TRACKR_UNLEASHED

PROJECT REPORT

Submitted by

KARTHIK S.	**Reg No.:13BEC064**
PERUMAL RAJ S.R.	**Reg No.:13BEC103**
RAKESH CHOUDHARY D.	**Reg No.:13BEC119**
MUHAMATHU NATHIM M.	**Reg No.:13BEC241**

In partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION

KUMARAGURU COLLEGE OF TECHNOLOGY

ANNA UNIVERSITY: CHENNAI

APRIL 2017

BONAFIDE CERTIFICATE

Certified that this project report **"TRACKR_UNLEASHED"** is the bonafide work of **Mr. KARTHIK S. [13BEC064], Mr. PERUMAL RAJ S.R. [13BEC103], Mr. RAKESH CHOUDHARY D. [13BEC119] and Mr. MUHAMATHU NATHIM M. [13BEC241]** who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other project or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE	**SIGNATURE**
Ms. Ashwini A Metri M.Tech,	Dr. K. Malarvizhi M.E., Ph.D.,
PROJECT SUPERVISOR	**HEAD OF THE DEPARTMENT**
Department of ECE	Department of ECE
Kumaraguru College of Technology	Kumaraguru College of Technology
Coimbatore-641049	Coimbatore-641049

The candidates with Register No: 13BEC064, 13BEC103, 13BEC119 and 13BEC241 are examined by us in the project viva-voce examination held on ……………………….

INTERNAL EXAMINAR **EXTERNAL EXAMINAR**

ACKNOWLEDGEMENT

We express our sincere thanks to the Management of Kumaraguru College of Technology and Joint Correspondent **Shri. ShankarVanavarayar** for the kind support and for providing necessary facilities to carry out the project work.

We would like to express our sincere thanks to our beloved Principal **Dr. R.S. Kumar M.E., Ph.D.,** Kumaraguru College of Technology, who encouraged using each and every step of the project.

We would like to thank **Dr. K. Malarvizhi M.E., Ph.D.,** Head of the Department, Electronics and Communication Engineering, for her kind support and for providing necessary facilities to carry out the project work.

We wish to thank with everlasting gratitude to our Project Coordinator **Dr. A. Vasuki M.E., Ph.D.,** Department of Electronics and Communication Engineering for her consistent support throughout the course of this project work.

We are greatly privileged to express our deep sense of gratitude and heartfelt thanks to our Project Guide **Ms. Ashwini A Metri M.Tech.,** Department of Electronics and Communication Engineering for her expert counseling and guidance to make this project to a great deal of success and also we wish to convey our regards to all teaching and non-teaching staff of ECE Department for their help and cooperation.

Finally, we thank our parents and our family members for giving us the moral support and abundant blessings in all of our activities and our dear friends who helped us to endure our difficult times with their unfailing support and warm wishes.

ABSTRACT

The needs for surveillance have an increased demand nowadays. The Project is a rover which can navigate in all directions and can be controlled remotely through Internet. It consists of a GPS Receiver Module for obtaining GPS coordinates and a camera to capture images at regular intervals. The coordinates are then posted to the Google Maps application and it can be used to identify the location of the rover. The images captured are used to detect the objects present in front using Google Vision API. It responds back with the text containing the detected objects. The user dashboard will contain the location of the rover as well as the objects present in front of the rover. Since data on surveillance features is available in real time, machines can be manipulated and perform some action based on the obtained results. It will also be useful in exploring the unexplored places.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

Rpi	Raspberry Pi
GPS	Global Positioning System
API	Application Program Interface
UART	Universal Asynchronous Receiver/Transmitter
SPI	Serial Peripheral Interface
JTAG	Joint Test Action Group
PPM	One part per million
IoT	Internet of Things
CISC	Complex Instruction Set Computer
ARM	Advanced risk machine
GSM	Global System for mobile communication
SoC	System on Chip
GPU	Graphical Processing Unit

1. INTRODUCTION:

The prototype is a rover which is to be used for the surveillance purpose. The rover can be controlled manually through internet using the created app or through a web page. The rover also consists of a GPS receiver which posts to the web API (consisting of Google Maps). It also tracks the path of the rover from the starting point of the traversal to the present location of the rover. The rover also has some intelligent features which can be used for recognizing the objects in the image captured by it. It also posts the recognized result to the web API. This runs in parallel such that the web API shows the current location of the rover and also the objects in front of it.

Open Weather Application is used to determine the temperature, wind speed and the humidity at that particular location. Using its python wrapper function, the details of temperature, wind speed and humidity can be obtained by passing the latitude and longitude values to it. The obtained results are continuously updated in the created dashboard.

This rover can be used by the army people to explore the unexplored places. It can be used to examine conditions and communicate to other robotic machines in obtaining a proper conclusion about a place.

The Development board chosen for this project is Raspberry Pi. It gets powered from the Power Bank with 10000mAh and can supply output current of 2100mA. The raspberry pi consumes 1200mA at the maximum. So, the Raspberry pi can be supported by the fully charged power bank for duration of 8.19 hours. The motors are powered by a 12v Battery. They have a stall torque of 4.515 kg/cm for 1.2A. So, the estimated current consumption is 230 mA. The calculated load of the project is 862g. As per the calculation, the battery can power up motors for duration of 5.13 hours. Thus, the prototype is designed for the working duration of 5.13 hours.

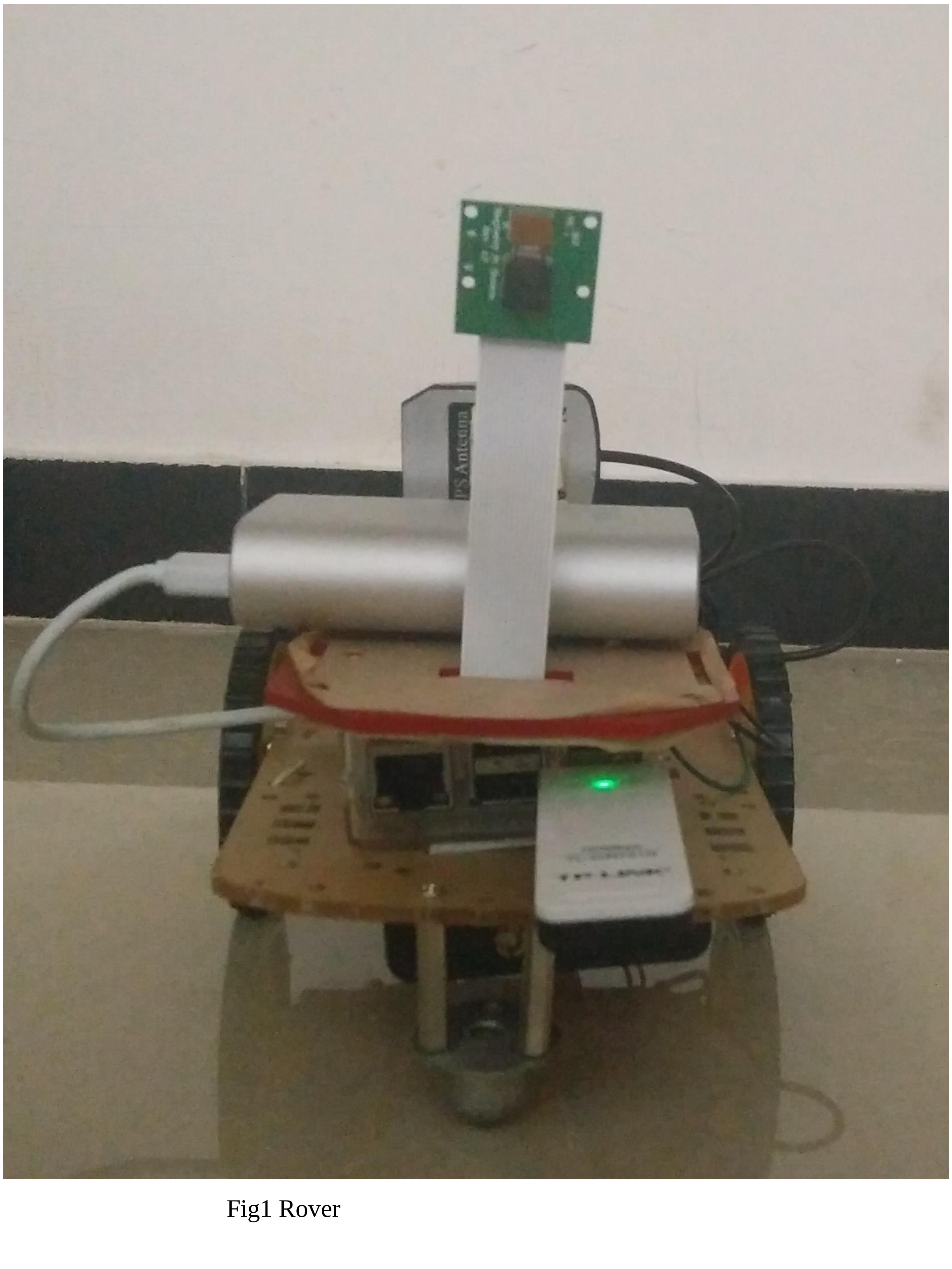


Fig1 Rover

2. BLOCK DIAGRAM:

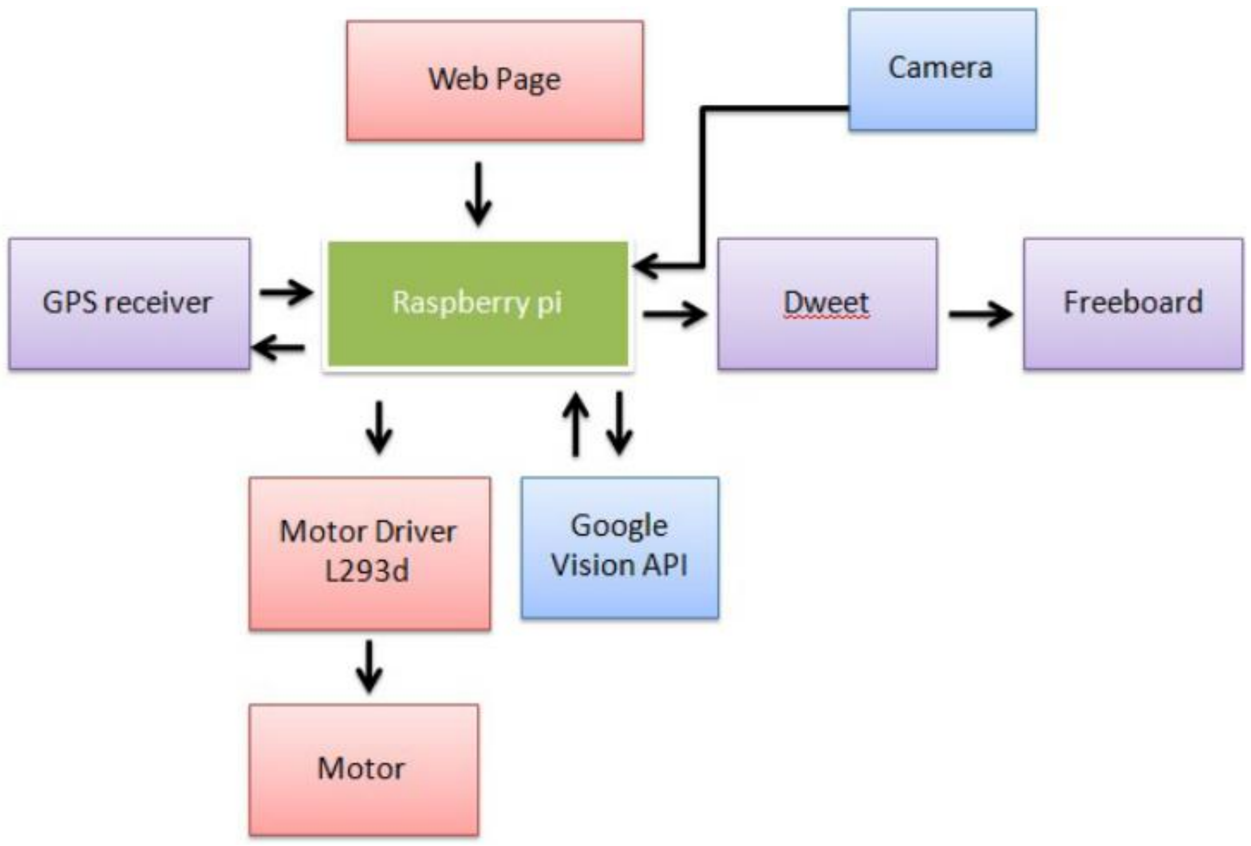


Fig 2.1 BLOCK DIAGRAM FOR WORKING

The webpage contains the buttons to control the direction of motion of rover. When the button is clicked, it triggers the L239d to execute the corresponding function which makes the rover to move in that particular direction. The GPS Receiver sends the GPS coordinates to the Raspberry Pi which dweets that to the Google Maps API in the freeboard dashboard. Raspberry Pi Camera takes picture at periodic intervals and sends that to the Google Vision API. The Google Vision API sends back the detected objects in the image to the Raspberry Pi which is then dweeted to the dashboard in the freeboard.

3. HARDWARE DESCRIPTION

The hardware components used are:

- Raspberry Pi
- L239D Motor Driver
- RPI Camera
- GPS Receiver Module

3.1 RASPBERRY PI

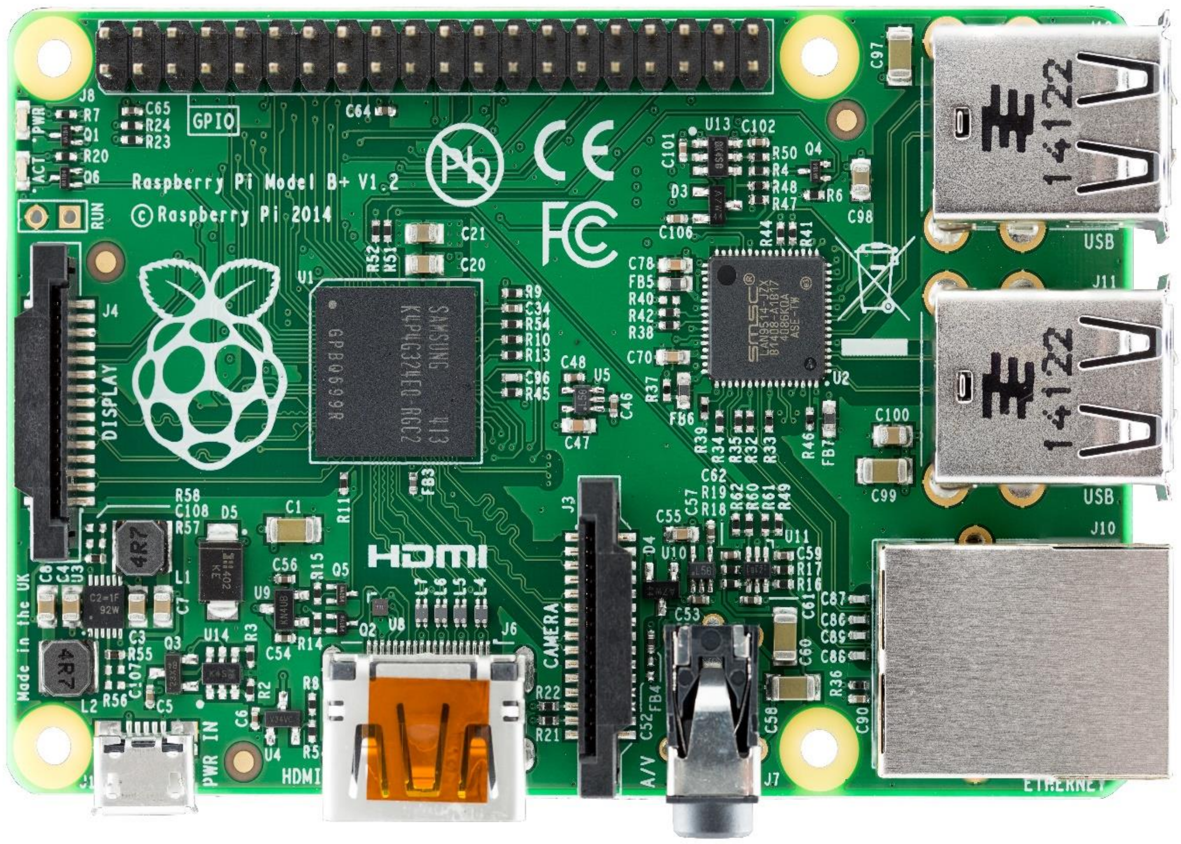


Fig3.1 Raspberry Pi Top View

3.1.1 DESCRIPTION:

The Raspberry Pi is a series of credit card sized single-board computers developed in UK by the Raspberry Pi Foundation with the intend of promoting the teaching of basic computer science in schools.

The original Raspberry Pi is based on the Broadcom BCM2835 system on a chip (SoC),which includes an ARM1176JZF-S 700 MHz processor, Video Core IV GPU and was originally shipped with 256 megabytes of RAM, later upgraded (models B and B+) to 512 MB. The system has Secure Digital (SD) (models A and B) or MicroSD (models A+ and B+) sockets for boot media and persistent storage.

3.1.2 SPECIFICATIONS:

3.1.2.1 PROCESSOR

The SoC used in the first generation Raspberry Pi is somewhat equivalent to the chip used in older smartphones (such as iPhone / 3G / 3GS). The Raspberry Pi is based on the Broadcom BCM2835 system on a chip (SoC), which includes a 700 MHz ARM1176JZF-S processor, Video Core IV GPU and RAM. It has a Level 1 cache of 16 KB and a Level 2 cache of 128 KB. The Level 2 cache is used primarily by the GPU. The SoC is stacked underneath the RAM chip, so only its edge is visible.

3.1.2.2 RAM

On the older beta model B boards, 128 MB was allocated by default to the GPU, leaving 128 MB for the CPU. On the first 256 MB release model B (and model A), three different splits were possible. The default split was 192 MB (RAM for CPU), which should be sufficient for standalone 1080p video decoding, or for simple 3D, but probably not for both together. 224 MB was for

Linux only, with just a 1080p framebuffer, and was likely to fail for any video or 3D. 128 MB was for heavy 3D, possibly also with video decoding (e.g. XBMC). Comparatively the Nokia 701 uses 128 MB for the Broadcom Video Core IV. For the new model B with 512 MB RAM initially there were new standard memory split files released(arm256_start.elf, arm384_start.elf, arm496_start.elf) for 256 MB, 384 MB and

496 MB CPU RAM (and 256 MB, 128 MB and 16 MB video RAM). But a week or so later the RPF released a new version of start.elf that could read a new entry in config.txt (gpu_mem=xx) and could dynamically assign an amount of RAM (from 16 to 256 MB in 8 MB steps) to the GPU, so the older method of memory splits became obsolete, and a single start.elf worked the same for 256 and 512 MB Pis. The second generation has 1 GB of RAM.

3.1.2.3 NETWORKING

Though the model A and A+ do not have an 8P8C ("RJ45") Ethernet port, they can be connected to a network using an external user-supplied USB Ethernet orWiFi adapter. On the model B and B+ the Ethernet port is provided by a built-in USB Ethernet adapter.

3.1.2.4 PHERIPERALS

Generic USB keyboards and mice are compatible with the Raspberry Pi.

3.2 GPS RECEIVER L80:

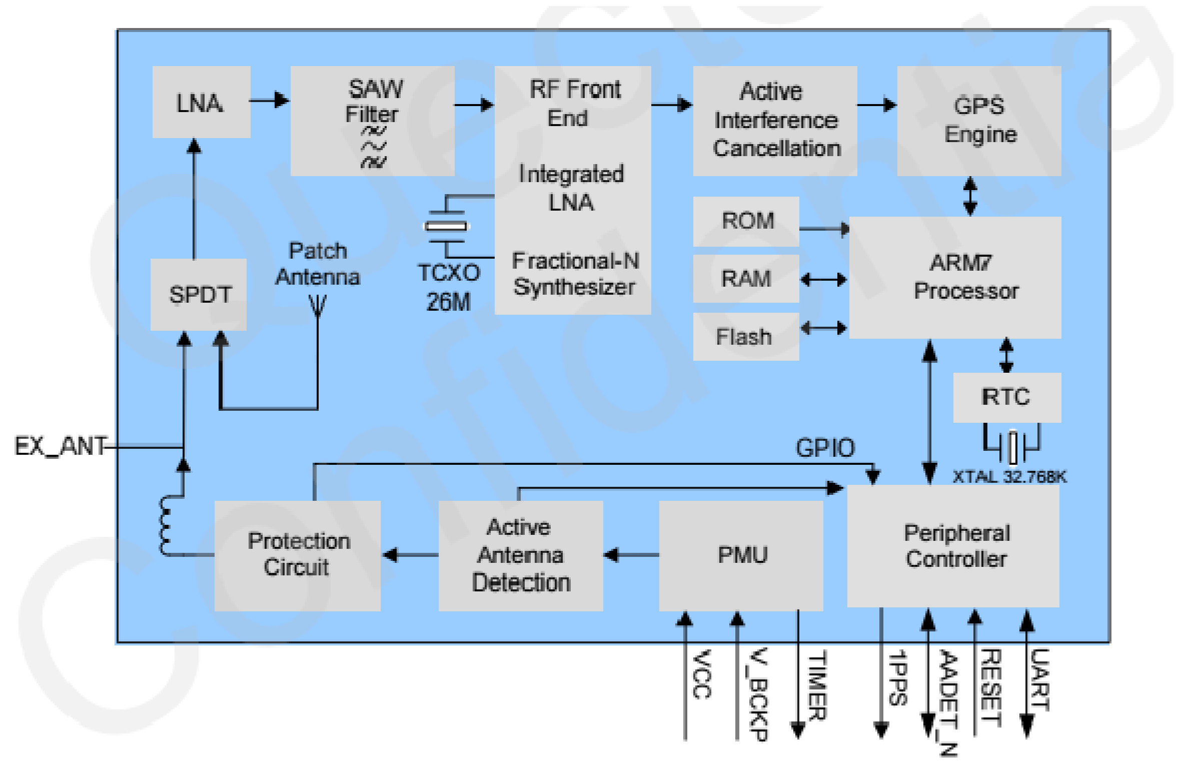


Fig3.2GPS Receiver Module

It consists of a single chip GPS IC which includes RF part and Baseband part, a SPDT, a patch antenna, a LNA, a SAW filter, a TCXO, a crystal oscillator, short protection and antenna detection circuit for active antenna.

3.2.1 PIN DIAGRAM

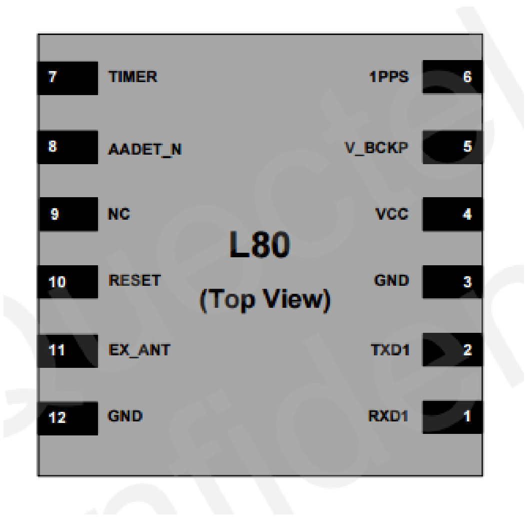
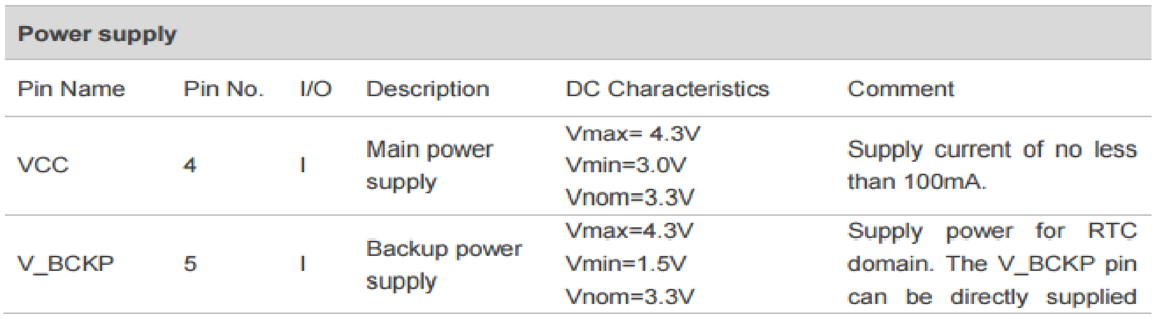
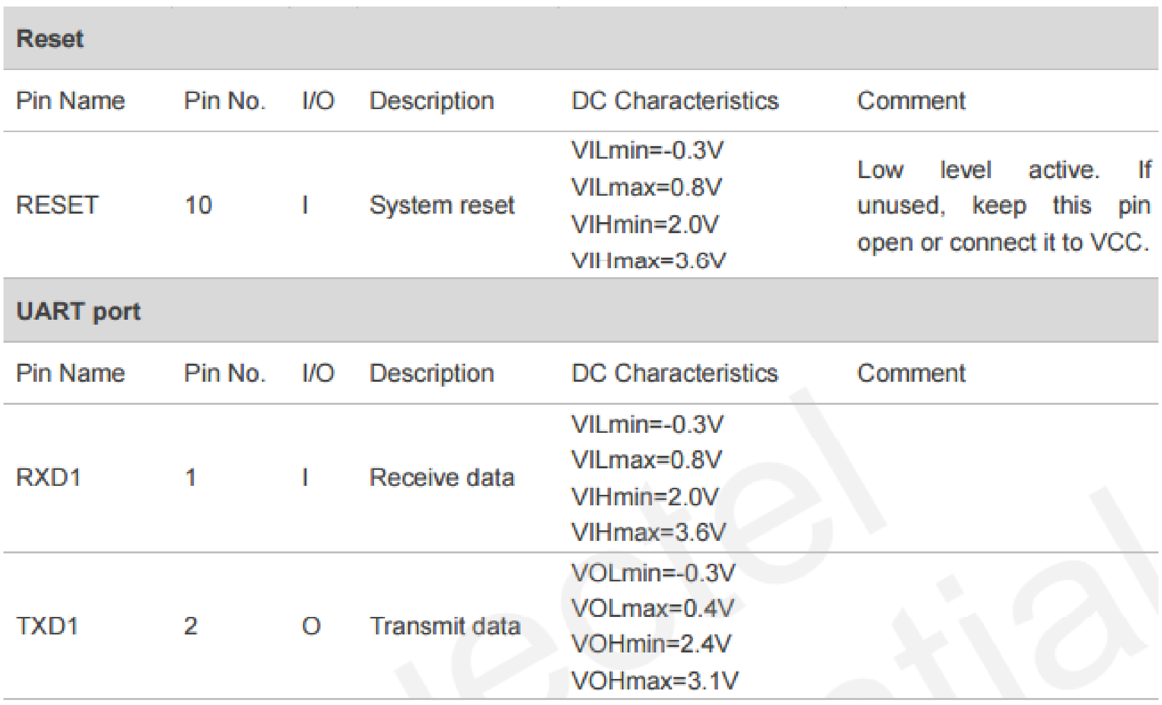


Fig3.2.1GPS Receiver L80 Pin Diagram

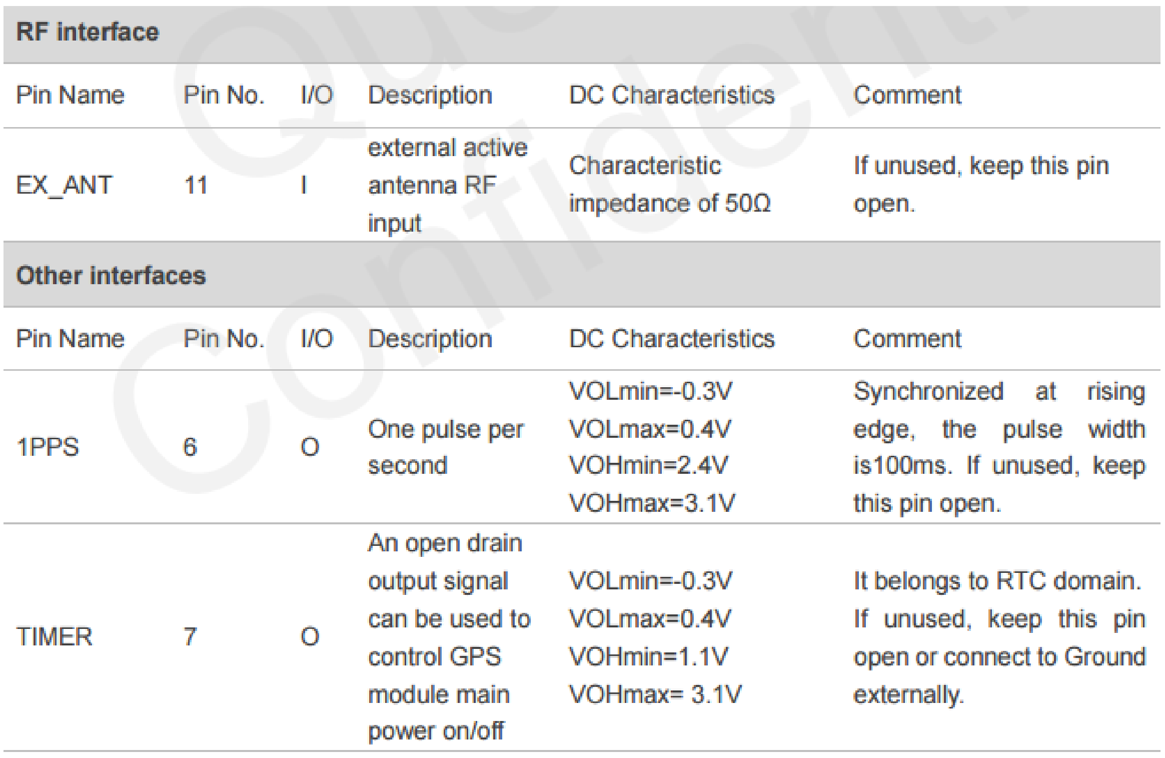
3.2.2 PIN DESCRIPTION



Power supply					
Pin Name	Pin No.	I/O	Description	DC Characteristics	Comment
VCC	4	I	Main power supply	Vmax= 4.3V Vmin=3.0V Vnom=3.3V	Supply current of no less than 100mA.
V_BCKP	5	I	Backup power supply	Vmax=4.3V Vmin=1.5V Vnom=3.3V	Supply power for RTC domain. The V_BCKP pin can be directly supplied



Reset					
Pin Name	Pin No.	I/O	Description	DC Characteristics	Comment
RESET	10	I	System reset	VILmin=-0.3V VILmax=0.8V VIHmin=2.0V VIHmax=3.6V	Low level active. If unused, keep this pin open or connect it to VCC.
UART port					
Pin Name	Pin No.	I/O	Description	DC Characteristics	Comment
RXD1	1	I	Receive data	VILmin=-0.3V VILmax=0.8V VIHmin=2.0V VIHmax=3.6V	
TXD1	2	O	Transmit data	VOLmin=-0.3V VOLmax=0.4V VOHmin=2.4V VOHmax=3.1V	



RF interface					
Pin Name	Pin No.	I/O	Description	DC Characteristics	Comment
EX_ANT	11	I	external active antenna RF input	Characteristic impedance of 50Ω	If unused, keep this pin open.
Other interfaces					
Pin Name	Pin No.	I/O	Description	DC Characteristics	Comment
1PPS	6	O	One pulse per second	VOLmin=-0.3V VOLmax=0.4V VOHmin=2.4V VOHmax=3.1V	Synchronized at rising edge, the pulse width is100ms. If unused, keep this pin open.
TIMER	7	O	An open drain output signal can be used to control GPS module main power on/off	VOLmin=-0.3V VOLmax=0.4V VOHmin=1.1V VOHmax= 3.1V	It belongs to RTC domain. If unused, keep this pin open or connect to Ground externally.

3.2.3 GPS WORKING

The Global Positioning System (GPS) is a network of about 30 satellites orbiting the Earth at an altitude of 20,000 km. Wherever you are on the planet, at least four GPS satellites are ‘visible’ at any time. Each one transmits information about its position and the current time at regular intervals. These signals, travelling at the speed of light, are intercepted by your GPS receiver, which calculates how far away each satellite is based on how long it took for the messages to arrive. Once it has information on how far away at least three satellites are, your GPS receiver can pinpoint your location using a process called Trilateration.

3.2.4 TRILATERATION

Data from a single satellite pinpoints position to a large area of the earth's surface. Adding data from a second satellite narrows position down to the region where the two spheres overlap. Adding data from a third satellite provides a relatively accurate position and data from a fourth satellite or more than four satellites enhances precision. It also determines accurate elevation or, in the case of aircraft, altitude. GPS receivers routinely track four to seven satellites or even more simultaneously and use trilateration to analyse it.

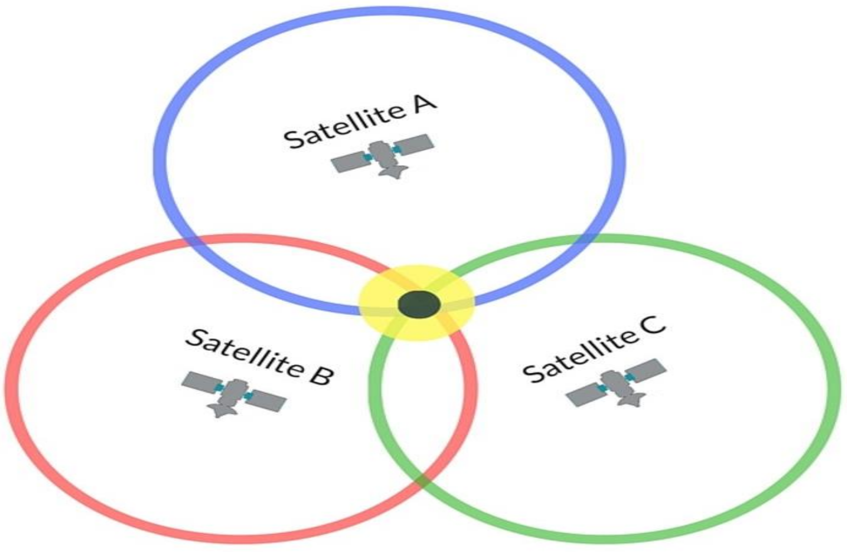


Fig3.3Trilateration

3.2.5 NMEA MESSAGE FORMAT

The GPS coordinates are obtained from the receiver as an NMEA Message.

***$GPGGA,181908.00,3404.7041778,N,07044.3966270,
W,4,13,1.00,495.144,M,29.200,M,0.10,0000*40***

All NMEA messages start with the $ character, and each data field is separated by a comma.

GP represent that it is a GPS position (GL would denote GLONASS).

181908.00 is the time stamp: UTC time in hours, minutes and seconds.

3404.7041778 is the latitude in the DDMM.MMMMM format. Decimal places are variable.

N denotes north latitude.

07044.3966270 is the longitude in the DDDMM.MMMMM format. Decimal places are variable.

W denotes west longitude.

4 denotes the Quality Indicator:

1 = Uncorrected coordinate

2 = differentially correct coordinate (e.g., WAAS, DGPS)

4 = RTK Fix coordinate (centimeter precision)

5 = RTK Float (decimeter precision)

13 denotes number of satellites used in the coordinate.

1.0 denotes the HDOP (horizontal dilution of precision).

495.144 denotes altitude of the antenna.

M denotes units of altitude (eg. Meters or Feet)

29.200 denotes the geoidal separation (subtract this from the altitude of the antenna to arrive at the Height above Ellipsoid (HAE).

M denotes the units used by the geoidal separation.

1.0 denotes the age of the correction (if any).

0000 denotes the correction station ID (if any).

***40** denotes the checksum.

$GPGSA – Detailed GPS DOP and detailed satellite tracking information (eg. individual satellite numbers). $GNGSA for GNSS receivers.

$GPGSV – Detailed GPS satellite information such as azimuth and elevation of each satellite being tracked. $GNGSV for GNSS receivers.

$GPVTG – Speed over ground and tracking offset.

3.3 L239D:

Fig3.4L239D silicon

3.3.1 DESCRIPTION

L293D is a typical Motor driver or Motor Driver IC which allows DC motor to drive on either direction. L293D is a 16-pin IC which can control a set of two DC motors simultaneously in any direction. It means that you can control two DC motor with a single L293D IC. Dual H-bridge Motor Driver integrated circuit (IC).

3.3.2 PIN DIAGRAM

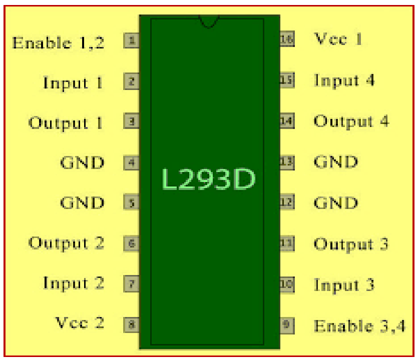


Fig 3.5L239D pin diagram

3.3.3 SPECIFICATIONS

RATING	MAX VALUE	UNIT
Maximum Supply voltage	36	V
Input voltage	7	V
Output supply current	24	mA
Logic supply current	60	mA
Operating temperature range	0 to +70	C

3.3.4WORKING OF L239D

It works on the concept of H-bridge. H-bridge is a circuit which allows the voltage to be flown in either direction. As voltage need to change its direction for being able to rotate the motor in clockwise or anticlockwise direction, this IC is ideal for driving a DC motor. In a single L293D chip there are two H-Bridge circuit inside the IC which can rotate two dc motor independently. Due its size it is very much used in robotic application for controlling DC motors.

There are two Enable pins on l293d. Pin 1 and pin 9, for being able to drive the motor, the pin 1 and 9 need to be high. For driving the motor with left H-bridge you need to enable pin 1 to high. And for right H-Bridge you need to make the pin 9 to high. If anyone of the either pin1 or pin9 goes low then the motor in the corresponding section will suspend working. It’s like a switch.

There are 4 input pins for l293d, pin 2, 7 on the left and pin 15, 10 on the right as shown on the pin diagram. Left input pins will regulate the rotation of motor connected across left side and right input for motor on the right hand side. The motors are rotated on the basis of the inputs provided across the input pins as LOGIC 0 or LOGIC 1.In simple you need to provide Logic 0 or 1 across the input pins for rotating the motor.

3.3.5 L239D LOGIC

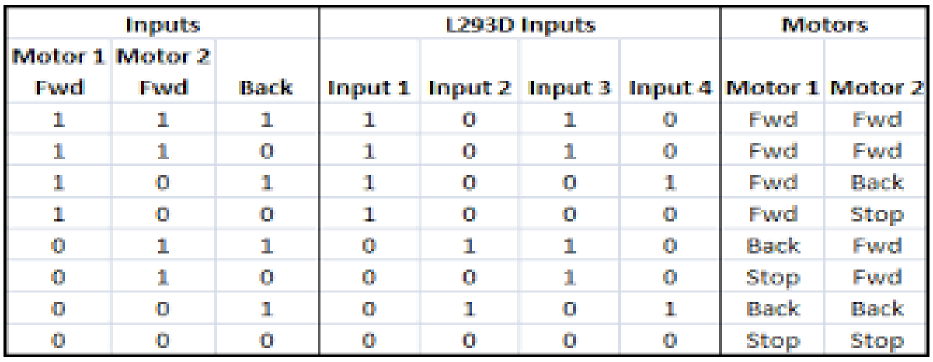
For rotating the motor in clockwise direction the input pins has to be provided with Logic 1 and Logic 0.

• Pin 2 = Logic 1 and Pin 7 = Logic 0 | Clockwise Direction

• Pin 2 = Logic 0 and Pin 7 = Logic 1 | Anticlockwise Direction

• Pin 2 = Logic 0 and Pin 7 = Logic 0 | Idle [No rotation] [Hi-Impedance state]

• Pin 2 = Logic 1 and Pin 7 = Logic 1 | Idle [No rotation].



Inputs			L293D Inputs				Motors	
Motor 1 Fwd	Motor 2 Fwd	Back	Input 1	Input 2	Input 3	Input 4	Motor 1	Motor 2
1	1	1	1	0	1	0	Fwd	Fwd
1	1	0	1	0	1	0	Fwd	Fwd
1	0	1	1	0	0	1	Fwd	Back
1	0	0	1	0	0	0	Fwd	Stop
0	1	1	0	1	1	0	Back	Fwd
0	1	0	0	0	1	0	Stop	Fwd
0	0	1	0	1	0	1	Back	Back
0	0	0	0	0	0	0	Stop	Stop

3.3.6 VOLTAGE SPECIFICATION

VCC is the voltage that it needs for its own internal operation 5v; L293D will not use this voltage for driving the motor. For driving the motors it has a separate provision to provide motor supply VSS (V supply). L293d will use this to drive the motor. It means if you want to operate a motor at 9V then you need to provide a Supply of 9V across VSS Motor supply.

The maximum voltage for VSS motor supply is 36V. It can supply a max current of 600mA per channel. Since it can drive motors Up to 36v hence you can drive pretty big motors with this l293d.VCC pin 16 is the voltage for its own internal Operation. The maximum voltage ranges from 5v and up to 36v.

3.4 RASPBERRY PI CAMERA

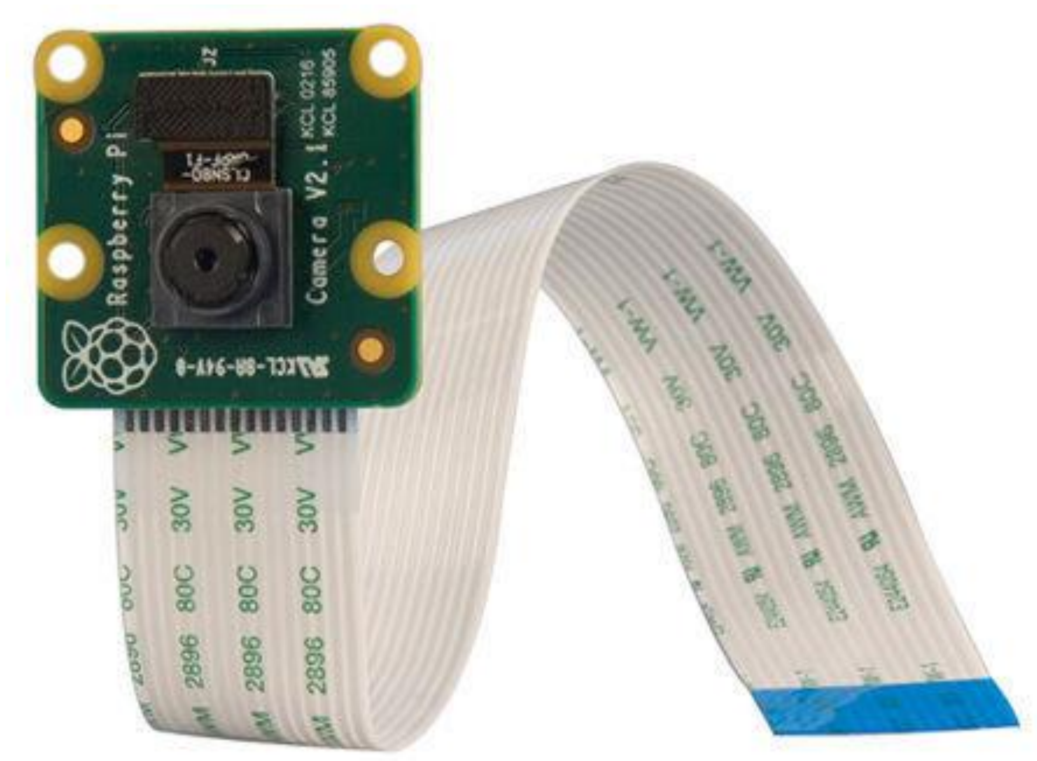


Fig 3.6Rpi Camera

3.4.1 SPECIFICATON

	Camera Module v2
Net price	$25
Weight	3g
Still resolution	8 Megapixels
Video modes	1080p30, 720p60 and 640 × 480p60/90
Linux integration	V4L2 driver available
C programming API	OpenMAX IL and others available
Sensor	Sony IMX219
Sensor resolution	3280 × 2464 pixels
Sensor image area	3.68 x 2.76 mm (4.6 mm diagonal)
Pixel size	1.12 µm x 1.12 µm
Optical size	1/4"
Focal length	3.04 mm
Horizontal field of view	62.2 degrees
Vertical field of view	48.8 degrees
Focal ratio (F-Stop)	2.0

3.4.2 SOFTWARE FEATURES

Picture formats	JPEG (accelerated), JPEG + RAW, GIF, BMP, PNG, YUV420, RGB888
Video formats	raw h264 (accelerated)
Effects	negative, solarise, poster, whiteboard, blackboard, sketch, remove noise, emboss, hatch, pastel, watercolour, film, blur, saturation
Exposure modes	auto, night, nightpreview, backlight, spotlight, sports, snow, beach, verylong, fixedfps, anti-shake, fireworks
Metering modes	average, spot, backlit, matrix
Automatic white balance modes	off, auto, sun, cloud, shade, tungsten, fluorescent, incandescent, flash, horizon
Triggers	Keypress, UNIX signal, timeout
Extra modes	demo, circular buffer, video with motion vectors, segmented video, live preview on 3D models

4. SOFTWARE

4.1 Dweet.io

The goal of IoT messaging is for a device to send a chunk of data to a server where it’s stored for some period of time. That data can then be retrieved by one or mutliple devices for analysis, display, storage or whatever. And here’s where it starts to get complex: What happens when a message from a source device gets lost? When it gets duplicated? When two messages arrive out of sequence? Dweet ignores these issues and simply makes a best effort attempt to receive and send messages so any messages integrity assurance has to be handled by the devices at either end (in other words, the devices have to add sequence numbers and or timestamps when they send messages and detect missing, duplicated, and out of sequence messages when they retrieve messages).

So, how easy is it to use dweet? Sending a message:

https://dweet.io/dweet/for/gearhead?hello=world

That’s it; no authentication required, no fussing around, you just make a GET request. The “thing” here that’s dweeting is named “gearhead” and it is dweeting the value of “world” for the key “hello”. Here’s the JSON response to that request:

```
{
"this": "succeeded",
"by": "dweeting",
"the": "dweet",
"with": {
"thing": "gearhead",
"created": "2016-10-21T19:23:36.899Z",
"content": {
"hello": "world"
},
"transaction": "2b6f25c1-d0b2-4319-8160-9f4e42caa794"
}
}
```

Dweet.io is a simple time series storage service designed for IoT devices. It comes in a free and paid variety. The free edition only can store up to 24 hours. First, a unique name has to be created for the thing. Now, try and request the last 24 hours history from the API to see if the thing name has been used already.

Hopefully, It should return 404 back. This indicates the thing name chosen is not being used. If any other response is received, then choose another name rather than wrecking some other guy's data set. Next, post some data into the created thing's time series dataset. Decide the key value pairs to be stored. Dweet accepts values either as a post with the body containing JSON key value pairs or via a GET with the key value pairs in the query string. For simplicity, it is better to use the GET method. Now, one should get a 200 response from the Dweet API. When submitted the first set of data, the thing name will be registered and the first data will be stored in the created thing.

4.2 Freeboard.io

Freeboard was developed by Bug Labs, the creators of a powerful service called dweet, a messaging service designed for Internet of Things devices. Freeboard is a free, open source dashboard project with optional hosted subscriptions that's easy to integrate with diverse data sources, production ready, and elegantly designed. The Data sources menu at the top left of the configuration panel is where you add the sources of data for widgets to display. In the downloaded version you can include JSON data from a URL, call the Open Weather Map API and retrieve data from Bug Labs' dweet.io service.

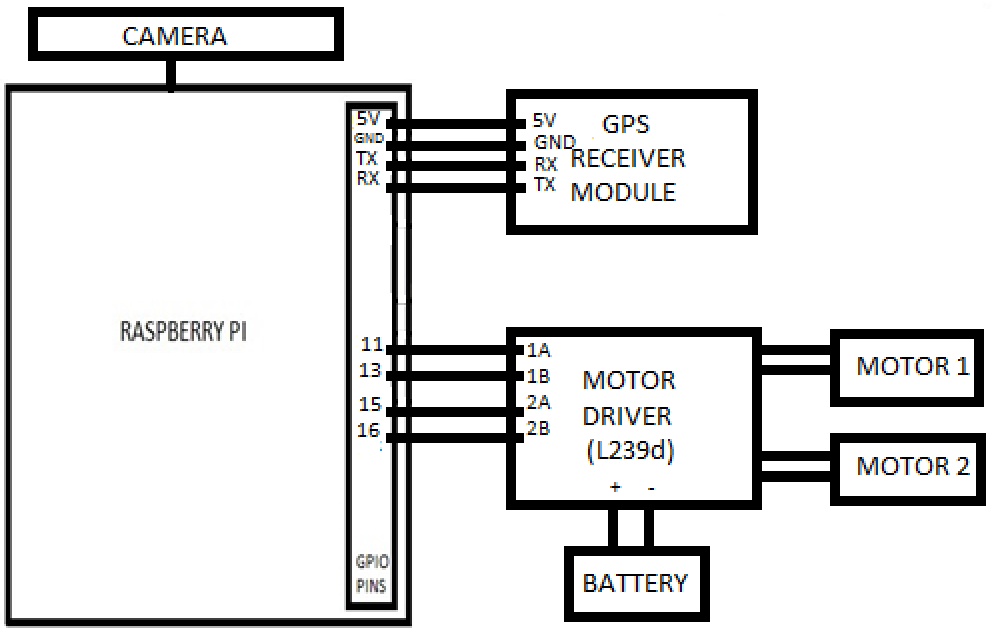
Freeboard.io is a simple web based dashboard service.One can create as many public dashboards as you like for free. Although if one wants to protect your dashboards, one should need a paid account. Head over to freeboard.io to sign up for an account, once confirmed login and create the first dashboard. Enter a name for the dashboard in the text box and click "Create Dashboard", the page will refresh and will be presented with a blank dashboard with editing tools in the pane at the top of the page. Before start building the dashboard one'll want to add the Dweet thing as a data source. Under the data source’s header click “add”, one'll notice that there are many types of data source which can be used from raw JSON to Dweet. Select Dweet.io as the data source and complete the form including a nickname for the data source, your thing name and so on, the key only applies if one has a paid Dweet.io account with a lock on your thing.

The widgets included in the download version of Freeboard are text, gauge, Sparkline, pointer, picture, indicator light (an on or off display, great for server and service status reporting), Google Maps, and HTML, while the online hosted version adds an historical chart widget to display either a bar chart or a line graph of a data source.

One can add a widget to the dashboard by clicking on the plus sign in a panel and selecting the type of widget, its title, size, and value. The value can be derived from a data source configured or add JavaScript to the widget to do whatever needed and set the widget accordingly. A nice feature of text-based widgets is an option to animate the display so that when a value changes, it uses smooth transitions. Once the widget is configured, it is immediately live.

When set up all of the widgets and dragged them to where one wants them located on the canvas, one can hide the top configuration panel by clicking on the caret icon at the bottom of the panel. A tidy, well designed, highly flexible responsive web dashboard with all of the configuration controls hidden except for a wrench icon to display the configuration panel. In the hosted version there is no user access to the configuration panel.

5. CIRCUIT DIAGRAM

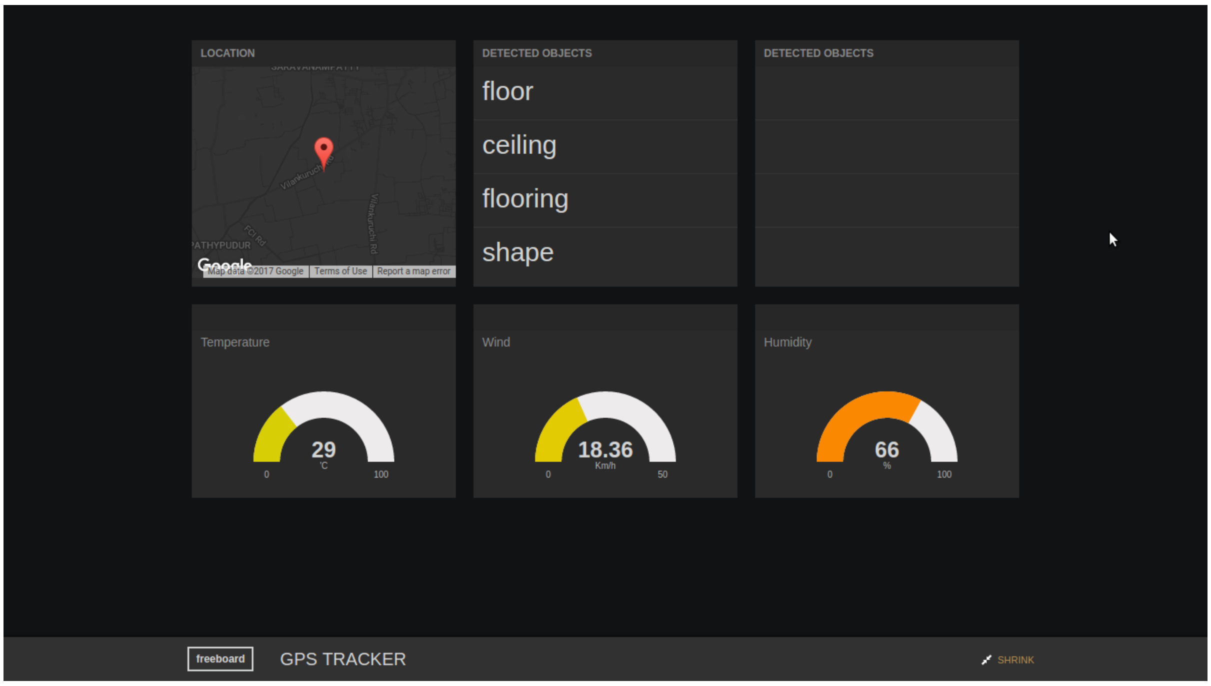


6. WORKING PRINCIPLES

Raspberry Pi is an ARM based processor. A HTTP Server called APACHE is installed in Raspberry pi so that the processor is made to act as a Server. Since it is made to act as a server, a cloud is created in which all the files related to project are stored. Anyone in the network of raspberry pi can access those files present in the cloud. PHP is a server scripting language. Front end of the webpage is created using PHP. 5 buttons namely STRAIGHT, LEFT, RIGHT, REVERSE and STOP are placed in that webpage. If a button is clicked, the clicked event triggers the corresponding python file to run. If a button is clicked, the corresponding python file triggers the corresponding GPIO connected to the motor driver and it controls the motor in the corresponding way. Now, the user can control the rover traversal remotely through Internet (IoT).

GPS Receiver Module is interfaced with Raspberry Pi through UART protocol. The GPS Receiver provides GPS Data in the NMEA Format. Using String operations, GPS Latitude and Longitude are extracted and dweeted to the dashboard in the Freeboard.

In Parallel, the camera captures pictures and the raspberry pi sends those to Google Vision API to analyse the objects in the picture. The returned responses are then dweeted to the dashboard in the Freeboard.



DETECTED OBJECTS

floor
ceiling
flooring
shape

DETECTED OBJECTS

Temperature 29 °C (0–100)

Wind 18.36 Km/h (0–50)

Humidity 66 % (0–100)

GPS TRACKER

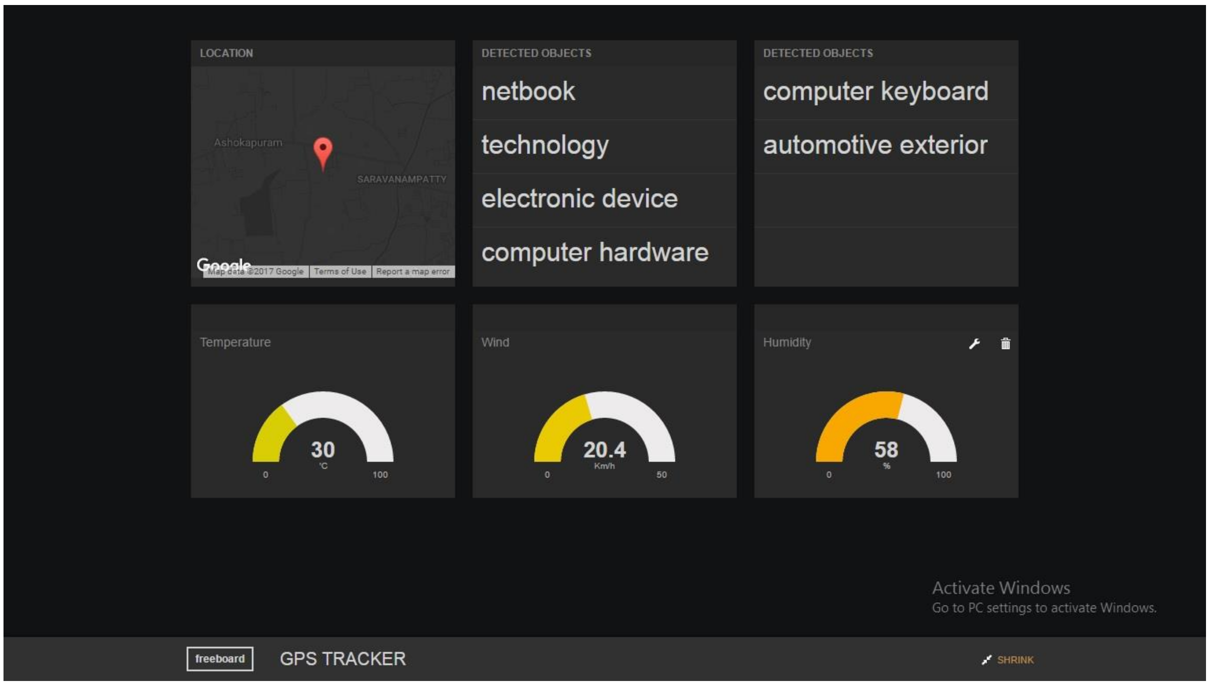


Fig 6.1Dashboard.

7. ADVANTAGES:

- Real time analysis about a particular location.
- Tracking the entire path traversed by the rover
- Reduced the risk of involving humans in exploring places which are dangerous.

8. DISADVANTAGES:

- Slow Speed Internet will affect the latency in posting the acquired details to the dashboard.

9. CONCLUSION:

Thus the rover can be remotely controlled through internet. Its location and the objects in front of the rover uploaded to the web dashboard in real time. This will be helpful in surveillance purposes to explore the unexplored places where it’s hard for humans to commute.

In Future, everything will be automated and machines communicate among themselves and decide by themselves. This rover will be the first step of that process which provides all its surveillance features as data which can be used by a machine that can characterize places on the result provided.

10. FUTURE SCOPE:

- Machine Learning algorithms can be added to the rover such that the robot navigates on its own.
- The life time of the prototype can be increased by installing addition of solar cells and battery which will store the trapped solar energy. When the power of the power bank gets low, the prototype will go to sleep state and the power bank is charged from the battery. When the power bank gets charged to certain level, the prototype will get back to its active state.
- When multiple TrackR prototypes are involved in exploring a place, machine to machine communication features can be added to attain the goal in shorter time.

}

APPENDIX

1.Front end of the webpage

```
<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<script src="http://code.jquery.com/jquery-1.11.0.min.js"></script>

<script type="text/javascript">

$(document).ready(function(){

$('#straight').click(function(){

var a= new XMLHttpRequest();

a.open("GET","straight.php");

a.onreadystatechange=function(){

if(a.readyState==4){

if(a.status==200)

{

document.getElementById("ajax").innerHTML ="STRAIGHT" ;

}

else{ alert("HTTP ERROR")

}

}
```

```
}
a.send();
});
$('#stop').click(function(){
var a= new XMLHttpRequest();
a.open("GET","stop.php");
document.getElementById("ajax").innerHTML ="STOP";
a.onreadystatechange=function(){
if(a.readyState==4){
if(a.status==200)       {
}
else {
alert("HTTP ERROR")
}
}
}
a.send();
$('#left').click(function(){
var a= new XMLHttpRequest();
a.open("GET","left.php");
document.getElementById("ajax").innerHTML ="LEFT";
a.onreadystatechange=function(){
```

```
if(a.readyState==4){
if(a.status==200)       {
}
else {
alert("HTTP ERROR")
}
}
}
a.send();
});
$('#right').click(function(){
var a= new XMLHttpRequest();
a.open("GET","right.php");
document.getElementById("ajax").innerHTML ="RIGHT";
a.onreadystatechange=function(){
if(a.readyState==4){
if(a.status==200)       {
}
else {
alert("HTTP ERROR")
}
}
```

```
}
a.send();
});
$('#reverse').click(function(){
 var a= new XMLHttpRequest();
 a.open("GET","reverse.php");
 document.getElementById("ajax").innerHTML ="REVERSE";
a.onreadystatechange=function(){
if(a.readyState==4){
if(a.status==200)       {
}
else {
alert("HTTP ERROR")
}
}
}
a.send();
});


});
});
</script>
```

```
<title>ROVER</title>

<style>

.button {

background-color :#4CAF50;

border: none;

color: white;

padding: 30px 32px;

test-align :center;

display: inline-block;

font-size: 32px;

cursor: pointer;

margin: auto;

width: 10%;

border: 3px solid #73AD21;

padding :10px;


}

</style>

</head>

<body>

<p align="center">

<b>ROVER</b>
```

```
<br>

<br>

<br>

 <button type="button" id="straight" name="straight"
class="button">STRAIGHT</button><br>

<br>

<button type="button" id="stop" name"stop"
class="button">STOP</button><br>

<br>

<button type="button" id="left" name"left"
class="button">LEFT</button><br>

<br>

<button type="button" id="right" name"right"
class="button">RIGHT</button><br>

<br>

<br>

 <button type="button" id="reverse" name ="reverse"
class="button">REVERSE</button><br>

<p align="center" id="ajax"></p>

</p>

</body>

</html>
```

2.Straight navigation

straight.php

```
<?php
        system("gpio -1 mode 11 out");
        system("gpio -1 mode 13 out");
        system("gpio -1 mode 15 out");
        system("gpio -1 mode 16 out");
        system("gpio -1 write 11 1");
        system("gpio -1 write 13 0");
        system("gpio -1 write 15 1");
        system("gpio -1 write 16 0");
echo '<p> STRAIGHT </p> '
?>
```

3. Left Navigation

left.php

```
<?php
        system("gpio -1 mode 11 out");
        system("gpio -1 mode 13 out");
        system("gpio -1 mode 15 out");

        system("gpio -1 mode 16 out");

        system("gpio -1 write 11 1");

        system("gpio -1 write 13 0");

        system("gpio -1 write 15 0");

        system("gpio -1 write 16 0");


echo '<p> LEFT </p> '

?>
```

.

4. Right Navigation

right.php

```
<?php
        system("gpio -1 mode 11 out");
        system("gpio -1 mode 13 out");
        system("gpio -1 mode 15 out");
        system("gpio -1 mode 16 out");
        system("gpio -1 write 11 0");
        system("gpio -1 write 13 0");
        system("gpio -1 write 15 1");
        system("gpio -1 write 16 0");
echo '<p> RIGHT </p> '
?>
```

5. Reverse Navigation

reverse.php

```
<?php
        system("gpio -1 mode 11 out");
        system("gpio -1 mode 13 out");
        system("gpio -1 mode 15 out");
        system("gpio -1 mode 16 out");
        system("gpio -1 write 11 0");
        system("gpio -1 write 13 1");
        system("gpio -1 write 15 0");
        system("gpio -1 write 16 1");

echo '<p> REVERSE </p> '
?>
```

6. Stop the Navigation

stop.php

```
<?php
        system("gpio -1 mode 11 out");
        system("gpio -1 mode 13 out");
        system("gpio -1 mode 15 out");
        system("gpio -1 mode 16 out");
        system("gpio -1 write 11 0");
        system("gpio -1 write 13 0");
        system("gpio -1 write 15 0");
        system("gpio -1 write 16 0");

echo '<p> STOP </p> '
?>
```

7. Post GPS Coordinates to Dashboard

```
        import serial
        import requests
        def GPS_9600():
                URL="http://dweet.io/dweet/for/gpstrack?"
                lat=None
                longt=None

ser=serial.Serial(port='/dev/ttyAMA0',baudrate=9600,parity=serial.PARITY_N
ONE,stopbits=serial.STOPBITS_ONE,bytesize=serial.EIGHTBITS,timeout=1)
                while(True):
                    rec=""
                    ser.timeout=20
                    rec=ser.readline()
                    s=str(rec)
                    if(len(rec)>0):
                        sp=s.split(',')[0]
                if((sp in "$GPGGA")|(sp in "$GPRMA")):
                    lat=float(s.split(',')[2])/100
                    longt=float(s.split(',')[4])/100
                    print lat
                    print longt
                elif((sp in "$GPRMC")):
                    lat=float(s.split(',')[3])/100
                    longt=float(s.split(',')[5])/100
                    print lat
                    print longt
                elif((sp in "$GPGPL")):
                    lat=float(s.split(',')[1])/100
                    longt=float(s.split(',')[3])/100
                    print lat
                    print longt
                PARAMS={'la':lat,'&lon':longt}
                r=requests.post(url=URL,params=PARAMS)


    if __name__=="__main__":
        GPS_9600()
```

8. Post Detected Objects to Dashboard

```
import commands
import requests
commands.getoutput("export
GOOGLE_APPLICATION_CREDENTIALS=trackr-6dc85e66833d.json")
```

```
while(True):
    URL="http://dweet.io/dweet/for/detectedobjects?"
    commands.getoutput('raspistill -o image.jpg')
    s=commands.getoutput('python python-docs-samples/vision/cloud-client/detect/detect.py labels image.jpg')
    detected=[None,None,None,None,None,None]
    s=s.split('\n')
    s=s[1:len(s)]
    detected[0:len(s)]=s
    PARAMS={'label':detected}
    r=requests.post(url=URL,params=PARAMS)
```

REFERENCE

1. Sathe Pooja,” Vehicle Tracking System Using GPS”, International Journal of Science and Research (IJSR), India Online ISSN: 2319-7064, 2013.
2. Albert Alexe, R. Ezhilarasie, "Cloud Computing Based Vehicle Tracking Information systems" , IJCST Vol 2, Issue 1, March 2011.
3. Magazine for Raspberry Pi users “The MagPi“ .
4. Raspberry Pi Architecture by Jon Holton and Tim Fratangelo “The Raspberry Pi Foundation”.
5. “https://cloud.google.com/vision/docs”.