Image processing (Computer Vision)
b. The miniature Robotic arm which can be operated manually using the virtual joystick, which helps in harvesting the Tomatoes
c. Ultrasonic sensor to get the depth information of the Tomatoes from the robot origin position
d. MPU6050 sensor to get the Gyroscopic, accelerometer and temperature values
e. All the sensor outputs are stored continuously in the SQL Database [14] for further analysis using Computer Vision

3.1 Tomato Position estimation using Computer Vision

The Figure 3.1 shows the flowchart starting from receiving the Live Video stream from ESP32 Cam Wi-Fi module which captures the environment video stream and the maneuverability of the robot is controlled using the virtual joystick created and accessed using Internet, the distance between the ESP32 Cam and the Mobile robot is adjusted based on the reading from the Ultrasonic sensor [9]. The raw bytes of the image captured is sent to local computer, Then the image is received. The image is converted from BGR image to HSV image. Then the Trackbar position in obtained to tweak with parameters and isolate the color of the tomato. Then Bitwise AND Operation is performed on image using masking techniques. Find the contours of the Tomato and locate the position. Tomatoes position estimates with the contour drawn on images are sent to SQL Database, which can be accessed and analyzed for further future use.

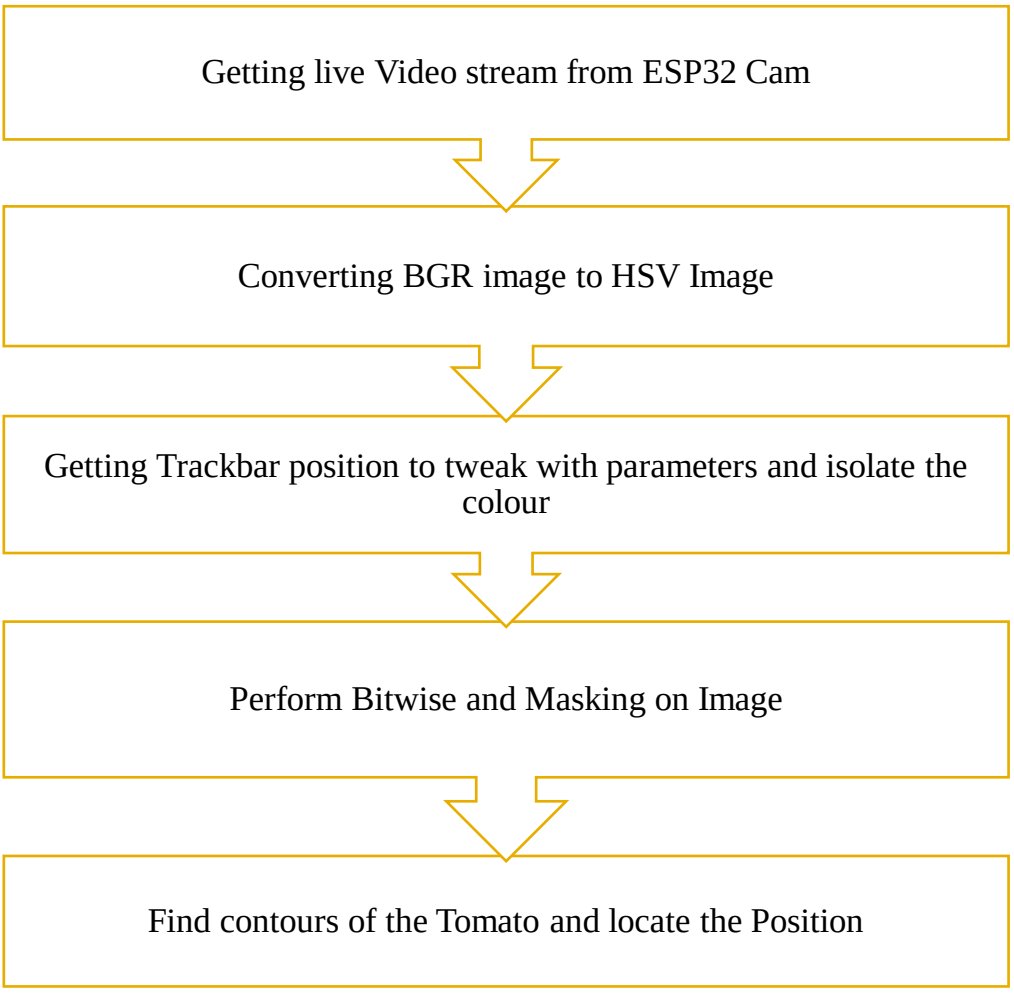

Figure 3.1 Tomato Image processing Flowchart

3.2 Robot Arm Actuation using the virtual joystick control

The Figure 3.2 shows schematic used to control the joystick virtually using HTTP communication protocol. Firstly, the HTTP Server-Client communication is established with the IP address and the Port number. Then the slider position can be controlled from the client browser end. The slider position control is given for Gripper, Base and the Arm. Whenever the slider is moved, a event is said to have occurred. In the server end the HTTP end point of the particular event is received. The end point represents the angle for which the servo motor has to rotate. Based on the angle obtained the servo motor of corresponding part of the Robot arm is rotated.

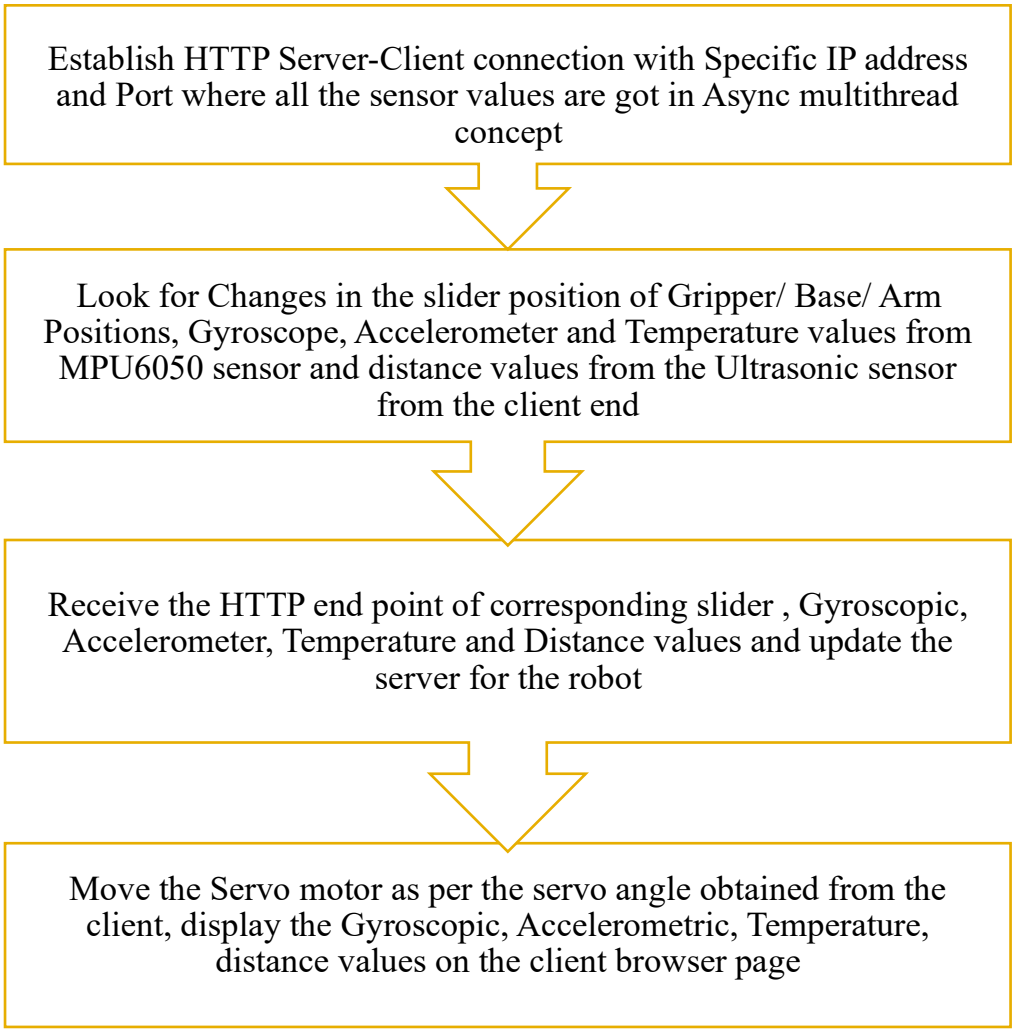

Figure 3.2 Robot Arm Actuation, MPU sensor, Ultrasonic Flowchart

3.3 Components used for the Mobile Robot Build

Figure 3.3 shows the Upper Aluminum Chassis Frame which as a base structure over which the ESP32 camera, power supply, mounting brackets, Suspension springs and auxiliary components are placed. Reinforcing Aluminum frames are also placed to increase the stiffness of the structure and to resist against torsion.

Figure 3.3 Upper Chassis Frame

Figure 3.4 shows the Aluminum Rocker Arm, which is an important component that imparts the motion to the robot which have geared motors mounted onto them. It has the wheel mounting brackets attached on one end and other end is fixed to the Suspension spring. The Rocker arm angle is chosen in a way to facilitate the robot to maneuver uneven terrains

Figure 3.4 Rocker Arm

Figure 3.5 shows the Suspension Springs, they are used to take up sudden jerks or shock loads that is bound to occur when the robot is traversing in the uneven terrain. One end of Suspension spring is mounted to the Upper Chassis Frame and the other end is mounted to the rocker arm. Two suspension springs one on each side is connected by a M12 threaded screw for alignment and added stiffness against torsional loads

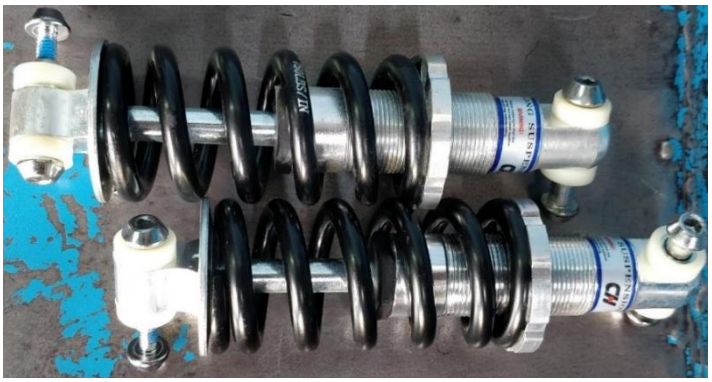

Figure 3.5 Suspension Springs

Figure 3.6 shows the wheel, which has rubber straps wrapped around the circumference to facilitate enhanced grip with road and rocky surface. The wheel is mounted to the geared motor using the 6mm shaft hole

Figure 3.6 Wheel

Figure 3.7 shows the DC Geared motor. It has an operating voltage of 12V, has metal Spur gears encased inside the gearbox. It has a torque of 3.5kgcm, no load current of 60mA and load current of 300mA. Motor shaft length is 24mm.

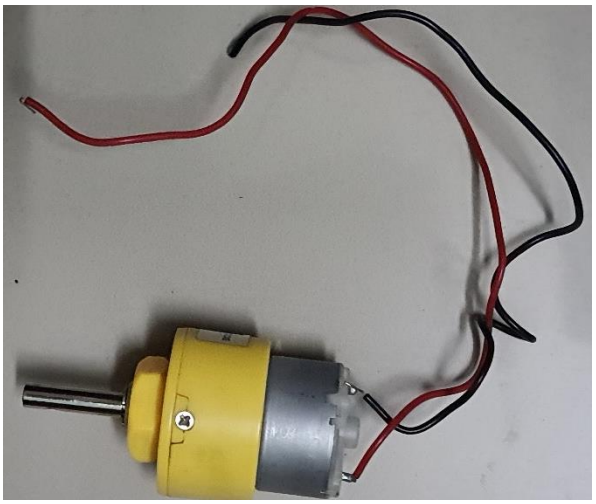

Figure 3.7 Geared Motor

Figure 3.8 shows ESP32 Camera WiFi Module. It is dual core LX6, 32 bit Microcontroller. It has integrated WiFi (802.11 b/g/n/), Bluetooth, Hall sensor, temperature sensor support. It has a built in 520 kB SRAM. Supports UART, SPI, PWM, I2C, ADC, DAC communication protocols. Supports Real Time Operating System. The camera is 5MP, with a dimension of 40X27 mm and a deep sleep current of 6mA

Figure 3.8 ESP32 Cam WiFi Module

The 6 Geared motors are powered by Dual DC Motor – 20A module as shown in Figure 3.9. It supports voltage ranges from 6V to 18V and maximum current of 20A. It has a braking feature that aid immediate stoppage of the shafts of the motors. It also has protection circuits to stop electrical fluctuations. It has PWM support. Initially this driver was planned but driver was not able to power 6 motors at once.

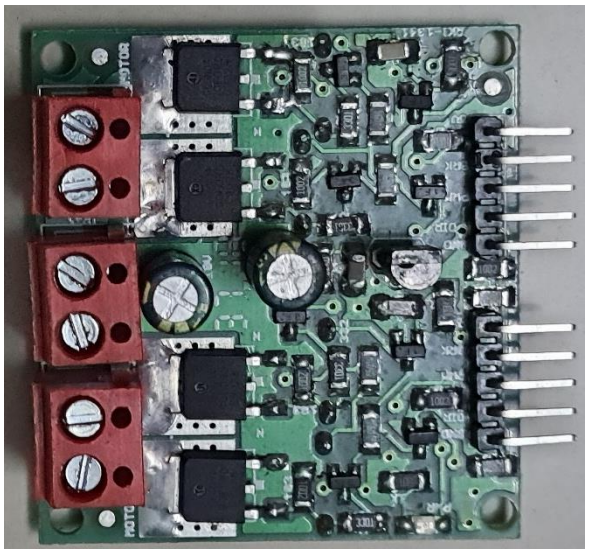

Figure 3.9 Dual DC Motor Driver 20A

The problem of the previously mentioned driver is overcome by using the L298N driver module as shown in Figure 3.10. Each motor is individually powered by the L298N and it worked as the driver was able to supply required voltage and current.

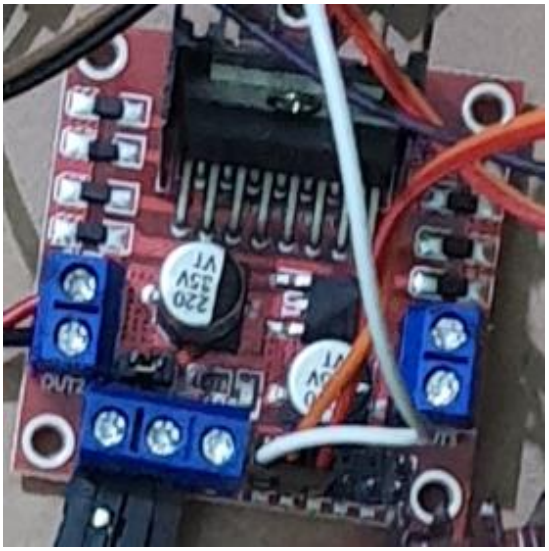

Figure 3.10 L298N Motor Driver Module

Figure 3.11 shows DC to DC convertor that converts 12V input voltage into 3.3V and 5V. It works on Buck convertor (Step Down Voltage Converter) principle. This Module is used to stepdown the voltage which is to be used as power source to the ESP32 Microcontroller. It outputs 3.3V at 800mA maximum current, 5V at 800mA maximum current. It supports 5.5mm Power jack.

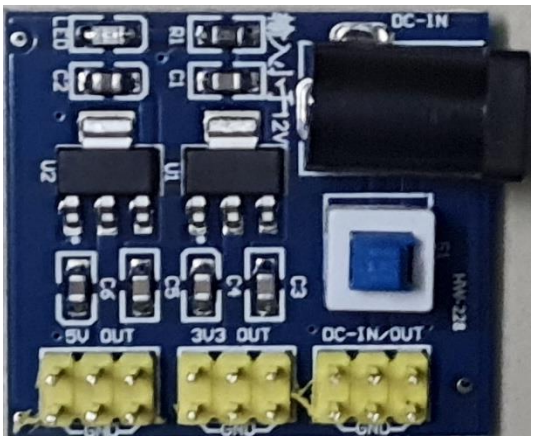

Figure 3.11 DC-DC 12V to 3.3V and 5V Power Module Multi Output Voltage converter

Figure 3.12 shows 8000mAh Lithium Polymer Battery Pack which supplies a voltage of 11.1V, maximum continuous discharge of 30C (240A) and maximum burst discharge of 60C (480A). It is a 3 cell battery pack. It supports heavy duty discharge leads that minimize resistance and can handle high current loads. It weighs around 600 grams.

Figure 3.12 8000mAh Lithium Polymer Battery Pack

Figure 3.13 shows the DC Power Jack Male connector 2.1X5.5 mm size. It is used to supply 12V DC input from LiPo battery to the DC-DC 12v to 3.3V and 4V Power Module. The jack can fit 5.5mm barrel jacks and have 2.1mm center pole.

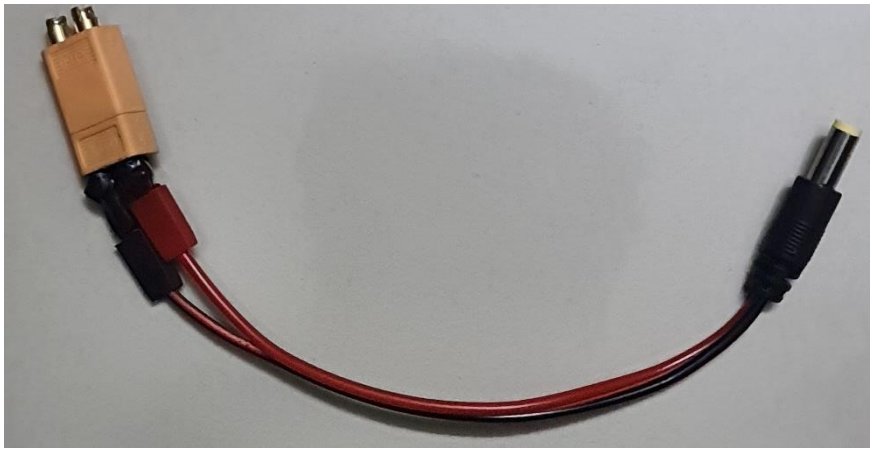

Figure 3.13 DC Power Jack Male connector

Figure 3.14 shows Two channel Logic Level converter. It is bidirectional logic level converter for stepping down 5V signals to 3.3V and stepping up 3.3V to 5V at same time. It works also with 2.8V and 1.8V devices. It supports I2C and UART communication protocols.

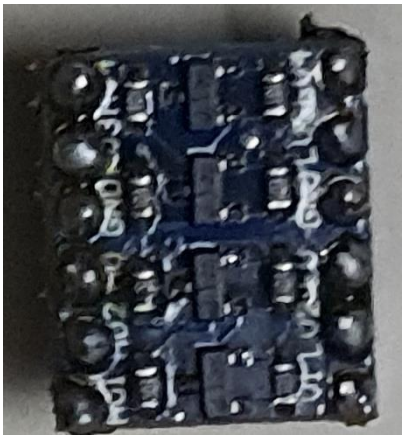

Figure 3.14 Two Channel Logic Level Converter

Figure 3.15 shows M12 threaded rod. It is used to provide as a guiding alignment locator for the two rocker arms and rear Aluminum tubes. It also provides extra torsional stiffness for the chassis frame. The rocker arm without these guiding rod lead to dislocation of the arms and thereby the two rocker arms were rotating about suspension springs which lead to issue of the robot not traversing straight path when moved front or back

Figure 3.15 M12 Threaded Guiding rod

Figure 3.16 shows mounting brackets that connects the Upper Chassis frame with the lower Suspension Springs also it holds the rear Aluminum tubes. It is a C shaped bracket that snuggly fits the Suspension springs and the aluminum frame thereby arresting the relative rotational movement.

Figure 3.16 Mounting brackets connecting Upper chassis with Suspension spring

Figure 3.17 shows Suspension Spring – Rocker arm Bracket. It is a connecting member that connects the Suspension Spring on one side and Rocker arm on other side. It is used to firmly secure the two components and also to arrest the rotation of the rocker arm about its axis.

Figure 3.17 Suspension Spring – Rocker arm Bracket

Figure 3.18 shows Geared motor mounting bracket. It firmly secures the Geared motor at one end and wheel is attached to motor on the other end. The mounting bracket is attached to the aluminum tubes using bolts.

Figure 3.18 Geared Motor mounting bracket

Figure 3.19 shows Aluminum Reinforcing tubes, which adds extra Bending and Torsional stiffness to the Chassis frame thereby avoiding the deflection of the frame during higher loads.

Figure 3.19 Reinforcing Aluminum tube

Figure 3.20 shows USB to UART TTL 5V 3.3V FT232RL Download cable to Serial Adapter module. It is used primarily for downloading the program codes from the computer to the ESP32 Cam WiFi microcontroller, since this module doesn't have internal programmer. The USB power has a current protection using 500mA self restore fuse. It has RXD/TXD transceiver for communication. It supports 3.3V and 5V

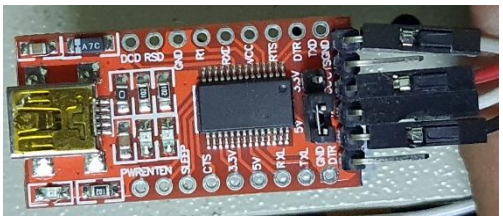

Figure 3.20 USB TO UART TTL 5V 3.3V FT232RL Download Cable To Serial Adapter Module

Figure 3.21 shows the assembled setup of the miniature robot arm. It has gripper arm to grab and hold the fruit in place. The Base plate is where the robot other links are mounted. The links are used to transfer motion for the robots like lifting the arm to raise the gripper position to grab the fruit.

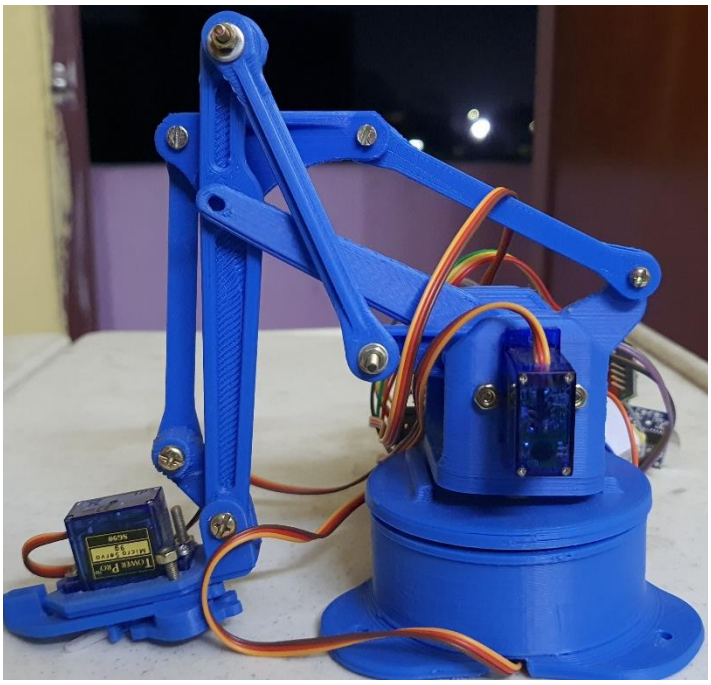

Figure 3.21 Miniature Robot Arm in assembled condition

Figure 3.22 shows the SG90 servo motor which is a three pole ferrite motor with nylon gears. It has an operating voltage of about 4.8 volts to 6.0 volts. It has operating speed of 0.12 sec/60 degree and a output torque of around 1.6kgcm at 4.8 voltage.

Figure 3.22 Tower Pro SG90 Servo motor

Figure 3.23 shows the Ultrasonic Range Finder sensor (model HC-SR04). The ultrasonic sensor has an operating voltage value of 5 Volts and a operating current is around 15mA. The effectual angle is less than 15°. The resolution is around 0.3 cm. Measuring angle is about 30°. The triggering pulse width is 10uS TTL pulse. The sensing sonar range varies between 2 cm to 400 cm. The maximum sensing range for the sensor is 450 cm. The operating frequency range is 40 kHz. The Echo signal output is TTL pulse which is proportional to the range of the distance.

Figure 3.23 Ultrasonic Range Finder Sensor (HC-SR04)

Figure 3.24 shows the MPU6050 sensor. It has a three axis accelerometer included in it along with the three axis gyroscope. The sensor uses the I2C protocol to establish connection between the microcontroller and the sensor. It operates at an input voltage range of about 2.3 volts to 3.4 volts. The three axis gyroscope has a sensitivity upto a value of about 131 LSBs/dps and has full operating range of ±250, ±500, ±1000 and ±2000 dps. The three axis accelerometer have a full scale programmable range of about ±2g, ±4g, ±8g and ±16g values. It has support for run time bias and calibration of compass. Along with Gyroscope and accelerometer it has support for the temperature sensing.

Figure 3.24 MPU6050 Sensor

Figure 3.25 shows the ESP32 microcontroller WROOM32 model. It has an inbuilt WiFi and BLE support. It has USB-UART port to upload program on to the board as it has inbuilt boot loader. Also, it has reset, boot modes, LDO regulator. It has greater support for the LWIP protocols and FreeRTOS. It also supports three modes namely AP, STA, AP+STA. It has 4MB Flash memory. 80mA current. It supports an input voltage range of 2.2 volts to 3.6 volts. Data Rate is about 54 Mbps and operating frequency of 2.4 GHz.

Figure 3.25 ESP32 Microcontroller

CHAPTER 4

RESULTS AND DISCUSSION

4.1 VIRTUAL JOYSTICK CONTROL OF MOBILE ROBOT

The Figure 4.1 shows the Mobile Robot prototype that traverse around the environment with the inputs given from the joystick control using the internet. It shows the live camera feed of the environment which helps to move around in the Mining environment setup

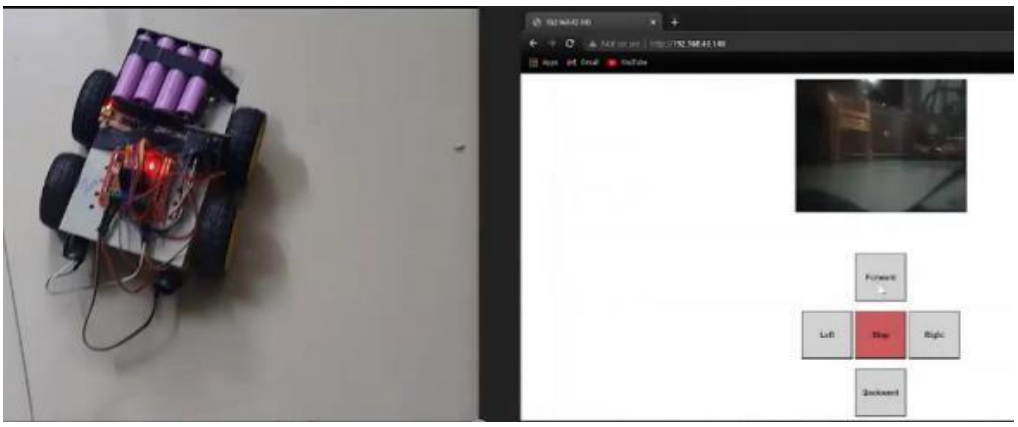

Figure 4.1 A demonstration of the mobile robot movement using the virtual joystick

Figure 4.2 shows in details how the simple dashboard is built. The Dash board is created using Java script which works on the Hyper Text Markup Language (HTTP) protocol to actuate the motor with 5 functions namely

- Forward
- Backward
- Left
- Right

Additional functionality of Led Lights On and Led Lights Off are provided to operate mobile robot under the dark Environment.

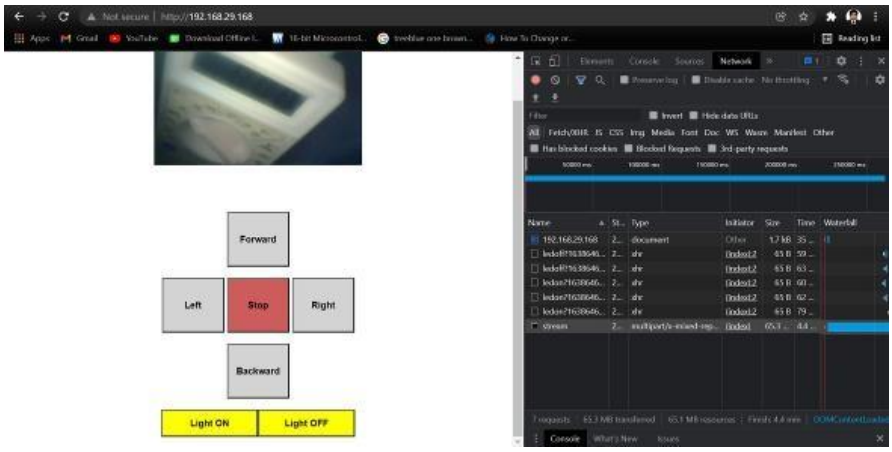

Figure 4.2 Virtual Joystick control of the Mobile Robot using Laptop

The Algorithm to control the Mobile Robot is by controlling the logic level of the PWM and the Direction control. Table 4.1 shows the logic sequence to be carried out in the logic pins namely Pulse Width Modulation and Direction control pins

Table 4.1 Table indicating the Logic Sequence for Moving the Robot

Condition	Logic Pins	Logic criteria
At Start	Left-Front	Low
	Left-Back	Low
	Right-Front	Low
	Right-Back	Low
Forward	Left-Front	High
	Left-Back	Low
	Right-Front	High
	Right-Back	Low
Backward	Left-Front	Low
	Left-Back	High
	Right-Front	Low
	Right-Back	High
Right	Left-Front	High
	Left-Back	Low
	Right-Front	Low
	Right-Back	High
Left	Left-Front	Low
	Left-Back	High
	Right-Front	High
	Right-Back	Low
Stop	PWM_1	Low
	PWM_2	Low

Figure 4.3 shows the interface for controlling the Mobile robot movement from the Android phone instead of the web browser of laptop. By entering the IP address and port number. All the options obtained in the desktop mode browser is also accessible in the mobile version.

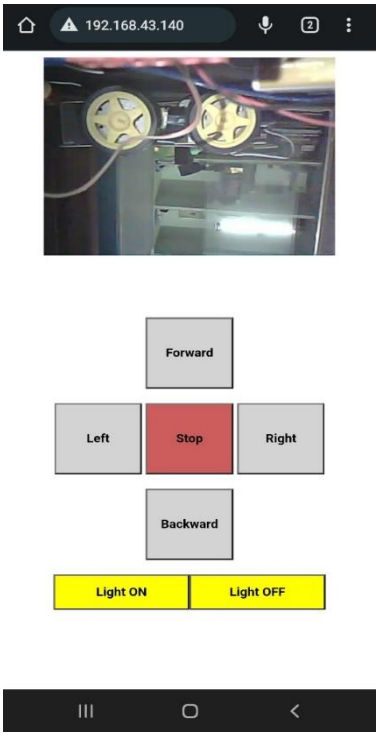

Figure 4.3 Virtual Joystick control of the Mobile Robot using Mobile

4.2 SIMULATION AND PROTOTYPE

The different terrain traversal of mobile robot [15] is simulated in solidworks motion study to access the manoeurability of the robot. Figure 4.4 shows Rocker Boggie model with suspension

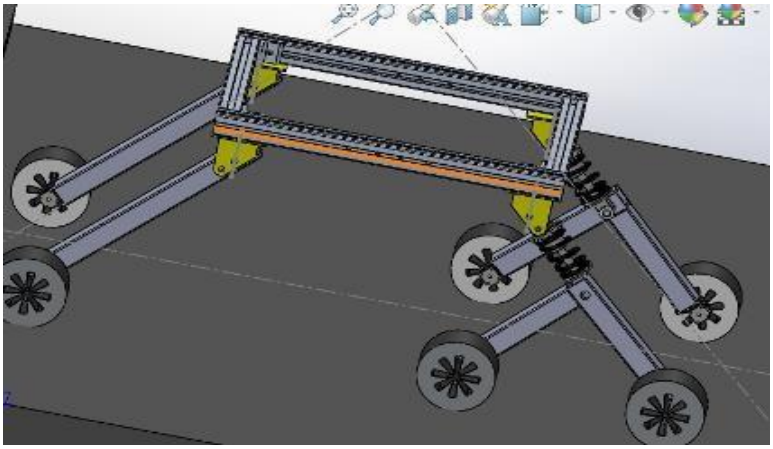

Figure 4.4 Rocker Boggie Model with suspension

Figure 4.5 shows the Motion study analysis done in different bumpy terrain condition

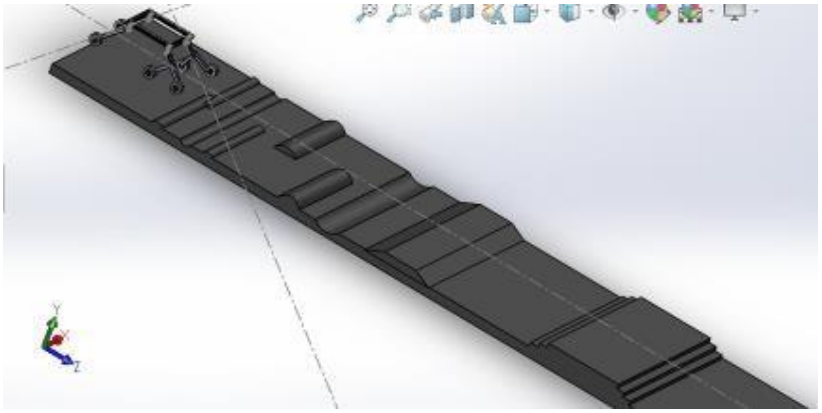

Figure 4.5 Motion study analysis of Model done in different bumpy conditions

Figure 4.6 shows the Motion study analysis done in High and Low bumps terrain condition

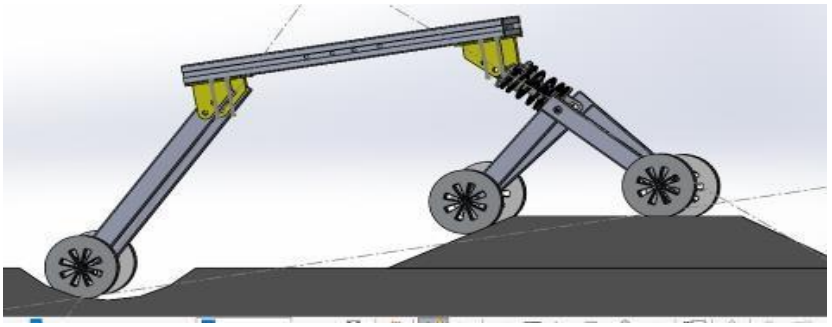

Figure 4.6 High and Low bump Traversal of Mobile Robot

Figure 4.7 shows the robot climbing up the stairs by using the rocker boggie and suspension mechanism.

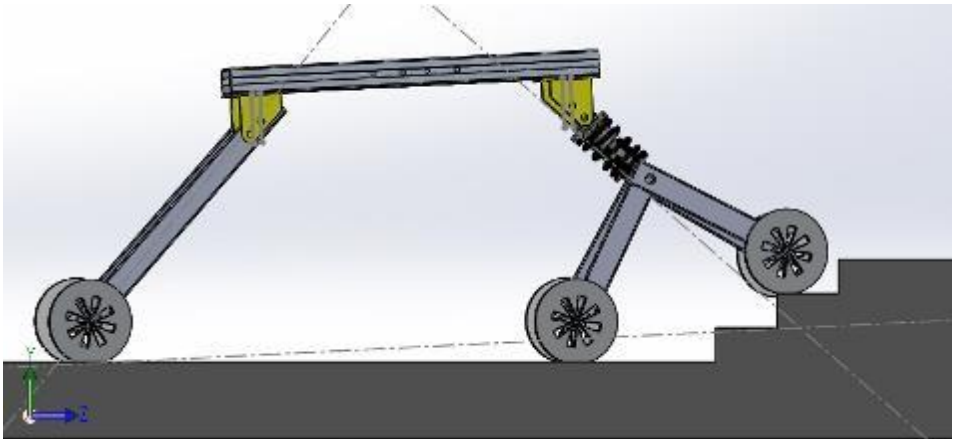

Figure 4.7 Step traversal of Robot – Climbing up

Figure 4.8 shows the Robot climbing down the stairs

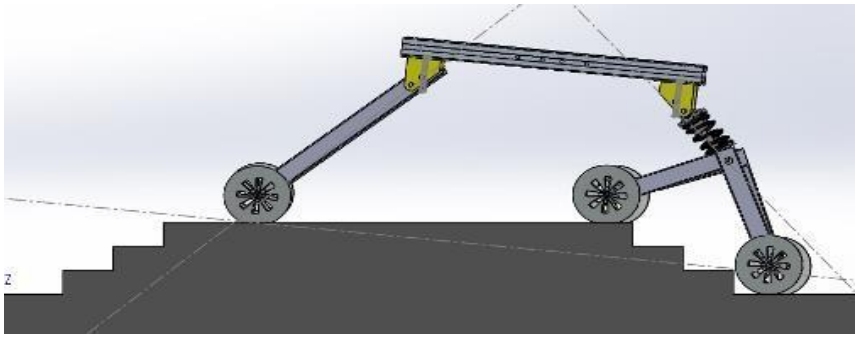

Figure 4.8 Step traversal of Robot – Climbing down

Figure 4.9 shows the Prototype build with the PVC pipe and wood.

Figure 4.9 Rocker Boggie Mobile Robot prototype

Figure 4.10 shows the Forces acting on the Mobile Robot

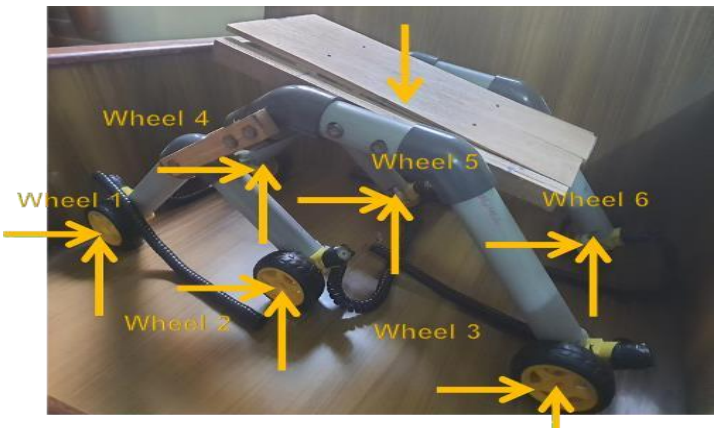

Figure 4.10 Forces acting on the Mobile robot

Figure 4.11 shows the Overall Electric Schematic Connection. The program is uploaded to ESP32CAM via TTL, then the two channel logic converter steps up 3.3V from microcontroller to 5V, which then powers Motor Driver. All these are powered by LiPo Battery. At last the DC geared motor rotates based on joystick command.

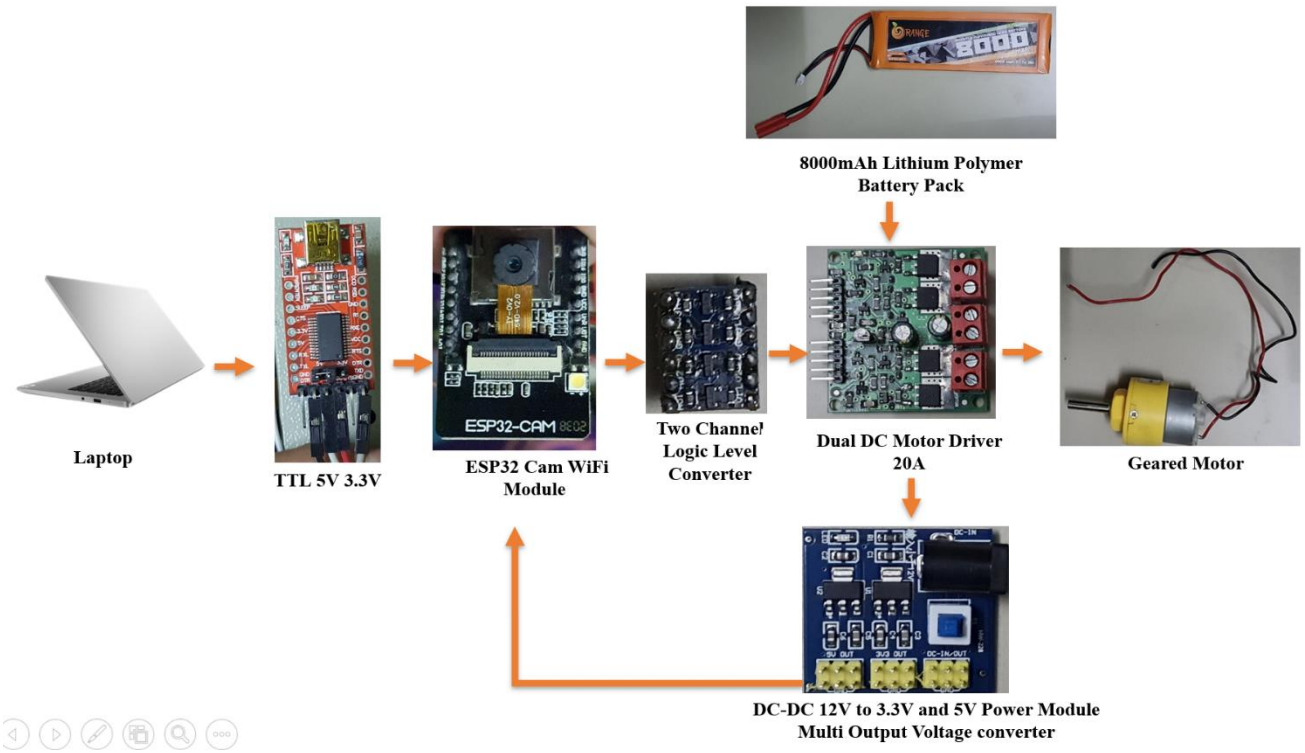

Figure 4.11 Overall Electric Schematic Connection – Previous version

Figure 4.12 shows the Johnson geared motor which was initially planned to be used for the project to drive the mobile robot. But there was a problem related to the design/ manufacturing of the motor. The image shows that the gear used in motor is overlapping with the hole which is meant for using the mounting screws. Since this problem existed the motor wasn't able to rotate due to the interference between the screw and the geared motor. To address this issue a new geared motor was replaced.

Figure 4.12 Johnson geared motor - problem

Figure 4.13 shows the overall Electric schematic connection after replacing the Dual DC motor driver 20A with L298N and the new

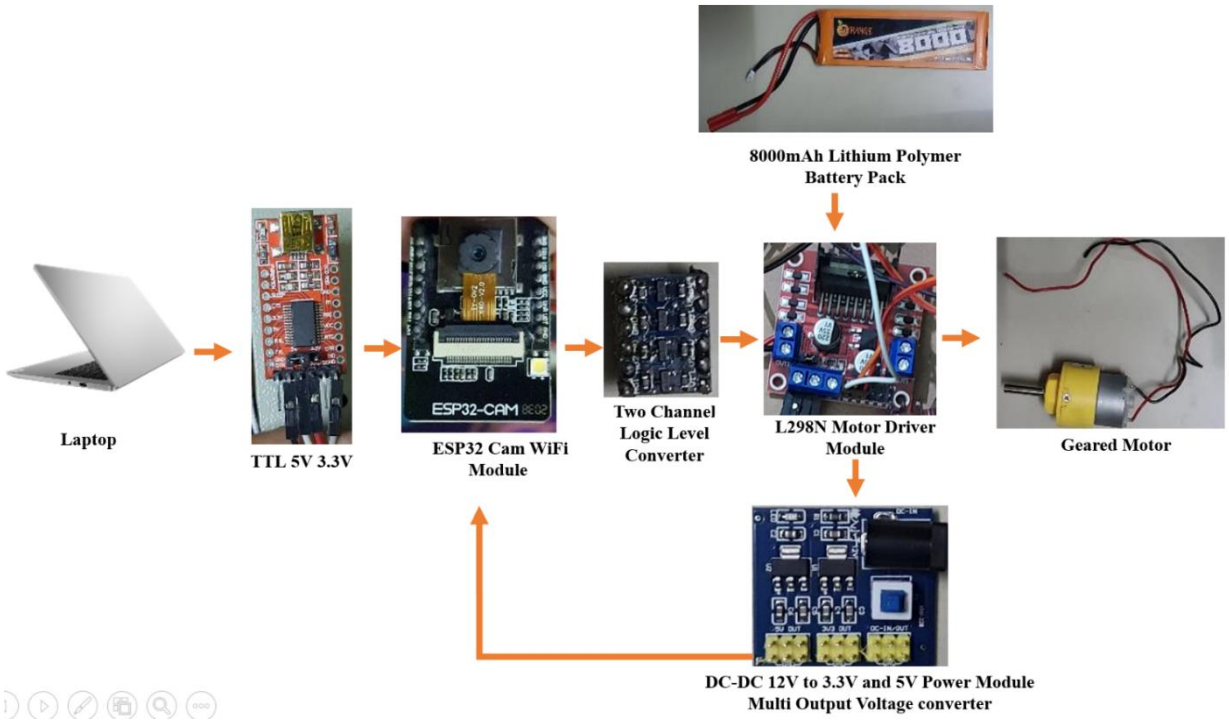

Figure 4.13 Overall Electric Schematic Connection – Final version

Figure 4.14 shows the overall Actual Electric connection. As seen from the image, individual L298N Motor drivers are used to connect.

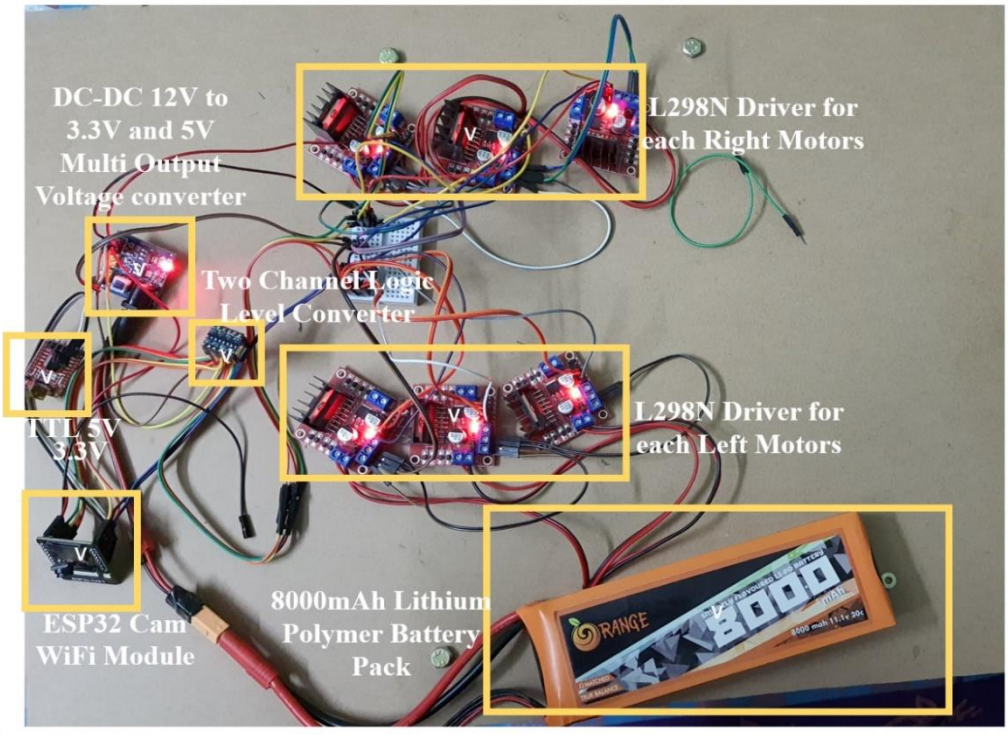

Figure 4.14 Actual Electric Connection

4.3 FINAL MODEL

Figure 4.15 shows the forward control of the Mobile robot from the mobile phone

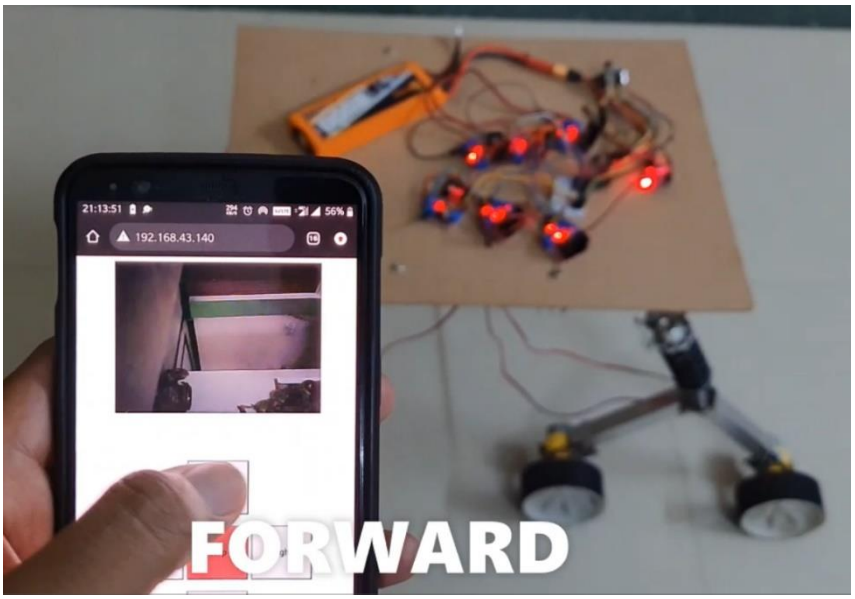

Figure 4.15 Forward Control of Robot

Figure 4.16 shows the Backward control of the Mobile robot from the mobile phone

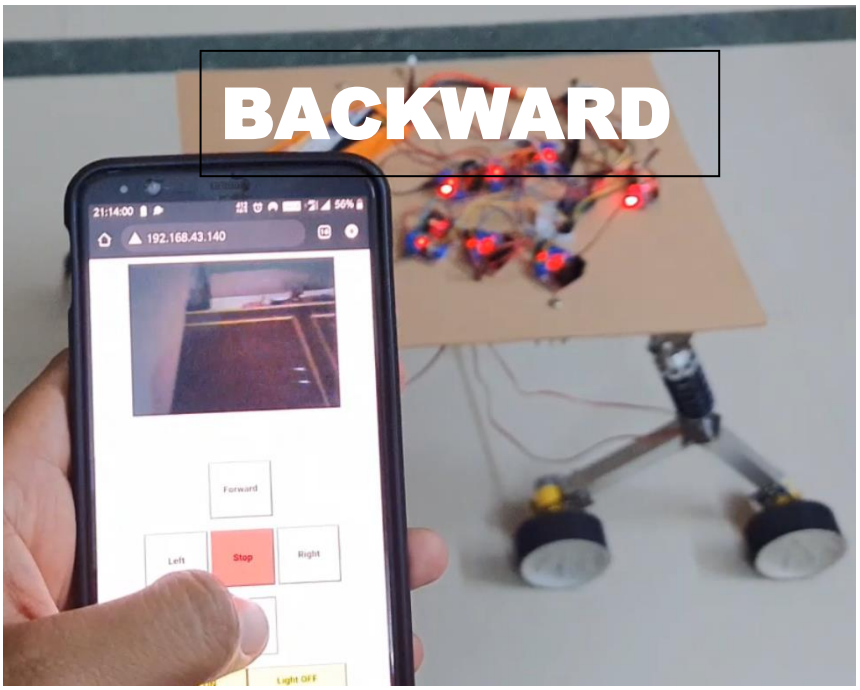

Figure 4.16 Backward Control of Robot

Figure 4.17 shows the LED On control of the Mobile robot from the mobile phone

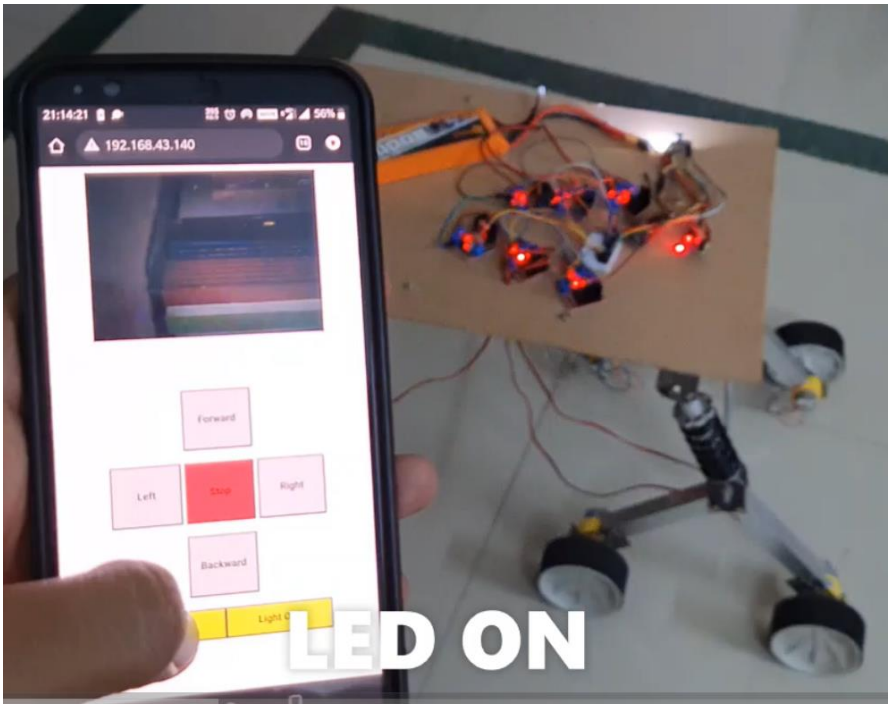

Figure 4.17 LED On control

Figure 4.18 shows the LED Off control of the Mobile robot from the mobile phone

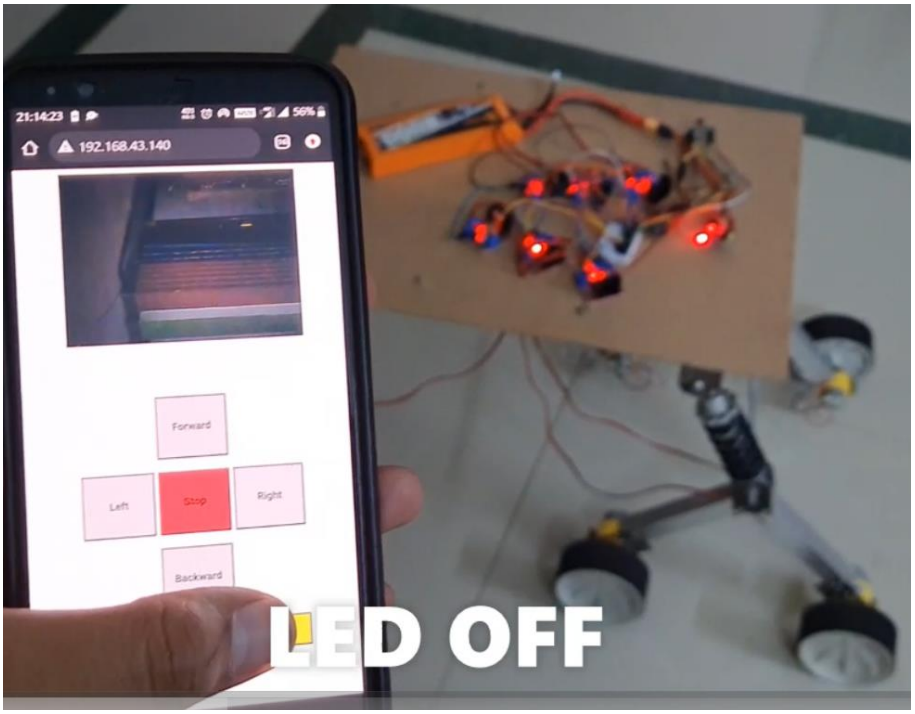

Figure 4.18 LED Off control

Figure 4.19 shows Slope Climb control of the Mobile robot from the mobile phone

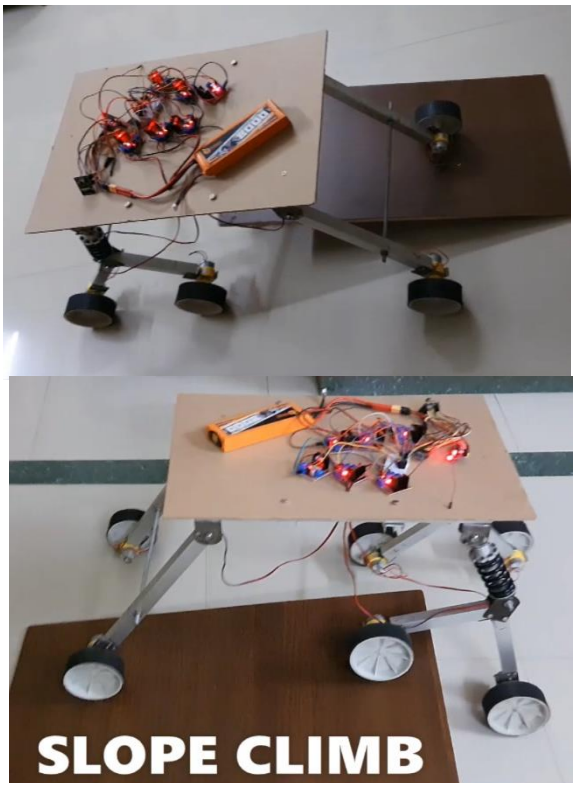

Figure 4.19 Slope Climb Control

Figure 4.20 shows Left and Right control of the Mobile robot from the mobile phone.

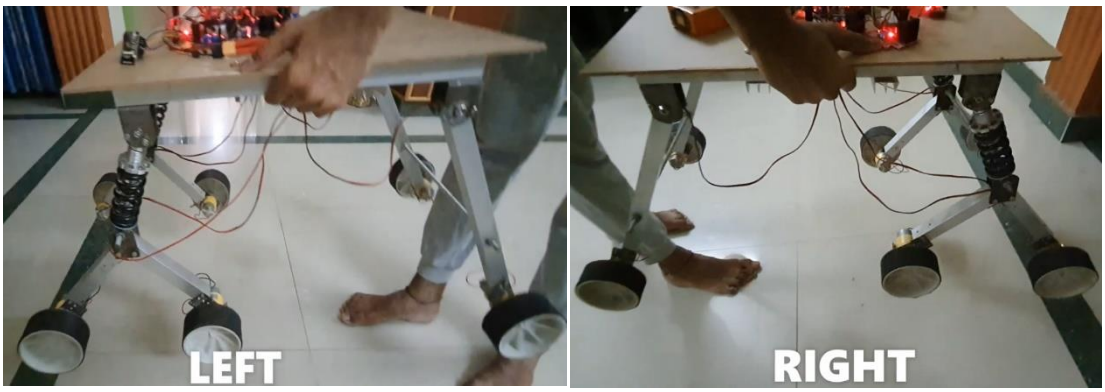

Figure 4.20 Left and Right Control

Figure 4.21 shows Bumpy Mud Terrain ride of the Mobile Robot from the mobile phone.

Figure 4.21 Bumpy Mud Terrain ride

4.4 IMAGE PROCESSING FOR TOMATO DETECTION

This image is sent as a byte array to the local computer as shown below in Figure 4.22. From the byte array the needed image byte array is trimmed (spliced) from the header messages and image processing is done using Computer vision, after applying all the filters

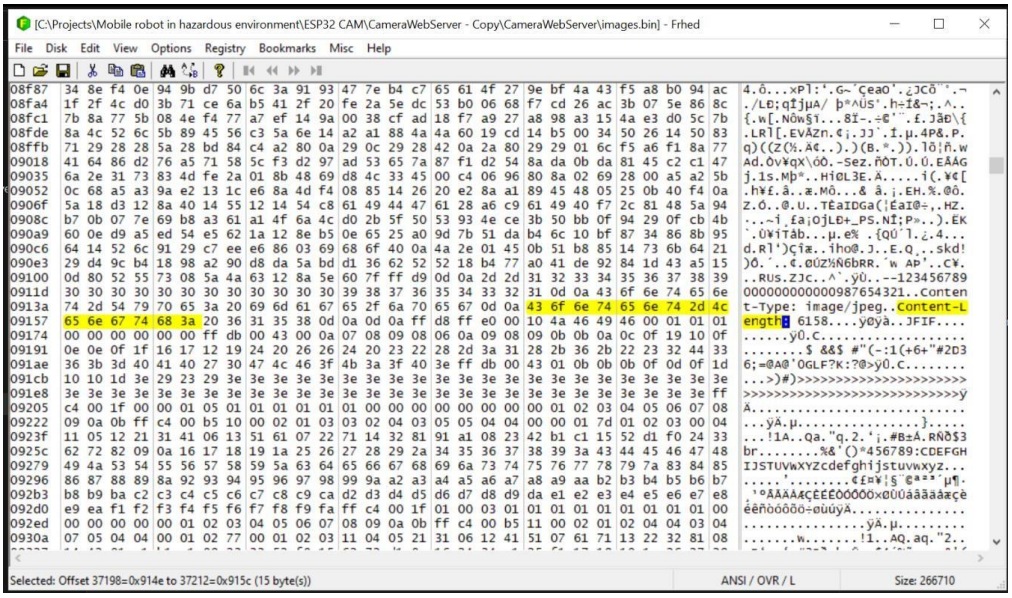

Figure 4.22 Image Byte array sent from ESP32 Cam to Local computer

Figure 4.23 shows the trackbar position that can be tweaked to isolate the red colour of the Tomato. The figure shows Hue minimum and maximum, Saturation Minimum and Maximum, Value Minimum and Maximum adjusting trackbars, adjusting by which the red colour can alone be separated.

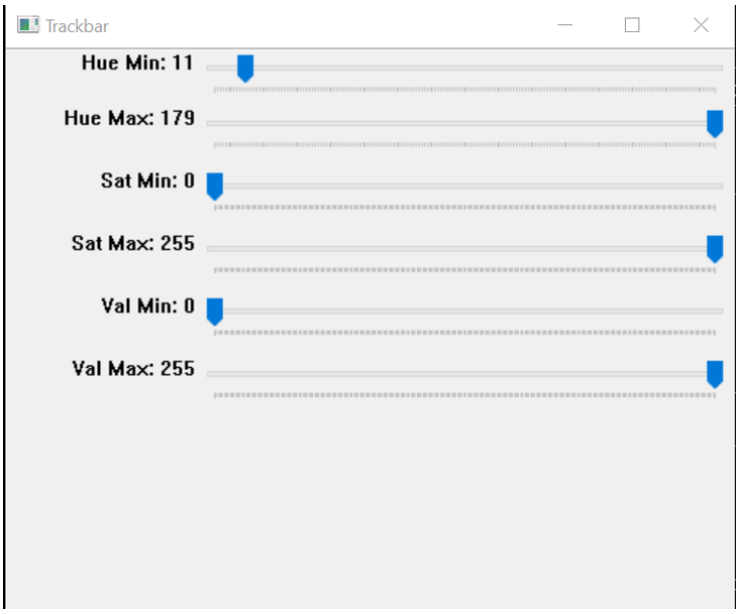

Figure 4.23 Hyper-parameters to isolate the colour

Figure 4.24 shows the image which is obtained from ESP32 Cam WiFi module which is in BGR which is converted to HSV which is used for further image processing

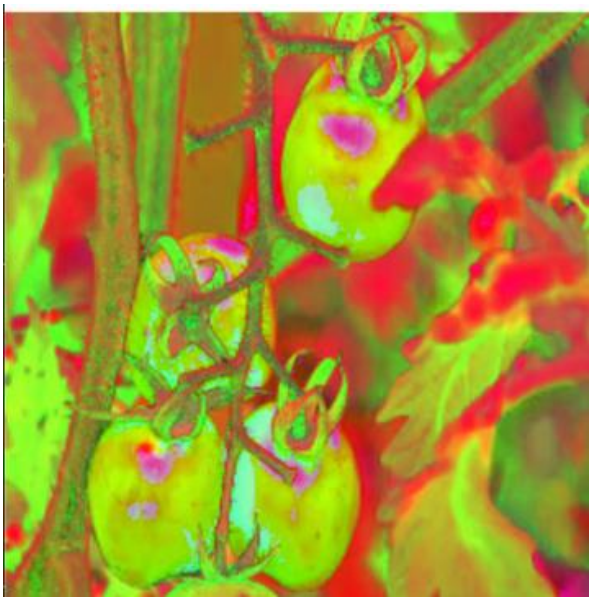

Figure 4.24 HSV Converted image from BGR

Figure 4.25 shows the image with Bitwise AND mask applied based on the HSV values noted for isolating the red colour of tomatoes.

Figure 4.25 Bitwise AND Mask applied to isolate the colour of interest

Figure 4.26 shows the image with Red colour of the Tomato isolated separately whose position is to be located in next step

Figure 4.26 Image with Red colour isolated

Figure 4.27 shows the image with Tomatoes' position located by the algorithm and indicated using the bounding box, also giving the 2D size of the Tomatoes.

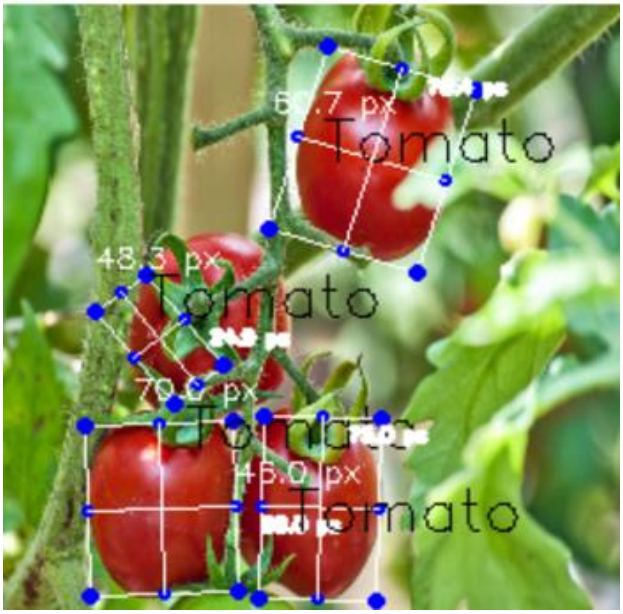

Figure 4.27 Tomatoes' position located and indicated using bounding box

Figure 4.28 shows the schematic image of the Multi core processing. Here the image is got in one processor from ESP32 Cam WiFi microcontroller and the image processing is done in another processor to make the work parallel and efficiently using octa core of the computer. It takes less time for processing a set of instructions and hence efficiency is maximum.

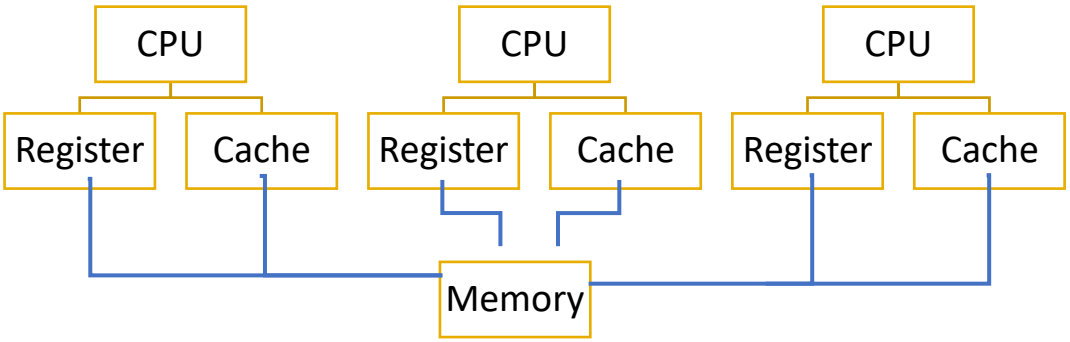

Figure 4.28 Multi Processing schematic

4.5 ROBOTIC ARM ACTUATION

Figure 4.29 shows the Robotic Arm which are actuated by using the servo motors. There are three servo motors in total. One motor is to control Gripper adjustment, another servo motor is to control the base of the robot and third servo motor to control link lift adjustments [7]. The MPU sensor is used to get the Gyroscopic, Accelerometer and Temperature values from the sensor. The ultrasonic sensor is used to get the distance information from the fruits and the mobile robot [12]. The entire components are controlled by the ESP32 microcontroller.

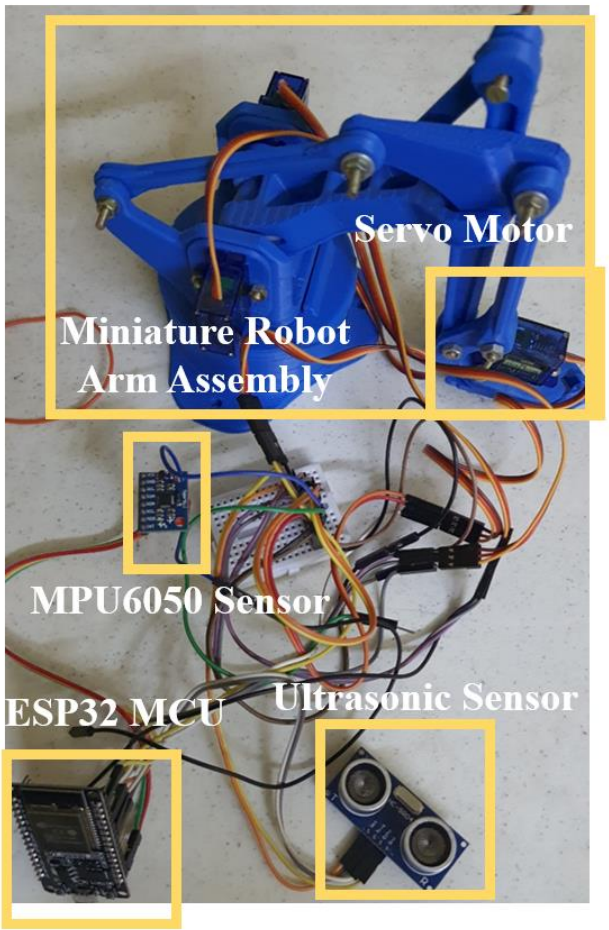

Figure 4.29 Overall Electric connection between miniature robotic arm, MPU6050 and Ultrasonic sensor

Figure 4.30 shows the Robotic Arm joint angle Control using the Gripper, Base, Arm servo motors along with Gyroscopic, accelerometer and temperature reading from MPU6050 [10] all these values are displayed in the client browser using the HTML, CSS, JS, C++ codes [11].

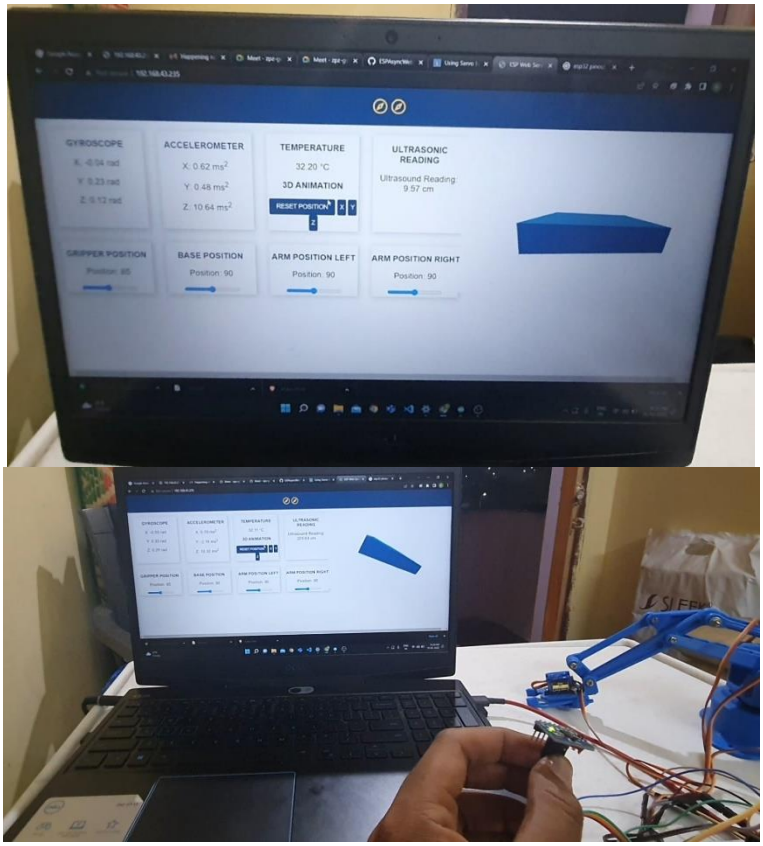

Figure 4.30 Robotic Arm Control with Gyroscopic, Accelerometer and Temperature, Ultrasonic readings

Figure 4.31 shows the sensors reading recorded in the SQL Database [13] viewed in the DB Browser software along with the time stamp.

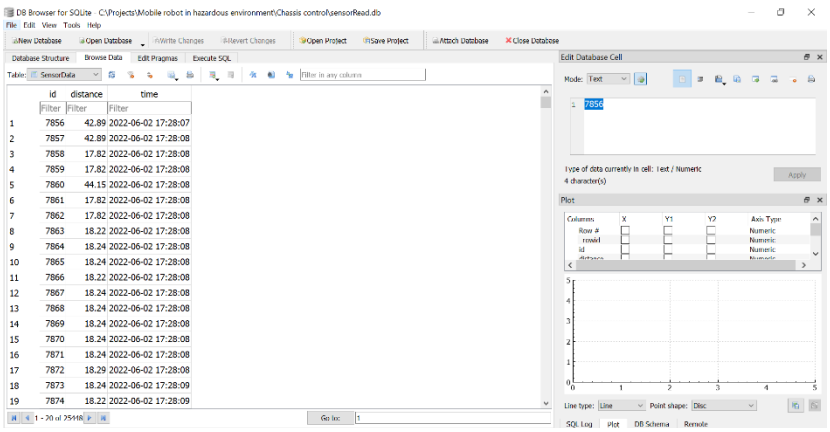

Figure 4.31 SQL Database to record Sensor Data

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

- In the Phase 2 of the Project, the objective of the project was changed from Mining Application to the Agricultural Application
- Full scale model of the All Terrain Mobile Robot was done
- Image Processing Algorithm to detect Tomatoes was done
- SQL Database was developed to store the sensor values to be used for future reference
- Robotic Arm Control Algorithm was developed to move the arms to desired position
- In the future, an Algorithm to automate the robotic fruit detection and pick process is to be developed
- The full scale robot arm is to be developed
- Present Robot has less Motor torques and Flat wheels which is to be replaced with high torque motors and bigger wheels to make left and right control more precise

CHAPTER 6

APPENDICES

6.1 APPENDIX 1- CODE FOR VIRTUAL JOYSTICK CONTROL OF MOTOR

```
#include "esp_camera.h"
#include <WiFi.h>
#define CAMERA_MODEL_AI_THINKER

const char* ssid = "";      //SSID WIFI Name
const char* password = "";      //WIFI Password


#if defined(CAMERA_MODEL_WROVER_KIT)
#define PWDN_GPIO_NUM       -1
#define RESET_GPIO_NUM      -1
#define XCLK_GPIO_NUM       21
#define SIOD_GPIO_NUM       26
#define SIOC_GPIO_NUM       27

#define Y9_GPIO_NUM         35
#define Y8_GPIO_NUM         34
#define Y7_GPIO_NUM         39
#define Y6_GPIO_NUM         36
#define Y5_GPIO_NUM         19
#define Y4_GPIO_NUM         18
#define Y3_GPIO_NUM          5
#define Y2_GPIO_NUM          4
#define VSYNC_GPIO_NUM      25
#define HREF_GPIO_NUM       23
#define PCLK_GPIO_NUM       22


#elif defined(CAMERA_MODEL_AI_THINKER)
#define PWDN_GPIO_NUM        32
#define RESET_GPIO_NUM       -1
#define XCLK_GPIO_NUM         0
#define SIOD_GPIO_NUM        26
#define SIOC_GPIO_NUM        27

#define Y9_GPIO_NUM          35
#define Y8_GPIO_NUM          34
#define Y7_GPIO_NUM          39
#define Y6_GPIO_NUM          36
#define Y5_GPIO_NUM          21
```

```
#else
#error "Camera model not selected"
#endif

// GPIO Setting
extern int gpLb =   2; // Left 1
extern int gpLf = 14; // Left 2
extern int gpRb = 15; // Right 1
extern int gpRf = 13; // Right 2
int ena = 0; // MotorA
int enb = 16; // MotorB
extern int gpLed =   4; // Light
extern String WiFiAddr ="";

void startCameraServer();

void setup() {
  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.println();

  pinMode(gpLb, OUTPUT); //Left Backward
  pinMode(gpLf, OUTPUT); //Left Forward
  pinMode(gpRb, OUTPUT); //Right Forward
  pinMode(gpRf, OUTPUT); //Right Backward
  pinMode(gpLed, OUTPUT); //Light

  //initialize
  digitalWrite(gpLb, LOW);
  digitalWrite(gpLf, LOW);
  digitalWrite(gpRb, LOW);
  digitalWrite(gpRf, LOW);
  digitalWrite(gpLed, LOW);
  digitalWrite(ena, HIGH);
  digitalWrite(enb, HIGH);
  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
```

```
config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;
  //init with high specs to pre-allocate larger buffers
  if(psramFound()){
     config.frame_size = FRAMESIZE_UXGA;
     config.jpeg_quality = 10;
     config.fb_count = 2;
  } else {
     config.frame_size = FRAMESIZE_SVGA;
     config.jpeg_quality = 12;
     config.fb_count = 1;
  }

  // camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
     Serial.printf("Camera init failed with error 0x%x", err);
     return;
  }

  //drop down frame size for higher initial frame rate
  sensor_t * s = esp_camera_sensor_get();
  s->set_framesize(s, FRAMESIZE_CIF);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
     delay(500);
     Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  startCameraServer();
  Serial.print("Camera Ready! Use 'http://");
  Serial.print(WiFi.localIP());
  WiFiAddr = WiFi.localIP().toString();
  Serial.println("' to connect");
```

```
}

void loop() {
  }

#include "esp_http_server.h"
#include "esp_timer.h"
#include "esp_camera.h"
#include "img_converters.h"
#include "camera_index.h"
#include "Arduino.h"
extern int gpLb;
extern int gpLf;
extern int gpRb;
extern int gpRf;
extern int gpLed;
extern String WiFiAddr;
void WheelAct(int nLf, int nLb, int nRf, int nRb);

typedef struct {
        size_t size; //number of values used for filtering
        size_t index; //current value index
        size_t count; //value count
        int sum;
        int * values;
} ra_filter_t;

typedef struct {
        httpd_req_t *req;
        size_t len;
} jpg_chunking_t;

#define PART_BOUNDARY "123456789000000000000987654321"
static const char* _STREAM_CONTENT_TYPE =
"multipart/x-mixed-replace;boundary=" PART_BOUNDARY;
static const char* _STREAM_BOUNDARY = "\r\n--" PART_BOUNDARY
"\r\n";
static const char* _STREAM_PART = "Content-Type:
image/jpeg\r\nContent-Length: %u\r\n\r\n";
static ra_filter_t ra_filter;
httpd_handle_t stream_httpd = NULL;
httpd_handle_t camera_httpd = NULL;

static ra_filter_t * ra_filter_init(ra_filter_t * filter, size_t sample_size){
    memset(filter, 0, sizeof(ra_filter_t));
```

```
if(!filter->values){
        return NULL;
    }
    memset(filter->values, 0, sample_size * sizeof(int));

    filter->size = sample_size;
    return filter;
}

static int ra_filter_run(ra_filter_t * filter, int value){
    if(!filter->values){
        return value;
    }
    filter->sum -= filter->values[filter->index];
    filter->values[filter->index] = value;
    filter->sum += filter->values[filter->index];
    filter->index++;
    filter->index = filter->index % filter->size;
    if (filter->count < filter->size) {
        filter->count++;
    }
    return filter->sum / filter->count;
}

static size_t jpg_encode_stream(void * arg, size_t index, const void* data,
size_t len){
    jpg_chunking_t *j = (jpg_chunking_t *)arg;
    if(!index){
        j->len = 0;
    }
    if(httpd_resp_send_chunk(j->req, (const char *)data, len) != ESP_OK){
        return 0;
    }
    j->len += len;
    return len;
}

static esp_err_t capture_handler(httpd_req_t *req){
    camera_fb_t * fb = NULL;
    esp_err_t res = ESP_OK;
    int64_t fr_start = esp_timer_get_time();

    fb = esp_camera_fb_get();
    if (!fb) {
        Serial.printf("Camera capture failed");
        httpd_resp_send_500(req);
```

```
}

    httpd_resp_set_type(req, "image/jpeg");
    httpd_resp_set_hdr(req, "Content-Disposition", "inline;
filename=capture.jpg");

    size_t fb_len = 0;
    if(fb->format == PIXFORMAT_JPEG){
        fb_len = fb->len;
        res = httpd_resp_send(req, (const char *)fb->buf, fb->len);
    } else {
        jpg_chunking_t jchunk = {req, 0};
        res = frame2jpg_cb(fb, 80, jpg_encode_stream,
&jchunk)?ESP_OK:ESP_FAIL;
        httpd_resp_send_chunk(req, NULL, 0);
        fb_len = jchunk.len;
    }
    esp_camera_fb_return(fb);
    int64_t fr_end = esp_timer_get_time();
    Serial.printf("JPG: %uB %ums", (uint32_t)(fb_len), (uint32_t)((fr_end -
fr_start)/1000));
    return res;
}

static esp_err_t stream_handler(httpd_req_t *req){
    camera_fb_t * fb = NULL;
    esp_err_t res = ESP_OK;
    size_t _jpg_buf_len = 0;
    uint8_t * _jpg_buf = NULL;
    char * part_buf[64];

    static int64_t last_frame = 0;
    if(!last_frame) {
        last_frame = esp_timer_get_time();
    }

    res = httpd_resp_set_type(req, _STREAM_CONTENT_TYPE);
    if(res != ESP_OK){
        return res;
    }

    while(true){
        fb = esp_camera_fb_get();
        if (!fb) {
            Serial.printf("Camera capture failed");
            res = ESP_FAIL;
```

```
 bool jpeg_converted = frame2jpg(fb, 80, &_jpg_buf, &_jpg_buf_len);
                        esp_camera_fb_return(fb);
                        fb = NULL;
                        if(!jpeg_converted){
                              Serial.printf("JPEG compression failed");
                              res = ESP_FAIL;
                        }
                  } else {
                        _jpg_buf_len = fb->len;
                        _jpg_buf = fb->buf;
                  }
            }
            if(res == ESP_OK){
                  size_t hlen = snprintf((char *)part_buf, 64, _STREAM_PART,
_jpg_buf_len);
                  res = httpd_resp_send_chunk(req, (const char *)part_buf, hlen);
            }
            if(res == ESP_OK){
                  res = httpd_resp_send_chunk(req, (const char *)_jpg_buf,
_jpg_buf_len);
            }
            if(res == ESP_OK){
                  res = httpd_resp_send_chunk(req, _STREAM_BOUNDARY,
strlen(_STREAM_BOUNDARY));
            }
            if(fb){
                  esp_camera_fb_return(fb);
                  fb = NULL;
                  _jpg_buf = NULL;
            } else if(_jpg_buf){
                  free(_jpg_buf);
                  _jpg_buf = NULL;
            }
            if(res != ESP_OK){
                  break;
            }
            int64_t fr_end = esp_timer_get_time();

            int64_t frame_time = fr_end - last_frame;
            last_frame = fr_end;
            frame_time /= 1000;
            uint32_t avg_frame_time = ra_filter_run(&ra_filter, frame_time);
            Serial.printf("MJPG: %uB %ums (%.1ffps), AVG: %ums (%.1ffps)"
                  ,(uint32_t)(_jpg_buf_len),
                  (uint32_t)frame_time, 1000.0 / (uint32_t)frame_time,
                  avg  frame  time, 1000.0 / avg  frame  time
```

```
}

    last_frame = 0;
    return res;
}

static esp_err_t cmd_handler(httpd_req_t *req){
    char*    buf;
    size_t buf_len;
    char variable[32] = {0,};
    char value[32] = {0,};

    buf_len = httpd_req_get_url_query_len(req) + 1;
    if (buf_len > 1) {
        buf = (char*)malloc(buf_len);
        if(!buf){
            httpd_resp_send_500(req);
            return ESP_FAIL;
        }
        if (httpd_req_get_url_query_str(req, buf, buf_len) == ESP_OK) {
            if (httpd_query_key_value(buf, "var", variable, sizeof(variable))
== ESP_OK &&
                httpd_query_key_value(buf, "val", value, sizeof(value)) ==
ESP_OK) {
            } else {
                free(buf);
                httpd_resp_send_404(req);
                return ESP_FAIL;
            }
        } else {
            free(buf);
            httpd_resp_send_404(req);
            return ESP_FAIL;
        }
        free(buf);
    } else {
        httpd_resp_send_404(req);
        return ESP_FAIL;
    }

    int val = atoi(value);
    sensor_t * s = esp_camera_sensor_get();
    int res = 0;
    if(!strcmp(variable, "framesize")) {
        if(s->pixformat == PIXFORMAT_JPEG) res = s->set_framesize(s,
(framesize_t)val);
```

```
}
    else if(!strcmp(variable, "quality")) res = s->set_quality(s, val);
    else if(!strcmp(variable, "contrast")) res = s->set_contrast(s, val);
    else if(!strcmp(variable, "brightness")) res = s->set_brightness(s, val);
    else if(!strcmp(variable, "saturation")) res = s->set_saturation(s, val);
    else if(!strcmp(variable, "gainceiling")) res = s->set_gainceiling(s,
(gainceiling_t)val);
    else if(!strcmp(variable, "colorbar")) res = s->set_colorbar(s, val);
    else if(!strcmp(variable, "awb")) res = s->set_whitebal(s, val);
    else if(!strcmp(variable, "agc")) res = s->set_gain_ctrl(s, val);
    else if(!strcmp(variable, "aec")) res = s->set_exposure_ctrl(s, val);
    else if(!strcmp(variable, "hmirror")) res = s->set_hmirror(s, val);
    else if(!strcmp(variable, "vflip")) res = s->set_vflip(s, val);
    else if(!strcmp(variable, "awb_gain")) res = s->set_awb_gain(s, val);
    else if(!strcmp(variable, "agc_gain")) res = s->set_agc_gain(s, val);
    else if(!strcmp(variable, "aec_value")) res = s->set_aec_value(s, val);
    else if(!strcmp(variable, "aec2")) res = s->set_aec2(s, val);
    else if(!strcmp(variable, "dcw")) res = s->set_dcw(s, val);
    else if(!strcmp(variable, "bpc")) res = s->set_bpc(s, val);
    else if(!strcmp(variable, "wpc")) res = s->set_wpc(s, val);
    else if(!strcmp(variable, "raw_gma")) res = s->set_raw_gma(s, val);
    else if(!strcmp(variable, "lenc")) res = s->set_lenc(s, val);
    else if(!strcmp(variable, "special_effect")) res = s->set_special_effect(s,
val);
    else if(!strcmp(variable, "wb_mode")) res = s->set_wb_mode(s, val);
    else if(!strcmp(variable, "ae_level")) res = s->set_ae_level(s, val);
    else {
        res = -1;
    }

    if(res){
        return httpd_resp_send_500(req);
    }

    httpd_resp_set_hdr(req, "Access-Control-Allow-Origin", "*");
    return httpd_resp_send(req, NULL, 0);
}
static esp_err_t status_handler(httpd_req_t *req){
    static char json_response[1024];

    sensor_t * s = esp_camera_sensor_get();
    char * p = json_response;
    *p++ = '{';
    p+=sprintf(p, "\"framesize\":%u,", s->status.framesize);
    p+=sprintf(p, "\"quality\":%u,", s->status.quality);
    p+=sprintf(p, "\"brightness\":%d,", s->status.brightness);
```

```
p+=sprintf(p, "\"saturation\":%d,", s->status.saturation);
        p+=sprintf(p, "\"special_effect\":%u,", s->status.special_effect);
        p+=sprintf(p, "\"wb_mode\":%u,", s->status.wb_mode);
        p+=sprintf(p, "\"awb\":%u,", s->status.awb);
        p+=sprintf(p, "\"awb_gain\":%u,", s->status.awb_gain);
        p+=sprintf(p, "\"aec\":%u,", s->status.aec);
        p+=sprintf(p, "\"aec2\":%u,", s->status.aec2);
        p+=sprintf(p, "\"ae_level\":%d,", s->status.ae_level);
        p+=sprintf(p, "\"aec_value\":%u,", s->status.aec_value);
        p+=sprintf(p, "\"agc\":%u,", s->status.agc);
        p+=sprintf(p, "\"agc_gain\":%u,", s->status.agc_gain);
        p+=sprintf(p, "\"gainceiling\":%u,", s->status.gainceiling);
        p+=sprintf(p, "\"bpc\":%u,", s->status.bpc);
        p+=sprintf(p, "\"wpc\":%u,", s->status.wpc);
        p+=sprintf(p, "\"raw_gma\":%u,", s->status.raw_gma);
        p+=sprintf(p, "\"lenc\":%u,", s->status.lenc);
        p+=sprintf(p, "\"hmirror\":%u,", s->status.hmirror);
        p+=sprintf(p, "\"dcw\":%u,", s->status.dcw);
        p+=sprintf(p, "\"colorbar\":%u", s->status.colorbar);
        *p++ = '}';
        *p++ = 0;
        httpd_resp_set_type(req, "application/json");
        httpd_resp_set_hdr(req, "Access-Control-Allow-Origin", "*");
        return httpd_resp_send(req, json_response, strlen(json_response));
}

static esp_err_t index_handler(httpd_req_t *req){
        httpd_resp_set_type(req, "text/html");
        String page = "";
          page += "<meta name=\"viewport\" content=\"width=device-width,
initial-scale=1.0, maximum-scale=1.0, user-scalable=0\">\n";
  page += "<script>var xhttp = new XMLHttpRequest();</script>";
  page += "<script>function getsend(arg) { xhttp.open('GET', arg +'?' + new
Date().getTime(), true); xhttp.send() } </script>";
  //page += "<p align=center><IMG SRC='http://" + WiFiAddr + ":81/stream'
style='width:280px;'></p><br/><br/>";
  page += "<p align=center><IMG SRC='http://" + WiFiAddr + ":81/stream'
style='width:300px; transform:rotate(180deg);'></p><br/><br/>";

  page += "<p align=center> <button
style=background-color:lightgrey;width:90px;height:80px
onmousedown=getsend('go') onmouseup=getsend('stop')
ontouchstart=getsend('go')
ontouchend=getsend('stop') ><b>Forward</b></button> </p>";
  page += "<p align=center>";
```

```
page += "<button style=background-color:lightgrey;width:90px;height:80px;
onmousedown=getsend('left') onmouseup=getsend('stop')
ontouchstart=getsend('left')
ontouchend=getsend('stop')><b>Left</b></button> ";
  page += "<button style=background-color:indianred;width:90px;height:80px
onmousedown=getsend('stop')
onmouseup=getsend('stop')><b>Stop</b></button> ";
  page += "<button style=background-color:lightgrey;width:90px;height:80px
onmousedown=getsend('right') onmouseup=getsend('stop')
ontouchstart=getsend('right')
ontouchend=getsend('stop')><b>Right</b></button>";
  page += "</p>";

  page += "<p align=center><button
style=background-color:lightgrey;width:90px;height:80px
onmousedown=getsend('back') onmouseup=getsend('stop')
ontouchstart=getsend('back')
ontouchend=getsend('stop') ><b>Backward</b></button></p>";

  page += "<p align=center>";
  page += "<button style=background-color:yellow;width:140px;height:40px
onmousedown=getsend('ledon')><b>Light ON</b></button>";
  page += "<button style=background-color:yellow;width:140px;height:40px
onmousedown=getsend('ledoff')><b>Light OFF</b></button>";
  page += "</p>";

      return httpd_resp_send(req, &page[0], strlen(&page[0]));
}

static esp_err_t go_handler(httpd_req_t *req){
      WheelAct(HIGH, LOW, HIGH, LOW);
      Serial.println("Go");
      httpd_resp_set_type(req, "text/html");
      return httpd_resp_send(req, "OK", 2);
}
static esp_err_t back_handler(httpd_req_t *req){
      WheelAct(LOW, HIGH, LOW, HIGH);
      Serial.println("Back");
      httpd_resp_set_type(req, "text/html");
      return httpd_resp_send(req, "OK", 2);
}

static esp_err_t left_handler(httpd_req_t *req){
      WheelAct(LOW, HIGH, HIGH, LOW);
      Serial.println("Left");
      httpd_resp_set_type(req, "text/html");
```

```
WheelAct(HIGH, LOW, LOW, HIGH);
    Serial.println("Right");
    httpd_resp_set_type(req, "text/html");
    return httpd_resp_send(req, "OK", 2);
}

static esp_err_t stop_handler(httpd_req_t *req){
    WheelAct(LOW, LOW, LOW, LOW);
    Serial.println("Stop");
    httpd_resp_set_type(req, "text/html");
    return httpd_resp_send(req, "OK", 2);
}

static esp_err_t ledon_handler(httpd_req_t *req){
    digitalWrite(gpLed, HIGH);
    Serial.println("LED ON");
    httpd_resp_set_type(req, "text/html");
    return httpd_resp_send(req, "OK", 2);
}
static esp_err_t ledoff_handler(httpd_req_t *req){
    digitalWrite(gpLed, LOW);
    Serial.println("LED OFF");
    httpd_resp_set_type(req, "text/html");
    return httpd_resp_send(req, "OK", 2);
}

void startCameraServer(){
    httpd_config_t config = HTTPD_DEFAULT_CONFIG();

    httpd_uri_t go_uri = {
        .uri       = "/go",
        .method    = HTTP_GET,
        .handler   = go_handler,
        .user_ctx  = NULL
    };

    httpd_uri_t back_uri = {
        .uri       = "/back",
        .method    = HTTP_GET,
        .handler   = back_handler,
        .user_ctx  = NULL
    };

    httpd_uri_t stop_uri = {
        .uri       = "/stop",
        .method    = HTTP_GET,
```

```
.handler     = stop_handler,
            .user_ctx    = NULL
        };

        httpd_uri_t left_uri = {
            .uri            = "/left",
            .method         = HTTP_GET,
            .handler        = left_handler,
            .user_ctx       = NULL
        };

        httpd_uri_t right_uri = {
            .uri            = "/right",
            .method         = HTTP_GET,
            .handler        = right_handler,
            .user_ctx       = NULL
        };

        httpd_uri_t ledon_uri = {
            .uri            = "/ledon",
            .method         = HTTP_GET,
            .handler        = ledon_handler,
            .user_ctx       = NULL
        };

        httpd_uri_t ledoff_uri = {
            .uri            = "/ledoff",
            .method         = HTTP_GET,
            .handler        = ledoff_handler,
            .user_ctx       = NULL
        };

        httpd_uri_t index_uri = {
            .uri            = "/",
            .method         = HTTP_GET,
            .handler        = index_handler,
            .user_ctx       = NULL
        };
        httpd_uri_t status_uri = {
            .uri            = "/status",
            .method         = HTTP_GET,
            .handler        = status_handler,
            .user_ctx       = NULL
        };

        httpd_uri_t cmd_uri = {
```

```
.uri           = "/control",
            .method     = HTTP_GET,
            .handler    = cmd_handler,
            .user_ctx   = NULL
        };

        httpd_uri_t capture_uri = {
            .uri           = "/capture",
            .method     = HTTP_GET,
            .handler    = capture_handler,
            .user_ctx   = NULL
        };

      httpd_uri_t stream_uri = {
            .uri           = "/stream",
            .method     = HTTP_GET,
            .handler    = stream_handler,
            .user_ctx   = NULL
        };

        ra_filter_init(&ra_filter, 20);
        Serial.printf("Starting web server on port: '%d'", config.server_port);
        if (httpd_start(&camera_httpd, &config) == ESP_OK) {
            httpd_register_uri_handler(camera_httpd, &index_uri);
            httpd_register_uri_handler(camera_httpd, &go_uri);
            httpd_register_uri_handler(camera_httpd, &back_uri);
            httpd_register_uri_handler(camera_httpd, &stop_uri);
            httpd_register_uri_handler(camera_httpd, &left_uri);
            httpd_register_uri_handler(camera_httpd, &right_uri);
            httpd_register_uri_handler(camera_httpd, &ledon_uri);
            httpd_register_uri_handler(camera_httpd, &ledoff_uri);
        }
        config.server_port += 1;
        config.ctrl_port += 1;
        Serial.printf("Starting stream server on port: '%d'", config.server_port);
        if (httpd_start(&stream_httpd, &config) == ESP_OK) {
            httpd_register_uri_handler(stream_httpd, &stream_uri);
        }
}
void WheelAct(int nLf, int nLb, int nRf, int nRb)
{
  digitalWrite(gpLf, nLf);
  digitalWrite(gpLb, nLb);
  digitalWrite(gpRf, nRf);
  digitalWrite(gpRb, nRb);
}
```

6.2 APPENDIX 2- TOMATO DETECTION CODES

```
import cv2
from cv2 import waitKey

def imageDisplay(tuningPara, rawImageQueue, outPutImageQueue):
	while True:
		# print("Hello world from get1")
		img = rawImageQueue.get()
		img2 = outPutImageQueue.get()
		cv2.imshow('rawImageDisplay',img)
		cv2.imshow('outPutImageDisplay',img2)
		if cv2.waitKey(1) & 0xFF == ord('q'):
			Break

import multipartHTTPget
import imageDisplay

# importing the multiprocessing module
from multiprocessing import Process, Queue

if __name__ == '__main__':
	# Queue
	rawImageQueue = Queue()
	outPutImageQueue    = Queue()

	# print("Hello world from main")
	# creating processes
	p1 = Process(target=multipartHTTPget.imageProcessor, args=(10,
rawImageQueue, outPutImageQueue))
	p2 = Process(target=imageDisplay.imageDisplay, args=(10,
rawImageQueue, outPutImageQueue))

	# print("Hello world from main")
	# starting process 1
	p1.start()
	# starting process 2
	p2.start()

	# wait until process 1 is finished
	p1.join()
	# wait until process 2 is finished
	p2.join()

	# both processes finished
	print("Done!")
```

```
 img_original = img
                        # img_original = cv2.resize(img, (700, 700))
                        imghsv = cv2.cvtColor(img,cv2.COLOR_BGR2HSV)

                        # Getting Trackbar control
                        # h_min = cv2.getTrackbarPos("Hue Min","Trackbar")
                        # h_max = cv2.getTrackbarPos("Hue Max","Trackbar")
                        # s_min = cv2.getTrackbarPos("Sat Min","Trackbar")
                        # s_max = cv2.getTrackbarPos("Sat Max","Trackbar")
                        # v_min = cv2.getTrackbarPos("Val Min","Trackbar")
                        # v_max = cv2.getTrackbarPos("Val Max","Trackbar")
                        h_min = 11
                        h_max = 179
                        s_min = 0
                        s_max = 255
                        v_min = 0
                        v_max = 255
                        lower = np.array([h_min, s_min, v_min])
                        upper = np.array([h_max, s_max, v_max])
                        mask = cv2.inRange(imghsv, lower, upper)
                        imgResult = cv2.bitwise_and(img, img, mask=mask)
                        mask = cv2.bitwise_not(mask, mask)

                        # Image contour detectin and bounding box creation
                        conts = cv2.findContours(mask, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
                        # print(conts)
                        conts = imutils.grab_contours(conts)
                        cont_img = np.zeros(img.shape)
                        cont_img = cv2.drawContours(img_original, None , -1,
(0,0,0), 1)

                        #midpoint definition
                        def midPoint(ptA, ptB):
                              return ((ptA[0] + ptB[0])/2, (ptA[1] + ptB[1])/2)

                        # #loop over all the contour coordinates
                        for c in conts:
                              # extract box points
                              box = cv2.minAreaRect(c) #(left, top), (right, bottom),
accuracy
                              # print(box)
                              box = cv2.boxPoints(box)
                              #convert box points to integer
                              box = np.array(box, dtype='int')
```

```
# cv2.drawContours(cont_img, [c], -1, (0,255,0), 1)
                        cv2.drawContours(cont_img, [box], -1, (255,255,255),
1)

                        #print(box)
                        for (x,y) in box:
                            cv2.circle(cont_img, (x, y), 2, (255, 0, 0), 2)
                            (tl, tr, br, bl) = box

                            #calculate midpoints for top-bottom of rectangle
                            (tlX, trX) = midPoint(tl, tr)
                            (brX, blX) = midPoint(br, bl)

                            #draw midpoint dots for top and bottom
                            cv2.circle(cont_img, (int(tlX), int(trX)), 1, (255,
0, 0), 2)
                            cv2.circle(cont_img, (int(brX), int(blX)), 1, (255,
0, 0), 2)

                            #connect the midpoints using line
                            cv2.line(cont_img, (int(tlX), int(trX)), (int(brX),
int(blX)), (255, 255, 255), 1)

                            #calculate the distance based on midpoints
                            dA = dist.euclidean((tlX, trX), (brX, blX))


                            #print the size in pixel in each contour rectangle
                            cv2.putText(cont_img, "{:.1f} px".format(dA),
(int(tlX-10), int(trX-10)),

cv2.FONT_HERSHEY_SIMPLEX, 0.4, (255,255,255), 1)
                            cv2.putText(cont_img, "Tomato", (int(tlX+10),
int(trX+10)),

cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0,0,0), 1)


                            #calculate midpoints for left-right of rectangle
                            (tlX, trX) = midPoint(tl, bl)
                            (brX, blX) = midPoint(tr, br)


                            #draw midpoint dots for left and right
                            cv2.circle(cont_img, (int(tlX), int(trX)), 1, (255,
0, 0), 2)
```

```
cv2.line(cont_img, (int(tlX), int(trX)), (int(brX), int(blX)), (255, 255, 255), 1)

                    #calculate the distance based on midpoints
                    dB = dist.euclidean((tlX, trX), (brX, blX))

                    #print the size in pixel in each contour rectangle
                    cv2.putText(cont_img, "{:.1f} px".format(dB), (int(brX+10),
int(blX+10)),
                                cv2.FONT_HERSHEY_SIMPLEX, 0.25,
(255,255,255), 2)


            # Showing image


        cv2.imshow("Original",cont_img)
        cv2.resizeWindow("Original",250,250)
        outPutImageQueue.put(cont_img)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
        # cv2.imshow("HSV",imghsv)
        # cv2.resizeWindow("HSV",250,250)
        # cv2.imshow("mask",mask)
        # cv2.resizeWindow("mask",250,250)
        # cv2.imshow("imgResult",imgResult)
        # cv2.resizeWindow("imgResult",250,250)

        # cv2.waitKey(1)
```

APPENDIX 3- Codes for Robotic Arm Control

```
#include <WiFi.h>
#include <Servo.h>

Servo myservo_gripper;    // create servo object to control a servo
Servo myservo_base;
Servo myservo_arm_left;
Servo myservo_arm_right;

// GPIO the servo is attached to
static const int servoP_gripper = 13;
static const int servoP_base = 12;
static const int servoP_arm_left = 14;
static const int servoP_arm_right = 27;

// Replace with your network credentials
const char* ssid     = "";
const char* password = "";

// Set web server port number to 80
WiFiServer server(80);

// Variable to store the HTTP request
String header;

// Decode HTTP GET value
String valueString_gripper = String(4);
String valueString_base = String(5);
String valueString_arm_left = String(6);
String valueString_arm_right = String(6);

int pos1 = 0;
int pos2 = 0;
int pos3 = 0;
int pos4 = 0;
int pos5 = 0;
int pos6 = 0;
int pos7 = 0;
int pos8 = 0;

// Current time
unsigned long currentTime = millis();
// Previous time
unsigned long previousTime = 0;
// Define timeout time in milliseconds (example: 2000ms = 2s)
const long timeoutTime = 2000;
```

```
void setup() {
  Serial.begin(115200);

  myservo_gripper.attach(servoP_gripper);  // attaches the servo on the
servoP_gripperin_gripper to the servo object
  myservo_base.attach(servoP_base);
  myservo_arm_left.attach(servoP_arm_left);
  myservo_arm_right.attach(servoP_arm_right);

  // Connect to Wi-Fi network with SSID and password
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  // Print local IP address and start web server
  Serial.println("");
  Serial.println("WiFi connected.");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  server.begin();
}

void loop(){
  WiFiClient client = server.available();   // Listen for incoming clients

  if (client) {                             // If a new client connects,
    currentTime = millis();
    previousTime = currentTime;
    Serial.println("New Client.");          // print a message out in the
serial port
    String currentLine = "";                // make a String to hold
incoming data from the client
    while (client.connected() && currentTime - previousTime <=
timeoutTime) { // loop while the client's connected
      currentTime = millis();
      if (client.available()) {             // if there's bytes to read from the
client,
        char c = client.read();             // read a byte, then
        Serial.write(c);                    // print it out the serial
monitor
        header += c;
        if (c == '\n') {                    // if the byte is a newline
character
```

```
// and a content-type so the client knows what's coming, then a blank line:
                    client.println("HTTP/1.1 200 OK");
                    client.println("Content-type:text/html");
                    client.println("Connection: close");
                    client.println();

                    // Display the HTML web page
                    client.println("<!DOCTYPE html><html>");
                    client.println("<head><meta name=\"viewport\"
content=\"width=device-width, initial-scale=1\">");
                    client.println("<link rel=\"icon\" href=\"data:,\">");
                    // CSS to style the on/off buttons

                    client.println("<style>body { text-align: center; font-family:
\"Trebuchet MS\", Arial; margin-left:auto; margin-right:auto;}");
                    client.println(".slider_gripper { width: 300px; }");
                    client.println(".slider_base { width: 300px; }");
                    client.println(".slider_arm_left { width: 300px; }");
                    client.println(".slider_arm_right { width: 300px; }</style>");
                    client.println("<script
src=\"https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js\"></scrip
t>");

                    // Web Page for gripper
                    client.println("</head><body><h1>Gripper Position</h1>");
                    client.println("<p>Position: <span
id=\"servoP_gripper\"></span></p>");
                    client.println("<input type=\"range\" min=\"0\" max=\"180\"
class=\"slider_gripper\" id=\"servoslider_gripper\"
onchange=\"servo_gripper(this.value)\"
value=\""+valueString_gripper+"\"/>");

                    client.println("<script>var slider_gripper =
document.getElementById(\"servoslider_gripper\");");
                    client.println("var servoP_gripper =
document.getElementById(\"servoP_gripper\"); servoP_gripper.innerHTML =
slider_gripper.value;");
                    client.println("slider_gripper.oninput = function()
{ slider_gripper.value = this.value; servoP_gripper.innerHTML =
this.value; }");
                    client.println("$.ajaxSetup({timeout:1000}); function
servo_gripper(pos_gripper) { ");
                    client.println("$.get(\"/gripper?value=\" + pos_gripper + \"&\");
{Connection: close};}</script>");

                  // Web Page for base
```

```
client.println("<input type=\"range\" min=\"0\" max=\"180\"
class=\"slider_base\" id=\"servoslider_base\"
onchange=\"servo_base(this.value)\" value=\""+valueString_base+"\"/>");

                client.println("<script>var slider_base =
document.getElementById(\"servoslider_base\");");
                client.println("var servoP_base =
document.getElementById(\"servoP_base\"); servoP_base.innerHTML =
slider_base.value;");
                client.println("slider_base.oninput = function()
{ slider_base.value = this.value; servoP_base.innerHTML = this.value; }");
                client.println("$.ajaxSetup({timeout:1000}); function
servo_base(pos_base) { ");
                client.println("$.get(\"/base?value=\" + pos_base + \"&\");
{Connection: close};}</script>");


              // Web Page for arm_left
                client.println("</head><<h1>Arm Position left</h1>");
                client.println("<p>Position: <span
id=\"servoP_arm_left\"></span></p>");
                client.println("<input type=\"range\" min=\"0\" max=\"180\"
class=\"slider_arm_left\" id=\"servoslider_arm_left\"
onchange=\"servo_arm_left(this.value)\"
value=\""+valueString_arm_left+"\"/>");

                client.println("<script>var slider_arm_left =
document.getElementById(\"servoslider_arm_left\");");
                client.println("var servoP_arm_left =
document.getElementById(\"servoP_arm_left\"); servoP_arm_left.innerHTML
= slider_arm_left.value;");
                client.println("slider_arm_left.oninput = function()
{ slider_arm_left.value = this.value; servoP_arm_left.innerHTML =
this.value; }");
                client.println("$.ajaxSetup({timeout:1000}); function
servo_arm_left(pos_arm_left) { ");
                client.println("$.get(\"/arm_left?value=\" + pos_arm_left +
\"&\"); {Connection: close};}</script>");


                  // Web Page for arm_right
                client.println("</head><<h1>Arm Position right</h1>");
                client.println("<p>Position: <span
id=\"servoP_arm_right\"></span></p>");
                client.println("<input type=\"range\" min=\"0\" max=\"180\"
class=\"slider  arm  right\" id=\"servoslider  arm  right\"
```

```
                client.println("<script>var slider_arm_right =
document.getElementById(\"servoslider_arm_right\");");
                client.println("var servoP_arm_right =
document.getElementById(\"servoP_arm_right\");
servoP_arm_right.innerHTML = slider_arm_right.value;");
                client.println("slider_arm_right.oninput = function()
{ slider_arm_right.value = this.value; servoP_arm_right.innerHTML =
this.value; }");
                client.println("$.ajaxSetup({timeout:1000}); function
servo_arm_right(pos_arm_right) { ");
                client.println("$.get(\"/arm_right?value=\" + pos_arm_right +
\"&\"); {Connection: close};}</script>");

                client.println("</body></html>");



                //GET /?value=180& HTTP/1.1
                if(header.indexOf("GET /gripper?value=")>=0) {
                   pos1 = header.indexOf('=');
                   pos2 = header.indexOf('&');
                   valueString_gripper = header.substring(pos1+1, pos2);

                   //Rotate the servo
                   myservo_gripper.write(valueString_gripper.toInt());
                   Serial.println(valueString_gripper);
                }

                if(header.indexOf("GET /base?value=")>=0) {
                   pos3 = header.indexOf('=');
                   pos4 = header.indexOf('&');
                   valueString_base = header.substring(pos3+1, pos4);

                   //Rotate the servo
                   myservo_base.write(valueString_base.toInt());
                   Serial.println(valueString_base);
                }

                  if(header.indexOf("GET /arm_left?value=")>=0) {
                   pos5 = header.indexOf('=');
                   pos6 = header.indexOf('&');
                   valueString_arm_left = header.substring(pos5+1, pos6);

                   //Rotate the servo
                   myservo_arm_left.write(valueString_arm_left.toInt());
                   Serial.println(valueString  arm  left);
```

```
  if(header.indexOf("GET /arm_right?value=")>=0) {
                        pos7 = header.indexOf('=');
                        pos8 = header.indexOf('&');
                        valueString_arm_right = header.substring(pos7+1, pos8);

                        //Rotate the servo
                        myservo_arm_right.write(valueString_arm_right.toInt());
                        Serial.println(valueString_arm_right);
                    }

                    // The HTTP response ends with another blank line
                    client.println();
                    // Break out of the while loop
                    break;
                  }

                  else { // if you got a newline, then clear currentLine
                    currentLine = "";
                  }
                } else if (c != '\r') {    // if you got anything else but a carriage return character,
                  currentLine += c;          // add it to the end of the currentLine
                }
              }
            }
            // Clear the header variable
            header = "";
            // Close the connection
            client.stop();
            Serial.println("Client disconnected.");
            Serial.println("");
          }
        }
```

APPENDIX 4- Codes for SQL Database

```
from flask import Flask
from flask import request
import sys
import csv
import sqlite3

# create sqlite3 database
db = sqlite3.connect('./sensorRead.db')

# create table with columns for distance and time and Date
db.execute('''CREATE TABLE IF NOT EXISTS SensorData
                    (id INTEGER PRIMARY KEY AUTOINCREMENT,
                    distance INTEGER,
                    gripperAngle INTEGER,
                    time TEXT)''')


# create flask app
app=Flask(__name__)
#create get end point
@app.route('/update_distance')
def update_distance():
        #get ultrasonic distance argument
        ultrasonicDistance=request.args.get('ultrasonicDistance')
        gripperAngle=request.args.get('gripperAngle')

        dataBase = sqlite3.connect('./sensorRead.db')
        #write to db
        dataBase.execute('INSERT INTO SensorData (distance, gripperAngle,
time) VALUES (?, ?, datetime("now"))', (ultrasonicDistance, gripperAngle))
        dataBase.commit()
        dataBase.close()

        return "200"

  if __name__ == '__main__':

        app.run(host='0.0.0.0', debug = True, port=1880)
```

```
#include <WiFi.h>
#include <Servo.h>
#include <HTTPClient.h>

Servo myservo_gripper;    // create servo object to control a servo
Servo myservo_base;
Servo myservo_arm_left;
Servo myservo_arm_right;

// GPIO the servo is attached to
static const int servoP_gripper = 13;
static const int servoP_base = 12;
static const int servoP_arm_left = 14;
static const int servoP_arm_right = 27;

// Replace with your network credentials
const char* ssid          = "OnePlus 5T";
const char* password = "c813da74egys";

String serverName = "http://192.168.43.85:1880/update_distance";

// Set web server port number to 80
WiFiServer server(80);

// Variable to store the HTTP request
String header;

// Decode HTTP GET value
String valueString_gripper = String(4);
String valueString_base = String(5);
String valueString_arm_left = String(6);
String valueString_arm_right = String(6);

int pos1 = 0;
int pos2 = 0;
int pos3 = 0;
int pos4 = 0;
int pos5 = 0;
int pos6 = 0;
int pos7 = 0;
int pos8 = 0;

// Current time
unsigned long currentTime = millis();
// Previous time
unsigned long previousTime = 0;
```

```
// Define timeout time in milliseconds (example: 2000ms = 2s)
const long timeoutTime = 2000;
const int trigPin = 5;
const int echoPin = 18;
//define sound speed in cm/uS
#define SOUND_SPEED 0.034
#define CM_TO_INCH 0.393701
long duration;
float distanceCm;
float distanceInch;

void setup() {
  Serial.begin(115200);
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
  myservo_gripper.attach(servoP_gripper);    // attaches the servo on the
servoP_gripperin_gripper to the servo object
  myservo_base.attach(servoP_base);
  myservo_arm_left.attach(servoP_arm_left);
  myservo_arm_right.attach(servoP_arm_right);
  // Connect to Wi-Fi network with SSID and password
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  // Print local IP address and start web server
  Serial.println("");
  Serial.println("WiFi connected.");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  server.begin();
}
void sendSensorData(){
  String gripperAngle;
  // Ultrasonic code
  HTTPClient http;
  // Clears the trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
```

```
// Reads the echoPin, returns the sound wave travel time in microseconds
   duration = pulseIn(echoPin, HIGH);
      // Calculate the distance
   distanceCm = duration * SOUND_SPEED/2;
      // Prints the distance in the Serial Monitor
   Serial.print("Distance (cm): ");
   Serial.println(distanceCm);
   gripperAngle = valueString_gripper;
   String serverPath = serverName + "?ultrasonicDistance=" + distanceCm ";
   }
void loop(){
   sendSensorData();
   WiFiClient client = server.available();      // Listen for incoming clients

   if (client) {                                          // If a new client connects,
            currentTime = millis();
      previousTime = currentTime;
      Serial.println("New Client.");                 // print a message out in the
serial port
      String currentLine = "";                          // make a String to hold
incoming data from the client
      while (client.connected() && currentTime - previousTime <=
timeoutTime) { // loop while the client's connected
         currentTime = millis();
         if (client.available()) {                    // if there's bytes to read from the
client,
            char c = client.read();                    // read a byte, then
            Serial.write(c);                             // print it out the serial
monitor
            header += c;
            if (c == '\n') {                             // if the byte is a newline
character
           // if the current line is blank, you got two newline characters in a row.
               // that's the end of the client HTTP request, so send a response:
               if (currentLine.length() == 0) {
                  // HTTP headers always start with a response code (e.g.
HTTP/1.1 200 OK)
                  // and a content-type so the client knows what's coming, then a
blank line:
                  client.println("HTTP/1.1 200 OK");
                  client.println("Content-type:text/html");
                  client.println("Connection: close");
                  client.println();
                    // Display the HTML web page
                  client.println("<!DOCTYPE html><html>");
```

```
client.println("<head><meta name=\"viewport\" content=\"width=device-width
initial-scale=1\">");
                client.println("<link rel=\"icon\" href=\"data:,\">");
                // CSS to style the on/off buttons
                // Feel free to change the background-color and font-size
attributes to fit your preferences
                client.println("<style>body { text-align: center; font-family:
\"Trebuchet MS\", Arial; margin-left:auto; margin-right:auto;}");
                client.println(".slider_gripper { width: 300px; }");
                client.println(".slider_base { width: 300px; }");
                client.println(".slider_arm_left { width: 300px; }");
                client.println(".slider_arm_right { width: 300px; }</style>");
                client.println("<script
src=\"https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js\"></scrip
t>");
          // Web Page for gripper
                client.println("</head><body><h1>Gripper Position</h1>");
                client.println("<p>Position: <span
id=\"servoP_gripper\"></span></p>");
                client.println("<input type=\"range\" min=\"0\" max=\"180\"
class=\"slider_gripper\" id=\"servoslider_gripper\"
onchange=\"servo_gripper(this.value)\"
value=\""+valueString_gripper+"\"/>");
                client.println("<script>var slider_gripper =
document.getElementById(\"servoslider_gripper\");");
                client.println("var servoP_gripper =
document.getElementById(\"servoP_gripper\"); servoP_gripper.innerHTML =
slider_gripper.value;");
                client.println("slider_gripper.oninput = function()
{ slider_gripper.value = this.value; servoP_gripper.innerHTML =
this.value; }");
                client.println("$.ajaxSetup({timeout:1000}); function
servo_gripper(pos_gripper) { ");
                client.println("$.get(\"/gripper?value=\" + pos_gripper + \"&\");
{Connection: close};}</script>");
               // Web Page for base
                client.println("</head><<h1>Base Position</h1>");
                client.println("<p>Position: <span
id=\"servoP_base\"></span></p>");
                client.println("<input type=\"range\" min=\"0\" max=\"180\"
class=\"slider_base\" id=\"servoslider_base\"
onchange=\"servo_base(this.value)\" value=\""+valueString_base+"\"/>");

                client.println("<script>var slider_base =
document.getElementById(\"servoslider  base\");");
```

```
client.println("var servoP_base = document.getElementById(\"servoP_base\");
servoP_base.innerHTML = slider_base.value;");
                client.println("slider_base.oninput = function()
{ slider_base.value = this.value; servoP_base.innerHTML = this.value; }");
                client.println("$.ajaxSetup({timeout:1000}); function
servo_base(pos_base) { ");
                client.println("$.get(\"/base?value=\" + pos_base + \"&\");
{Connection: close};}</script>");
                                // Web Page for arm_left
                client.println("</head><<h1>Arm Position left</h1>");
                client.println("<p>Position: <span
id=\"servoP_arm_left\"></span></p>");
                client.println("<input type=\"range\" min=\"0\" max=\"180\"
class=\"slider_arm_left\" id=\"servoslider_arm_left\"
onchange=\"servo_arm_left(this.value)\"
value=\""+valueString_arm_left+"\"/>");

                client.println("<script>var slider_arm_left =
document.getElementById(\"servoslider_arm_left\");");
                client.println("var servoP_arm_left =
document.getElementById(\"servoP_arm_left\"); servoP_arm_left.innerHTML
= slider_arm_left.value;");
                client.println("slider_arm_left.oninput = function()
{ slider_arm_left.value = this.value; servoP_arm_left.innerHTML =
this.value; }");
                client.println("$.ajaxSetup({timeout:1000}); function
servo_arm_left(pos_arm_left) { ");
                client.println("$.get(\"/arm_left?value=\" + pos_arm_left +
\"&\"); {Connection: close};}</script>");
                  // Web Page for arm_right
                client.println("</head><<h1>Arm Position right</h1>");
                client.println("<p>Position: <span
id=\"servoP_arm_right\"></span></p>");
                client.println("<input type=\"range\" min=\"0\" max=\"180\"
class=\"slider_arm_right\" id=\"servoslider_arm_right\"
onchange=\"servo_arm_right(this.value)\"
value=\""+valueString_arm_right+"\"/>");

                client.println("<script>var slider_arm_right =
document.getElementById(\"servoslider_arm_right\");");
                client.println("var servoP_arm_right =
document.getElementById(\"servoP_arm_right\");
servoP  arm  right.innerHTML = slider  arm  right.value;");
```

```
  client.println("slider_arm_right.oninput = function() { slider_arm_right.value
= this.value; servoP_arm_right.innerHTML = this.value; }");
                    client.println("$.ajaxSetup({timeout:1000}); function
servo_arm_right(pos_arm_right) { ");
                    client.println("$.get(\"/arm_right?value=\" + pos_arm_right +
\"&\"); {Connection: close};}</script>");
                    client.println("</body></html>");
                    sendSensorData();
                      //GET /?value=180& HTTP/1.1
                    if(header.indexOf("GET /gripper?value=")>=0) {
                       pos1 = header.indexOf('=');
                       pos2 = header.indexOf('&');
                       valueString_gripper = header.substring(pos1+1, pos2);

                       //Rotate the servo
                       myservo_gripper.write(valueString_gripper.toInt());
                       Serial.println(valueString_gripper);
                    }
                       if(header.indexOf("GET /base?value=")>=0) {
                       pos3 = header.indexOf('=');
                       pos4 = header.indexOf('&');
                       valueString_base = header.substring(pos3+1, pos4);

                       //Rotate the servo
                       myservo_base.write(valueString_base.toInt());
                       Serial.println(valueString_base);
                    }
                      if(header.indexOf("GET /arm_left?value=")>=0) {
                       pos5 = header.indexOf('=');
                       pos6 = header.indexOf('&');
                       valueString_arm_left = header.substring(pos5+1, pos6);
                       //Rotate the servo
                       myservo_arm_left.write(valueString_arm_left.toInt());
                       Serial.println(valueString_arm_left);
                    }

                       if(header.indexOf("GET /arm_right?value=")>=0) {
                       pos7 = header.indexOf('=');
                       pos8 = header.indexOf('&');
                       valueString_arm_right = header.substring(pos7+1, pos8);

                       //Rotate the servo
                       myservo_arm_right.write(valueString_arm_right.toInt());
                       Serial.println(valueString_arm_right);
                    }
```

```
   // The HTTP response ends with another blank line
                    client.println();
                    // Break out of the while loop
                    break;
                  }
                   else { // if you got a newline, then clear currentLine
                    currentLine = "";
                  }
                } else if (c != '\r') {    // if you got anything else but a carriage return
character,
                  currentLine += c;           // add it to the end of the currentLine
                }
              }
            }
            // Clear the header variable
            header = "";
            // Close the connection
            client.stop();
            Serial.println("Client disconnected.");
            Serial.println("");

   }

   }
```

REFERENCES

1. Anmol Singh, PK Jain, “A study on Rocker-Bogie Suspension for a Planetary Rover Prototype”. (2020), Volume 8 Issue 3 230-238 Applications”, NDT.net -, Vol. 7 No.09
2. Bakkum, Peter, and Kevin Skadron. "Accelerating SQL database operations on a GPU with CUDA." In Proceedings of the 3rd workshop on general-purpose computation on graphics processing units, pp. 94-103. 2010.
3. Baohua Zhang, Yuanxin Xie, Jun Zhou, Kai Wang, Zhen Zhang, “State-of-the-art robotic grippers, grasping and control strategies, as well as their applications in agricultural robots: A review, Computers and Electronics in Agriculture”, Volume 177, 2020, ISSN 0168-1699
4. Boaz Arad1, Jos Balendonck2, Ruud Barth2, Ohad Ben-Shahar1, Yael Edan3, Thomas Hellström4, Jochen Hemming2, “Development of a sweet pepper harvesting robot”. (2020), Journal of Field Robotics, ISSN 1556-4959, E-ISSN 1556-4967, Vol. 37, no 6, pages. 1027-1039.
5. Dandil. E and Cevik, "Computer Vision Based Distance Measurement System using Stereo Camera View," *2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, 2019, pp. 1-4, doi: 10.1109/ISMSIT.2019.8932817.
6. Date, C. J. "A critique of the SQL database language." ACM SIGMOD Record 14, no. 3 (1984): 8-54.
7. G. Horng, M. Liu and C. Chen, "The Smart Image Recognition Mechanism for Crop Harvesting System in Intelligent Agriculture," in *IEEE Sensors Journal*, vol. 20, no. 5, pp. 2766-2781, 1 March1, 2020, doi: 10.1109/JSEN.2019.2954287.
8. Harrington, Brian D., and Chris Voorhees. "The challenges of designing the rocker-bogie suspension for the mars exploration rover." (2004).
9. N. Li and B. Zhang, "The Design and Implementation of Responsive Web Page Based on HTML5 and CSS3," 2019 International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI), 2019, pp. 373-376, doi: 10.1109/MLBDBI48998.2019.00084.
10. Oliveira, Luiz F.P., António P. Moreira, and Manuel F. Silva. 2021. "Advances in Agriculture Robotics: A State-of-the-Art Review and Challenges Ahead" *Robotics* 10, no. 2: 52. https://doi.org/10.3390/robotics10020052
11. Simon Birrell, Josie Hughes,Julia Y. Cai,Fumiya Iida, “A field-tested robotic harvesting system for iceberg lettuce”, (2019), Journal of Field Robotics Volume 37, Issue 2 p. 225-245

12. Tatsuki Kamata, Ali Roshanianfard, Noboru Noguchi, “Heavy weight Crop Harvesting Robot Controlling Algorithm”, 2018 International Federation of Automatic Control hosting by Elsevier Ltd., Volume 51, Issue 17, Pages 244-249.

13. Vikram Raja, Bindu Bhaskaran, Koushik Karan Geetha Nagaraj, Jai Gowtham Sampathkumar, Shri Ram Senthilkumar, “Agricultural harvesting using integrated robot system”, (2022) Indonesian Journal of Electrical Engineering and Computer Science , Volume 25,No.1,January 2022, pp. 152-158

14. Yuki Onishi, Takeshi Yoshida, Hiroki Kurita, Takanori Fukao, Hiromu Arihara, Ayako Iwai (2019) “An Automated Fruit Harvesting Robot using Deep Learning”, Robomech J 6,13 (2019)

15. Zhao, Wei, Tianxin Li, Bozhao Qi, Qifan Nie, and Troy Runge. 2021. "Terrain Analytics for Precision Agriculture with Automated Vehicle Sensors and Data Fusion" *Sustainability* 13, no. 5: 2905. https://doi.org/10.3390/su13052905.

RE-2022-28816-plag-report

by Research Experts - Turnitin Report

Submission date: 23-Jun-2022 10:21PM (UTC-0500)
Submission ID: 1862107852
File name: RE-2022-28816.pdf (5.78M)
Word count: 6782
Character count: 32968

RE-2022-28816-plag-report

ORIGINALITY REPORT

10%	7%	4%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Indian Institute of Technology, Madras Student Paper	2%
2	Submitted to University of Central Florida Student Paper	1%
3	robu.in Internet Source	1%
4	robomechjournal.springeropen.com Internet Source	1%
5	Wataru Yamanouchi. "Real-world force feedback control for mobile-hapto", 2008 13th International Power Electronics and Motion Control Conference, 09/2008 Publication	1%
6	ijeecs.iaescore.com Internet Source	<1%
7	Tatsuki Kamata, Ali Roshanianfard, Noboru Noguchi. "Heavy-weight Crop Harvesting Robot - Controlling Algorithm", IFAC-PapersOnLine, 2018	<1%