Sudan University of Science and Technology

College of Engineering

Electrical Engineering

SERVO ROBOTIC-ARM USING SMART CAR

ذراع آلية بإستخدام عربة ذكية

A Project Submitted in Partial Fulfillment for the Requirements of the Degree of B.Sc. (Honor) In Electrical Engineering (control)

Prepared by:

1. **Abdalla Osama Abdalla Ouda.**
2. **ABBAS HAG ADAM BABIKER HAG ADAM.**
3. **Mojahid Noureldeen Hussein Hassan.**
4. **MUSTAFA ALA ELDIN ABD ALSALAM ELTAHIR.**

Supervised by:

U. Hanaa gaafer

October 2018

الآية

قال تعالى:

(لَا يُكَلِّفُ اللَّهُ نَفْسًا إِلَّا وُسْعَهَا ۚ لَهَا مَا كَسَبَتْ وَعَلَيْهَا مَا اكْتَسَبَتْ ۗ رَبَّنَا لَا تُؤَاخِذْنَآ إِنْ نَّسِينَآ أَوْ أَخْطَأْنَا ۚ رَبَّنَا وَلَا تَحْمِلْ عَلَيْنَآ إِصْرًا كَمَا حَمَلْتَهُۥ عَلَى الَّذِينَ مِنْ قَبْلِنَا ۚ رَبَّنَا وَلَا تُحَمِّلْنَا مَا لَا طَاقَةَ لَنَا بِهِۦ ۖ وَاعْفُ عَنَّا وَاغْفِرْ لَنَا وَارْحَمْنَآ ۚ أَنْتَ مَوْلَٰنَا فَانْصُرْنَا عَلَى الْقَوْمِ الْكَٰفِرِينَ)

سورة البقرة الاية (286)

DEDICATION

To those who give us love and tenderness…to the symbols of love and healing balm…to the pure white hearts…to the grace of god in the earth our mothers to the lights that illuminate our life paths to those who taught us to endure no matter how the circumstances change our dear fathers to each one whose mind`s light illuminate the minds of others or given the correct answer to the confusion of his clients and he showed with his grace the humility of the scholars to friend who stands with us till end Ali Mohammed and Mustafa Mohammed to our university and teachers lastly.

to all of you

ACKNOWLEDGMENT

Our skies are always shining stars, this light does not fade from us for one moment, we look forward to it and we delight in its brightness in our skies every hour and it is fitting that it should be lifted up in our eyes with all love and fulfillment and the thinnest words of thanks and praise. It is the hearts filled with brotherhood. words of praise for your right thank you for your offers.

ABSTRACT

The linchpin examines the typical structure of designing and programming a robot, which consist of an arm, car, and a process links it with a phone to control its motion.

The arm has been designed according to three degrees of freedom, the care to ease the movement and locomotion, and all requirements to fulfill the small missions in a precise way. For example, the swift material of the arm is provided by “servo engine” which links the arms, additionally the performance of the arm movements, also the care-made of aluminum pointed as countrified mildness that reduces the countrified- has been provided by continues motor stream to give the wanted motion.

The controller that drives all motors has ability to refit the situation where interpreting process is on “Raspberry Pi” controller where “Python” is what has been programed by, and the robot is controlled by “Android” device.

In the world of today, the machinery arm fetches the republic use, besides robots. This kind of robots is designed by the available applications in different fields.

المستخلص

تتمحور الدراسة حول تصميم وبرمجة الروبوت ؛ الذي يحتوي علي ذراع الية وعربة وعملية تربط بين الربوت وجهاز الهاتف للتحكم في حركته .

تم تصميم الذراع الآلية وفقا لثلاث درجات حرية , عربة لتسهيل الحركة والانتقال , وكل المطلوبات لتفيذ المهام الصغيرة بصورة دقيقة , كمثال المادة الخفيفة التي تصنع منها الذراع الالية مزودة بمحرك "سيرفو" الذي يربط بين أجزاء الذراع , بالإضافة الى أداء حركة الذراع وتم تصميم العربة –المصنوعة من مادة الألمونيوم الذي يمتاز بخفة وزنه مما يخفف الوزن الكلي– عن طريق محرك التيار المستمر لاعطاء الحركة المطلوبة.

المتحكم الذي يقود جميع المحركات له القدرة على اصلاح الوضع حيث تتم عمليات الترجمة والتفسير على متحكم "الراسبيري–باي" , بالاضافة لاستخدام "بايثون" كلغة برمجية , ويتم التحكم في الربوت باستخدام جهاز اندرويد.

ففي عالم اليوم , تبين أن تقنية الذراع الالية جالبة للمنافع الإستخدام العام بجانب الروبوتات. تم تصميم هذا النوع من الربوتات عن طريق برامج متاحة في جميع المجالات .

TABLE OF CONTENTS

MAIN CIRCUIT DESIGN AND PROGRAMMING

LIST OF FIGURES

LIST OF ABBREVIATIONS

IOT	internet of things
UIDS	unique identifiers
RFID	radio frequency ID
MEMS	microelectromechanical systems
IT	information technology
OT	operational technology
M2M	machine-to-machine
DC	Direct Current
IC	integrated circuit
ATX	Advanced Technology extended
GND	ground
USB	Universal Serial Bus
DOF	Degree of freedom
PWM	Pulse Width Modulation
OOP	Object Oriented Programming

CHAPTER ONE
INTRODUCTION

1.1 General Concepts

Robotic arm can be used for various tasks such as welding, drilling, spraying and many more. A robotic arm car is fabricated by using components like controller, sensors and actuators. This increases the speed of operation, easy to control and reduces complexity. It also increases productivity. The main part of the design is Raspberry pi controller which coordinates and controls the product’s position. This specific controller is used in various types of embedded applications.

1.2 Problem Statement

Implementing of manipulating robotic arm with smart car using smartphone orientation sensor over internet to web server offer the opportunity to facilitate controlling of arm with less effort and equipment and also helpful in tasks that are considered too dangerous to be performed by humans.

1.3 Objectives

- To design a robotic arm with a smart cart.
- To implement the robot.
- To build a connection between robot and android smartphone.

1.4 Methodology

The project is practical application of easy control, over internet and smartphone angles, the design steps are:

1.4.1 Robotic arm case:

- Studying the mechanism of smartphone angles.
- Studying the mechanical movements of robotic-arm.
- Studying the mechanism of compass sensor.
- Implementing between android smartphone and compass sensor to give the right angles of the arm robot.

1.4.2 Building a robotic car with camera:

- Design a smart car to carry the weight of the robotic arm and the component of application.
- Studying the mechanism of controlling smart car over the internet using android.
- Connect the camera to give to easy the control.

1.4.3 Programming and assembling the subsystems:

- Programming each of the subsystems in Python.
- Assembling the subsystem into a Python package.
- Building a controller program for the manipulating robotic arm and smart car.

1.5 Project Layout

This project consists of five chapters: Chapter One gives an introduction to the project. Chapter Two discusses the theoretical background of robot’s internet of things, motors, raspberry pi controller and electrical sensors. Chapter Three describes the system fabrication, control circuit, software and hardware. Chapter Four handles the system implementation and the experimental results. Finally, Chapter Five provides the conclusions, recommendations.

CHAPTER TWO
LITERATURE REVIEW

2.1 Introduction

A Robot is a virtually intelligent agent capable of carrying out tasks robotically with the help of some supervision. Practically, a robot is basically an electro-mechanical machine that is guided by means of computer and electronic programming. Robots can be classified as autonomous, semiautonomous and remotely controlled. Robots are widely used for variety of tasks such as service stations, cleaning drains, and in tasks that are considered too dangerous to be performed by humans. A robotic arm is a robotic manipulator, usually programmable, with similar functions to a human arm. The robots interact with their environment, which is an important objective in the development of robots. This interaction is commonly established by means of some sort of arm and gripping device or end effectors. In the robotic arm, the arm has a few joints, similar to a human arm, in addition to shoulder, elbow, and wrist, coupled with the finger.

2.2 Advantage and disadvantage on robotic arm

2.2.1 The advantage

- Increase productivity.
- Use equipment effectively.
- Reduce working costs.
- Flexibility at work.

- Get the job done in the shortest time.
- Provide good returns on investment, Better accuracy in performance.
- Ability to work in risky ways and make it more safe. [5]

2.2.2 The disadvantage

- Cause unemployment for manual workers.
- High initial cost.
- designed Arm to perform specific tasks and not comparable to the human hand.
- Difficulty programmed to perform Accurate tasks.
- Needed a large number of sensors and high accuracy to perform the Complex tasks.
- And other technical problems, "especially in the fields of artificial intelligence and Machine vision".
- When the Robotic arm break down the production line will go off in the factories. [5]

2.2.3 Knowledgebase for robotics

Typical knowledgebase for the design and operation of robotics systems are [2]:

- Dynamic system modeling and analysis.
- Feedback control.
- Sensors and signal conditioning.
- Actuators (muscles) and power electronics.
- Hardware/computer interfacing.
- Computer programming.

2.3 Internet OF Things (IOT)

The internet of things, or(IOT), is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.

Figure 2.1: Internet of things

2.3.3 History OF IOT

Kevin Ashton, co-founder of the Auto-ID Center at MIT, first mentioned the internet of things in a presentation he made to Procter & Gamble (P&G) in 1999. Wanting to bring radio frequency ID (RFID) to the attention of P&G's senior management, Ashton called his presentation "Internet of Things" to incorporate the cool new trend of 1999: the internet. MIT professor Neil Gershenfeld's book, When Things Start to think, also appearing in 1999, didn't use the exact term but provided a clear vision of where IOT was headed. IOT has evolved from the convergence of wireless technologies, microelectromechanical systems (MEMS), micro services and the internet. The convergence has helped tear down the silos between operational technology (OT) and information technology (IT), enabling unstructured machine-generated data to be analyzed for insights to drive improvements. Although Ashton's was the first mention of the internet of things, the idea of connected devices has been around since the 1970s, under the monikers embedded internet and pervasive computing. The first internet appliance, for example, was a Coke machine at Carnegie Mellon University in the early 1980s. Using the web, programmers could check the status of the machine and determine whether there would be a cold drink awaiting them, should they decide to make the trip to the machine. IOT evolved from machine-to-machine (M2M) communication, i.e., machines connecting to each other via a network without human interaction. M2M refers to connecting a device to the cloud, managing it and collecting data. Taking M2M to the next level, IOT is a sensor network of billions of smart devices that connect people, systems and other applications to collect and share data. As its foundation, M2M offers the connectivity that enables IOT. The internet of things is also a natural extension of SCADA (supervisory control and data acquisition), a category of software application program for process control, the gathering of data in real time from remote locations to control equipment and conditions. SCADA systems include hardware and software components. The hardware gathers and feeds data into a computer that has SCADA software installed, where it is then processed and presented it in a timely manner. The evolution of SCADA is such that late-generation SCADA systems developed into first-generation IOT systems. The concept of the IOT ecosystem, however, didn't really come into its own until the middle of 2010 when, in part, the government of China said it would make IOT a strategic priority in its five-year plan.

2.3.2 Work of IOT

An IOT ecosystem consists of web-enabled smart devices that use embedded processors, sensors and communication hardware to collect, send and act on data they acquire from their environments. IOT devices share the sensor data they collect by connecting to an IOT gateway or other edge device where data is either sent to the cloud to be analyzed or analyzed locally. Sometimes, these devices communicate with other related devices and act on the information they get from one another. The devices do most of the work without human intervention, although people can interact with the devices -- for instance, to set them up, give them instructions or access the data. The connectivity, networking and communication protocols used with these web-enabled devices largely depend on the specific IOT applications deployed.

2.3.3 Benefit of IOT

The internet of things offers a number of benefits to organizations, enabling them to:

i. monitor their overall business processes.
ii. improve the customer experience.
iii. save time and money.
iv. enhance employee productivity.
v. integrate and adapt business models.
vi. make better business decisions.
vii. Generate more revenue.

2.4 Raspberry pi controller

The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote the teaching of basic computer science in schools and in developing countries. The original model became far more popular than anticipated, selling outside its target market for uses such as robotics. It does not include peripherals (such as keyboards and mice) and cases. However, some accessories have been included in several official and unofficial bundles. Raspberry pi controller represents a complement computer made of single electronic chip consist of traditional computer component which it's data processor or omnidirectional central processing unit with 700 MH speed, binuclear Graphical user interface with 250 MH speed fit to play HD movies and 3D games with Random Access Memory till 512 Mbyte. Additionally, digital output control have an ability to control on electrical and electronic pieces like Microcontrollers all that known as system on chip this tiny computer works with Linux open source systems. raspberry pi have 40 pin extended General Purpose Input Output, 4 x USB 2 ports, 4 pole Stereo output and Composite video port, Full size HDMI, CSI camera port for connecting the Raspberry Pi camera, display port for connecting the Raspberry Pi touch screen display and Micro SD port for loading your operating system and storing data.

i. Application of raspberry pi

Raspberry pi used as any traditional computer to check out the internet, sending e-mails and even edit liber office bundles also used to convert any TV you have to an entertainment home system connected to internet and also you can make fascinating electronic control project and use raspberry as a very developed alternative instead of microcontroller.

- Smart Home Automation.
- Making Robots, ROV and UAV.
- Remote Monitor.
- Smart streamers.
- Smart TV.
- Supercomputers.
- Balloon Satellites (weather balloon).

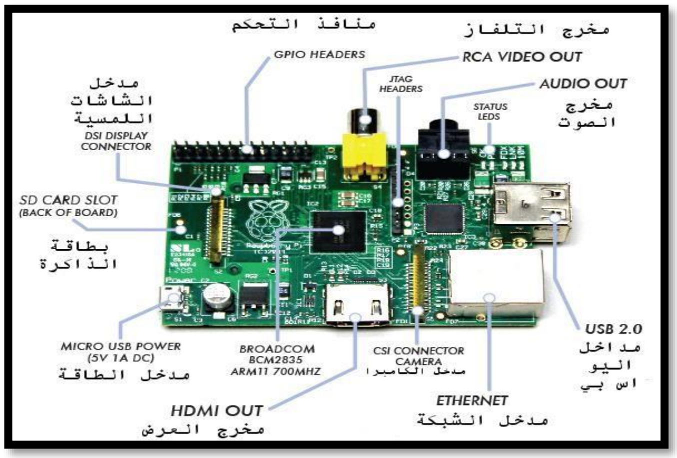


Figure 2.2: Raspberry pi controller

2.5 L293D Driver

L293D is a dual H-bridge motor driver integrated circuit (IC). Motor drivers act as current amplifiers since they take a low-current control signal and provide a higher-current signal. This higher current signal is used to drive the motors.

L293D contains two inbuilt H-bridge driver circuits. In its common mode of operation, two DC motors can be driven simultaneously, both in forward and reverse direction. The motor operations of two motors can be controlled by input logic at pins 2 & 7 and 10 & 15. Input logic 00 or 11 will stop the corresponding motor. Logic 01 and 10 will rotate it in clockwise and anticlockwise directions, respectively.

Enable pins 1 and 9 (corresponding to the two motors) must be high for motors to start operating. When an enable input is high, the associated driver gets enabled. As a result, the outputs become active and work in phase with their inputs. Similarly, when the enable input is low, that driver is disabled, and their outputs are off and in the high-impedance state. The figure shows pin diagram.

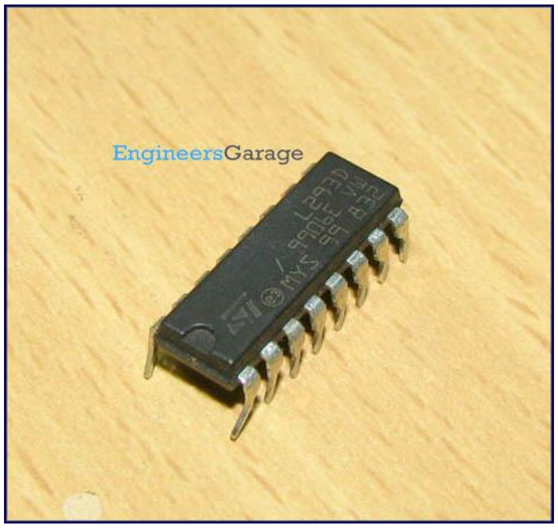


Figure 2.3: L293D driver pin diagram

2.6 Theory of DC Servo Motor

As we know that any electrical motor can be utilized as servo motor if it is controlled by servomechanism. Likewise, if we control a DC motor by means of servomechanism, it would be referred as DC servo motor.
So we can define a servo motor as a type of motor who's the output shaft can be moved to specific angular position by sending coded signal. The servo mechanism that use position feedback to control of motion and final position. The measured position of output by sensor is compared to the command signal

and be input to the controller. If the output position differs from required, an error signal is generated which then causes the motor rotate to bring the output shaft to appropriate position. The servo motor which is DC or AC motor depends on the power supplied to it. The DC servo motor consists of separately excited DC motor or permanent magnet DC motor and the armature is designed to have large resistance so that torque-speed characteristics are linear and have a large negative slope. The motors which are utilized as DC servo motors, generally have separate DC source for field winding and armature winding. The control can be archived either by controlling the field current or armature current. Field control has some specific advantages over armature control and on the other hand armature control has also some specific advantages over field control. Which type of control should be applied to the DC servo motor, is being decided depending upon its specific applications.

i. Field Controlled DC Servo Motor

The direction of rotation can be changed by changing polarity of the field. The direction of rotation can also be altered by using split field DC motor, where the field winding is divided into two parts, one half of the winding is wound in clockwise direction and other half in wound in anticlockwise direction. The amplified error signal is fed to the junction point of these two halves of the field as shown below. The magnetic field of both halves of the field winding opposes each other. During operation of the motor, magnetic field strength of one half dominates other depending upon the value of amplified error signal fed between these halves. Due to this, the DC servo motor rotates in a particular direction according to the amplified error signal voltage.

i. Servo motor operation

A servo consists of a motor (DC or AC), a potentiometer, gear assembly and a controlling circuit. First of all, we use gear assembly to reduce RPM and to increase torque of motor. Say at initial position of servo motor shaft, the position of the potentiometer knob is such that there is no electrical signal generated at the output port of the potentiometer. Now an electrical signal is given to another input terminal of the error detector amplifier. Now difference between these two signals, one comes from potentiometer and another comes from other source, will be processed in feedback mechanism and output will be provided in term of error signal. This error signal acts as the input for motor and motor starts rotating. Now motor shaft is connected with potentiometer and as motor rotates so the potentiometer and it will generate a signal. So as the potentiometer's angular position changes, its output feedback signal changes. After sometime the position of potentiometer reaches at a position that the output of potentiometer is same as external signal provided. At this condition, there will be no output signal from the amplifier to the motor input as there is no difference between external applied signal and the signal generated at potentiometer, and in this situation motor stop rotating.

ii. DC servo motor (MG 996R)

Direct Current servo motor (MG 996R) as shown in Figure 2.4, is a heavy-duty metal gear, digital servo with 180°wide angle, high torque power, improved stability and durability. The servo is able to work with 6V and deliver a strong torque power of over 9.4Kg. This (MG 996R) servo demonstrates a maximum torque of 11Kg without much vibration or

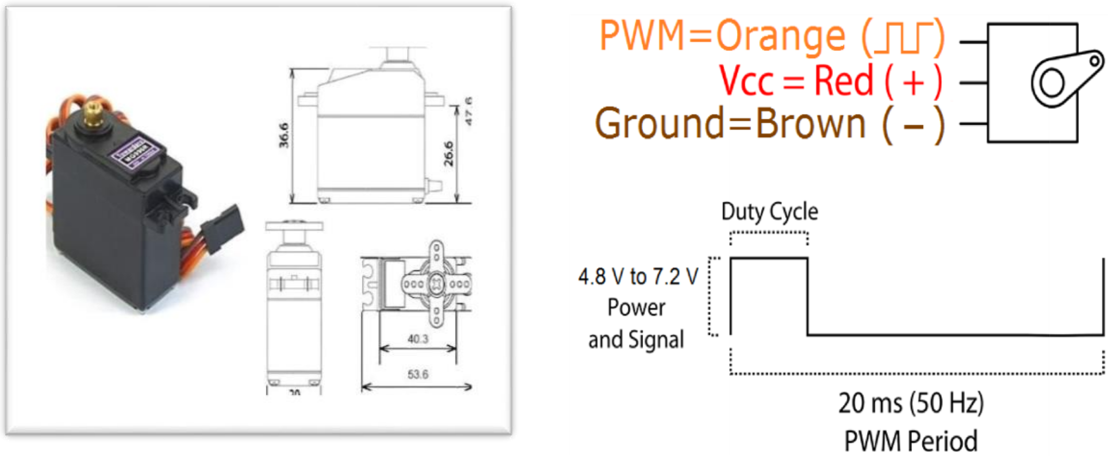


Figure 2.4: Servo motor and wire color diagram

2.7 Power supply

A power supply is an electronic device that supplies electric energy to an electrical load. The primary function of a power supply is to convert one form of electrical energy to another and, as a result, power supplies are sometimes referred to as electric power converters. Some power supplies are discrete, stand-alone devices, whereas others are built into larger devices along with their loads. All power supplies have a power input, which receives energy from the energy source, and a power output that delivers energy to the load. shows some information about Advanced Technology extended ATX computer power supply which used as power supply for feeding the circuit. The ATX is the most common supply out there and is in use in most desktop computers today.

2.8 Compass Sensor

The Compass Module is designed for low-field magnetic sensing with a digital interface and perfect to give precise heading information. This compact sensor fits into small projects such as UAVs and robot navigation

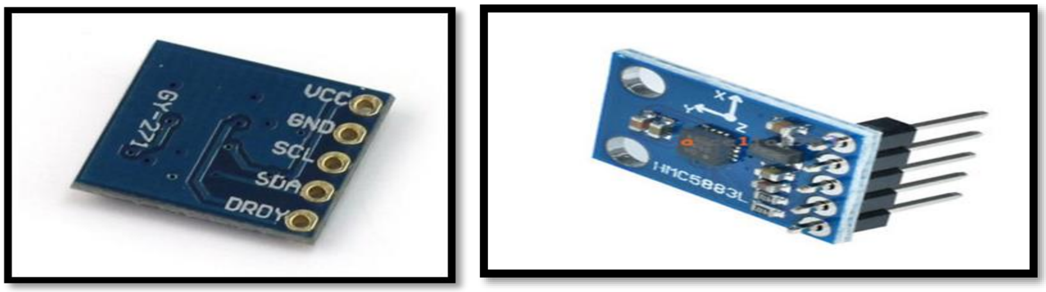


Figure 2.5: Compass Sensor (HMC5883L)

systems. The sensor converts any magnetic field to a differential voltage output on 3 axes. This voltage shift is the raw digital output value, which can then be used to calculate headings or sense magnetic fields coming from different directions.

i. Specifications

- Power 3V-5V DC.
- Chipset HMC5883L.
- Communication via I2C protocol.
- Measuring range: ± 1.3-8 Gauss.
- Dimensions 14.8 x 13.5 x 3.5mm.

ii. Pin Configuration

- VCC: 3V-5V DC.
- GND: ground.
- SCL: analog input (A5).
- SDA: analog input (A4).
- DRDY: not connected.

2.8.3 RPI camera

The Raspberry Pi Camera Board plugs directly into the CSI connector on the Raspberry Pi. It's able to deliver a crystal clear 5MP resolution image, or 1080p HD video recording at 30fps! Latest Version 1.3! Custom designed and manufactured by the Raspberry Pi Foundation in the UK, the Raspberry Pi Camera Board features a 5MP (2592?1944 pixels) Omni vision 5647 sensor in a fixed focus module. The module attaches to Raspberry Pi, by way of a 15 Pin Ribbon Cable, to the dedicated 15-pin MIPI Camera Serial Interface (CSI), which was designed especially for interfacing to cameras. The CSI bus is capable of extremely high data rates, and it exclusively carries pixel data to the BCM2835 processor. The board itself is tiny, at around 25mm x 20mm x 9mm, and weighs just over 3g, making it perfect for mobile or other applications where size and weight are important. The sensor itself has a native resolution of 5 megapixels, and has a fixed focus lens onboard. In terms of still images, the camera is capable of 2592 x 1944-pixel static images, and also supports 1080p @ 30fps, 720p @ 60fps and 640x480p 60/90 video recording. The camera is supported in the latest version of Raspbian, the Raspberry Pi's preferred operating system. The Raspberry Pi Camera Board Features:

- Fully Compatible with Both the Model A and Model B Raspberry Pi 5MP Omnivision 5647 Camera Module
- Still Picture Resolution: 2592 x 1944
- Video: Supports 1080p @ 30fps, 720p @ 60fps and 640x480p 60/90 Recording
- 15-pin MIPI Camera Serial Interface - Plugs Directly into the Raspberry Pi Board
- Size: 20 x 25 x 9mm
- Weight 3g
- Fully Compatible with many Raspberry Pi cases

CHAPTER THREE
RASPBERRY PI SUBSYSYEM AND INTERFACE

3.1. System interface

In this chapter the study was tackled the system communication of each component with the raspberry pi, discus about port subsystems according to the given specification of each component. motors, drivers, sensors, power supply and also the controller itself in case of detail.

3.2. Raspberry pi controller

The Raspberry pi is a series of small single-board computers used to control the dc motors in the car, and the servo motors in the arm according to the signals received from the mobile and also the readings of compass sensor signals

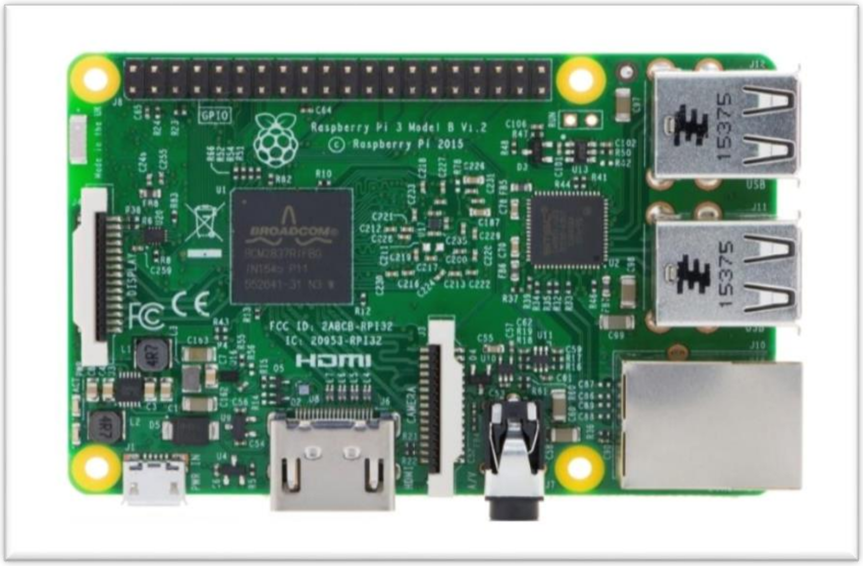


Figure3.1: Raspberry pi controller

3.2.1.Technical Specification

- Broadcom BCM2837 64bit ARMv7 Quad Core Processor powered Single Board Computer running at 1.2 GHz.
- 1GB RAM.
- BCM43143 Wi-Fi on board.
- Bluetooth Low Energy (BLE) on board.
- 40pin extended GPIO.
- 4 x USB 2 ports.
- 4 pole Stereo output and Composite video port.
- Full size HDMI.
- CSI camera port for connecting the Raspberry Pi camera.
- DSI display port for connecting the Raspberry Pi touch screen display.
- Micro SD port for loading your operating system and storing data.
- Upgraded switched Micro USB power source (now supports up to 2.4 Amps).
- Expected to have the same form factor has the Pi 2 Model B, however the LEDs will change position.

3.2.2. Feature of Raspberry Pi

i. Broadcom BCM2835 SOC

Multimedia processor which it contain CPU (central processing unit) acts as a brain of the raspberry pi controller and give the whole actions and operation to the other devices that connected to, the CPU specifications are ARM 1176JZF-S (armv6k) 700MHz, RISC Architecture and low power draw and Not compatible with traditional PC software.

The graphical user interface GPU which it's specification are broad com Video IV and specialized graphical instruction sets. The Broadcom BCM2835 SOC also contain random access memory RAM 512MB (Model B rev.2) 256 MB (Model A, Model B rev.1)

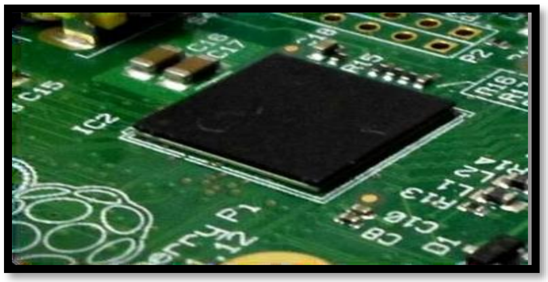


Figure 3.2: Broadcom BCM2835 SOC

ii. Connecting a Display and Audio

1. **HDMI**

   – Digital signal

   – Video and audio signal

   – DVI cannot carry audio signal

   – Up to 1920x1200 resolution

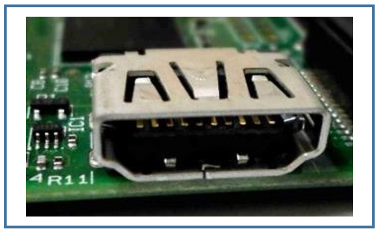
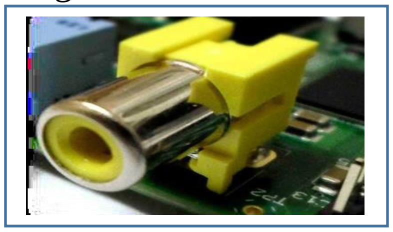


Figure 3.3:HDMI

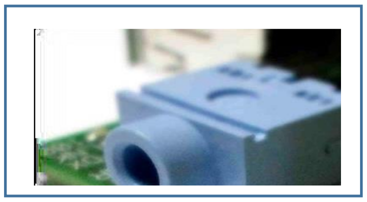


2. **Composite RCA**

   – Analog signal

   – 480i, 576i resolution

Figure 3.4: RCA



3. **Audio jack**

   3.5mm

- The Audio Jack is a 3.5 mm standard

Figure 3.5: Jack

iii. Universal Serial Bus

- Two USB 2.0 ports in RPI.
- Buy a powered USB hub

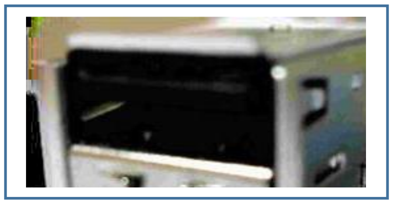


Figure 3.6: USB

iv. Storage: Secure Digital (SD)

Figure 3.7: SD card

- **Form factor**
  – SD, Mini SD, Micro SD
- **Types of Card**
  – SDSC (SD): 1MB to 2GB
  – SDHC: 4GB to 32 GB
  – SDXD up to 2TB

v. Networking – wireless

Figure 3.8: Ethernet

1. **Protocols**
   - 802.11 b, up to 11Mbps
   - 802.11 g, up to 54Mbps
   - 802.11 n, up to 300Mbps
   - 802.11 ac (draft), up to 1Gbps
2. **Frequency band**
   - 2.4GHz, 5GHz.

iv. Low Speed Peripherals

Which content 40 Pins.

•General Purpose Input/output (GPIO)

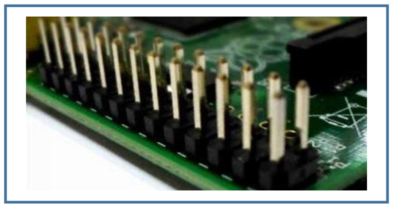


Figure 3.9: RPI Pins

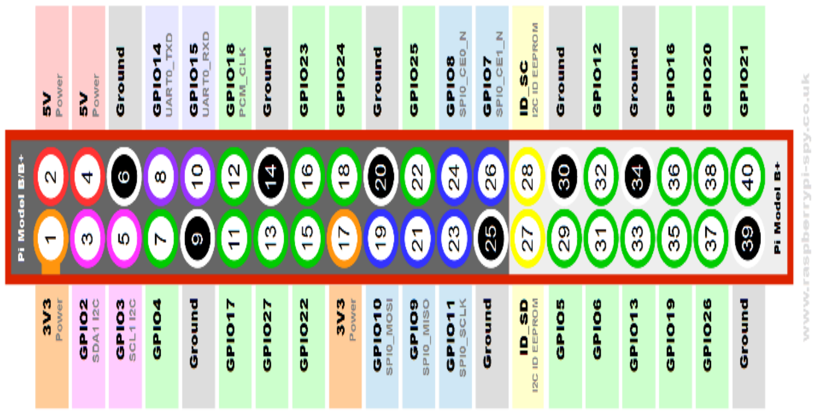


Figure 3.10: GPIO

3.2.3 The GPIO

Pins can be configured to be input/output, Reading from various environmental sensors Writing output to dc motors, LEDs for status.

- 3.3V Pins (1,17)
- 5V Pins (2,4).
- GND Pins (6,9,14,20,30,39).
- Sending 5V to a pin may kill the Pi.
- Maximum permitted current draw from a 3.3V Pin is 50mA.

3.3 Port subsystems

In port subsystem we define the branch of devices connected to the raspberry pi processor which are:

3.3.1 L293D Driver

There are sixteen pins in Driver L293D. Pin (1, 8, 16 and 9) represent the Enable1, VCC1, VCC2 and Enable2 of the driver and it's connected to the power supply (High voltage) (5 volt) , pin(4, 5, 12 and 13) represent the ground of the driver it's connected to the power supply ground pin (Low voltage) (0 volt) ,pin(3, 6, 11 and 14) represent the output of the driver and the input of motors at same time, pin (3 and 6) connected to first motor and pin (11 and 14)) connected to second motor, pin(2, 7, 10 and 15) represent the input of the driver and connected to the output of raspberry pi, pin (2 and 7) connected to pin(11 and 13) in Raspberry Pi and Pin (10 and 15) connected to pin(16 and 18) in Raspberry Pi to give the specification of control on the motors and reverse it's movement of it.

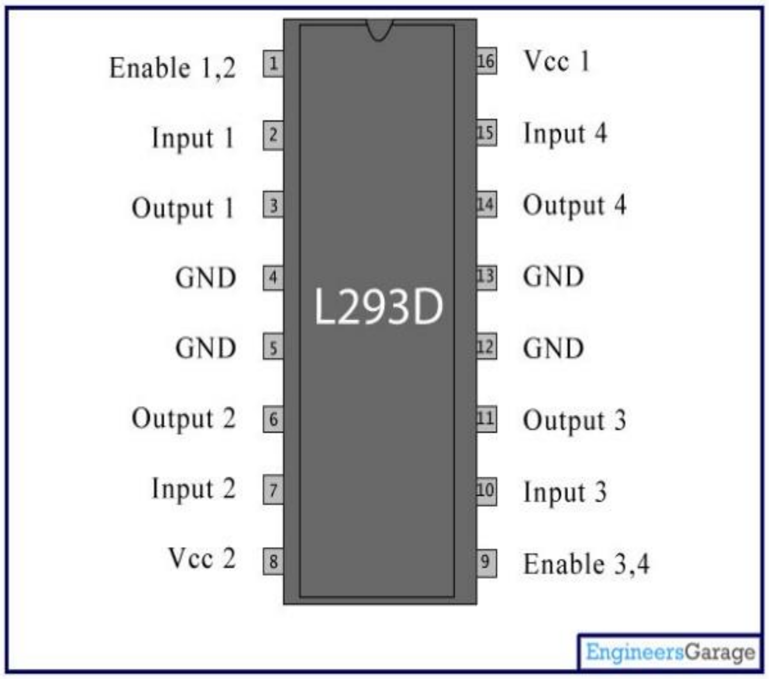


Figure 3.11: Driver L293D

i. Pin Description

Pin No	Function	Name
1	Enable pin for Motor 1; active high	Enable 1,2
2	Input 1 for Motor 1	Input 1
3	Output 1 for Motor 1	Output 1
4	Ground (0V)	Ground
5	Ground (0V)	Ground
6	Output 2 for Motor 1	Output 2
7	Input 2 for Motor 1	Input 2
8	Supply voltage for Motors; 9-12V (up to 36V)	VCC $_2$
9	Enable pin for Motor 2; active high	Enable 3,4
10	Input 1 for Motor 1	Input 3
11	Output 1 for Motor 1	Output 3
12	Ground (0V)	Ground
13	Ground (0V)	Ground
14	Output 2 for Motor 1	Output 4
15	Input2 for Motor 1	Input 4
16	Supply voltage; 5V (up to 36V)	VCC $_1$

Table 3.1: Pin Description of Driver L293D

3.3.2 Compass Sensor

The compass sensor Module shown in figure is designed for low-field magnetic sensing with a digital interface and perfect to give precise heading information. This compact sensor fits into small projects such as UAVs and robot navigation systems. The sensor converts any magnetic field to a differential voltage output on 3 axes (x, y and z). This voltage shift is the raw digital output value, which can then be used to calculate headings or sense magnetic fields coming from different directions. Compass sensor contain five pins which are (VCC, GND, SCL, SDA and DRDY). The VCC pin connected to pin (1) in Raspberry Pi which represent the 3.3 volt . The GND pin connected to pin (6 or 9 or 14 or 20 or 39) in Raspberry Pi. The SCL connected to pin (5) in Raspberry Pi. The SDA connected to pin (3) in Raspberry Pi, and the DRDY is not connected.

The compass pin description are power 3V-5V DC, Chipset HMC5883L Communication via I2C protocol, Measuring range: ± 1.3-8 Gauss. Dimensions 14.8 x 13.5 x 3.5mm. and the Pin Configuration of compass sensor are VCC: 3V-5V DC GND: ground SCL: analog input (A5) SDA: analog input (A4) DRDY: not connected

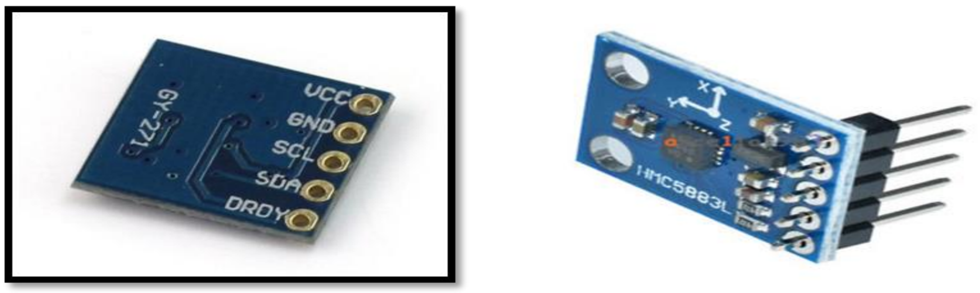


Figure 3.12: Compass Sensor (HMC5883L)

3.3.3 Servo motor

The Servo Robotic Arm has three servo motors. The first one used for a roller movement of the arm which is connected to Pin (12) in Raspberry Pi, the ground and to power supply. The second servo motor used for horizontal movement of the arm, which is connected to ground and power supply but the signal terminal connected to Pin (22) in Raspberry Pi. The third servo motor used for gripper (to collect things) which is connected to ground and power supply but the signal terminal connected to Pin (7) in Raspberry Pi.

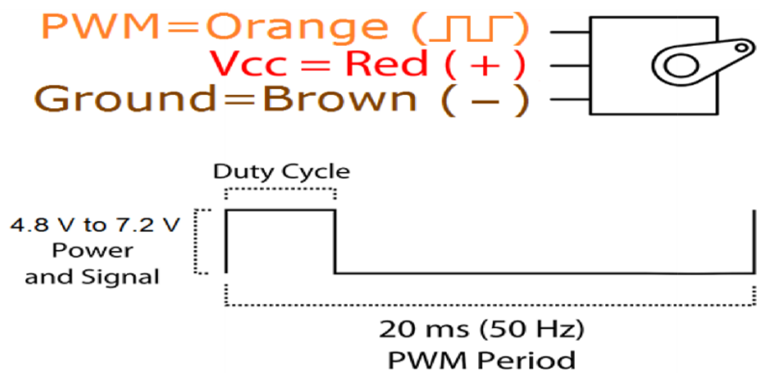


Figure 3.13: Servo motor and wire color diagram

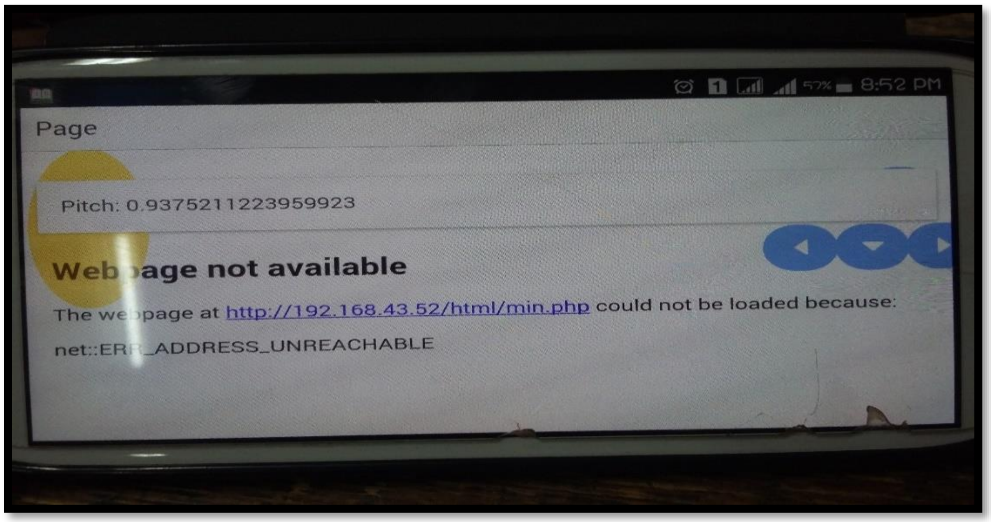
i. Specifications

- Weight: 55 g
- Dimension: 40.7 x 19.7 x 42.9 mm approx.
- Stall torque: 9.4 kgf.cm (4.8 V), 11 kgf.cm (6 V)

- Operating speed: 0.17 s/60º (4.8 V), 0.14 s/60º (6 V)
- Operating voltage: 4.8 V a 7.2 V
- Running Current 500 mA – 900 mA (6V)
- Stall Current 2.5 A (6V)
- Dead band width: 5 µs
- Stable and shock proof double ball bearing design
- Temperature range: 0 ºC – 55 ºC

3.4 Smart phone application

The smart phone application designed with special programming language like java script, PHP and web design language which are HTML and CSS to give the desired buttons and phone sensors reading to be able to send over the web to the raspberry pi controller to recognize and then gives the order to the devices connected to it to give us the motion and control over robot with easy way.



Pitch: 0.9375211223959923

Web page not available

The webpage at http://192.168.43.52/html/min.php could not be loaded because:

net::ERR_ADDRESS_UNREACHABLE

Figure 3.14: Application in the smart phone

3.5 Raspberry pi operating system

3.5.2 Linux

GNU-Linux is a modular Unix-like operating system, deriving much of its basic design from principles established in Unix during the 1970's and 1980's It is a well-known and tested free and open source operating system. Many of today OS's such as Ubuntu, Fedora, Android - are derived from GNU-Linux, which are called (Distributions). There are several Operating Systems developed specifically for the Raspberry Pi Computer such as:

i. Raspbian OS

- Raspbian is the official OS for the Raspberry Pi, it is derived from Linux Debian.
- It is supported by the Raspberry Pi Foundation and its online community.
- Its backed with a large collection of programs available at raspbian.org

ii. Openelec

- Is a special OS for media and home entertainment.
- It is based on the XBMC media manager.
- And it facilitates hacking the Raspberry Pi into a media center.

iii. Adafruit

- A special OS for controlling advanced control applications

iv. Kali Linux

- Kali-linux is the strongest OS used for Network security and hacking.
- It provides tools to hack computer networks and websites as well as wireless networks.

v. ROC

- Robotics Operating System is used especially for robotics applications.
- It contains collections of programs preinstalled on ubuntu and Debian OS's

vi. Arch Linux

- Arch Linux is a fast OS specialized for Linux experts who are welling to control every little and great in their Linux distributions.
- It is also one of the smallest Linux Distributions.

vii. NOOBS

- NOOBS is a collection of 6 raspberry pi operating systems contained in a single image, it is made to help beginners to easily install a Raspberry Pi OS.

viii. RISC OS

- RISC OS is not a Linux Distribution, and it is designed to be lightweight to run on ARM architecture.

CHAPTER FOUR
MAIN CIRCUIT DESIGN AND PROGRAMMING

4.1 System structure

The first step of designing a robotic arm with a car is to detect the dimension and workspace configuration according to the requirements. The next step is to decide the specification of each actuator [4]. The robot arm was designed with three degrees of freedom and talented to accomplish accurately simple tasks, such a slight material handling.

The structure of the robotic arm and the car is built with aluminum in order to decrease the overall weight of the robot. The arm is attached to the base of the car. All parts of the robot and the car including the parts for shoulder, gripper and etc., were cut accurately. Some processes where applied to the aluminum to make the necessary holes and cuts to connect the parts to each other and to keep the actuators tightly. Physical movement of the robot is done by using servo motors and DC motors for the car. All the parts were cut and drilled properly according to the design template.

It is important to mention that the base ought to have considerably heavy weight in order to maintain the general balance of the robotic arm in case of grabbing an object.

4.2 Block Diagram

Figure 4.1 show the block diagram of the project.

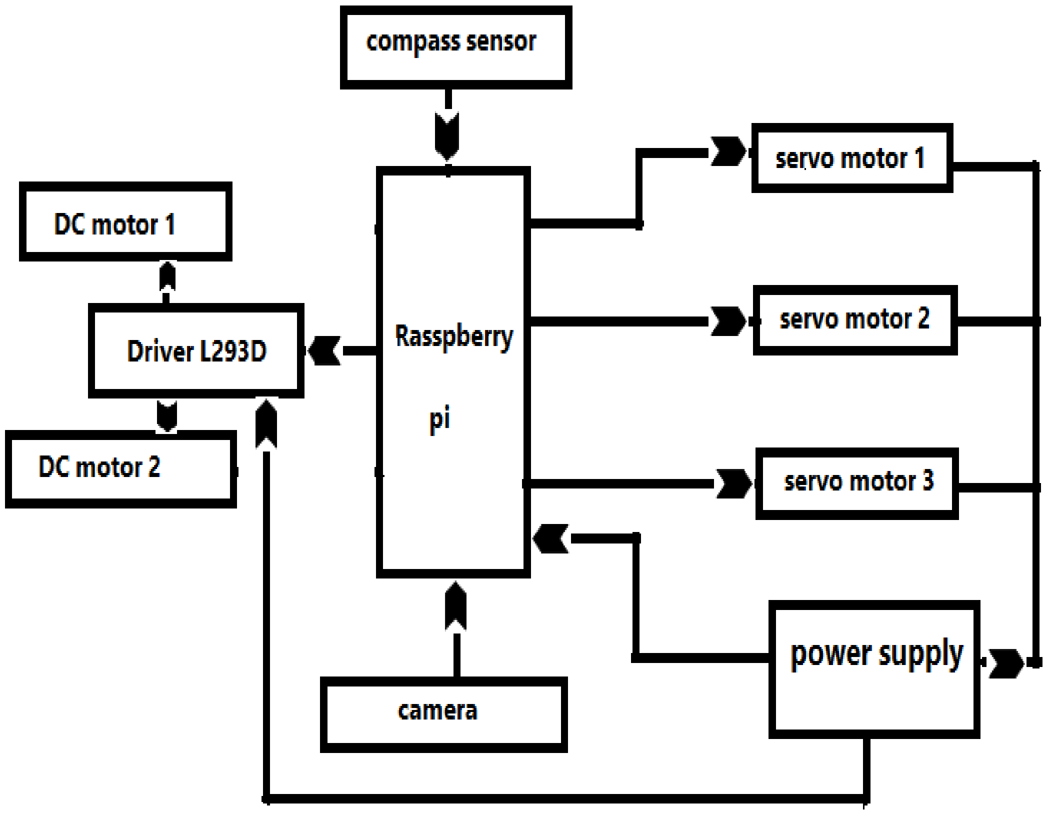


Figure 4.1: System block diagram

-The different components involved in our project are:

4.2.1 Raspberry pi

Is used to control the DC motors in the car, and the servo motors in the arm according to the signals received from the mobile and also reads compass sensor signals

4.2.2 Motors

The DC motors used to move the car forward and backward and to change the direction to the left or the right, and the servo motors used in the arm for the movement up and down and left or right allowing the arm to move toward the object, and one for the gripper to pick and drop the object.

4.2.3 The Camera

It is used to observe the object which the arm is going to pick, and to supervise the road for the car as well.

4.2.4 Compass Sensor

It is chipset HMC5883L it designed for low-field magnetic sensing with a digital interface; it is used as a compass to determine the car direction

4.2.5. Power supply

A regulated power supply is an electronic circuit, it is function to supply a stable voltage to the system components. This is used to supply the power to the microcontroller and the motors.

4.3 Wiring diagram

The wiring diagram below in figure 4.2 describes how the wires connected between the raspberry and the other components:

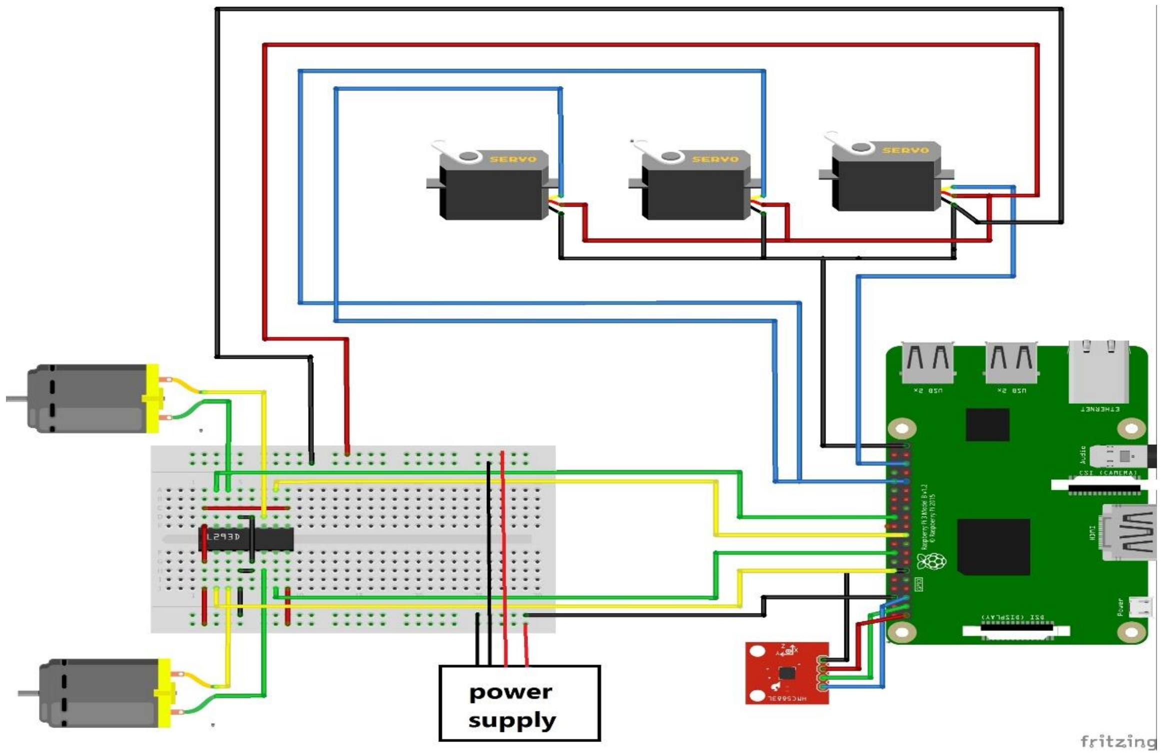


Figure 4.2: Wiring diagram

4.4 Design

The design of the model consists of three parts, the robotic arm, Controlling unit, and the car. The design of each of the arm, the control unit and the car will be specified as follow:

4.4.1The arm design

The mechanical design of the robot arm is based on a robot manipulator with similar functions to a human arm.

It consists of:

1. Three servo motors.
2. Aluminum links.
3. End effectors (Gripper).

in constructing the arm, we made use of three servo motors (including the gripper).there is a servo motor at the base , which allows for angular movement of the whole structure ; and the second to allow the upward and downward movement of the arm ,while the last one for gripping the object.
a motor is toggled in the aluminum links as the body of the arm; the reason of choosing the aluminum because of it's lightweight and it is also strong enough to keep and hold the whole parts tightly together, the lighter the body is the more less load the motors will bear. The degree of freedom, or DOF, is a very important term to understand. Each degree of freedom is a joint on the arm, a place where it can bend or rotate or translate. You can typically identify the number of degree of by the number of actuators on the robot arm. Now this is very important- when building a robotic arm you need as few degrees of freedom allowed for the application; because each degree of freedom requires a motor, and that exponentially the cost. In this model there are three degrees of freedom, and three actuators (servo motors). These three degrees of freedom make controlling the arm more accurate, but it make the movement of the arm in the arm workspace freer than the two degree of freedom. The workspace or sometimes known as reachable space is all places that the end effector (gripper) can reach. The workspace is dependent on the DOF angle/translation limitations, the arm link strength, the angle oat which something must be picked up at, etc. The workspace is highly dependent on the arm configuration. The end effector in this project is a gripper, this gripper acts like the fingers of a human hand. The actuator of the gripper is a servo motor linked with a rope that causes the movement of the gripper. The rope transfer the rotation of the motor to linear movement, when the motor rotate in the clockwise the two parts of the gripper close, and when it rotate anti

clockwise the two parts open. So controlling the gripper is by controlling the angle of the servo motor using the microcontroller.

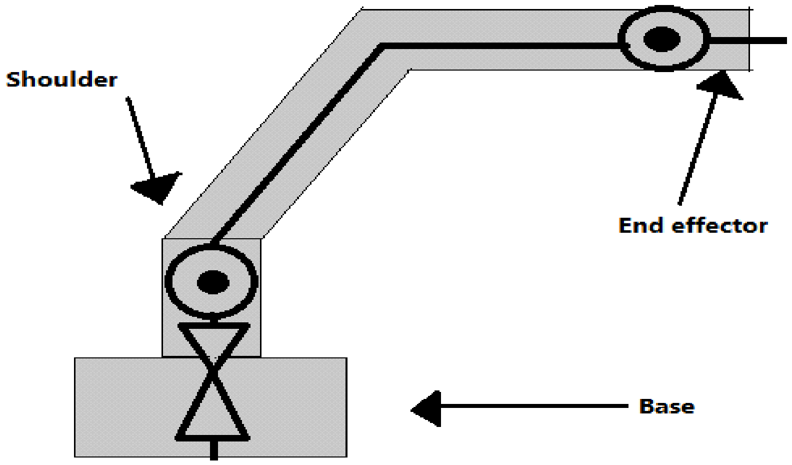


Figure 4.3: Show free body diagram of the robot arm

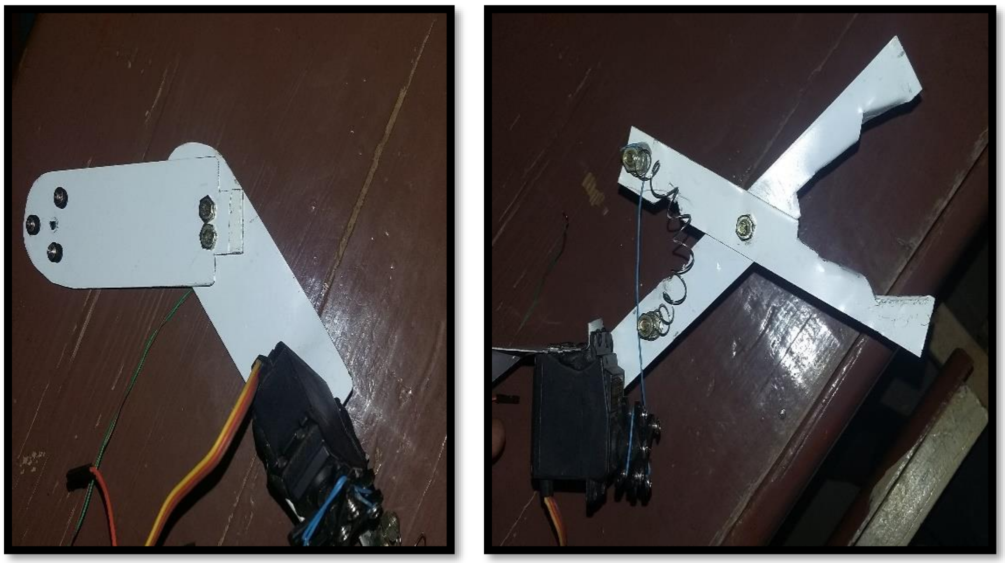


Figure 4.4: Show the arm parts

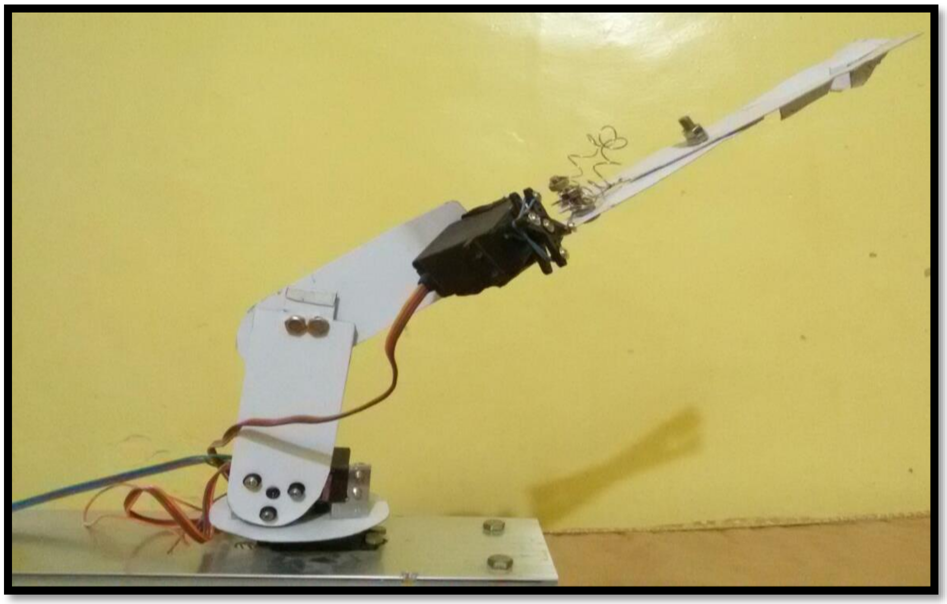


Figure 4.5: Show the arm design

4.4.2 Control unit

The control unit is the board that control the robotic arm, and it consist of the microcontroller. The microcontroller used in this project is raspberry pi which controls the motors in the arm and the motor that controls the car using the mobile phone. The servo motor has three pins, the red and the black pins are for the power and the yellow one is for the signal, this signal pin is connected directly to one of PWM (pulse width modulation) pins as an output from the microcontroller.

The controlling of each motor in the arm is achieved by a program stored in the memory of the microcontroller. This program sets the angle of the servo motor to move the arm to the desired point in the work space.

The figure 4.6 below shows the control unit.

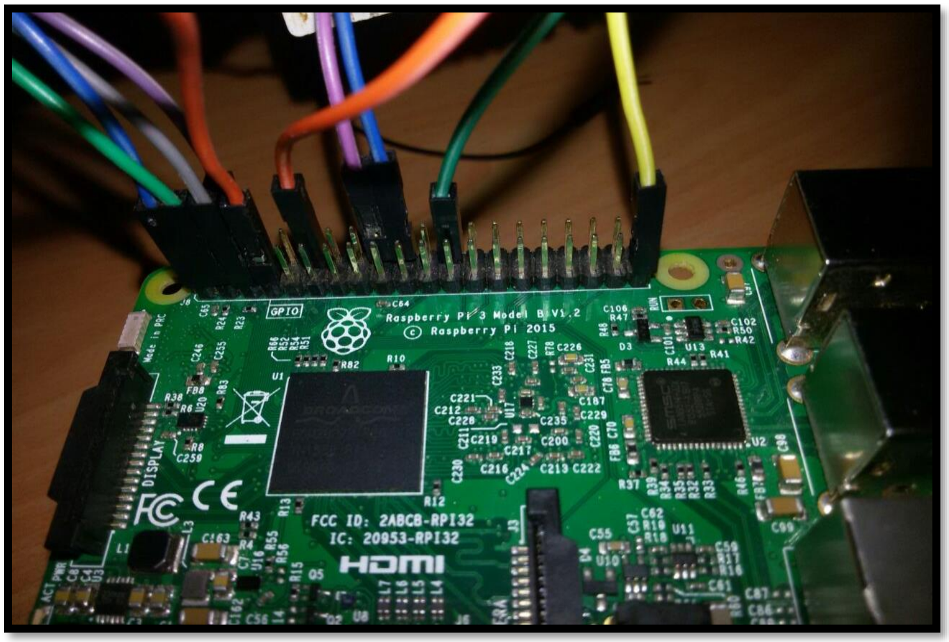
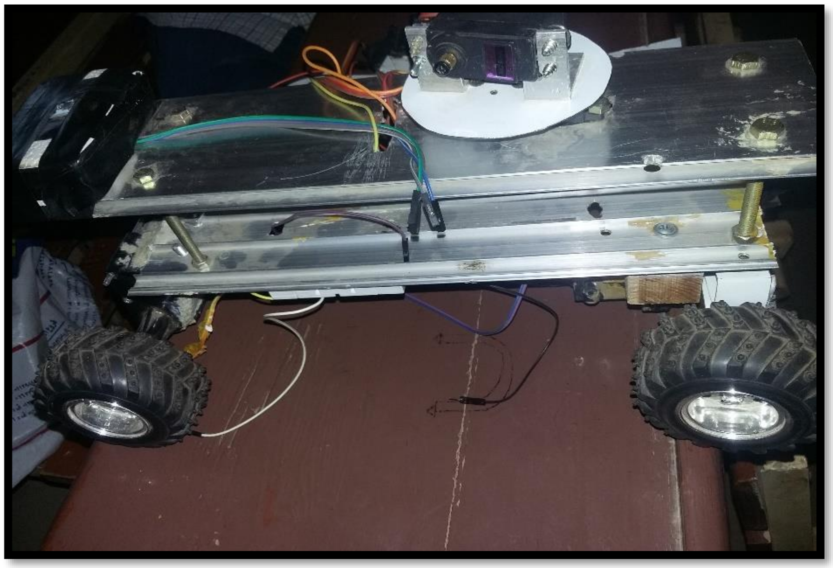


Figure 4.6: Show the control unit

4.4.3 The car design

In constructing the car, we made use of two DC motors. There is a DC motor in the back of the car for the movement forward and backward, and the other one placed in the front and it is used for the direction either left or right . The body of the car is made of aluminum because of it is light weight and strength, there are two pieces one over the other and there is a gap between them where we put the controller (the raspberry), and the bread board which consist the driver for the DC motors and the wires for connection. On the top of the car we have the compass sensor and the robotic arm.



.

Figure 4.7: The car design

4.5 Robot Workspace

The workspace of a robotic manipulator is the total volume swept out by the end effector as the manipulator executes all possible motions. The workspace is determined by the geometry of the manipulator and the limits of the joint motions. It is more specific to define the reachable workspace as the total locus of points at which the end effector can be placed and the dexterous workspace. [8]

It should be noted that it does not include the DOF which controls the wrist orientation as the workspace is independent of orientation variable.

The shoulders rotate a maximum of 90 degrees. To determine the workspace, trace all locations that the end effector (gripper) can reach as in the image below. Now rotating that by the base joints another 180 degrees. This creates a workspace of a shelled quarter sphere as shown below.

If you change the link lengths you can get very different sizes of workspaces, but this would be the general shape. Any location outside of this space is a location the arm can’t reach. If there are objects in the way of the arm, the workspace can get even more complicated.

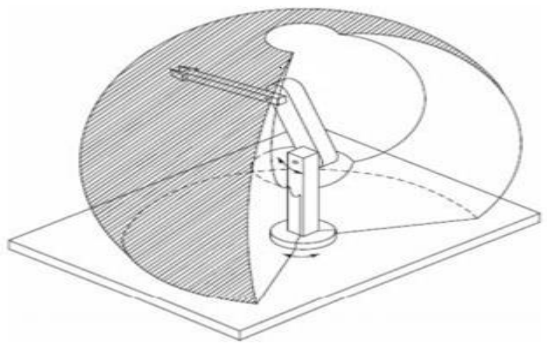


Figure 4.8: Work region of the robotic arm

The arm is a three degree of freedom system. Two DOF control the position of the arm in the Cartesian space, and the third servo for actuating gripper.

The robot features in the control GUI or teach pendant are base rotation, shoulder, and a functional gripper. The base of the robotic arm and the links are made up of Aluminum.

Servo motors serve as the actuators at various joint. These motors feature 180 degree rotation in clockwise direction. The motors are controlled by raspberry pi board upon receiving commands from a host mobile via internet.

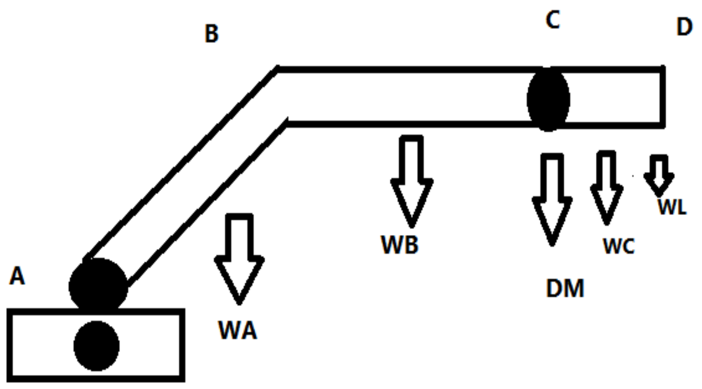


Figure 4.9: Force diagram of robot arm

The values used for the torque calculations:

WC =0.020 kg (weight of link CD)

WB = 0.020 kg (weight of link BC)

WA = 0 .020kg (weight of link AB)

WL = 0.25 kg (load)

Dm = 0.055 kg (weight of motor)

LAB = 0.1 m (length of link AB)

LBC = 0.1 m (length of link BC)

LCD = 0.1 m (length of link CD)

Performing the sum of forces in the Y axis, using the loads as shown in Figure, and solving for CY and CB, see Equations (1). (4). Similarly, performing the sum of moments around point C, Equation (5), and point B, Equation (6), to obtain the torque e in C and B, Equations (7) and (8), respectively

$\sum Fy= (WL+WC+DM)*g - Cy=0$ (1)

$CY= (0.325 kg)\ 9.8m/s^2= 3.185\ N$ (2)

$\sum Fy=(WL+ WC+DM+WB+WA)*g – BY=0$ (3)

$BY= (0.365 kg)\ 9.8m/s^2= 3.577\ N$ (4)

$\sum Mc = -WC\ (LCD/2)-WL\ (LCD) +MC=0$ (5)

$\sum MA=-WL\ (LAB+LBC+LCD)-WC(LAB+LBC+LCD/2)$

$-WA\ (LAB/2)-DM\ (LAB+LBC)-WB\ (LAB+LBC/2) =0$ (6)

MC= 0.026 Nm

MA= 0.095Nm

The servo motor that was selected, based on the calculations, is the TOWARDPRO MG996R, which has a torque of 2Nm this motor was recommended because it is much cheaper than any other motor with same specifications.[4]

4.6 Python

Python is a general-purpose programming language that is often applied in scripting roles. It is commonly defined as an object-oriented scripting language a definition that blends support for OOP with an overall orientation toward scripting roles. If pressed for a one-liner, I'd say that Python is probably better known as a general purpose programming language that blends procedural, functional, and object oriented paradigms a statement that captures the richness and scope of today's Python.

Python comes with standard Internet modules that allow Python programs to perform a wide variety of networking tasks, in client and server modes. Scripts can communicate over sockets; extract form information sent to server-side CGI scripts; transfer files by FTP; parse and generate XML and JSON documents; send, receive, compose, and parse email; fetch web pages by URLs; parse the HTML of fetched web pages; communicate over XML-RPC, SOAP, and Telnet; and more. Python's libraries make these tasks remarkably simple and because our systems use IOT python was the perfect programming language beside these features: [3]

4.6.1 Software quality

For many, Python’s focus on readability, coherence, and software quality in general sets it apart from other tools in the scripting world. Python code is designed to be readable, and hence reusable and maintainable much more so than traditional scripting languages. The uniformity of Python code makes it easy to understand, even if you did not write it. In addition, Python has deep support for more advanced software reuse mechanisms, such as object-oriented (OO) and function programming. [3]

4.5.2 Developer productivity

Python boosts developer productivity many times beyond compiled or statically typed languages such as C, C++, and Java. Python code is typically one-third to one-fifth the size of equivalent C++ or Java code. That means there is less to type, less to debug, and less to maintain after the fact. Python programs also run immediately, without the lengthy compile and link steps required by some other tools, further boosting programmer speed.

4.5.3 Program portability

Most Python programs run unchanged on all major computer platforms. Porting Python code between Linux and Windows, for example, is usually just a matter of copying a script’s code between machines. Moreover, Python offers multiple options for coding portable graphical user interfaces, database access programs, web based systems, and more. Even operating system interfaces, including program launches and directory processing, are as portable in Python as they can possibly be. [3]

4.6.4 Support libraries

Python comes with a large collection of prebuilt and portable functionality, known as the standard library. This library supports an array of application-level programming tasks, from text pattern matching to network scripting. In addition, Python can be extended with both homegrown libraries and a vast collection of third-party application support software. Python's third-party domain offers tools.

For website construction, numeric programming, serial port access, game development, and much more (see ahead for a sampling). The NumPy extension, for instance, has been described as a free and more powerful equivalent to the Mat lab numeric programming system. [3]

4.6.5 Component integration

Python scripts can easily communicate with other parts of an application, using a variety of integration mechanisms. Such integrations allow Python to be used as a product customization and extension tool. Today, Python code can invoke C and C++ libraries, can be called from C and C++ programs, can integrate with Java and NET components, can communicate over frameworks such as COM and Silverlight, can interface with devices over serial ports, and can interact over networks with interfaces like SOAP, XML-RPC, and CORBA. It is not a standalone tool. [3]

4.7 Procedure

When connecting the power supply to the raspberry, servo motors in the arm, and the dc motors in the car starting the application on the mobile phone, the camera will work and he view in front of the car will appears on the mobile screen, then the system will be ready to pick up any object.

The control operation is as follows the car move forward, backward, left, or right according to the button pushed on the mobile screen when pushing any button, a signal will be send to the raspberry and the dc motor in the car will work and this will move the car to the object location, once the object is in the workspace of the arm the button should be released and the car will stop. Then we can start controlling the arm throw moving the mobile , and it is different from the car control because here we use the mobile angles instead of buttons so when we left the mobile up the raspberry read the angels of the phone and the arm goes up with the same amount and if we move the mobile to the right the arm moves right and so on , once the arm in the right position to pick the object there is a button on the screen for the gripper which allows us to pick the object , after that we move the car to the desired position and there is another button to release the object on the desired place.

CHAPTER FIVE
CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

The main aims of the project is to design and implement the robot and build the connection between the robot and the android smart phone The focus of this work was to design and programming robotic arm. the robot arm was designed with three degrees of freedom and talented to accomplish accurately simple tasks, such as light material handling. The robot arm is equipped with several servo motors which do links between arms and perform arm movements. The controller drives the servo motors and the car with the capability of modifying position. The programming is done on Raspberry pi controller using python programming language. The compass sensor is also used to detect the angles of rotation and compare the signals between the phone and the car to modulate the right position and show the video on the screen.

5.2 Recommendations

- Use artificial neural network to train and program this project instead of giving the orders by your self’s.
- Using smart phone screen to view video’s of raspberry pi camera instead of showing it on personal computer screen.

5.3 References

[1] H. Bunke, “FUNDAMENTALS OF ROBOTICS”, Bern, Switzerland, 2003.

[2] Fareed Shakhatreh, “THE BASICS OF ROBOTICS”, Syksy, 2011.

[3] Mark Lutz, “Learning Python”, O’Reilly Media, Inc., 1005 Graven stein Highway North, Sebastopol, CA 95472, United States of America.

[4] Park, I. D., Park, C., Do, H., Choi, T., Kyung, J., "Design And Analysis of Dual Arm Robot Using Dynamic Simulation" IEEE 10th International Conference on Ubiquitous

[5]Robots and Ambient Intelligence (URAI), pp. 681-682, 2013

[6]John J. Craig “ Introduction to Robotics”, Third Edition © 2005 Pearson Education, Inc. United States of America.

[7]Abdallah Ali Abdallah “Simply raspberry pi” © 2010.

[8] Bruno Siciliano, Oussama Khatib (Eds.),”Springer Handbook of Robotics”, Springer-Verlag Berlin Heidelberg,2008

5.4 Appendix A

Raspberry pi motors code

```
from flask import Flask, abort, request
import json
import ast
from subprocess import call
import time
from threading import *
import smbus
import math

SL=1
speed=100
app = Flask(__name__)




def forward(forward_hold):
    call (["echo P1-11={}% > /dev/servoblaster".format(speed)], shell=True)
    call (["echo P1-13=0% > /dev/servoblaster"], shell=True)
    if not(forward_hold):
        time.sleep(SL)
        call (["echo P1-11=0% > /dev/servoblaster"], shell=True)
        call (["echo P1-13=0% > /dev/servoblaster"], shell=True)


def backward(forward_hold):

    call (["echo P1-13={}% > /dev/servoblaster".format(speed)], shell=True)
    call (["echo P1-11=0% > /dev/servoblaster"], shell=True)




    if not(forward_hold):
        time.sleep(SL)
        call (["echo P1-13=0% > /dev/servoblaster"], shell=True)
          call (["echo P1-11=0% > /dev/servoblaster"], shell=True)

def right():



    call (["echo P1-16=80% > /dev/servoblaster"], shell=True)
    call (["echo P1-18=0% > /dev/servoblaster"], shell=True)
    time.sleep(0.15)
    call (["echo P1-16=0% > /dev/servoblaster"], shell=True)
    call (["echo P1-18=0% > /dev/servoblaster"], shell=True)




def left():

    call (["echo P1-18=80% > /dev/servoblaster"], shell=True)
    call (["echo P1-16=0% > /dev/servoblaster"], shell=True)
    time.sleep(0.15)
    call (["echo P1-18=0% > /dev/servoblaster"], shell=True)
    call (["echo P1-16=0% > /dev/servoblaster"], shell=True)

def applyAngle(angle,pin):
    if angle>180:
        angle=180
    if angle<0:
        angle=0
    us=500+(2100-500)/180*angle
    call("echo P1-{}={}us > /dev/servoblaster".format(pin,us) , shell=1 )

@app.after_request
def after_request(response):
    response.headers.add('Access-Control-Allow-Origin', '*')
    response.headers.add('Access-Control-Allow-Headers',                'Content-
Type,Authorization')
```

```
    response.headers.add('Access-Control-Allow-Methods',
'GET,PUT,POST,DELETE')
    return response

@app.route('/foo', methods=['POST'])
def foo():
    if 'yaw' in request.json:
        global imu_event
        imu_event.wait()
        global c
        r=request.json['yaw']
        p=request.json['pitch']
        if c<=90 and (r>=270 and r<=360):
            servo1Angle=90-(180-(r-270)-(90-c))
            applyAngle(servo1Angle,12)
        elif r<=90 and (c>=270 and c<=360):
            servo1Angle=90-(180-(c-270)-(90-r))
            applyAngle(180-servo1Angle,12)
        else:
            servo1Angle=r-c+90
            applyAngle(180-servo1Angle,12)
        applyAngle(-1*(p+5),22)
    if 'x' in request.json and request.json['x']=='h':
        global isClosed
        if isClosed:
            applyAngle(180,7)
        else:
            applyAngle(0,7)
        isClosed=not isClosed




    # if (request.json['yaw']<120)and(request.json['yaw']>40):
        # us=(120-float(request.json['yaw']))*1000/80+700
        # print us
        # call (["echo 4="+str(us)+"us > /dev/servoblaster"], shell=True)
    # else:
        # donothing=4
    # if (request.json['pitch']>-90)and(request.json['pitch']<10):
        # us1=2000+(float(request.json['pitch'])-10)*1000/100
        # print us1
        # call (["echo 3="+str(us1)+"us > /dev/servoblaster"], shell=True)
    # else:
        # donothing=4

    return json.dumps({"hi":5})

def imu(imu_event):
    global c
    bus = smbus.SMBus(1)
    address = 0x0d
    print "working"
```

```
while 1:
    def read_byte(adr): #communicate with compass
        return bus.read_byte_data(address, adr)

    def read_word(adr):
        low = bus.read_byte_data(address, adr)
        high = bus.read_byte_data(address, adr+1)
        val = (high<< 8) + low
        return val

    def read_word_2c(adr):
        val = read_word(adr)
        if (val>= 0x8000):
            return -((65535 - val)+1)
        else:
            return val

    def write_byte(adr,value):
        bus.write_byte_data(address, adr, value)
    write_byte(11, 0b00000001)
    write_byte(10, 0b00100000)
    write_byte(9, 0b00000000|0b00000000|0b00001100|0b00000001 )
    scale = 0.92
    x_offset = -10
    y_offset = 10
```

```
        x_out = (read_word_2c(0)- x_offset+2) * scale #calculating x,y,z
coordinates
        y_out = (read_word_2c(2)- y_offset+2)* scale
        z_out = read_word_2c(4) * scale
        bearing = math.atan2(y_out, x_out)+.48 #0.48 is correction value
        if(bearing < 0):
            bearing += 2* math.pi
        c=math.degrees(bearing)-15
        if c<0:
            c+=360
        c=c+90
        if c>360:
            c=c-360
        print c
        imu_event.set()
        imu_event.clear()
        print "Bearing:", c
    #   print "x: ", x_out
    #   print "y: ", y_out
    #   print "z: ", z_out
        time.sleep(0.2)

global isClosed
isClosed=True
global imu_event
imu_event=Event()
```

```
th=Thread(target=imu,kwargs={'imu_event':imu_event})
th.start()
applyAngle(110,15)
applyAngle(0,7)
print "hi"

if __name__ == '__main__':

    app.run(host='0.0.0.0', port=5000, debug=False)

    call (["sudo ~/PiBits/ServoBlaster/user/./servod --min=0% --max=100%"],
shell=True)
```

5.5 Appendix B

Stream client raspberry pi code

```
"""
Reference:
PiCamera documentation
https://picamera.readthedocs.org/en/release-1.10/recipes2.html

"""


import io
import socket
import struct
import time
import picamera



# create socket and bind host
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect(('192.168.1.100', 8000))
connection = client_socket.makefile('wb')


try:
    with picamera.PiCamera() as camera:
        camera.resolution = (320, 240)      # pi camera resolution
        camera.framerate = 10               # 10 frames/sec
        time.sleep(2)                   # give 2 secs for camera to initilize
        start = time.time()
        stream = io.BytesIO()

        # send jpeg format video stream
        for foo in camera.capture_continuous(stream, 'jpeg', use_video_port =
True):
            connection.write(struct.pack('<L', stream.tell()))
            connection.flush()
            stream.seek(0)
            connection.write(stream.read())
            if time.time() - start > 600:
                break
            stream.seek(0)
            stream.truncate()
    connection.write(struct.pack('<L', 0))
finally:
    connection.close()
    client_socket.close()
```

5.6 Appendix C

Stream server test pc code

```
__author__ = 'zhengwang'


import numpy as np

import cv2

import socket




class VideoStreamingTest(object):
    def __init__(self):


        self.server_socket = socket.socket()
        self.server_socket.bind(('192.168.1.100', 8000))
        self.server_socket.listen(0)
        self.connection, self.client_address = self.server_socket.accept()
        self.connection = self.connection.makefile('rb')
        self.streaming()

    def streaming(self):


        try:
            print "Connection from: ", self.client_address
            print "Streaming..."
            print "Press 'q' to exit"
            stream_bytes = ' '
            while True:
                stream_bytes += self.connection.read(1024)
                first = stream_bytes.find('\xff\xd8')
                last = stream_bytes.find('\xff\xd9')
                if first != -1 and last != -1:
                    jpg = stream_bytes[first:last + 2]
                    stream_bytes = stream_bytes[last + 2:]
                    #image   =   cv2.imdecode(np.fromstring(jpg,   dtype=np.uint8),
cv2.CV_LOAD_IMAGE_GRAYSCALE)
                    image   =   cv2.imdecode(np.fromstring(jpg,   dtype=np.uint8),
cv2.CV_LOAD_IMAGE_UNCHANGED)
                    cv2.imshow('image', image)

                    if cv2.waitKey(1) & 0xFF == ord('q'):
                        break
        finally:
            self.connection.close()
            self.server_socket.close()
if __name__ == '__main__':
    VideoStreamingTest()
```