ABSTRACT

Disasters are unexpected events which shatter our lives. There lies a thin line between success and failure while considering management of disasters. Any calamity, whether natural or otherwise, does not give the luxury of time, while being handled. Disaster response teams must make decisions that can save innumerable number of lives instantaneously. The disaster management teams must rely on first hand insights of the affected zone and data must be highly accurate and dependable. To this end, robots and machines are extremely valuable. We will build a small compact four wheeled robot which can easily enter into unknown and hazardous territory where humans cannot. A robot that is both effective and advantageous in terms of use, efficiency, design and cost. It is instrumental in environment and situation analysis during the unfortunate case of an occurrence of a disaster. The bot will give camera view to the user, which is normal view. The control of the bot is manual and the robot will give many details to the user like temperature and gas sensor etc. The orientation of the cameras can be set by the two servo motors which control horizontal and vertical positions respectively . The controller used in the robot is Raspberry Pi 3B. A webpage will be developed where a user can watch the processed video and can also control the bot through keyboard and mouse.

**LIST OF TABLES**

**TABLE TITLE PAGE**

**NO NO**

4.1 Components of the Project 16

4.2 Characteristics of Raspberry Pi Camera Module V2 17

4.3 Gas Concentrations 19

4.4 Characteristics of temperature sensor 21

5.1 Features and specifications of Raspberry Pi 24

8.1 Truth table for working of the robot 37

**LIST OF FIGURES**

**FIGURE TITLE PAGE**

**NO NO**

3.1 Block Diagram of rescue robot 14

4.1 Working positions of servo motor 18

4.2 a) MQ309A Operating conditions & output signal

b) Temperature dependency characteristics

c) Response of sensor resistance and temperature 20

5.1 Raspberry Pi 3 Model B 23

6.1 Video streaming Architecture 26

6.2 a) Terminal view for VLC

b) Opening stream in VLC

c) Streamed video in VLC 27

6.3 HTTP live streaming architecture 28

7.1 Wiring Pi showing status of GPIO pins 29

7.2 Web control of robot movement 30

7.3 Creating PWM signal for Servo control 32

7.4 Streaming of video in MJPG Streamer 33

7.5 Web Page Created 34

8.1 Operation of H-Bridge circuit 36

10.1 Darkice configuration 40

10..2 Recording audio 42

**TABLE OF CONTENTS**

**CHAPTER TITLE PAGE**

**NO NO**

1 INTRODUCTION 11

1.1 Different types of rescue Bots 11

1.2 Overview of Project 12

2 LITERARY REVIEW 13

3 BLOCK DIAGRAM OF PROJECT 14

4 COMPONENTS 16 4.1 Camera modules 17

4.2 Motors 17

4.3 Servo motors 18

4.4 Sensors 19

4.4.1 Gas Sensor 19

4.4.2 Temperature Sensor 21

4.4.3 Ultrasonic Sensor 21

4.5 Analog to Digital Converter 22

5 RASPBERRY PI 23

5.1 Introduction to pi 23

5.2 Operating system 23

5.3 Working of Raspberry pi 23

5.4 Features and specifications 24

6 VIDEO TRANSMISSION 25

6.1 Overview of video streaming &

Communication Applications 25

6.1.1 Video streaming by HTTP protocol 26

6.2 HTTP Live Streaming Architecture on Webpage 28

7 CREATING WEBPAGE FOR THE CONTROL OF BOT 29

7.1 Wiring Pi 29

7.2 Common Gateway Interface 30

7.3 Servo Blaster 31

7.4 MJPG Streamer 32

7.5 HTML coding for creating webpage 33

8 INTERFACING WITH MOTORS 35

8.1 Introduction 35

8.2 GPIO pin usage 35

8.3 Working of Motor Driver 35

8.4 Working of Motors 37

9 INTERFACING SERVO MOTORS 38

9.1 Introduction 38

9.2 GPIO pin usage 38

9.3 Working of Servo Motors 38

10 AUDIO STREAMING 39

10.1 Introduction 39

10.2 Darkice 39

10.3 Icecast 2.0 40

10.4 Interfacing microphone 41

10.5 Recording Audio 41

REFERENCES 43

ANNEXURE 1 44

Chapter 1

INTRODUCTION

Saving victims of natural and human made disaster requires an organized rescue planning from a government. The aim is to get out to the disaster areas, find the victims and help them as fast as possible. Disasters come with many obstacles for the rescue team that makes it hard for them to reach the victims, for example rainstorms, collapsed buildings, obstructions and dangerous substances. The rescue team must fast and securely find information of the disaster areas, a task that is both hard and dangerous Rescue robots, that is a type of field robots, can serve as appreciated tools for human teams under disasters. They can reach places between rubble and hazardous places that humans cannot, and effectively gather important information. The robots can also reach victims through narrow spaces and apply them with fluids and medication. To be useful tools the rescue robots must be smart and dynamic so they do not become obstacles for the rescue team. Because of the unstructured and dynamic environment that occurs during disasters, the robots are nowadays in a degree teleoperated, which demands that the robots have a good communication with the rescue team.

1.1 DIFFERENT TYPES OF RESCUE ROBOTS

There are mainly four types of rescue robots that can be categorized like this:

Unmanned Ground Vehicles (UGVs):These robots work on the ground and can help rescuers to find and interact with trapped or hurt victims, in areas were human cannot enter.

Unmanned Aerial Vehicles (UAVs): These robots can easily transport medical treatments to victims and can give the responders a bird view of the situation.

Unmanned Underwater Vehicles (UUVs): These robots can search through water and find victims, dangerous subject or substance. Cannot use GPS signals yet.

Unmanned Surface Vehicles (USVs): These robots work on the water surface, and can help rescuers to locate and bring the right equipment to the victims.

The rescue robots can further be divided in three groups, depending on the model size. When choosing the size, it is important to know what the robot should be capable of and how soon after a disaster it might be used. The sizes are Man-packable, Man-portable and Maxi-sized.

Man-packable is typically small and they can easily be carried to the disasters hot zone immediately. Man-portable is little bigger in size and may need two persons, or a vehicle to be carried. These robots often need to wait before the path is curved up. Maxi-sized robots are the biggest type and they need trolleys for transportation, and they can be used in the hot zone and directly on the rubble.

1.2 OVERVIEW OF THE PROJECT:

The robot that we designed is a four wheeled UGV, used for finding the people under the collapsed buildings and interact with them through audio transmission. The bot will sense the details like temperature, light, gas, movement, audio and video and this data can be provided for the rescue team.

The controller used for the robot is Raspberry Pi 3B and the Pycamera will capture the view of the objects. Audio taken by the microphone is streamed through the server. The raspberry pi will be controlled from the connected laptop/computer. The data from the raspberry pi can also be seen on the screen of the laptop. The camera’s spatial orientation will be controlled by two servo motors which controls both the horizontal and vertical placement. The raspberry pi is connected to the laptop by common Wi-Fi. The inputs from the system is transmitted through Wi-Fi as commands. The Raspberry Pi is coded to act accordingly to the commands. There will be a flash light available which can be controlled by the raspberry pi when needed.The whole transmission of data between laptop and the robot will happen through webserver.

Chapter 2

LITERARY REVIEW

The different type of sensors are needed for the project. The various sensors what we used like PIR, temperature and gas sensors have various specifications. And these specifications are studied from the journal proposed by A. Joshi, C. Nagarjun and R. Srinivas[1].

The main reliability of the robot depends on the mechanical design of the robot. The various methods for the movement of the bot and the torque selection of the motors used are designed by taking reference of journal proposed by Zhang Guowei[2].

The interfacing of temperature sensor with raspberry pi is done by Q. Luo and M. Xie in their temperature and humidity detection project which is referred from reference[4]

By interfacing PIR sensor with Raspberry Pi S. Mathur, B. Subramanian, S. Jain, K. Choudhary and D. Prabha had developed a human detector and counter by using three PIR’s and this process is referred from the reference[5]

Smart gas Detecting System in the areas of natural gas storage is proposed by A. Ilie and C. Vaccaro, where the gas sensor is interfaced with raspberry pi and this is referred from reference[6]

The interfacing of the Pycamera with the raspberry pi is studied from the reference [8]

As the orientation of the camera will be controlled by the servo motors, the controlling of servo motors with the Raspberry pi using PWM method is studied from the reference[9]

Different types of Rescue bots and their working mechanisms were studied from the reference[10]

The basic information about Raspberry Pi 3b and its interfacing with accessories is learnt from the reference[11]

Chapter 3

BLOCK DIAGRAM OF A RESCUE ROBOT

The different blocks or components needed for the project are shown in the below fig1

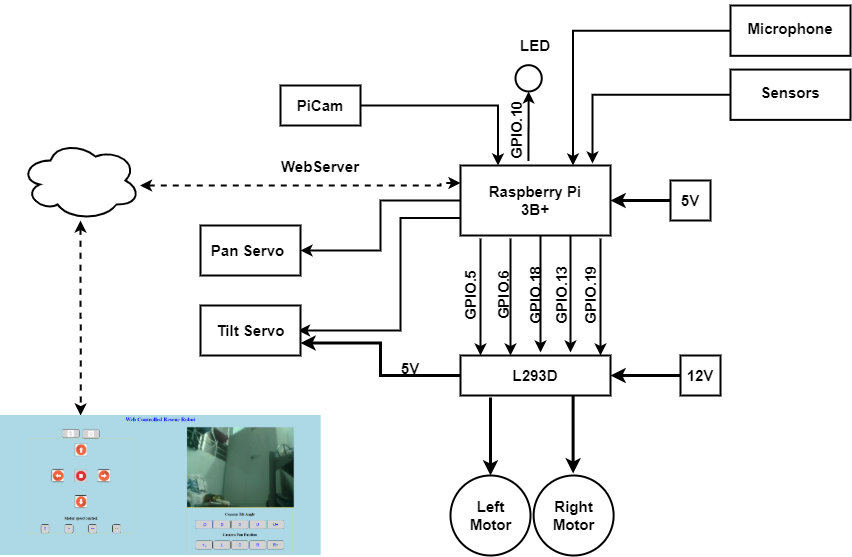


Fig 3.1 Block diagram of Rescue Bot

Raspberry Pi is the controller in this project. This will serve as web server for the communication between the robot and the user. The signals from raspberry pi is given to L298D motor drive which runs the four motors which can be modelled as Left Motor and a Right Motor. A 12V LiPo Battery is connected to the motor driver through which power will be channeled to the two motors. The enable signal from the Raspberry pi will control the speed of the motors. This speed control is based on the duty cycle of the PWM signal generated by the controller. The Pi is powered by a 5V Power source. The environment in which the bot is there can be observed by the user through the Picam. The video from this camera will be streamed directly to the user wirelessly through webserver. The Orientation of the Picam is set by the two servo motors connected to the Pi. The servo motor control signals are PWM signals generated by the controller. The duty cycle of the PWM signals will decide the angle at which the servo should be placed. One Led is also controlled by the Controller which helpfulin dark premises. A USB microphone is connected to the Raspberry Pi, through which the sounds of the environment can be captured by the robot and send it to the user. There are sensors mounted on the robot like gas sensor, ultrasonic sensor, temperature sensor. These sensor values will be transmitted to the user. The functioning of the bot is done by using the ‘.cgi’ scripts. By using HTML coding we will create a web page where the controls for the robot is given through as buttons. The video, audio and sensor values will be visible directly on the webpage to the user.

Chapter 4

COMPONENTS OF THE PROJECT

As explained in the overview of the project, the project requires the following list of components as specified in table 1

Table 4.1 Components of the project

|  |  |  |
| --- | --- | --- |
| COMPONENTS | MODEL SPECIFICATION | QUANTITY |
| Raspberry Pi | Model 3B | 1 |
| Raspberry Pi Camera | Module V2 | 1 |
| Servo motors |  | 2 |
| High Torque Motors |  | 4 |
| Gas sensor | MQ309 | 1 |
| Temperature sensor | LM35 | 1 |
| LED light |  | 1 |
| Battery | LiPo (12V) | 1 or 2 |
| USB microphone for Raspberry Pi |  | 1 |
| Motor driver | L298N | 1 |
| ADC | ADC0808 | 1 |

4.1 CAMERA MODULE:

The bot needs normal camera view for the movement of the robot. So Raspberry Pi Camera Module V2 (Pycam) is used. The raspberry Pi camera board plugs directly into the CSI connecter on the Raspberry Pi. It’s able to deliver a crystal clear 5MP resolution image or 1080p video recording at 30fps.The module attaches to the Raspberry pi through 15- pin ribbon cable, to the dedicated 15-pin camera serial interface(CSI) and the CSI bus is capable of extremely high data rates.

Raspberry pi camera module V2:

Table 4.2 Characteristics of Raspberry Pi Camera Module V2

|  |  |
| --- | --- |
| **PROPERTY** | **VALUE** |
| Resolution | 5MP |
| Interface Type | CSI (Camera Serial Interface) |
| Supported Video Formats | 1080p @ 30fps  720p @ 60fps  640x480p @ 90fps |

Fully Compatible with Raspberry Pi 3 Model B.

4.2 MOTORS:

These motorsare used for high torque medium speed applications. We use these motors for the movement of the bot.

CALCULATIONS FOR MOTOR DRIVER:

we consider, mass of the bot=5kg(Max) (5 kg/4 wheels=1.25kg on each wheel)

radius of wheel=4.5cm

* Torque= (Mass\*acceleration due to gravity\*radius)/100 = mgr/100

=5\*9.8\*4.5/100

=2.20725 Nm.

* Speed(ꞷ)= 2\*3.14\*N/60 {2\*pi\*N/60}

we want N=300 rpm

ꞷ=2\*3.14\*300/60

=31.5 radian/sec.

* Power=T\*ꞷ

=2.20725\*31.5

=69.528W.(approx. 70W)

If 12V battery is used, it draws 1.44A current per motor. Hence the maximum total current that motor driver should withstand is 1.5\*4=6A (max).

According to the requirements, Dual DC Motor Driver (20A) of model RKI-1341 motor driver is available in the market having the capacity of 6V to 18V and 20A operating current.

CALCULATIONS FOR MOTOR TORQUE:

Torque in kg-cm available in market:

since Mass=5kg(max)

radius=4.5cm

Torque = mass\*acceleration due to gravity\*radius = m\*g\*r = 5\*9.81\*4.5 =2.20725 N-m.

Torque/motor=220.75/4= 0.55125 N-m = 22.5 Kg-cm.

Speed of the motor is 300rpm.

For this specifications, the model available in market is: ET-CGM95A-12100

whose Torque is 30kgcm.

4.3 SERVO MOTOR:

Servos are small motors with embedded control circuitry that can turn up to 180 degrees. You control the servo by turning one of the GPIO pins on and off at an incredibly fast rate. The length of the pulses (also known as pulse width) is what controls which direction the servo is pointing in. The servo shaft can be positioned to specific angular positions by sending the servo a coded signal. As long as the coded signal exists on the input line, the servo will maintain the angular position of the shaft.

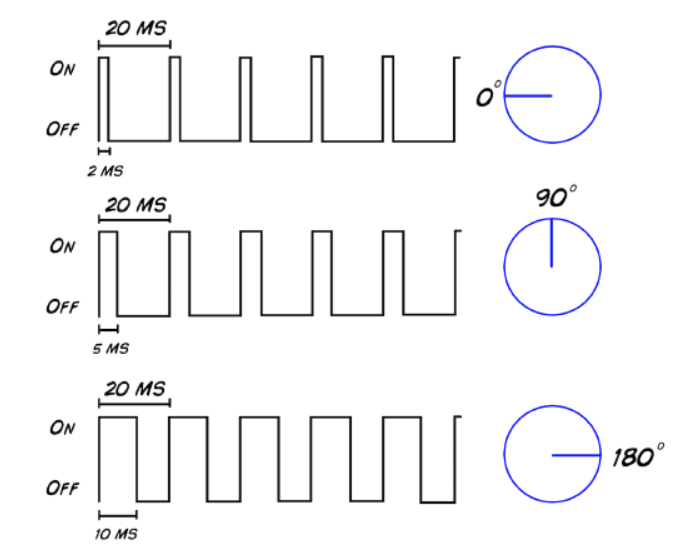


Fig 4.1Working positions of the servomotor

4.4 SENSORS:

4.4.1 GAS SENSOR:

The initial aim in a disaster scenario rescue is the localization of victims and the detection of dangerous situations such as gas leaks etc. The rescuers were not able to immediately determine the environmental dangers that the gas concentrations had on victims and rescuers. In past search and rescue scenarios, rescuers requested that air sensory meters be attached to the robots. These gas meters were difficult to attach to the robots and gas concentrations in the environment were only known once the robots returned from search mission. In addition to search and rescue operations, preventive operation should be taken depending on the gas levels. Underground mines, especially in the coal industry, are characterized by tough working conditions and constitutes hazardous environment. These environments involve accidents resulting from variety of causes, including sudden rise in toxicants such as carbon monoxide (CO), dangerous flammable gases such as methane (CH4) leading to insufficient oxygen levels. Semiconductor based gas sensor technologies are suitable for such environments.

Most sensors give an output of gas concentration (in ppm) that correlates with an analog value. This data is required to be interpreted for further analysis to be made. There are three concentrations that are considered to be important. These values are shown in table. With this information the rescuers could determine whether to risk their lives to enter an environment.

Table 4.3. Gas concentrations

|  |  |  |  |
| --- | --- | --- | --- |
| ***GAS*** | ***Unsafe Concentration(ppm)*** | ***Deadly Concentration(ppm)*** | ***Flammable (ppm)*** |
| *CO2* | 5,000 | 40,000 | Non flammable |
| *CO* | 25 | 1,000 | 1,20,000 |
| *H2S* | 10 | 100 | 43,000 |
| *Methane* | 1,000 | 5,000 | 50,000 |
| *Oxygen* | No concentration | No concentration | Non flammable |

In combustible gases, the use of natural gas has been increasing and methane detection is important to prevent explosions caused by gas leaks. CO is produced by incomplete combustion in boilers or other gas /oil combustion appliances, and causes many CO poisoning.

4.4.1.1 CONSTRUCTIVE PROPERTIES OF MQ309A/SB-95 SENSOR

Gas sensitive semiconductor material is a mini bead type and a heater coil and electrode wire is embedded in the element. The sensing element is installed in the metal housing which uses double stainless steel mesh in the path of gas flow. This sensor is placed in an external housing which contains an active charcoal filter.

4.4.1.2 GAS ANALYSIS METHOD

In the case of semiconductor the measurement is based on the variation of the induced electrical conductivity, when the gas is absorbed, on the surface of a metallic oxide. By controlling a heated filament it is possible to raise the temperature of the sensitive element to a desired value. The sensitivity of the SnO2 to various gases is dependent on the temperature. This is chosen so that the maximum sensitivity for the methane and carbon monoxide gases is obtained. For methane the maximum sensitivity of the sensor is at a temperature of 400°C and for the CO at a temperature of 90°C.

For simultaneous detection of the two gases we have to establish a thermostatic cycle of the sensitive element at the two temperature values and measure the resistivity of the sensing element during each gas period. The filament powered by a 0.9V voltage for 3 seconds can heat up the filament to 400°C and for 7 seconds at 0.2V will heat the sensor up to 90°C.



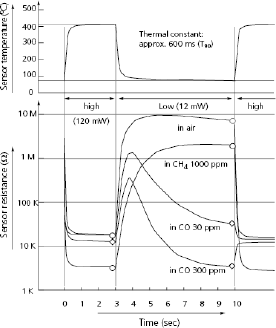
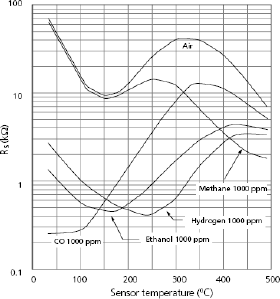


Fig 4.2 a) MQ309A Operating conditions and output signal b) Temperature dependency characteristics c) Response of sensor resistance and temperature.

4.4.2 TEMPERATURE SENSOR:

Temperature sensors are used to measure the temperature of the region surrounding it. It has various applications in the region where temperature monitoring is important, like monitoring of CPU temperature, monitoring of battery temperature, etc.

There are many variants of temperature sensors and among them the most commonly used one is the LM35 sensor, since the LM35 series provide better accuracy compared to other sensors. It is an IC whose output voltage is proportional to the surrounding temperature. This sensor can be operated with both single power supply and dual power supplies.

Table 4.4 Characteristics of temperature sensor

|  |  |
| --- | --- |
| **PROPERTY** | **VALUE** |
| Calibration | In ⁰Celsius |
| Scaling Factor | +10mV/⁰C |
| Accuracy | 0.5⁰C(at 25⁰C) |
| Operating voltage | 4V to 30V |
| Operating current | 60µA |
| Output impedance | 0.1Ω |

4.4.3 ULTRASONIC SENSOR

The ultrasonic sensor that is used in this project is HC-SR04. This will give a pulse when the triggering signal is given to it. The GPIO pins of number 7 is set as “Trigger”, and number 40 is set as “Echo”. A signal of 10µs is developed from trigger and is given to raspberry pi. The HC-SR04 will give a digital high when it got a triggering pulse and will generate a 40kHz pulse. When it again senses back the 40kHz signal, the digital high signal from sensor will go low. The on time of the cycle is taken by the pulse to travel two times the distance between sensor and surface. The start time and stop time is calculated by “time.time()” function. This difference will give the on time. As the speed of sound is 340m/s and the distand covered is twice between the sensor and surface. The distance is calculated by

Distance=(stop-start)\*17000 cms

The ultrasonic sensor will give a digital high of 5V, where as the GPIo pins of raspberry pi will work at 3.3V. So a voltage divider is used to give only a digital high of 3.3V is given to the raspberry pi.

4.5 ANALOG TO DIGITAL CONVERTER

Gas sensor return analog signals which cannot be easily read by the raspberry pi. But we can use ADC which can be read out via IC2 bus. ADS1015 is a 12- bit ADC with 4 channels. It has a programmable logic gain of 2/3x to 16x.so you can amplify small signals and read them with high precision.

Logic level converter:

Raspberry pi pins work with 3.3V logic level and are not 5V tolerant. If you apply 5V to a GPIO pins, then pi gets damaged.

Chapter 5

RASPBERRY PI

5.1. INTRODUCTION TO PI:

Raspberry Pi is a credit card–sized computer that plugs into PC, a keyboard and mouse.

It is a capable little device that enables to explore computing, and to help program in languages like Python. Raspberry Pi has the ability to interact with the outside world and has been used in a wide array of digital projects.

5.2 OPERATING SYSTEM:

Raspbian OS is used for the application. It employs the Open box stacking window manager and the Pi Improved XWindows Environment Lightweight coupled with a number of pre-installed software which includes Mine craft Pi, Java, Mathematical, and Chromium. Raspbian is the Raspberry foundation’s official supported OS and is capable of accomplishing any task you throw at it.

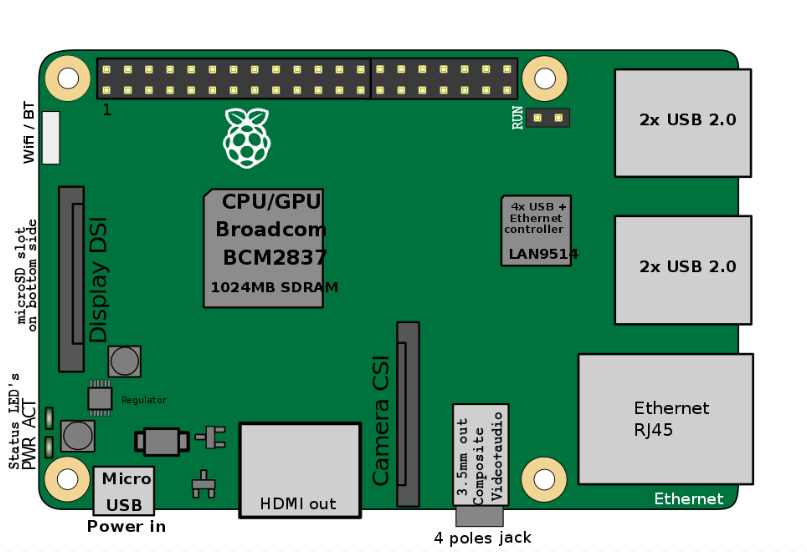


Fig 5.1 Raspberry Pi 3 MODEL B

5.3 WORKING OF RASPBERRY PI:

An SD card inserted into the slot on the board acts as the hard drive for the Raspberry Pi. It is powered by USB and the video output can be hooked up to RCA TV set, a more modern monitor, or even a TV using the HDMI port. It also has an extremely low power consumption of about 3 watts. For this particular application Raspberry Pi 3b has been used.

5.4 FEATURES AND SPECIFICATIONS:

Table5.1Features and specifications of Raspberry pi

|  |  |
| --- | --- |
| FEATURES | SPECIFICATION |
| SoC | BCM2837 |
| CPU | Quad Cortex A53 @ 1.2GHz |
| Instruction Set | ARMv8-A |
| GPU | 400MHz VideoCore IV |
| RAM | 1GB SDRAM |
| Storage | micro-SD |
| Ethernet | 10/100 |
| Wireless Bluetooth 4.0 | 802.11n |
| Video Output | HDMI/Composite |
| Audio Output | HDMI/Composite |
| GPIO | 40 |

Chapter 6

VIDEO TRANSMISSION

6.1 OVERVIEW OF VIDEO STREAMING AND COMMUNICATION APPLICATIONS

1. Point to point, multi cast, broad cast communication

The most popular form of video communication is one-to-many communication or broadcast communication, e.g. broadcast television. Another common form of communication is point-to-point or one-to-one communication, e.g. videophone and unicast streaming over the Internet. Another form of communication with properties that lie between point-to-point and broadcast is multicast. Multicast is a one-to-many communication, but it is not as in broadcast. E.g. IP-Multicast over the Internet.

1. Real-time encoding versus pre-encoded video

Video may be captured and encoded for real-time communication, or it may be pre-encoded and stored for later viewing. Interactive applications are one example of applications which require real time encoding, e.g. videophone, video conferencing. In many applications video content is pre-encoded and stored for later viewing.

1. Static versus dynamic channels

Video communication system design varies significantly if the characteristics of the communication channel, such as bandwidth, delay and loss , are static or dynamic(time-varying). Examples of static channels include ISDN (which provide fixed bit rate and delay), and of dynamic channels include communication over wireless channels or over the Internet.

Pycamera is interfaced to the raspberry pi at the Camera Serial Interface(CSI) using the flex cable. The camera is enabled in the raspberry pi configurations. Then there are two different ways to transmit a video.

6.1.1 VIDEO STREAMING ON VLC THROUGH HTTP PROTOCOL

Video data is captured from Pycamera, compressed into MPEG format, transferred by the HTTP1 protocol under the control of Raspberry pi, then the monitor client will receive the compressed data frames to restructure and recompose video images.

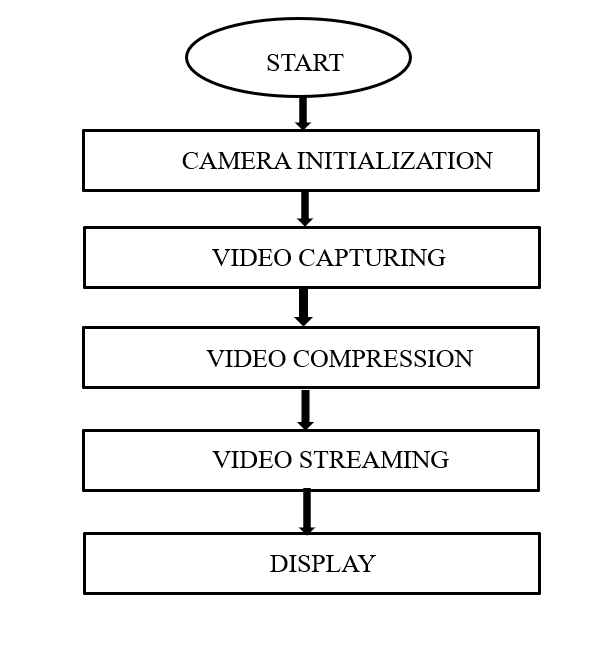


Fig 6.1 Video streaming architecture

The application related shell script is made to run in the terminal which will execute the application in board resulting video streaming on VLC using HTTP protocol, entering a static IP address by user in any wireless device which is in local network can view the remote location.This method is used to transmit the video through the IP address and destination number which is coded in the script. First VLC should be installed in raspberry pi by command:

* $sudo apt-get install vlc

After this the command for invoking the camera is used:

* $raspivid -o - -t 0 -hf -vf -w 800 -h 400 -fps 24 | cvlc -vvvstream:///dev/stdin –sout’#standard{access=http,mux=ts,dst=:8160}’ :demux=h264

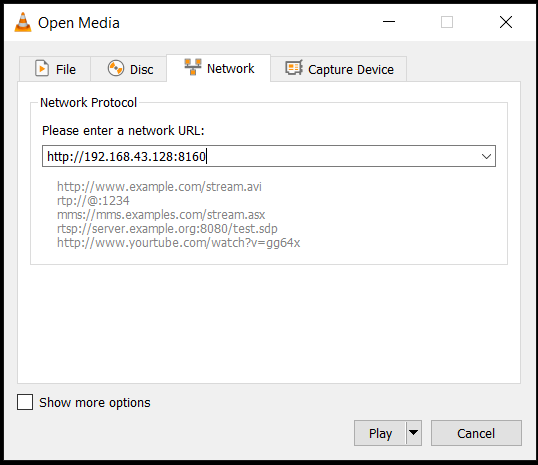
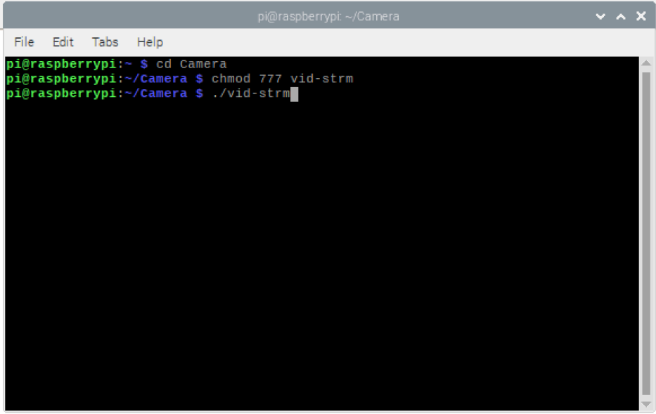
This code has all the parameters for the video which is to be transmitted like, delay,horizontal flip, vertical flip, width, height, frames per second,streaming destination etc. Instead of typing this much long code every time, we can store this command in a file called “vid-strm” and can run that file by typing the command:

* $chmod 777 vid-strm #*for giving permission to access the file*
* $./vid-strm

Then open VLC in the system in which video has to be transmitted and open Network Stream. Then type the IP address and the destination code given in the terminal.

<http://192.168.0.24:8160>

Then the video will be transmitted based on the above specifications but a delay of 5 to 6 seconds is observed, which is not applicable for the real time applications. Therefore this method is not suitable for this project.



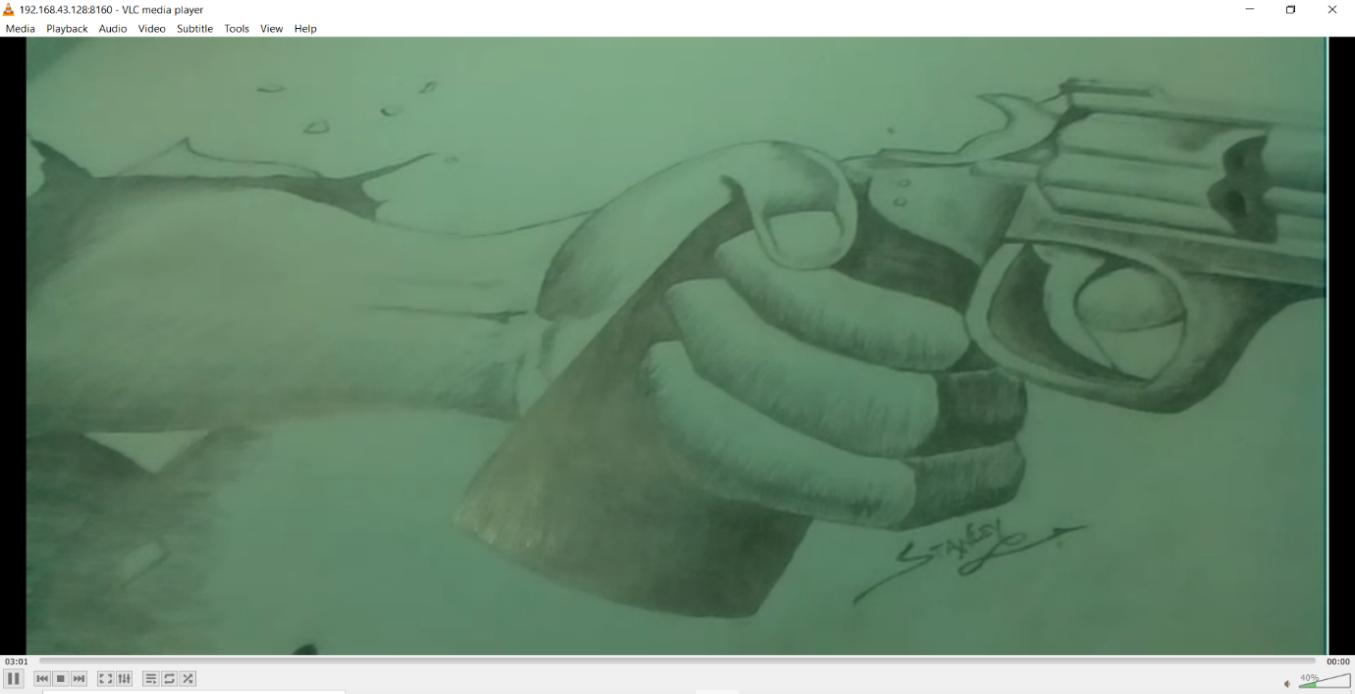


Fig 6.2 a) Terminal view for VLC b) Opening stream in VLC c) Streamed video in VLC

6.2 HTTP LIVE STREAMING ARCHITECTIRE ON WEB PAGE

HTTP live streaming consists of three parts: server component , distribution component , client.

A) Server component

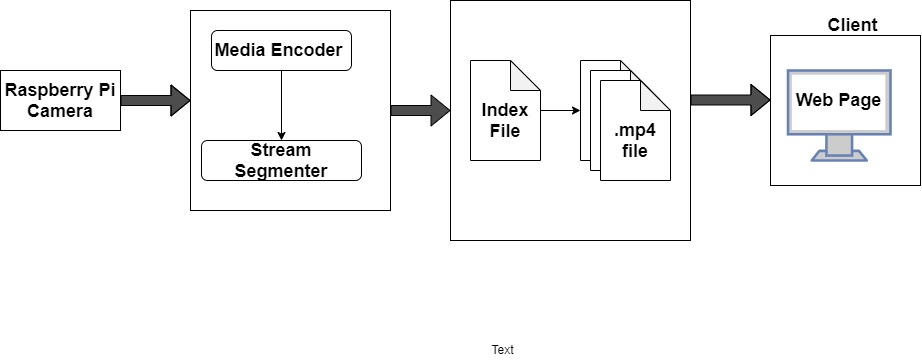
The server component is responsible for taking video input from the raspberry pi camera and encoding them digitally. It compresses the video files in h.264 or mp4 format and is encapsulated by MPEG-2 transport stream. Later the encapsulated files are divided into fragments of equal length. In order to reassemble the media files in acurrent sequence an index file is created .

B) Distribution component

The distribution system delivers the media files and index files to the client over HTTP. To deploy HTTP live streaming , you need to create an HTML page for browsers.

C)Client

The index file and media files are fetched using the URL of the streamer . Based on the sequence in the index file the media files are reassembled and streamed to the user.

Fig 6.3 HTTP live streaming Architecture

Chapter 7

CREATING WEBPAGE FOR THE CONTROL OF BOT

7.1 WIRING PI

Wiring Pi is a GPIO access library written in C for th Raspberry Pi. It is very easy to use and simplify a lot in any project involving RPi and electronics. This library is used to read and write the pins and even use it to control them from shell scripts. It can be used in scripts to manipulate the GPIO pins – set outputs and read inputs. It’s even possible to write entire programs just using the GPIO commands in shell scripts.

To set up the pin:

gpio -g mode 10 out // for setting the pin as output

gpio -g write 10 1 // for setting the pin as digital high

gpio -g mode 18 pwm // for setting the pin in pwm mode

gpio pwm 1 XXX [between 0 and 1023] //for setting the duty cycle

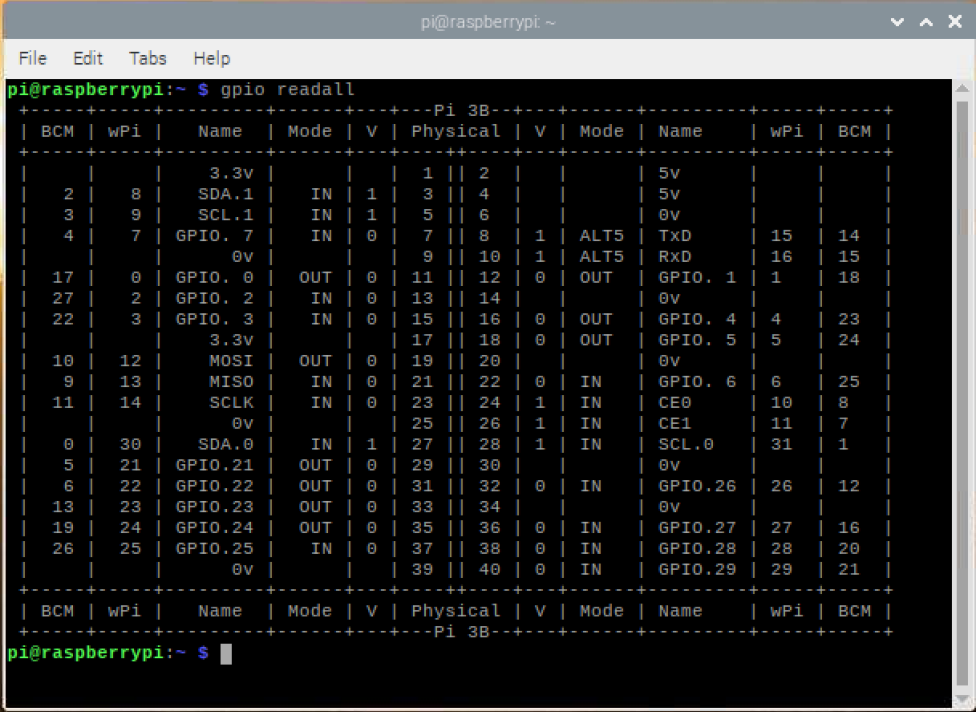


Fig. 7.1 Wiring Pi showing status of GPIO pins

7.2 COMMON GATEWAY INTERFACE

It is a interface for webservers to execute programs that execute like console applications.

CGI scripts execute at a time when a request is made.

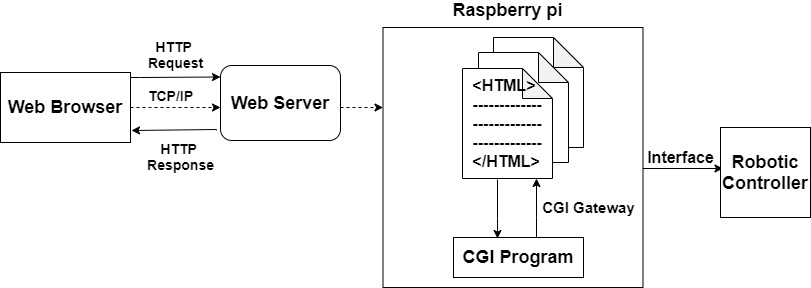


Fig 7.2. Web Control of robot movement

Every CGI script file is essentially plain text. When a text file is attempted to be executed, shells will parse through them for clues as to whether they’re scripts or not, and how to handle everything properly. Because of this, there are few guidelines.

* Every script should begin with “#!/bin/bash” {The Hash- Bang hack}
* Every new line is a new command
* Comment lines start with a #
* Commands are surrounded by ()

A shell script for forward is created,

sudo nano forward.cgi

#!/bin/bash

gpio -g write 5 1

gpio -g write 6 0

gpio -g write 13 1

gpio -g write 19 0

The script should be given permission by using the command

sudo chmod 755 forward.cgi

For executing the script

sudo ./forward.cgi

There are scripts which are created for stop, reverse, left, right, lighton, lightoff, nospeed, fullspeed, regularspeed, lowspeed.

7.3 SERVO BLASTER

Servo blaster is a great library to be used to control servo motors. This is software for the RaspberryPi, which provides an interface to drive multiple servos via the GPIO pins. We control the servo positions by sending commands to the driver saying what pulse width a particular servo output should use. The driver maintains that pulse width until you send a new command requesting some other width. By default 8 pins are configured to drive the servo. Servos typically need an active high pulse of somewhere between 0.5ms and 2.5ms, where the pulse width controls the position of the servo. The pulse should be repeated approximately every 20ms, although pulse frequency is not critical. The pulse width is critical, as that translates directly to the servo position.

sudo ./servod –p1pins=11,16

This code is used to configure the required pins as output.

echo P1-11=40% >/dev/servoblaster

echo P1-16=60% >/dev/servoblaster

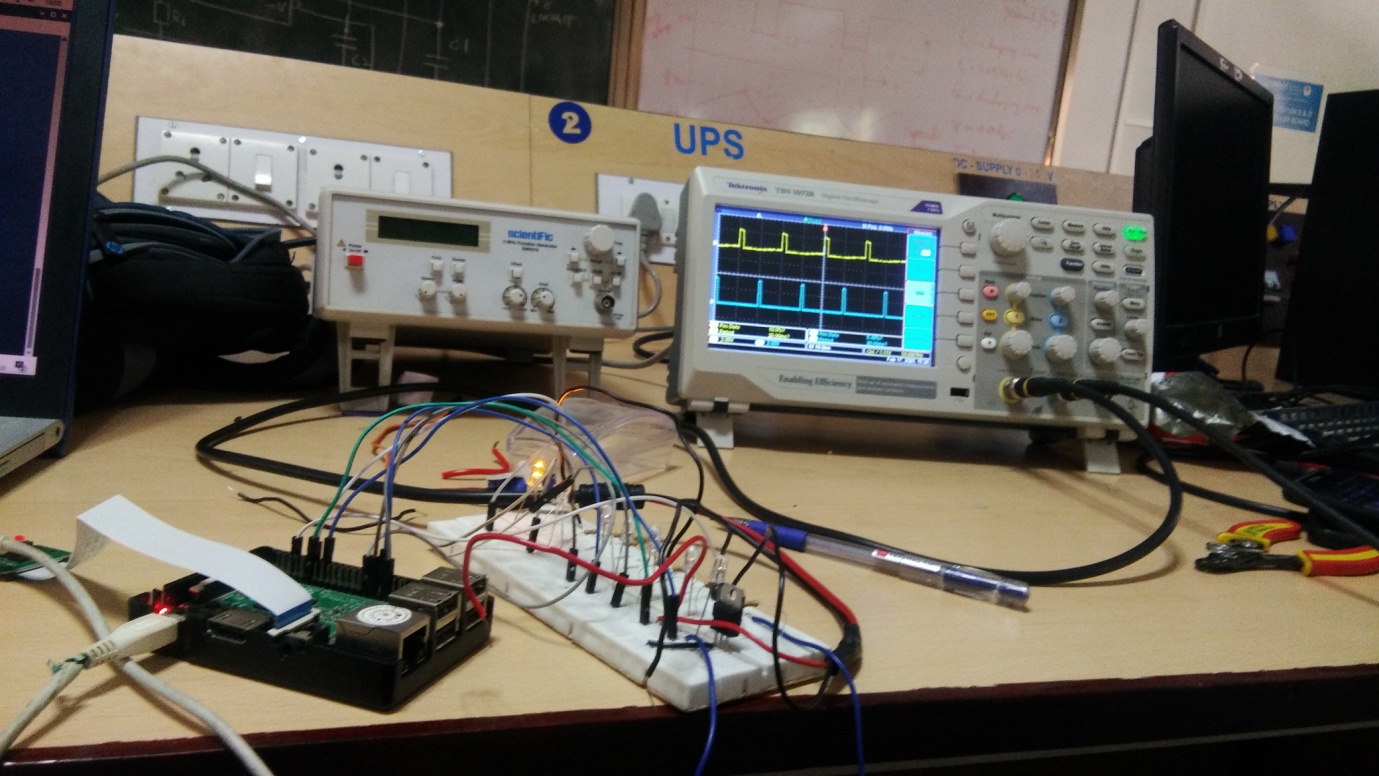


Fig 7.3 Creating PWM signals for servo control

These codes are used to set the duty cycle of the PWM signals which controls the position of the servomotor. The speed control scripts are also using the same method. There are five positions set for each of the servos i.e. pan servo and tilt servo. This gives a total of 25 different types of views.

1. leftpan.cgi ==> 90%
2. leftcenterpan.cgi ==> 76%
3. centerpan.cgi ==> 62%
4. rightcenterpan.cgi ==> 46%
5. rightpan.cgi ==> 30%
6. downtilt.cgi ==> 20%
7. downcentertilt.cgi ==> 40%
8. centertilt.cgi ==> 60%
9. upcentertilt.cgi ==> 75%
10. uptilt.cgi ==> 90%

7.4 MJPG STREAMER

This software is used to stream the video captured by the Pi cam and play it in the web server available at http://Ip Address:9000/stream.html

After the MJPG streamer is installed in RPi, a command should be run to start streaming.

LD\_LIBRARY\_PATH=/opt/mjpg-streamer//opt/mjpg-streamer/mjpg\_streamer-i "input\_raspicam.so -fps 15 -q 50 -x 640 -y 480 -vf -hf" -o "output\_http.so -p 9000 -w /opt/mjpg-streamer/www" &

The streaming is done successfully.

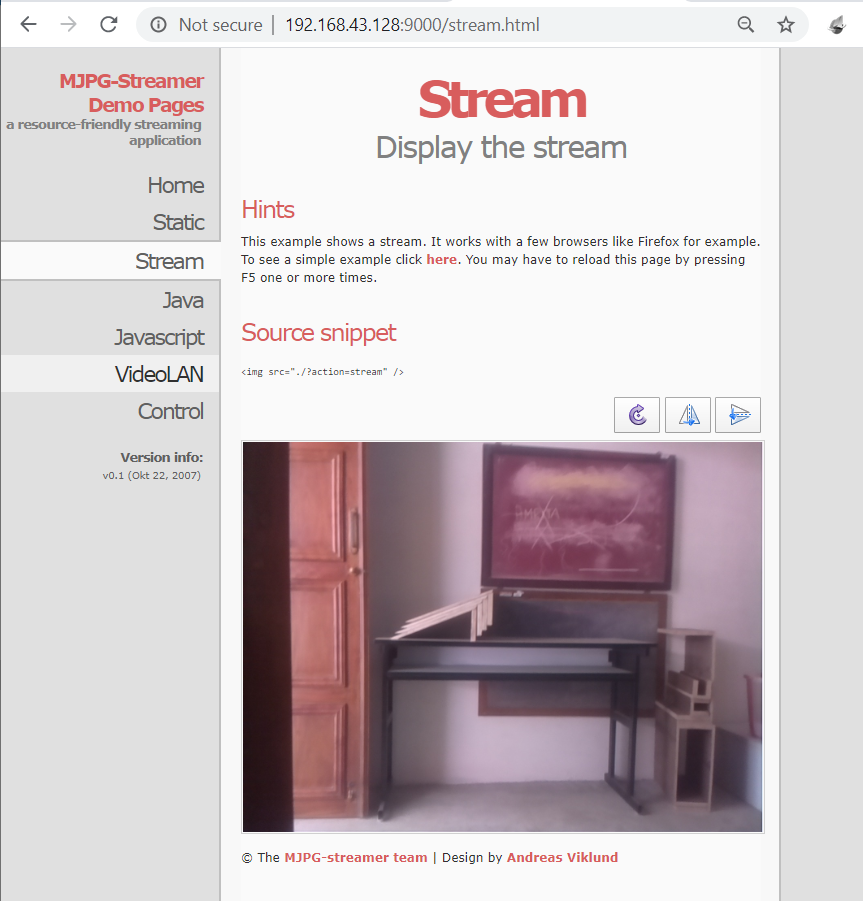


Fig 7.4 Streaming of video in MJPG Streamer

7.5 HTML CODING FOR CREATING WEBPAGE

The buttons are created by using <button> command and for each onclick the .cgi scripts in the RPi is called. The video frame in webpage is created by the command <iframe>, in which the url of streaming video is given here.

<iframe src="http://192.168.43.128:9000/javascript\_simple.html" frameborder="0"

align="middle" width="640" height="480" align="middle" scrolling="no"></iframe>

The height and width of the frame is variable and it is fixed 640\*480 here.

The buttons are created using the command

<button style="height: 50px; width: 100px" onclick="lighton()"><img style="height: 40px"src="/images/lighton.png"></button>

The HTML coding for webpage creation is stated in Annexure 1.

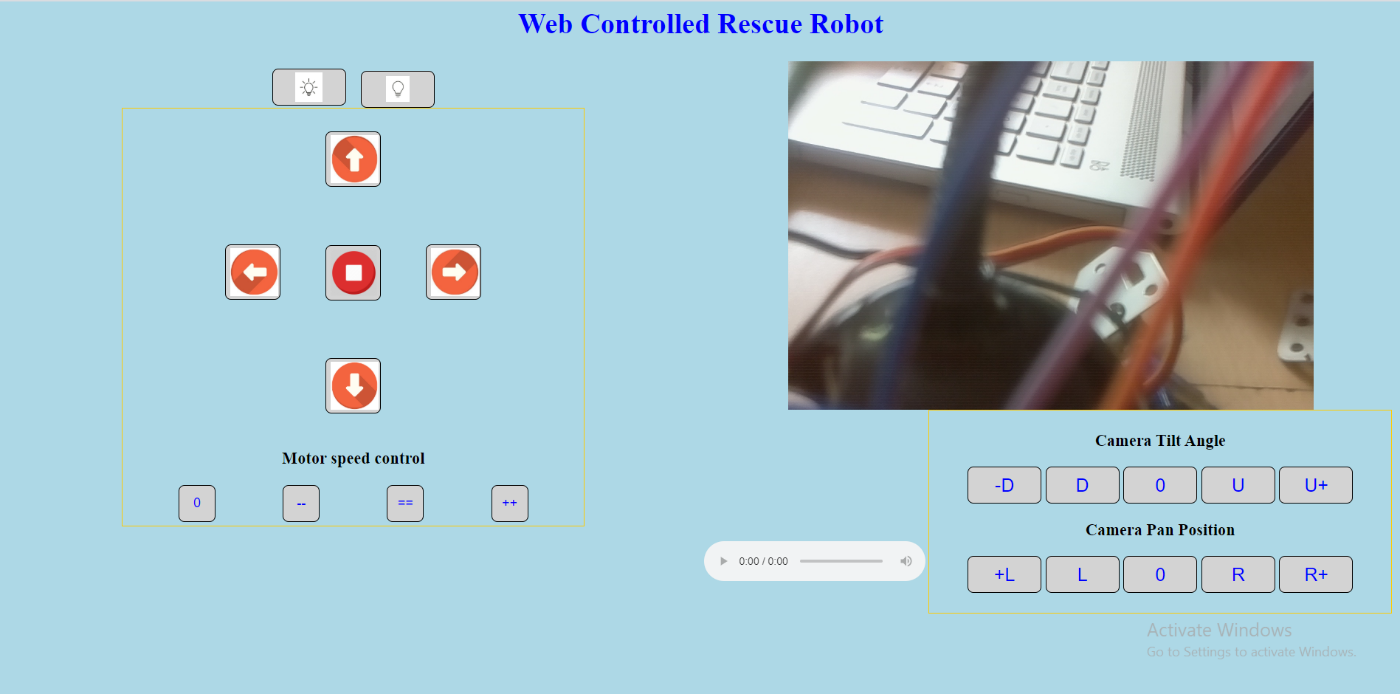


Fig 7.5. Web page created

Chapter 8

INTERFACING WITH MOTORS

8.1 INTRODUCTION

The motors of the bot can be modelled as the working of two motors. The motors in the right side of the bot will work as one motor. The left side can be configured as a single motor. The motors are connected to the motor drive and the controlling signals for the motor drive will be given from the raspberry pi.

8.2 GPIO PIN USAGE

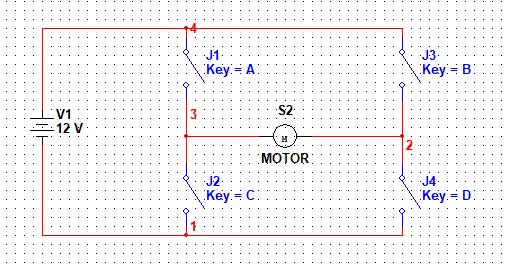
The GPIO pins used are configured as output.

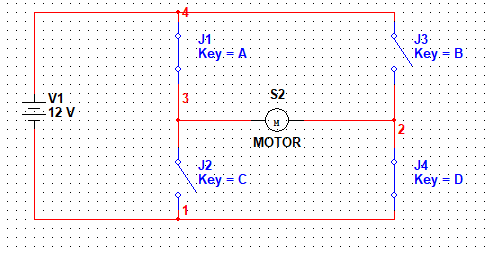
* Motor1A & 1B= GPIO pin 5 and 6
* Motor2A & 2B= GPIO pin 13 and 19

8.3 WORKING OF MOTOR DRIVER

You can drive a LED ON or OFF using a raspberry pi. There’s is no problem there because 20mA current is enough to LED. But you can’t drive loads that require much more current like a motor. This is where the L298N motor controller comes in.

The L298N motor controller follows the H-bridge configuration, which made easy to control the direction of rotation of a DC motor, The other benefit of using H-bridge is that you can provide a separate power supply to the motors. The schematic diagram of H-bride is shown in figure .





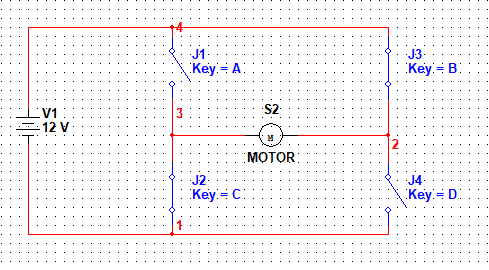


Fig 8.1 a) H-Bridge circuit, b)forward direction , c) reverse direction

Here, the motor rotates in the direction dictated by the switches, When S1 and S4 are ON, the left motor terminal is more positive than the right terminal, and the motor rotates in a certain direction. On the other hand, when S2 and S3 are ON, the right motor terminal is more positive than the left terminal, making the motor rotate in the other direction. Figure . Speed control is also possible with L298N motor driver. All you need is feed PWM signals to the motor enable pins. The speed of the motor will vary accordingly to the width of the pulses. The wider the pulses, the faster the motor rotates.

Table 8.1 Truth table for the working of the Robot

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| IN1 | IN2 | IN3 | IN4 | STATUS |
| 0 | 0 | 0 | 0 | STOP |
| 1 | 0 | 1 | 0 | FORWARD |
| 0 | 1 | 0 | 1 | BACKWARD |
| 1 | 0 | 0 | 1 | LEFT |
| 0 | 1 | 1 | 0 | RIGHT |

L298N motor driver IC has two input power pins Vss and Vs. Vs pin gets its power for driving the motors which can be 5 to 35V.Vss pin is used for driving the logic circuitry which can be 5 to 7V. And they both sink to the common ground GND.

Losses: The voltage drop of the L298N motor driver is about 2V. This is due to the internal voltage drop in the switching transistors in the H-bridge circuit.

8.4 WORKING OF MOTORS

Whenever the forward button is clicked in the webpage, it will call the forward.cgi script which has the wiring pi commands to directly control the gpio pins. The same procedure will happen if we click the left, right, reverse and stop buttons.

For example:

sudo nano forward.cgi

#!/bin/bash

gpio -g write 5 1

gpio -g write 6 0

gpio -g write 13 1

gpio -g write 19 0

Chapter 9

INTERFACING WITH SERVO MOTORS

9.1 INTRODUCTION

There are two servomotors used for the bot. These are used for the movement of camera both horizontally and vertically. The position of the servo motor shaft is controlled by controlling the duty cycle of the signal given to the servo.

9.2 GPIO PIN USAGE

There are only three pins for the servo motor. GPIO pin 11 is used for the generation of PWM signal for servo motor 1. GPIO pin 16 is used for the generation of PWM signal for servo motor 2.

* Power pin – 5V
* Ground
* Input – GPIO pin 11 and GPIO pin 16

9.3 WORKING OF SERVO MOTORS

Servo blaster is used to produce stable PWM signals for multiple servo motors simultaneously. The servo blaster command is kept in different .cgi scripts with different duty cycles. The camera position is controlled by two servos i.e. pan servo and tilt servo. Each servo is coded with five positions. This will give a total of 25 spatial orientations. The servo code is:

echo P1-11=40% >/dev/servoblaster

echo P1-16=60% >/dev/servoblaster

Chapter 10

AUDIO STREAMING

10.1 INTRODUCTION

There are many audio formats available at present. They are MP3, OGG, WAV etc. MP3 (formally MPEG-1 Audio Layer III or MPEG-2 Audio Layer III) is a coding format for digital audio. MP3 (or mp3) as a file format commonly designates files containing an elementary stream of MPEG-1 audio and video encoded data, without other complexities of the MP3 standard. The OGG Audio file format OGG is a free, open container format created under unrestricted software patents by the Xiph.Org Foundation (Ogg Vorbis). It allows users to stream and alter high quality digital multimedia files. WAV files are lossless uncompressed broadcast CD quality music files. Loops can be used to extend a full track or just by itself. WAV loops can also be easily processed with Flash for web animations. If you use Flash, the process is very easy. The streaming media which is used for transmission in this MP3 format.

10.2 DARKICE

DarkIce is a live audio streamer capable of recording audio from an audio interface (e.g., a sound card), encoding the audio, and sending it to a streaming server. This streamer will encode the audio recorded from USB microphone is converted to .mp3 format and send it to streaming server.

Darkice is installed in raspberry pi and a configuration file is created.

$ sudo nano darkice.cfg

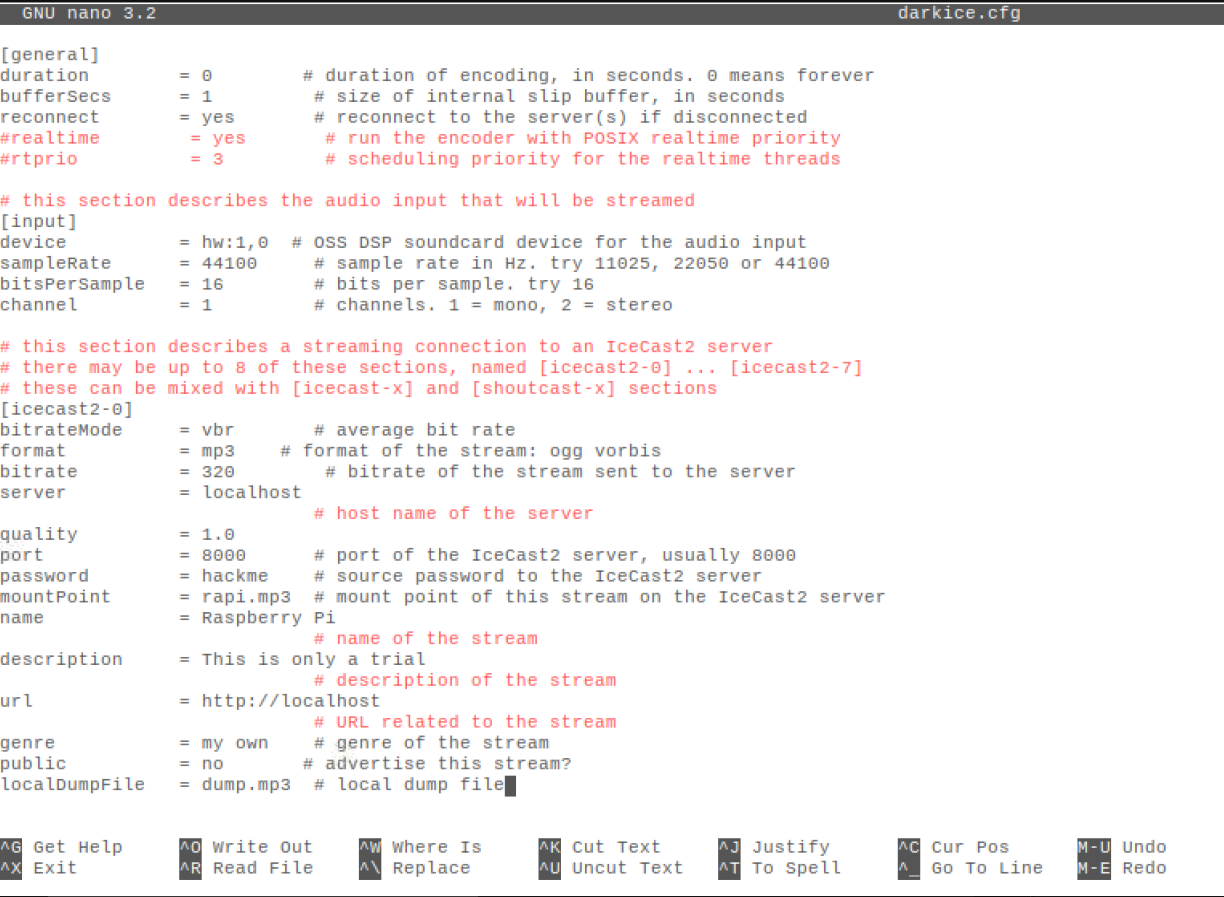


Fig 10.1 Darkice configuration

Duration will be infinite if its value is set to 0, until an external interrupt is occurred. Device is selected as (hw: 1,0) which can be found from ‘alsa.conf’ file in raspberry pi. The sample rate is the number of samples of audio recorded by the microphone per second which is set to ‘44100’ for better streaming quality. Bits per sample specifies the bit depth in every sample which is set to ‘16’. The format of audio is set to be mp3. Server name is localhost. The port for transmission of audio is 8000. For every streaming there will be a unique mountpoint which here is ‘rapi.mp3’.

10.3 ICECAST 2.0

Icecast is an audio/video streaming media server that can be used to create everything from Internet radio stations to privately running music playlists. The recorded audio from a USB microphone is encoded to mp3 format by Darkice and stream it to an Icecast server.

Icecast2 is installed in raspberry pi by the command

$ sudo apt-get install icecast2

The username and password which is given in the configuration file of darkice should be given here.

10.4 INTERFACING MICROPHONE

The microphone used here is external USB microphone. This microphone is selected by raspberry pi automatically, but sometimes it won’t recognize. Then the command ‘arecord -l’ will give the list of all recording devices connected to the raspberry pi. The USB device sound card number will be 1, and this should be set as default in ‘/usr/share/alsa/alsa.conf’.

Replace

defaults.ctl.card 0

defaults.pcm.card 0

with

defaults.ctl.card 1

defaults.pcm.card 1

and reboot to apply the appropriate changes.

10.5 RECORDING AUDIO

For live streaming of audio, first we need to start the server icecast first which can be done by the command:

$ /etc/init.d/icecast2 start

Then the required authentication should be given to server by entering the password and then start darkice to record the audio and encode it and stream it to icecast server by using the command.

$ darkice

This will stream the audio in the server with url “<http://192.168.43.128:8000>” and listening url “<http://localhost:8000/rapi.mp3>” where localhost is the server name and rapi.mp3 is the mountname given. This listening url is given in html code to stream audio in the webpage. And this will have a delay of nearly 10 seconds.

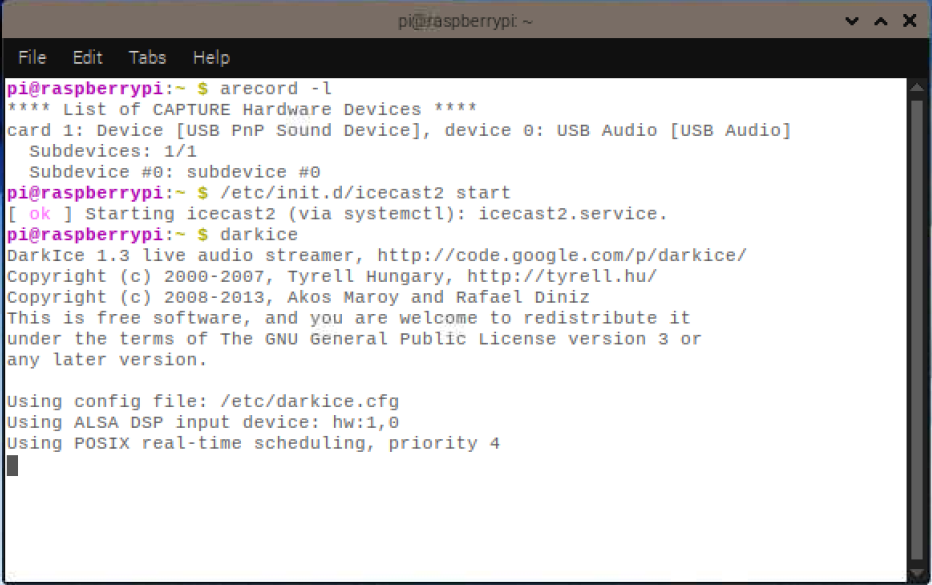


Fig 10.2 Recording Audio

REFERENCES:

[1]A. Joshi, C. Nagarjun and R. Srinivas, "The DRASB — Disaster Response And Surveillance Bot", *2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, 2017. Available: 10.1109/icecct.2017.8117959 [Accessed 4 August 2019].

[2]Zhang Guowei et al., "Development of robotic spreader for earthquake rescue", *2014 IEEE International Symposium on Safety, Security, and Rescue Robotics (2014)*, 2014. Available: 10.1109/ssrr.2014.7017679 [Accessed 4 August 2019].

[3]L. Anishchenko, A. Tataraidze, A. Bugaev and V. Razevig, "Automated long-term contactless temperature monitoring in animals via a thermographic camera", *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2017. Available: 10.1109/embc.2017.8037061 [Accessed 4 August 2019].

[4]Q. Luo and M. Xie, "Temperature and Humidity Detection System of Communication System Based on Raspberry Pi", *2018 International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS)*, 2018. Available: 10.1109/icitbs.2018.00062 [Accessed 20 August 2019].

[5] S. Mathur, B. Subramanian, S. Jain, K. Choudhary and D. Prabha, "Human detector and counter using raspberry Pi microcontroller", *2017 Innovations in Power and Advanced Computing Technologies (i-PACT)*, 2017. Available: 10.1109/ipact.2017.8244984 [Accessed 20 August 2019].

[6] A. Ilie and C. Vaccaro, "Design of a smart gas detection system in areas of natural gas storage", *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2017. Available: 10.1109/igarss.2017.8128365 [Accessed 20 August 2019].

[7]R. Stopforth and S. Davrajh, "Gas concentration and equation correlation: Of the Figaro sensors, used for dangerous environments", *2017 IEEE AFRICON*, 2017. Available: 10.1109/afrcon.2017.8095690 [Accessed 18 December 2019].

[8]E. Cordos et al., "Methane and Carbon Monoxide Gas Detection system based on semiconductor sensor", *2006 IEEE International Conference on Automation, Quality and Testing, Robotics*, 2006. Available: 10.1109/aqtr.2006.254633 [Accessed 18 December 2019].

[9] <https://docs.isy.liu.se/pub/VanHeden/DataSheets/amg8833.pdf>

[10][Getting started with the Camera Module - Introduction | Raspberry Pi Projects](https://projects.raspberrypi.org/en/projects/getting-started-with-picamera)

[11][Controlling a Servo from the Raspberry Pi · RPi Labs](https://rpi.science.uoit.ca/lab/servo/)

[12] <http://www.rroij.com/open-access/pdfdownload.php?download=rescue-robota-study.pdf&aid=43661>

[13] [http://makersportal.com/blog/2018/1/25/heat-mapping-with-a-64-pixel-infarared-sensor-and-raspberry-pi](http://makersportal.com/blog/2018/1/25/heat-mapping-with-a-64-pixel-infarared-sensor-and-raspberry-pi1)

[14] <http://www.raspberrypi.org.>

[15] http://www.pyimagesearch.com/

ANNEXURE 1

<html>

<head>

<style>

#leftbox {

float:left;

background:lightblue;

width:50%;

height:1000px;

}

#rightbox{

float:right;

background:light blue;

width:50%;

height:1000px;

}

</style>

</head>

<style>

body {background-color: lightblue}

h1 {color:blue}

button {

color: blue;

background:lightgrey;

border: 1px solid #000;

border-radius: 8px;

position: center;

}

</style>

<body>

<div id = "boxes">

<div style="text-align:center">

<h1> Web Controlled Rescue Robot </h1>

<div id = "leftbox">

<br>

<button style="height: 50px; width: 100px" onclick="lighton()" align="left"><img style="height: 40px"src="/images/lighton.jpg"></button>

<img hspace="1" style="padding-left: 10px">

<button style="height: 50px; width: 100px" onclick="lightoff()" align="left"><img style="height: 35px"src="/images/lightoff.jpg"></button>

<br>

<span style="display:inline-block;padding:5px;border:1px solid #fc0; font-size: 140%;font-weight: bold;">

<br>

<button style="height: 75px; width: 75px" onclick="forward()"><img style="height: 65px"src="/images/forward.jpg"></button>

<br><br><br><br>

<img hspace="10" style="padding-left: 5px">

<button style="height: 75px; width: 75px" onclick="left()"><img style="height: 65px"src="/images/left.jpg"></button>

<img hspace="20" style="padding-left: 10px">

<button style="height: 75px; width: 75px" onclick="stop()"><img style="height: 63px"src="/images/stop.png"></button>

<img hspace="20" style="padding-left: 10px">

<button style="height: 75px; width: 75px" onclick="right()"><img style="height: 65px"src="/images/right.jpg"></button>

<img hspace="10" style="padding-left: 5px">

<br><br><br><br>

<button style="height: 75px; width: 75px" onclick="reverse()"><img style="height: 65px"src="/images/reverse.png"></button>

<br><br>

<p>Motor speed control</p>

<img hspace="30" style="padding-left: 5px">

<button style="height: 50px; width: 50px; font-size: 18px" onclick="nospeed()">0</button>

<img hspace="30" style="padding-left: 20px">

<button style="height: 50px; width: 50px; font-size: 18px" onclick="lowspeed()">--</button>

<img hspace="30" style="padding-left: 20px">

<button style="height: 50px; width: 50px; font-size: 18px" onclick="regularspeed()">==</button>

<img hspace="30" style="padding-left: 20px">

<button style="height: 50px; width: 50px; font-size: 18px" onclick="highspeed()">++</button>

<img hspace="30" style="padding-left: 5px">

<br>

</span>

</div>

<div id = "rightbox">

<iframe src="http://192.168.43.128:9000/javascript\_simple.html" frameborder="0" align="middle" width="640" height="480" align="middle" scrolling="no"></iframe>

<audio controls src="http://192.168.43.128:8000/rapi.mp3">

<div style="border: 1px solid black ; padding: 8px ;">

</div>

</audio>

<span style="display:inline-block;padding:5px;border:1px solid #fc0; font-size: 140%;font-weight: bold;">

<p>Camera Tilt Angle</p>

<img hspace="18" style="padding-left: 5px">

<button style="height: 50px; width: 100px; font-size: 25px" onclick="downtilt()">-D</button>

<button style="height: 50px; width: 100px; font-size: 25px" onclick="downcentertilt()">D</button>

<button style="height: 50px; width: 100px; font-size: 25px" onclick="centertilt()">0</button>

<button style="height: 50px; width: 100px; font-size: 25px" onclick="upcentertilt()">U</button>

<button style="height: 50px; width: 100px; font-size: 25px" onclick="uptilt()">U+</button>

<img hspace="18" style="padding-left: 5px">

<br>

<p>Camera Pan Position</p>

<button style="height: 50px; width: 100px; font-size: 25px" onclick="leftpan()">+L</button>

<button style="height: 50px; width: 100px; font-size: 25px" onclick="leftCenterPan()">L</button>

<button style="height: 50px; width: 100px; font-size: 25px" onclick="centerpan()">0</button>

<button style="height: 50px; width: 100px; font-size: 25px" onclick="rightCenterPan()">R</button>

<button style="height: 50px; width: 100px; font-size: 25px" onclick="rightpan()">R+</button>

<p></p>

</span>

</div>

<script>

var xmlhttp;

xmlhttp=new XMLHttpRequest();

function lighton()

{

xmlhttp.open("GET","cgi-bin/lighton.cgi",true);

xmlhttp.send();

}

function lightoff()

{

xmlhttp.open("GET","cgi-bin/lightoff.cgi",true);

xmlhttp.send();

}

function forward()

{

xmlhttp.open("GET","cgi-bin/forward.cgi",true);

xmlhttp.send();

}

function stop()

{

xmlhttp.open("GET","cgi-bin/stop.cgi",true);

xmlhttp.send();

}

function left()

{

xmlhttp.open("GET","cgi-bin/left.cgi",true);

xmlhttp.send();

}

function right()

{

xmlhttp.open("GET","cgi-bin/right.cgi",true);

xmlhttp.send();

}

function reverse()

{

xmlhttp.open("GET","cgi-bin/reverse.cgi",true);

xmlhttp.send();

}

function lowspeed()

{

xmlhttp.open("GET","cgi-bin/lowspeed.cgi",true);

xmlhttp.send();

}

function regularspeed()

{

xmlhttp.open("GET","cgi-bin/regularspeed.cgi",true);

xmlhttp.send();

}

function highspeed()

{

xmlhttp.open("GET","cgi-bin/highspeed.cgi",true);

xmlhttp.send();

}

function nospeed()

{

xmlhttp.open("GET","cgi-bin/nospeed.cgi",true);

xmlhttp.send();

}

function downtilt()

{

xmlhttp.open("GET","cgi-bin/downtilt.cgi",true);

xmlhttp.send();

}

function downcentertilt()

{

xmlhttp.open("GET","cgi-bin/downcentertilt.cgi",true);

xmlhttp.send();

}

function centertilt()

{

xmlhttp.open("GET","cgi-bin/centertilt.cgi",true);

xmlhttp.send();

}

function upcentertilt()

{

xmlhttp.open("GET","cgi-bin/upcentertilt.cgi",true);

xmlhttp.send();

}

function uptilt()

{

xmlhttp.open("GET","cgi-bin/uptilt.cgi",true);

xmlhttp.send();

}

function leftpan()

{

xmlhttp.open("GET","cgi-bin/leftpan.cgi",true);

xmlhttp.send();

}

function leftCenterPan()

{

xmlhttp.open("GET","cgi-bin/leftcenterPan.cgi",true);

xmlhttp.send();

}

function centerpan()

{

xmlhttp.open("GET","cgi-bin/centerpan.cgi",true);

xmlhttp.send();

}

function rightCenterPan()

{

xmlhttp.open("GET","cgi-bin/rightcenterPan.cgi",true);

xmlhttp.send();

}

function rightpan()

{

xmlhttp.open("GET","cgi-bin/rightpan.cgi",true);

xmlhttp.send();

}

</script>

<div>

</body>

</html>