MILITARY TACTICAL VISOR

by

Ayush Raj (1401010038)

Department of Computer Science

United College Of Engineering And Research
UPSIDC industrial area, Naini, Allahabad

April, 2018

MILITARY TACTICAL VISOR

by
Ayush Raj (1401010038)

Submitted to the Department of Computer Science
in partial fulfillment of the requirements
for the degree of
Bachelor Of Technology
in
Computer Science and Engineering

United College Of Engineering and Research, Naini, Allahabad
APJ Abdul Kalam Technical University

April, 2018

TABLE OF CONTENTS

page no.

CERTIFICATE

This is to certify that Project Report entitled "Military Tactical Visor" which is submitted by Ayush Raj in partial fulfillment of the requirement for the award of degree B. Tech. in Department of Computer Science And Engineering of A.P.J. Abdul Kalam Technical University, is a record of the candidate's own work carried out by him under my/our supervision. The matter embodied in this thesis is original and has not been submitted for the award of any other degree.

Date: **Supervisor:**

DECLARATION

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Signature:

Name:

Roll no:

Date:

ACKNOWLEDGEMENTS

It gives me a great sense of pleasure to present the report of the B. Tech Project undertaken during B. Tech. Final Year. I owe special debt of gratitude to Mr. Om Prakash Agrahari, Department of Computer Science & Engineering, United College Of Engineering And Research, Naini, Allahabad for his constant support and guidance throughout the course of my work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for me. It is only his cognizant efforts that our endeavors have seen light of the day.

I also take the opportunity to acknowledge the contribution of Mr, Sanjay Pandey, Head Of Department, Computer Science & Engineering, United College Of Engineering And Research, Naini, Allahabad for his full support and assistance during the development of the project.

I also do not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind assistance and cooperation during the development of our project. Last but not the least, I acknowledge my friends for their contribution in the completion of the project.

Signature:

Name:

Roll no:

Date:

ABSTRACT

In this project development of a heads up display, which is supposed to aid military personnel in the battlefield, has been undertaken. A system which does not rely on external connections like internet, mobile networks etc has been developed keeping in mind the remote environment of battlefields. This system has facilities to sense its environment (temperature, humidity, GPS position, direction of movement) and display the sensed information to the user. The user can then view this information and use inbuilt radio communication facilities to relay the information to the base station.

LIST OF FIGURES

LIST OF ABBREVIATIONS

Pi	Raspberry Pi Zero WH v1.1 Development Board
GPIO	General Purpose Input Output
I2C	Inter Integrated Circuits
UART	Universal Asynchronous Receiver Transmitter
USB	Universal Serial Bus
GPS	Global Positioning System

CHAPTER 1

INTRODUCTION

In the battlefield having more information than the enemy may sometimes turn the tides in one's favor. Keeping this idea in mind this system has been developed to obtain information about the surroundings using electronic modules and relay them to the soldier as well as to the base station.

This system does not need any internet or mobile connectivity to operate and it has its own radio transmission system. An encrypted radio channel will ensure proper communication to and from the base station. It is also understood that such systems have already been implemented and are in use by the soldiers of the country. However, those systems are bulky and expensive. They can only be carried by at most one soldier in a group. The system which has been developed here has a very small signature, is cheap and can be supplied to each and every soldier in the party. Moreover, this system has the scope of addition of night vision system at a low cost. This system also makes use of Augmented Reality technology so that the soldiers can always view the information in front of them and not need to look at any display device separately.

The Augmented Reality system may, in future, enable us to add image recognition algorithms which may be able to identify threats from a distance (landmines, drones etc.). This has not been implemented in this project but this project has the scalabiliy to include image recognition algorithms which may help us achieve the mentioned functionalities.

THE PROPOSED SYSTEM

The heads up display system which has been developed in this project consists of a system on a chip computer which has the ability to control many electronic modules at a time. The heads up display also consists of a well arranged optical system which is necessary for the augmented reality display. A general purpose battery pack has also been used in the development of this system. However, if the project is to be used in actual battle situations one should consider the use of a better and longer lasting power source. The power source should strictly be a 5V 1A (max) power source.

The system has been implemented by using the components as mentioned below:

1. Raspberry Pi Zero WH v1.1 Development Board
2. 0.96" OLED display
3. QMC5883L Magnetometer module
4. Neo6m V2 GPS module
5. DHT11 temperature and humidity sensor module
6. C-media sound card
7. Tactile Switches
8. General purpose power bank
9. General purpose microphone
10. General purpose tools and equipments

The raspberry pi zero development board has the following pin configuration:

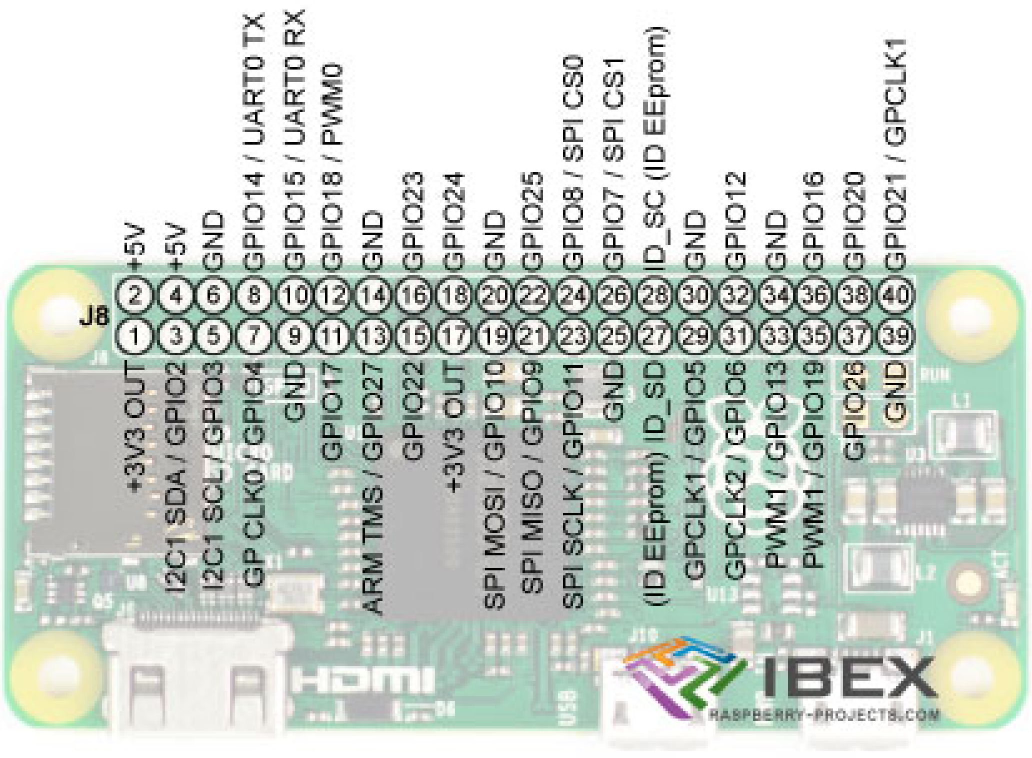


Illustration 1: Raspberry Pi Zero W pin labels

The pins which we have used in the development of this system are as follows: 1, 2, 3, 5, 6, 8, 10, 11, 13, 15, 39

The mentioned pins have the following uses:

Pin no. 1: 3.3 volt output given to connected electronic modules

Pin no. 2: 5 volt output given to connected electronic modules

Pin no. 3: SDA line of I2C bus

Pin no. 5: SCL/SCK line of I2C bus

A note about I2C*: I2C stands for Inter Integrated Circuits. This is a protocol which enables electronic devices to be connected in parallel like a bus. The electronic modules connected to I2C bus must be I2C compatible. Each*

module is given a unique address using which it can be accessed in the bus.
Pin no. 7: GPIO pin. This pin is used to send radio transmissions.
A note about GPIO: *GPIO stands for General Purpose Input Output. GPIO pins can be used to accept or supply voltage of 3.3 volt and can be used as data lines.*
Pin no. 6: Ground pin
Pin no. 8: UART TX line
Pin no. 10: UART RX line
A note about UART: *UART stands for Universal Asynchronous Receiver Transmitter. This is another protocol which helps in interfacing of two electronic systems. We will be using UART pins to connect GPS module to the Raspberry Pi.*
Pin no. 11: GPIO pin. Used for implementing a forward button. This forward button is used to scroll through the data displayed on the AR display.
Pin no. 13: GPIO pin. Used for implementing a backward button. This backward button is used to scroll through the data displayed on the AR display.
Pin no. 15: GPIO pin. Used for implementing a shutdown button. This button can be used to shut down the system when it is not in use in order to save power.
Pin no. 29: Ground pin.

Several libraries and programs have been used to control the connected modules. These libraries and programs have either been self written on taken from open source projects. Next, details about how each module has been connected to the Pi are given.

INTERFACING OF MODULES

We have used the following electronic modules:

1. QMC5883L Magnetometer module
2. Neo6m V2 GPS module
3. 0.96" OLED display
4. Tactile Switches (forward, backward and shutdown functions)
5. Microphone
6. Antenna for radio transmission
7. Power Bank

The intefacing of each module has been explained in detail in the following sections.

Interfacing of QMC5883L Magnetometer module

The QMC5883L magnetometer module is I2C compatible. Hence, it has been connected to the I2C bus on the Pi. The SDA pin of the module has been connected to the SDA pin of the Pi. The SCK/SCL pin of the module has been connected to the SCK/SCL pin of the Pi. In order to make the connections general purpose jumper wires have been used. However, in order to use the system on field one must consider soldering the wires instead of using jumper wires.

The QMC5883L module accepts a 3.3 volt power so the VCC/VDD pin has been connected to the 3.3v pin (pin no. 1) of the Pi. The ground pin of the module has been connected to any of the several ground pins available on the

Pi (in this case pin no. 39).

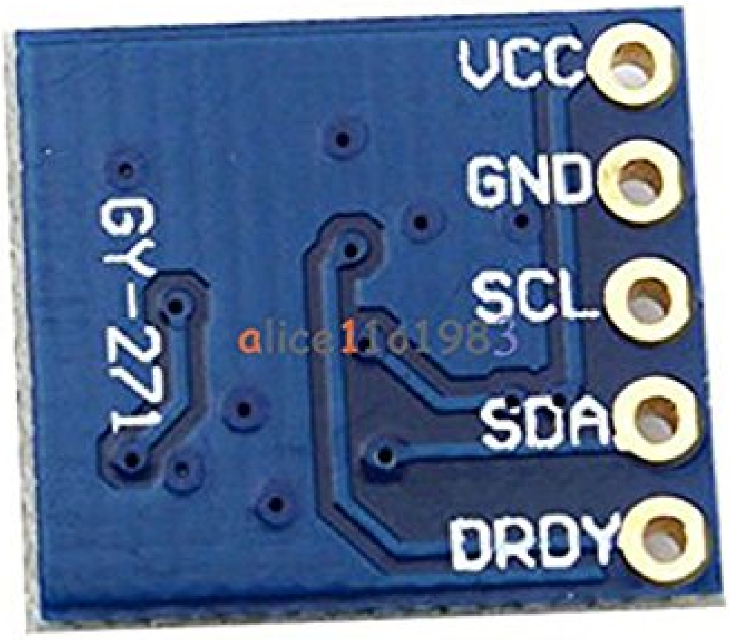


Illustration 2: QMC5883L magnetometer module

Interfacing of Neo6m V2 GPS module

The Neo6m V2 GPS module has a UART interface. Hence, we will connect it to the UART pins on the Pi. By default, the UART pins on the Pi are used to drive the Bluetooth functionality. Therefore, we will have to disable Bluetooth first in order to successfully use those pins for GPS. This has been discussed later in the software configuration portion.

The TX pin of the GPS module is connected to the RX pin of the Pi (this may

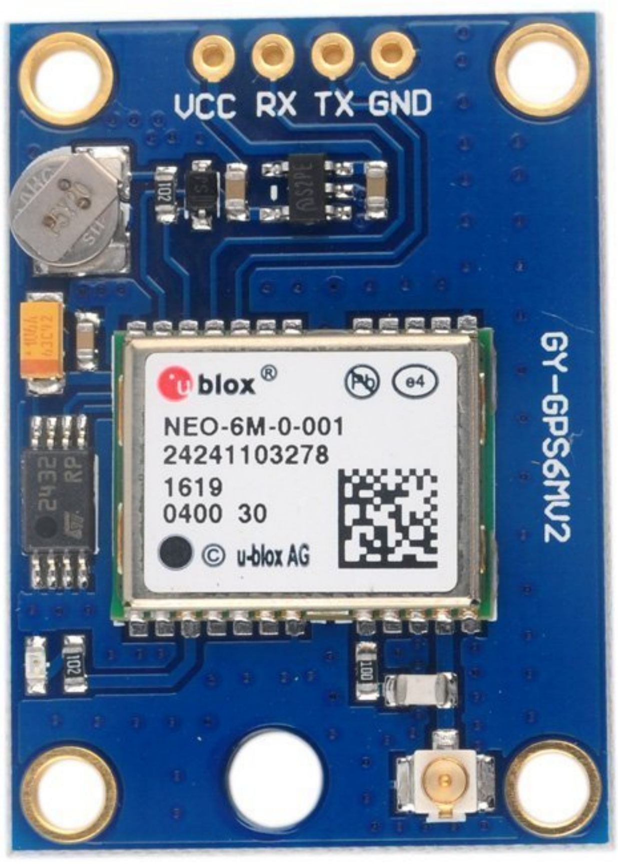


Illustration 3: GPS module

be counter intuitive at first but it is very logical to connect the receiving end of the Pi to the transmitting end of the GPS and vice versa. UART protocol works in this way only. This may not be true for other protocols like I2C or

SDA). The RX pin of the GPS module is connected to the TX pin of the Pi. GPS module can handle both a 3.3 volt supply as well as a 5 volt supply. Hence, we can connect it to any of the power pins. GPS, in this case, has been connected to a 5 volt power.The ground pin can be connected to any of the ground pins available on the Pi. In this case it has been connected to pin no. 6.

Interfacing of OLED display

The OLED display is I2C compatible. Hence, we will connect it to the I2C

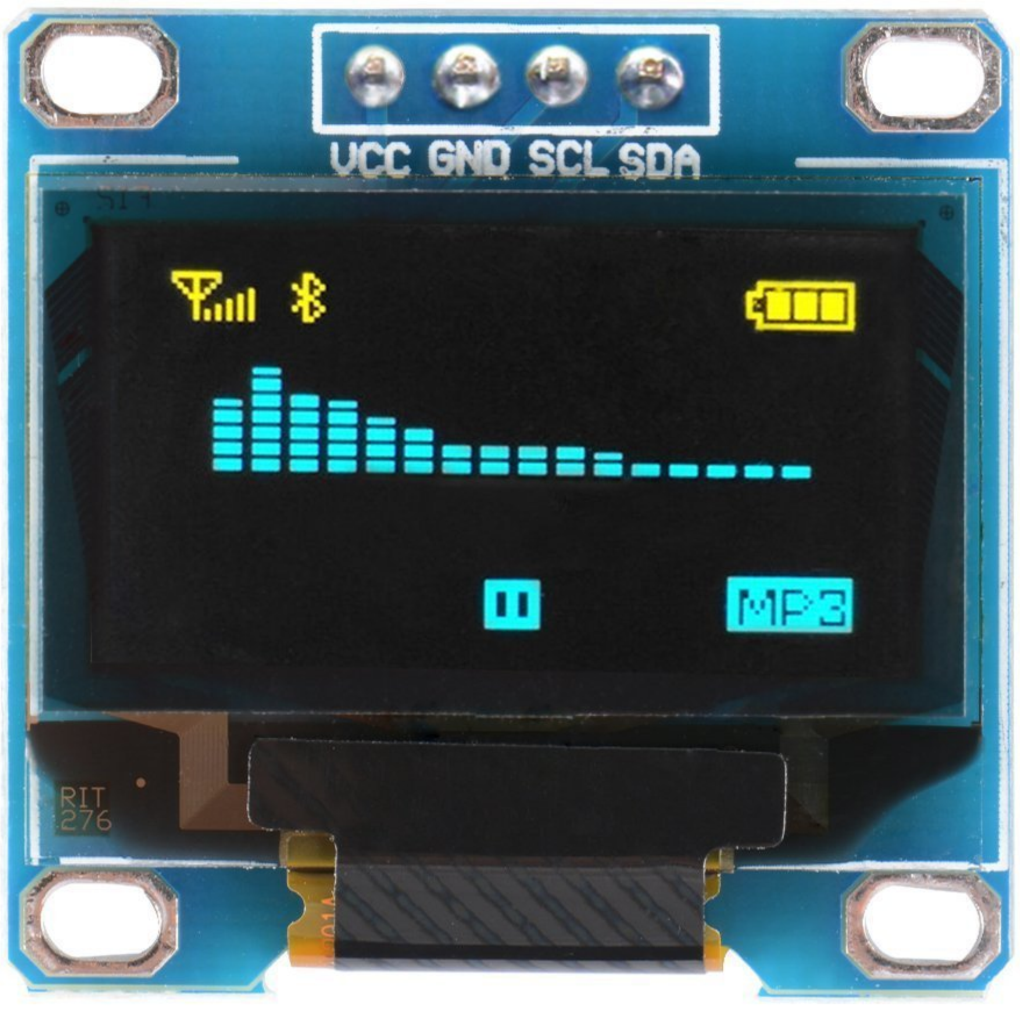


Illustration 4: OLED display

bus on the Pi. The SDA pin of the OLED is connected to the SDA pin of the Pi. The SCK/SCL pin of the OLED is connected to the SCK/SCL pin of the Pi. It should be noted here that the magnetometer module is also connected to the same pins. This is not a issue as we can connect both the modules in parallel. This is due to the fact that I2C protocol can be used to connected modules in the form of a bus. Each module can be accessed separately by using its address on the bus.

OLED display uses a 3.3 volt power so it has been connected to 3.3 volt pin (pin no. 1). The ground pin of the OLED has been connected to one of the many ground pins available on the Pi (in this case pin no. 39).

Interfacing of Tactile Switch (for forward button)

The tactile switch is an always off switch. When it is pressed then it goes to on state and when the force is released it automatically goes back to the off state. It is ideal for implementing key like functions similar to that available on a general purpose keyboard.

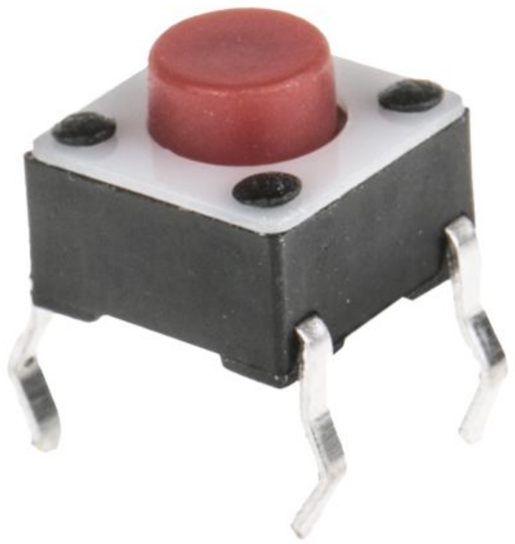


Illustration 5: Tactile Switch

Any one pair of diametrically opposite legs of a tactile switch are used to make a connection. In the case of the forward button implementation One pin of the switch has been connected to 3.3 volt pin of the raspberry pi. The diametrically opposite pin of the switch has been connected to the GPIO pin no. 15 of the Pi. When the switch is pressed then the diametrically opposite pins get connected and hence a closed ciruit is formed. This is used to send a 3.3 volt signal to the GPIO pin. This signal is processed in a program which will be discussed later.

Interfacing of Tactile Switch (for backward button)

In the case of the backward button the other diametrically opposite end of another tactile switch is connected to GPIO pin no. 13 of the Pi. Rest of the connections are same as mentioned above for the forward button.

Interfacing of Tactile Switch (for shutdown button)

In the case of the shutdown button the other diametrically opposite end of another tactile switch is connected to GPIO pin no. 11 of the Pi. Rest of the connections are same as mentioned above for the forward button.

Interfacing of a Microphone

The Raspberry Pi Zero does not have an in built audio jack. Therefore, an external USB sound card had to be used in order to connect a microphone to the Pi. C-media USB sound card has been used in this project and it is connected to the Pi via micro USB. The sound card has jacks for connecting microphones/speakers having a 3.5mm pin. The microphone has been connected to the sound card in this way. Later, the microphone is used to relay voice messages via radio transmission. Details of this will be discussed later.

Interfacing of Antenna

The GPIO pin no. 7 has been used as an antenna. A simple jumper cable has been connected to the pin no. 7 and it works as a very simple antenna with low range. In the future, this range can be increased by using a standard dipole antenna with a dedicated power supply.

Power Bank

The Raspberry Pi board is powered up using a general purpose power bank. This power bank provides a micro USB connection which can be connected to the micro USB power socket available in the Pi.

CHAPTER 2

THE OPERATING SYSTEM

Raspbian Stretch Operating System has been used in this project. This OS is available free of cost from the raspberry pi website. A micro SD card is required to make the Pi work because it is used as a hard disk. Hence, the OS is installed in a micro SD card and then the card is inserted into the Pi.

A .img file of the OS can be downloaded from the website and then a bootable SD card can be made. For doing this instructions can be found on the web.

In our windows environment the following softwares were used in chronological order in order to install the OS in the SD card:

1. SD card formatter (format the SD card)
2. Win32DiskImager (to install the .img file into the SD card)

After following the above two steps one can insert the SD into the Pi and power it on. The Pi will then be ready to function.

The above mentioned tools can be downloaded free of cost from the internet.

ACCESSING THE PI (SSH, FTP)

In order to communicate with the Pi and to send and receive files SSH and FTP protocols were used. However, in order to use these protocols one has to make some changes in the configuration of the Pi. This can be achieved by removing the SD card from the Pi and re-inserting it into any general purpose computer. From then on, one should open the SD card and follow the given steps:

1. Add an SSH file in the sd card

2. Add a wpa_supplicant.conf file in the sd card

3. In the wpa_supplicant add these lines:

```
country=IN
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1

network={
        ssid="testing"
        psk="testingPassword"
        scan_ssid=1
        key_mgmt=WPA-PSK
}
```

4. Go to config.txt in the sd card and add line at the end "enable_uart=1"

5. Go to config.txt in the sd card and add the line at the end "dtoverlay=dwc2"

6. Go to cmdline.txt in the sd card and add the line after rootwait (make sure to have only one space after rootwait otherwise the file may not get parsed) "modules-load=dwc2,g_ether"

7. Connect a USB cable to the USB end of raspberry pi zero and plug the other end to pc. SSH using terminal.

After following the above steps we can also use FTP to transfer files to and from the Pi. This can be achieved by using any FTP client (example, WinSCP).

PROGRAMS TO DRIVE THE MODULES

We have connected the modules to the Pi and have installed an OS in it. Now, we are ready to write our own programs in order to control the eletronic modules which we have connected to it. Along with writing of programs we may also need to change some settings of the Pi here and there. Everything will be discussed in details in the following sections.

Program for the magnetometer module

The QMC5883L magnetometer module is compatible with the I2C bus hence we have connected it to the I2C pins of the Pi. We will be using a JavaScript library in order to drive this module. This library goes by the name of Nodejs-QMC5883L and is freely available on Github. Using git we need to clone the repository in our Pi and then navigate to a file named test.js.

This file contains basic functions which can output the Azimuth value to the console. The magnetometer gives raw data at first. This raw data has to be transformed in some way in order to convert it to meaningful data showing the angle at which the magnetometer is facing. This transformed data is called the Azimuth value and it ranges from 0 to 360.

Taking readings from the I2C lines and converting raw data to the Azimuth value is being done by the library itself. Hence, not much attention has been paid to this. However, in place of logging the Azimuth value on the console, it is being saved in a file named azimuth.txt in string format. This file can then be accessed by any other program and the Azimuth value can be read.

Suppose the Azimuth value is represented by D. Then,

If D is greater than 337.25 degrees or less than 22.5 degrees – North

If D is between 292.5 degrees and 337.25 degrees – North-West

If D is between 247.5 degrees and 292.5 degrees – West

If D is between 202.5 degrees and 247.5 degrees – South-West

If D is between 157.5 degrees and 202.5 degrees – South

If D is between 112.5 degrees and 157.5 degrees – South-East

If D is between 67.5 degrees and 112.5 degrees – East

If D is between 0 degrees and 67.5 degrees – North-East

The above conditions will be applied to the program which will read the Azimuth values and tell the direction. The full code for the program discussed above will be provided in the CD.

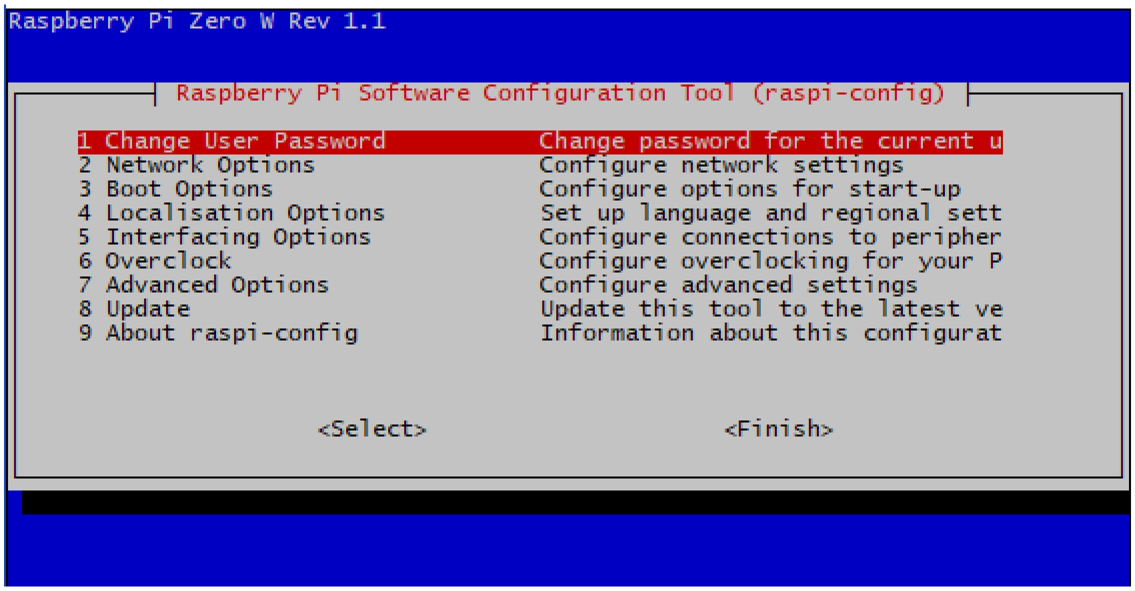
Program for GPS module

Since we have attached the GPS module to the UART pins we need to disable Pi's Bluetooth function. This can be done by logging into Pi using SSH and performing the following steps.

In the terminal type:

```
sudo raspi-config
```

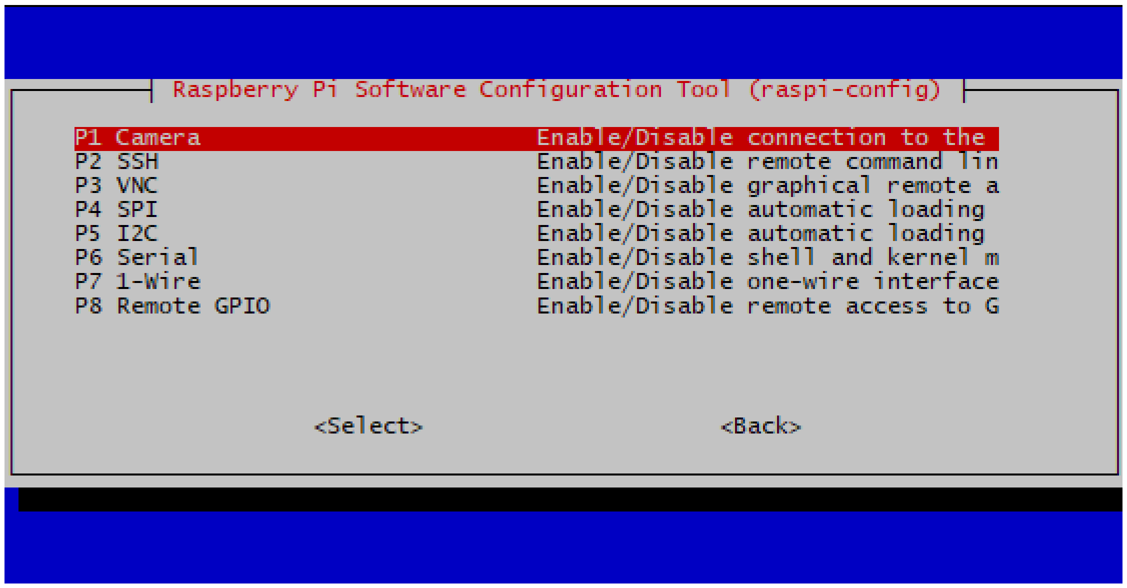
A window will open which will look like this:



```
Raspberry Pi Zero W Rev 1.1

        Raspberry Pi Software Configuration Tool (raspi-config)

  1 Change User Password          Change password for the current u
  2 Network Options               Configure network settings
  3 Boot Options                  Configure options for start-up
  4 Localisation Options          Set up language and regional sett
  5 Interfacing Options           Configure connections to peripher
  6 Overclock                     Configure overclocking for your P
  7 Advanced Options              Configure advanced settings
  8 Update                        Update this tool to the latest ve
  9 About raspi-config            Information about this configurat

                   <Select>                    <Finish>
```

Illustration 6: raspi-config screen 1

Go to interfacing options



```
        Raspberry Pi Software Configuration Tool (raspi-config)

  P1 Camera                       Enable/Disable connection to the
  P2 SSH                          Enable/Disable remote command lin
  P3 VNC                          Enable/Disable graphical remote a
  P4 SPI                          Enable/Disable automatic loading
  P5 I2C                          Enable/Disable automatic loading
  P6 Serial                       Enable/Disable shell and kernel m
  P7 1-Wire                       Enable/Disable one-wire interface
  P8 Remote GPIO                  Enable/Disable remote access to G

                   <Select>                    <Back>
```

Illustration 7: raspi-config screen 2

The select Serial. When Serial is selected then it will ask whether to enable serial login console. Select no to that. It will then ask whether to enable serial hardware. Select no to that too.

And this will disable the Bluetooth functionality at the UART lines and we can now use those lines to work with our GPS module.

Now, in the terminal we need to enter the following commands:

sudo apt-get update

sudo apt-get upgrade

sudo reboot

Then login again to SSH and enter more commands:

sudo nano /boot/config.txt

We need to change the contents of the config.txt file. At the bottom of the config.txt file add the followig lines:

```
dtparam=spi=on
dtoverlay=pi3-disable-bt
core_freq=250
enable_uart=1
force_turbo=1
```

Press Ctrl+x to exit and press y to save.

Type the following commands too:

sudo cp boot/cmdline.txt boot/cmdline_backup.txt

sudo nano /boot.cmdline.txt

Replace the entire contents of the cmdline.txt with the following:

```
dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline fsck.repair=yes rootwait quiet splash plymouth.ignore-serial-consoles modules-load=dwc2,g_ether
```

Now we need to disable the Pi getty service. To do this type the following commands:

sudo systemctl stop serial-getty@ttyS0.service

sudo systemctl disable serial-getty@ttyS0.service

We now must activate ttyAMA0 service. To do this we must type the following commands:

sudo systemctl enable serial-getty@ttyAMA0.service

When we read the data given by the GPS module we will receive NMEA sentences. These sentences contain various information that the GPS satellites relay to the GPS receivers. It is difficult to figure our the latitude and longitude just by looking at the NMEA sentences. Therefore, we will use a parser written in python in order to read these sentences.

In the terminal type:

sudo pip install pynmea

This will install the pynmea library which we can include in our python program in order to use the GPS module. Also note that we need to have pip installed in our Pi as well python serial library. To install the python serial library type the following commands in the terminal:

sudo pip install serial

The GPS program reads the data supplied by the GPS module and saves only the latitude and longitude part in a file called gps_data.txt in string format. This file can then be read by some other program which wishes to see this data. Entire program is provided in the CD.

Program for OLED display as well as the main program

By main program we mean the program which will be reading all the data written by rest of the programs and showing it to the user. This program will also include user interface functions discussed earlier that is, forward, backwards and shutdown.

The OLED is an I2C compatible eletronic module hence we will use the Pi's I2C bus to drive it. The tactile switches are connected to the GPIO pins. To use both these parts we need some libraries. Python libraries are being used in this project. In order to install them type the following commands in the terminal:

sudo pip install RPi.GPIO

sudo pip install smbus

sudo pip install luma.oled

Also clone the Adafruit_SSD1306 repository from github and install it. Both

luma.oled and Adafruit_SSD1306 has been used to drive the OLED display.
In order to communicate with the GPIO pins the RPi.GPIO library has been used.
The entire program is present in the accompanying CD.

OUT OF THE RABBIT HOLE

If one is not comfortable with following the above mentioned steps then one can just install the image file provided in the CD. That image file already has all the settings done and all the programs written and setup to start right after boot. This means that one just needs to install the .img file on an SD card. Insert the SD card in the Pi and do the physical interfacing of all the modules and the system is ready to go. One does not need to follow the steps given in the previous section if one does not want to.

CHAPTER 3

THE OPTICAL SYSTEM

The optical system which has been used to implement Augmented Reality feature in this project uses a mirror and a glass sheet. The arrangement of the mirror and the sheet is shown as follows:

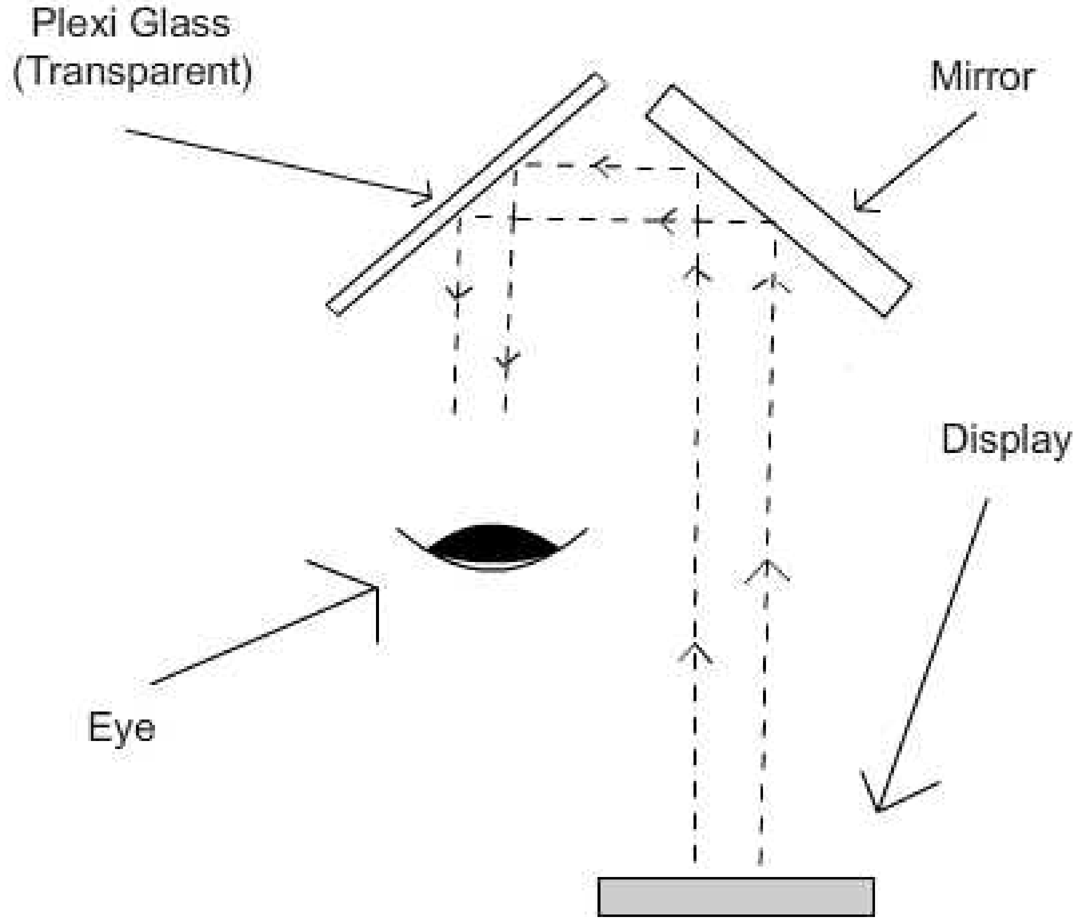


Illustration 8: Optical system

Here, light will originate at the display and will travel in straight line until it reaches the mirror. The mirror is kept at 45 degrees angle with respect to the

mirror. The light rays reflect off the mirror and reach the plexiglass. In doing so the image displayed by the display unit is inverted.
The plexiglass is placed at an angle of 45 degrees with respect to the display and 90 degrees with respect to the mirror. Here total internal reflection takes place and the light rays are bent towards the eye. The plexiglass is trasparent and therefore one can also see through it.
Due to the reason that the data displayed on the OLED is brighter than its surroundings, the light emerging due to the pixels only reach the eye noticeably. This creates an effect in which it seems that the text has been superimposed upon the environment.

SHORTCOMINGS OF THE OPTICAL SYSTEM (AND POSSIBLE WORKAROUNDS)

The mirror is placed at an angle of 45 degrees with respect to the display unit. When light from the display unit reflects from the mirror it forms a double image on the mirror. One image is brighter and the other is dull but is slightly shifted from the brighter image. This is a disadvantage of the optical system used in this project. This effect makes it harder to read the text. Further when light reaches the plexiglass it again undergoes the same effect but it is not so vivid as in the case of the mirror.

One possible way to overcome this is to replace the mirror and plexiglass arrangement with a right angled prism. This method was also tried in this project but it resulted in an even more shifted double image.

Finally, due to lack of time and resources, it was decided that it will be best to lay off the idea of augmented reality for the time being and mount the display on the wrist of the user (where the rest of the components will also be present). This method will result in a usable device which would not have been possible with the intricate optical system which was generating a double image. Future plans are to use an actual screen in front of the eyes and a camera close to the forehead. Both these components can then be used to effectively create an augmented reality experience.

CHAPTER 4
TESTING

Unit Testing

In this phase individual electronic components were tested.

Testing Raspberry Pi

After installing the OS in an SD card the rasperry pi was booted up and it was checked whether one can SSH into it or not. No problems were found in SSH connection as well as the session.

Testing OLED display

Since the OLED display cannot be tested separately hence it was interfaced with the Pi and then tested. It was seen that the OLED responded as intended. Sometimes it turned off due to exceptions encountered by the program. Exception handling code was written and this problem was removed.

Testing the GPS module

GPS module was particularly harder to test because it was required to let it connect to a satellite first. It took many days to make the GPS work. It was also first iterfaced with the Pi as it cannot be tested separately. It was found to work as intended.

Testing the magnetometer module

The magnetometer was also first intefaced with the Pi and then tested. It was found to have no problems.

Testing the sound card and microphone

The sound card was tested by making it play music. It was successfully able to do so. The mic was only tested after the FM transmission system was implemeted. The transmitter was programmed to transmit at 100.1 MHz. A general purpose radio was tuned to 100.1 MHz and voice input was given to the microphone. The voice given as input was heard at the general purpose radio. This feature was tested again and again by many personnel purely out of curiosity. The channel was not encrypted at that time hence any radio receiver could listen to the signal.

Integration Testing

All the components were combined in a single stage and tested. All programs were made to start after boot up of the system. Hence, in order to test the intergrated system one did not need to access the Pi over SSH.

The entire system was full of uncaught exceptions which caused a lot of problems. The programs were written again with exception handling code and this problem was solved.

Beta Testing

Members of the team tested the system one by one. The optical system was not working well. Hence, it was removed and the OLED alongwith the entire system was simply mounted on a general purpose glove which one can wear and use the system effectively.

The user interface was found to be problematic as the buttons had to be "long pressed" to have any effect. This problem was attributed to the the single core of Raspberry Pi Zero board and was not looked into much as it required overclocking the CPU which may have caused other problems (over heating and higher battery drain).

Conclusion of Testing

The system was found to be fit for civilian use. The system has not been tested for military use.

CHAPTER 5
FUTURE DEVELOPMENTS

MAP

If many users in a party are given this system and they use it in an unknown environment then they will be able to see each others' positions using the map feature which will be discussed here.
Each user will have the information about his/her cooordinates with the help of the GPS chip. This data can then be encrypted and relayed in all the directions using any radio transmitter (it is planned that NRF24L01 will be best for this purpose). This data can then be received by the peers of the user which are within range of the transmitter and they can decrypt it and obtain the coordinates of the user. Those peers who will not be in range will have the latest data of the user which was relayed when they were in range.

In this way every user in the party will relay his/her coordinates to one another. Using these coordinates one can easily design an algorithm which can plot the coordinates as points on a map. This map can then be displayed to the user. The user can then get a visual representation of the location of his/her peers. With the help of the map as well as the magnetometer the user will be able to locate any of his/her peer as he/she desires.
This feature will not depend on any external network or voice

communication. No human interference will be involved in sharing of coordinates as well plotting of the map.

NIGHT VISION

A better augmented reality system will make it possible to install a night vision system in this project. A camera and an IR source will be needed. The camera can be connected to the Raspberry Pi using a dedicated port for camera. IR source can just be an IR LED which can be powered by the 3.3 volt or the 5 volt power supply from the Pi itself.

As the IR illuminates the surroundings the camera can catch the IR and display its output to the AR system. This will lead to a cost effective implementation of night vision.

This system has not been installed in this project due to lack of time and resources.

A BETTER AUGMENTED REALITY SYSTEM

The augmented reality system which was planned for this project had shortcomings so it was not installed in the final prototype. However, plans are there to improve the AR system by using a display right infront of the eyes of the user and use convex lenses to help the use focus on the display. This system closely mimics the system used by Google Cardboard where a mobile phone with AR/VR capabilites is placed inside a specially designed cardboard box. This box can then be worn like glasses and then used. They use the mobile phone's camera and other sensors to successfully create the effect of augmented reality. The problem with Google Cardboard is that it is very bulky and the mobile phone with such capabilites is expensive. In this project a special screen will be designed and raspberry pi itself will be used for all the processing hence cutting the costs a lot.

IRNSS INSTEAD OF GPS

GPS is a system developed and maintained by the United States of America. This system depends on the network of satellites launched and maintained by the USA.

India is also developing its own network of satellites which will be used for positioning purposes. This system has been named IRNSS. IRNSS stands for Indian Regional Navigation Satellite System.

However, in order to use IRNSS the positioning module must be changed from GPS to whatever IRNSS provides in the future.

APPENDIX

NMEA Data

NMEA consists of sentences, the first word of which, called a data type, defines the interpretation of the rest of the sentence. Each Data type would have its own unique interpretation and is defined in the NMEA standard. The GGA sentence (shown below) shows an example that provides essential fix data. Other sentences may repeat some of the same information but will also supply new data. Whatever device or program that reads the data can watch for the data sentence that it is interested in and simply ignore other sentences that is doesn't care about. In the NMEA standard there are no commands to indicate that the gps should do something different. Instead each receiver just sends all of the data and expects much of it to be ignored. Some receivers have commands inside the unit that can select a subset of all the sentences or, in some cases, even the individual sentences to send. There is no way to indicate anything back to the unit as to whether the sentence is being read correctly or to request a re-send of some data you didn't get. Instead the receiving unit just checks the checksum and ignores the data if the checksum is bad figuring the data will be sent again sometime later.

There are many sentences in the NMEA standard for all kinds of devices that may be used in a Marine environment. Some of the ones that have applicability to gps receivers are listed below: (all message start with GP.)

- AAM - Waypoint Arrival Alarm
- ALM - Almanac data
- APA - Auto Pilot A sentence
- APB - Auto Pilot B sentence
- BOD - Bearing Origin to Destination
- BWC - Bearing using Great Circle route
- DTM - Datum being used.
- GGA - Fix information
- GLL - Lat/Lon data
- GRS - GPS Range Residuals
- GSA - Overall Satellite data
- GST - GPS Pseudorange Noise Statistics
- GSV - Detailed Satellite data
- MSK - send control for a beacon receiver
- MSS - Beacon receiver status information.
- RMA - recommended Loran data
- RMB - recommended navigation data for gps
- RMC - recommended minimum data for gps
- RTE - route message
- TRF - Transit Fix Data

- STN - Multiple Data ID
- VBW - dual Ground / Water Spped
- VTG - Vector track an Speed over the Ground
- WCV - Waypoint closure velocity (Velocity Made Good)
- WPL - Waypoint Location information
- XTC - cross track error
- XTE - measured cross track error
- ZTG - Zulu (UTC) time and time to go (to destination)
- ZDA - Date and Time

Some gps receivers with special capabilities output these special messages.

- HCHDG - Compass output
- PSLIB - Remote Control for a DGPS receiver

In addition some GPS receivers can mimic Loran-C receivers by outputing the LC prefix in some of their messages so that they can be used to interface to equipment that is expecting this prefix instead of the GP one.

The last version 2 iteration of the NMEA standard was 2.3. It added a mode indicator to several sentences which is used to indicate the kind of fix the receiver currently has. This indication is part of the signal integrity information needed by the FAA. The value can be A=autonomous, D=differential, E=Estimated, N=not valid, S=Simulator. Sometimes there can be a null value as well. Only the A and D values will correspond to an Active and reliable Sentence. This mode character has been added to the RMC, RMB, VTG, and GLL, sentences and optionally some others including the BWC and XTE sentences.

If you are interfacing a GPS unit to another device, including a computer program, you need to ensure that the receiving unit is given all of the sentences that it needs. If it needs a sentence that your GPS does not send then the interface to that unit is likely to fail. Here is a Link for the needs of some typical programs. The sentences sent by some typical receivers include:

NMEA 2.0

Name	Garmin	Magellan	Lowrance	SiRF	Notes:
GPAPB	N	Y	Y	N	Auto Pilot B
GPBOD	Y	N	N	N	bearing, origin to destination - earlier G-12's do not transmit this
GPGGA	Y	Y	Y	Y	fix data
GPGLL	Y	Y	Y	Y	Lat/Lon data - earlier G-12's do not transmit this
GPGSA	Y	Y	Y	Y	overall satellite reception data, missing on some Garmin models
GPGSV	Y	Y	Y	Y	detailed satellite data, missing on some Garmin models
GPRMB	Y	Y	Y	N	minimum recommended data when following a route
GPRMC	Y	Y	Y	Y	minimum recommended data
GPRTE	Y	U	U	N	route data, only when there is an active route. (this is sometimes bidirectional)
GPWPL	Y	Y	U	N	waypoint data, only when there is an active route (this is sometimes bidirectional)

NMEA 1.5 - some units do not support version 1.5. Lowrance units provide the ability to customize the NMEA output by sentences so that you can develop your own custom sentence structure.

Name	Garmin	Magellan	Notes:
GPAPA	N	Y	Automatic Pilot A
GPBOD	Y	N	bearing origin to destination - earlier G-12's do not send this
GPBWC	Y	Y	bearing to waypoint using great circle route.
GPGLL	Y	Y	lat/lon - earlier G-12's do not send this
GPRMC	Y	N	minimum recommend data
GPRMB	Y	N	minimum recommended data when following a route
GPVTG	Y	Y	vector track and speed over ground
GPWPL	Y	N	waypoint data (only when active goto)
GPXTE	Y	Y	cross track error

The NMEA 2.3 output from the Garmin Legend, Vista, and perhaps some others include the BWC, VTG, and XTE sentences.

The Trimble Scoutmaster outputs: APA, APB, BWC, GGA, GLL, GSA, GSV, RMB, RMC, VTG, WCV, XTE, ZTG.

The Motorola Encore outputs: GGA, GLL, GSV, RMC, VTG, ZDA and a proprietary sentence PMOTG.

Units based on the SiRF chipset can output: GGA, GLL, GSA, GSV, RMC, and VTG. What is actually output is based on which sentences are selected by the user or application program. See below for more details. Some implementations have enhanced the SiRF capabilities with other sentences as well by changing the firmware. For example, the u-blox receivers add ZDA and some proprietary sentences to the above list of sentences. Check your documentation for more details.

Garmin receivers send the following Proprietary Sentences:

- PGRME (estimated error) - not sent if set to 0183 1.5
- PGRMM (map datum)
- PGRMZ (altitude)
- PSLIB (beacon receiver control)

Note that Garmin converts lat/lon coordinates to the datum chosen by the user when sending this data. This is indicated in the proprietary sentence PGRMM. This can help programs that use maps with other datums but is not an NMEA standard. Be sure and set your datum to WGS84 on Garmin units when communicating to other NMEA devices.

Magellan also converts lat/lon coordinates to the datum chosen on the receiver but do not indicate this in a message. Magellan units use proprietary sentences for waypoint maintenance and other tasks. They use a prefix of PMGN for this data.

Most other units always output NMEA messages in the WGS84 datum. Be sure and check the user documentation to be sure.

It is possible to just view the information presented on the NMEA interface using a simple terminal program. If the terminal program can log the session then you can build a history of the entire session into a file. More sophisticated logging programs can filter the messages to only certain sentences or only collect sentences at prescribed intervals. Some computer programs that provide real time display and logging actually save the log in an ascii format that can be viewed with a text editor or used independently from the program that generated it.

NMEA input

Some units also support an NMEA input mode. While not too many programs support this mode it does provide a standardized way to update or add waypoint and route data. Note that there is no handshaking or commands in NMEA mode so you just send the data in the correct sentence and the unit will accept the data and add or overwrite the information in memory. If the data is not in the correct format it will simply be ignored. A carriage return/line feed sequence is required. If the waypoint name is the same you will overwrite existing data but no warning will be issued. The sentence construction is identical to what the unit downloads so you can, for example, capture a WPL sentence from one unit and then send that same sentence to another unit but be careful if the two units support waypoint names of different lengths since the receiving unit might truncate the name and overwrite a waypoint accidently. If you create a sentence from scratch you should create a correct checksum. Be sure you know and have set you unit to the correct datum. Many units support the input of WPL sentences and a few support RTE as well.

On NMEA input the receiver stores information based on interpreting the sentence itself. While some receivers accept standard NMEA input this can only be used to update a waypoint or similar task and not to send a command to the unit. Proprietary input sentences could be used to send commands. Since the Magellan upload and download maintenance protocol is based on NMEA sentences they support a modified WPL message that adds comments, altitude, and icon data.

Some marine units may accept input for alarms such as deep or shallow water based on the DPT sentence or MTW to read the water temperature. For example the Garmin Map76 supports DPT, MTW (temperature), and VHW (speed) input sentences. Other units may use NMEA input to provide initialization data via proprietary sentences, or to select which NMEA sentences to output.

Decode of selected position sentences

The most important NMEA sentences include the GGA which provides the current Fix data, the RMC which provides the minimum gps sentences information, and the GSA which provides the Satellite status data.

GGA - essential fix data which provide 3D location and accuracy data.

```
 $GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47

Where:
     GGA          Global Positioning System Fix Data
     123519       Fix taken at 12:35:19 UTC
     4807.038,N   Latitude 48 deg 07.038' N
     01131.000,E  Longitude 11 deg 31.000' E
     1            Fix quality: 0 = invalid
                               1 = GPS fix (SPS)
                               2 = DGPS fix
                               3 = PPS fix
                               4 = Real Time Kinematic
                               5 = Float RTK
                               6 = estimated (dead reckoning) (2.3 feature)
                               7 = Manual input mode
                               8 = Simulation mode
     08           Number of satellites being tracked
     0.9          Horizontal dilution of position
     545.4,M      Altitude, Meters, above mean sea level
     46.9,M       Height of geoid (mean sea level) above WGS84
                      ellipsoid
     (empty field) time in seconds since last DGPS update
     (empty field) DGPS station ID number
     *47          the checksum data, always begins with *
```

If the height of geoid is missing then the altitude should be suspect. Some non-standard implementations report altitude with respect to the ellipsoid rather than geoid altitude. Some units do not report negative altitudes at all. This is the only sentence that reports altitude.

GSA - GPS DOP and active satellites. This sentence provides details on the nature of the fix. It includes the numbers of the satellites being used in the current solution and the DOP. DOP (dilution of precision) is an indication of the effect of satellite geometry on the accuracy of the fix. It is a unitless number where smaller is better. For 3D fixes using 4 satellites a 1.0 would be considered to be a perfect number, however for overdetermined solutions it is possible to see numbers below 1.0.

There are differences in the way the PRN's are presented which can effect the ability of some programs to display this data. For example, in the example shown below there are 5 satellites in the solution and the null fields are scattered indicating that the almanac would show satellites in the null positions that are not being used as part of this solution. Other receivers might output all of the satellites used at the beginning of the sentence with the null field all stacked up at the end. This difference accounts for some satellite display programs not always being able to display the satellites being tracked. Some units may show all satellites that have ephemeris data without regard to their use as part of the solution but this is non-standard.

```
  $GPGSA,A,3,04,05,,09,12,,,24,,,,,2.5,1.3,2.1*39

Where:
     GSA      Satellite status
     A        Auto selection of 2D or 3D fix (M = manual)
     3        3D fix - values include: 1 = no fix
                                       2 = 2D fix
                                       3 = 3D fix
     04,05... PRNs of satellites used for fix (space for 12)
     2.5      PDOP (dilution of precision)
     1.3      Horizontal dilution of precision (HDOP)
     2.1      Vertical dilution of precision (VDOP)
     *39      the checksum data, always begins with *
```

GSV - Satellites in View shows data about the satellites that the unit might be able to find based on its viewing mask and almanac data. It also shows current ability to track this data. Note that one GSV sentence only can provide data for up to 4 satellites and thus there may need to be 3 sentences for the full information. It is reasonable for the GSV sentence to contain more satellites than GGA might indicate since GSV may include satellites that are not used as part of the solution. It is not a requirment that the GSV sentences all appear in sequence. To avoid overloading the data bandwidth some receivers may place the various sentences in totally different samples since each sentence identifies which one it is.

The field called SNR (Signal to Noise Ratio) in the NMEA standard is often referred to as signal strength. SNR is an indirect but more useful value that raw signal strength. It can range from 0 to 99 and has units of dB according to the NMEA standard, but the various manufacturers send different ranges of numbers with different starting numbers so the values themselves cannot necessarily be used to evaluate different units. The range of working values in a given gps will usually show a difference of about 25 to 35 between the lowest and highest values, however 0 is a special case and may be shown on satellites that are in view but not being tracked.

```
  $GPGSV,2,1,08,01,40,083,46,02,17,308,41,12,07,344,39,14,22,228,45*75

Where:
      GSV          Satellites in view
      2            Number of sentences for full data
      1            sentence 1 of 2
      08           Number of satellites in view

      01           Satellite PRN number
      40           Elevation, degrees
      083          Azimuth, degrees
      46           SNR - higher is better
           for up to 4 satellites per sentence
      *75          the checksum data, always begins with *
```

RMC - NMEA has its own version of essential gps pvt (position, velocity, time) data. It is called RMC, The Recommended Minimum, which will look similar to:

```
$GPRMC,123519,A,4807.038,N,01131.000,E,022.4,084.4,230394,003.1,W*6A

Where:
     RMC          Recommended Minimum sentence C
     123519       Fix taken at 12:35:19 UTC
     A            Status A=active or V=Void.
     4807.038,N   Latitude 48 deg 07.038' N
     01131.000,E  Longitude 11 deg 31.000' E
     022.4        Speed over the ground in knots
     084.4        Track angle in degrees True
     230394       Date - 23rd of March 1994
     003.1,W      Magnetic Variation
     *6A          The checksum data, always begins with *
```

Note that, as of the 2.3 release of NMEA, there is a new field in the RMC sentence at the end just prior to the checksum. For more information on this field see here.

GLL - Geographic Latitude and Longitude is a holdover from Loran data and some old units may not send the time and data active information if they are emulating Loran data. If a gps is emulating Loran data they may use the LC Loran prefix instead of GP.

```
  $GPGLL,4916.45,N,12311.12,W,225444,A,*1D

Where:
     GLL          Geographic position, Latitude and Longitude
     4916.46,N    Latitude 49 deg. 16.45 min. North
     12311.12,W   Longitude 123 deg. 11.12 min. West
     225444       Fix taken at 22:54:44 UTC
     A            Data Active or V (void)
     *iD          checksum data
```

Note that, as of the 2.3 release of NMEA, there is a new field in the GLL sentence at the end just prior to the checksum. For more information on this field see here.

VTG - Velocity made good. The gps receiver may use the LC prefix instead of GP if it is emulating Loran output.

```
  $GPVTG,054.7,T,034.4,M,005.5,N,010.2,K*48

where:
        VTG          Track made good and ground speed
        054.7,T      True track made good (degrees)
        034.4,M      Magnetic track made good
        005.5,N      Ground speed, knots
        010.2,K      Ground speed, Kilometers per hour
        *48          Checksum
```

Note that, as of the 2.3 release of NMEA, there is a new field in the VTG sentence at the end just prior to the checksum. For more information on this field see here.

Receivers that don't have a magnetic deviation (variation) table built in will null out the Magnetic track made good.

Decode of some Navigation Sentences

WPL - Waypoint Location data provides essential waypoint data. It is output when navigating to indicate data about the destination and is sometimes supported on input to redefine a waypoint location. Note that waypoint data as defined in the standard does not define altitude, comments, or icon data. When a route is active, this sentence is sent once for each waypoint in the route, in sequence. When all waypoints have been reported, the RTE sentence is sent in the next data set. In any group of sentences, only one WPL sentence, or an RTE sentence, will be sent.

```
$GPWPL,4807.038,N,01131.000,E,WPTNME*5C

With an interpretation of:

     WPL         Waypoint Location
     4807.038,N  Latitude
     01131.000,E Longitude
     WPTNME      Waypoint Name
     *5C         The checksum data, always begins with *
```

AAM - Waypoint Arrival Alarm is generated by some units to indicate the Status of arrival (entering the arrival circle, or passing the perpendicular of the course line) at the destination waypoint.

```
  $GPAAM,A,A,0.10,N,WPTNME*32

Where:
    AAM    Arrival Alarm
    A      Arrival circle entered
    A      Perpendicular passed
    0.10   Circle radius
    N      Nautical miles
    WPTNME Waypoint name
    *32    Checksum data
```

APB - Autopilot format B is sent by some gps receivers to allow them to be used to control an autopilot unit. This sentence is commonly used by autopilots and contains navigation receiver warning flag status, cross-track-error, waypoint arrival status, initial bearing from origin waypoint to the destination, continuous bearing from present position to destination and recommended heading-to-steer to destination waypoint for the active navigation leg of the journey.

Note: some autopilots, Robertson in particular, misinterpret "bearing from origin to destination" as "bearing from present position to destination". This is likely due to the difference between the APB sentence and the APA sentence. for the APA sentence this would be the correct thing to do for the data in the same field. APA only differs from APB in this one field and APA leaves off the last two fields where this distinction is clearly spelled out. This will result in poor performance if the boat is sufficiently off-course that the two bearings are different.

```
  $GPAPB,A,A,0.10,R,N,V,V,011,M,DEST,011,M,011,M*3C

where:
    APB     Autopilot format B
    A       Loran-C blink/SNR warning, general warning
    A       Loran-C cycle warning
    0.10    cross-track error distance
    R       steer Right to correct (or L for Left)
    N       cross-track error units - nautical miles (K for kilometers)
    V       arrival alarm - circle
    V       arrival alarm - perpendicular
    011,M   magnetic bearing, origin to destination
    DEST    destination waypoint ID
    011,M   magnetic bearing, present position to destination
    011,M   magnetic heading to steer (bearings could True as 033,T)
```

BOD - Bearing - Origin to Destination shows the bearing angle of the line, calculated at the origin waypoint, extending to the destination waypoint from the origin waypoint for the active navigation leg of the journey.

```
  $GPBOD,045.,T,023.,M,DEST,START*01

where:
        BOD          Bearing - origin to destination waypoint
        045.,T       bearing 045 True from "START" to "DEST"
        023.,M       bearing 023 Magnetic from "START" to "DEST"
        DEST         destination waypoint ID
        START        origin waypoint ID
        *01          checksum
```

BWC - Bearing & Distance to Waypoint using a Great Circle route. Time (UTC) and distance & bearing to, and location of, a specified waypoint from present position along the great circle path.

```
  $GPBWC,225444,4917.24,N,12309.57,W,051.9,T,031.6,M,001.3,N,004*29

where:
        BWC          Bearing and distance to waypoint - great circle
        225444       UTC time of fix 22:54:44
        4917.24,N    Latitude of waypoint
        12309.57,W   Longitude of waypoint
        051.9,T      Bearing to waypoint, degrees true
        031.6,M      Bearing to waypoint, degrees magnetic
        001.3,N      Distance to waypoint, Nautical miles
        004          Waypoint ID
        *29          checksum
```

RMB - The recommended minimum navigation sentence is sent whenever a route or a goto is active. On some systems it is sent all of the time with null data. The Arrival alarm flag is similar to the arrival alarm inside the unit and can be decoded to drive an external alarm. Note the use of leading zeros in this message to preserve the character spacing. This is done, I believe, because some autopilots may depend on exact character spacing.

```
  $GPRMB,A,0.66,L,003,004,4917.24,N,12309.57,W,001.3,052.5,000.5,V*20

where:
           RMB          Recommended minimum navigation information
           A            Data status A = OK, V = Void (warning)
           0.66,L       Cross-track error (nautical miles, 9.99 max),
                                steer Left to correct (or R = right)
           003          Origin waypoint ID
           004          Destination waypoint ID
           4917.24,N    Destination waypoint latitude 49 deg. 17.24 min. N
           12309.57,W   Destination waypoint longitude 123 deg. 09.57 min. W
           001.3        Range to destination, nautical miles (999.9 max)
           052.5        True bearing to destination
           000.5        Velocity towards destination, knots
           V            Arrival alarm  A = arrived, V = not arrived
           *20          checksum
```

RTE - RTE is sent to indicate the names of the waypoints used in an active route. There are two types of RTE sentences. This route sentence can list all of the waypoints in the entire route or it can list only those still ahead. Because an NMEA sentence is limited to 80 characters there may need to be multiple sentences to identify all of the waypoints. The data about the waypoints themselves will be sent in subsequent WPL sentences which will be sent in future cycles of the NMEA data.

```
  $GPRTE,2,1,c,0,W3IWI,DRIVWY,32CEDR,32-29,32BKLD,32-I95,32-US1,BW-32,BW-198*69

Where:
           RTE          Waypoints in active route
           2            total number of sentences needed for full data
           1            this is sentence 1 of 2
           c            Type c = complete list of waypoints in this route
                        w = first listed waypoint is start of current leg
           0            Route identifier
           W3IWI,...    Waypoint identifiers (names)
           *69          checksum
```

XTE - Measured cross track error is a small subset of the RMB message for compatibility with some older equipment designed to work with Loran. Note that the same limitations apply to this message as the ones in the RMB since it is expected to be decoded by an autopilot.

```
  $GPXTE,A,A,0.67,L,N*6F

Where:
           XTE          Cross track error, measured
           A            General warning flag V = warning
                                (Loran-C Blink or SNR warning)
           A            Not used for GPS (Loran-C cycle lock flag)
           0.67         cross track error distance
           L            Steer left to correct error (or R for right)
           N            Distance units - Nautical miles
           *6F          checksum
```

Other sentences that may be useful

ALM - GPS Almanac Data contains GPS week number, satellite health and the complete almanac data for one satellite. Multiple messages may be transmitted, one for each satellite in the GPS constellation, up to maximum of 32 messages. Note that these sentences can take a long time to send so they are not generally sent automatically by the gps receiver. (Sorry I don't have an exact example of the sentence.) Note that this sentence breaks the 80 character rule. Also note that this sentence is often accepted as input so that you can preload a new almanac in a receiver.

```
     $GPALM,A.B,C.D,E,F,hh,hhhh,...

Where:
       ALM   Almanac Data being sent
       A     Total number of messages
       B     Message number
       C     Satellite PRN number
       D     GPS week number (0-1023)
       E     Satellite health (bits 17-24 of message)
       F     eccentricity
       hh    t index OA, almanac reference time
       hhhh  sigma index 1, inclination angle
       ...   OMEGADOT rate of right ascension
             SQRA(A) root of semi-major axis
             Omega, argument of perigee
             Omega index 0, longitude of ascension node
             M index 0, mean anomaly
             a index f0, clock parameter
             a index f1, clock parameter
```

HCHDG - Compass output is used on Garmin etrex summit, vista , and 76S receivers to output the value of the internal flux-gate compass. Only the magnetic heading and magnetic variation is shown in the message.

```
  $HCHDG,101.1,,,7.1,W*3C

where:
     HCHDG    Magnetic heading, deviation, variation
     101.1    heading
     ,,       deviation (no data)
     7.1,W    variation
```

ZDA - Data and Time

```
  $GPZDA,hhmmss.ss,dd,mm,yyyy,xx,yy*CC
  $GPZDA,201530.00,04,07,2002,00,00*60

where:
        hhmmss    HrMinSec(UTC)
        dd,mm,yyy Day,Month,Year
        xx        local zone hours -13..13
        yy        local zone minutes 0..59
        *CC       checksum
```

MSK - Control for a Beacon Receiver

```
  $GPMSK,318.0,A,100,M,2*45

where:
       318.0      Frequency to use
       A          Frequency mode, A=auto, M=manual
       100        Beacon bit rate
       M          Bitrate, A=auto, M=manual
       2          frequency for MSS message status (null for no status)
       *45        checksum
```

MSS - Beacon Receiver Status

```
  $GPMSS,55,27,318.0,100,*66

where:
       55         signal strength in dB
       27         signal to noise ratio in dB
       318.0      Beacon Frequency in KHz
       100        Beacon bitrate in bps
       *66        checksum
```

Proprietary Sentences

Proprietary sentences can either be output from the gps or used as input to control information. They always start with P which is followed by a 3 character manufactures code and additional characters to define the sentence type.

Garmin

The following are Garmin proprietary sentences. "P" denotes proprietary, "GRM" is Garmin's manufacturer code, and "M" or "Z" indicates the specific sentence type. Note that the PGRME sentence is not set if the output is set to NMEA 1.5 mode.

```
  $PGRME,15.0,M,45.0,M,25.0,M*1C

where:
     15.0,M       Estimated horizontal position error in meters (HPE)
     45.0,M       Estimated vertical error (VPE) in meters
     25.0,M       Overall spherical equivalent position error


  $PGRMZ,93,f,3*21

where:
      93,f         Altitude in feet
      3            Position fix dimensions 2 = user altitude
                                           3 = GPS altitude
   This sentence shows in feet, regardless of units shown on the display.
   Note that for units with an altimeter this will be altitude computed
   by the internal altimeter.


  $PGRMM,NAD27 Canada*2F
     Currently active horizontal datum
```

PSLIB

Proprietary sentences are used to control a Starlink differential beacon receiver. (Garmin's DBR is Starlink compatible as are many others.) When the GPS receiver is set to change the DBR frequency or b/s rate, the "J" sentence is replaced (just once) by (for example): $PSLIB,320.0,200*59 to set the DBR to 320 KHz, 200 b/s.

```
      $PSLIB,,,J*22   Status request
      $PSLIB,,,K*23   configuration request
```

These two sentences are normally sent together in each group of sentences from the GPS. The three fields are: Frequency, bit Rate, Request Type. The value in the third field may be: J = status request, K = configuration request, or null (blank) = tuning message. The correct values for frequency range from 283.5-325.0 KHz while the bit rate can be set to 0, 25, 50, 100 or 200 bps.

Magellan

Magellan uses proprietary sentences to do all of their waypoint and route maintenance. They use the MGN prefix for their sentences. This use is documented in their interface specification and will not be repeated here. However, they also send proprietary sentences to augment the gps data just like Garmin does. Here is an example of a sentence sent by the GPS Companion product:

```
  $PMGNST,02.12,3,T,534,05.0,+03327,00*40

where:
      ST      status information
      02.12   Version number?
      3       2D or 3D
      T       True if we have a fix False otherwise
      534     numbers change - unknown
      05.0    time left on the gps battery in hours
      +03327  numbers change (freq. compensation?)
      00      PRN number receiving current focus
      *40    checksum
```

A tracklog on a Meridian is made up of propretary sentences that look like:

```
$PMGNTRK,4322.061,N,07948.473,W,00116,M,173949.42,A,,020602*67
$PMGNTRK,4322.058,N,07948.483,W,00090,M,174202.45,A,,020602*69.

where

      TRK       Tracklog
      4322.071  Latitude
      N         North or South
      07948.473 Longitude
      W         East or West
      00116     Altitude
      M         Meters or Feet
      173949.42 UTC time
      A         Active or Void
      ,,        Track Name
      020602    date
      *67       checksum
```

Motorola

The **PMOTG** is used by Motorola Oncore receivers to send a command to the receiver. This command is used to set the output of the sentence to a particular frequency in seconds (or to 0) or to switch the output formula to motorola binary, gps, or loran.

```
  $PMOTG,xxx,yyyy

where:
      xxx    the sentence to be controlled
      yyyy   the time interval (0-9999 seconds)

or $PMOTG,FOR,y

where:
      y    MPB=0, GPS=1, Loran=2
```

Rockwell International

The Rockwell chipset is used on a number of gps receivers. It outputs some proprietary sentences with the **PRWI** prefix and accepts input from some special sentences similar to the approach used by Magellan. It can also be switched to a separate binary mode using a proprietary sentence. The input sentence most used to initialize the unit is $PRWIINIT and one output sentence is $PRWIRID

```
  $PRWIRID,12,01.83,12/15/97,0003,*42

where:
     $PRWIRID
     12         12 channel unit
     01.83      software version
     12/15/97   software date
     0003       software options (HEX value)
                Bit 0 minimize ROM usage
                Bit 1 minimize RAM usage
     *42        checksum
```

An input sentence that will define which NMEA sentences are to be output from the Rockwell unit is:

```
  $PRWIILOG,GGA,A,T,1,0

where:
   $PRWIILOG
   GGA        type of sentence
   A          A=activate, V=deactivate
   T          cyclic
   1          every 1 second
   0          ??
```

The initialization sentence which can be input to speed up acquisition looks like:

```
$PRWIINIT,V,,,4308.750,N,07159.791,W,100.0,0.0,M,0.0,T,175244,230503*77

where:
   $PRWIINIT     INIT = initialization
   V             V = reset, A = no reset
   ,,            Reserved for future use
   4308.750      Latitude
   N             N = North, S = South
   07159.791     Longitude
   W             W = West, E = East
   100.0         Altitude in meters
   0.0           Speed
   M             M = m/s, N = knots, K = km/hr
   0.0           Heading
   T             T = True, M = Magnetic
   175244        UTC time (hour, min, sec)
   230503        UTC date (day, month, year)
   *77           Checksum
```

Note: Commas may be used to signify using existing data. If units are supplied then the data must be present. Speed and direction must be supplied together. Lat/Lon must be supplied together. UTC time and date must be supplied together. If heading is magnetic then lat/lon needs to be supplied along with UTC time and date.

The sentences available for the Rockwell Jupiter chipset are: GGA, GSA, GSV, VTG, RMC and some proprietary sentences.

SiRF

The SiRF line of chips support several input sentences that permit the user to customize the way the chip behaves. In addition SiRF has a binary protocol that is even more powerful permitting different implementations to behave entirely differently. However, most applications do not attempt to customize the behavior so a user will need to make sure that the any customization is compatible with the application they are planning to use. There are 5 input sentences defined that begin with $PSRF which is followed by three digits. Each sentence takes a fix amount of input fields which must exist, no null fields, and is terminated with the standard CR/LF sequence. The checksum is required.

The sentences 100 and 102 set the serial ports. 100 sets the main port A while 102 sets the DGPS input port B. 100 has an extra field that can be used to switch the interface to binary mode. Binary mode requires 8 bits, 1 stop bit, no parity. There is a command in binary mode that will switch the interface back to NMEA. Do not use the NMEA command to switch to binary mode unless you have the ability to switch it back. You could render your gps inoperative.

```
 $PSRF100,0,9600,8,1,0*0C
 $PSRF102,9600,8,1,0*3C

where
   $PSRF100
   0          0=SiRF, 1=NMEA  - This is where the protocol is changed.
   9600       b/s rate 4800, 9600, 19200, 38400
   8          7, 8 Databits
   1          0, 1 Stopbits
   0          0=none, 1=odd, 2=even Parity
   *0C        checksum
```

The sentences 101 and 104 can be used to initialize values to be used by the gps. Supplying these values can shorten the initial lock time. If the clock offset is set to 0 then an internal default will be used. Sentence 101 supplies data in the internal ECEF (Earth centered, Earth Fixed) format in meters while sentence 104 supplies the data in the traditional Lat / Lon format.

```
 $PSRF101,-2686700,-4304200,3851624,95000,497260,921,12,3*22
 $PSRF104,37.3875111,-121.97232,0,95000,237759,922,12,3*3A

where
   $PSRF104
   37.3875111 Latitude in degrees
   -121.97232 Longitude in degrees
   0          Ellipsoid Altitude in meters
   95000      Clock offset
   237759     GPS Time of Week in seconds
   922        GPS Week Number
   12         Channel count (1 to 12)
   3          Reset config where
              1 = warm start, ephemeris valid
              2 = clear ephemeris, warm start (First Fix)
              3 = initialize with data, clear ephemeris
              4 = cold start, clear all data
              8 = cold start, set factory defaults
   *3A        checksum
```

The sentence 103 is used to control which NMEA sentences are to be sent and how often. Each sentence type is controlled individually. If the query bit is set then the gps responds by sending this message in the next second no matter what the rate is set to. Note that if trickle power is in use (can only be set in binary mode) then the actual update rate will be the selected update rate times the trickle rate which could mean that the data will be sent less frequently than was set here.

```
 $PSRF103,05,00,01,01*20

where
   $PSRF103
   05         00=GGA
              01=GLL
              02=GSA
              03=GSV
              04=RMC
              05=VTG
   00         mode, 0=set rate, 1=query
   01         rate in seconds, 0-255
   01         checksum 0=no, 1=yes
   *20        checksum
```

The 105 sentence controls a debug mode which causes the gps to report any errors it finds with the input data. $PSRF105,1*3E would turn debug on while $PSRF105,0*3F would turn it off.

Magnavox

The old Magnavox system used mostly proprietary sentences. The Magnavox system was acquired by Leica Geosystems in 1994. Information on this system can be found at this site. The NMEA sentences themselves are described here. They all use the MVX prefix and include:

Control Port Input sentences

- $PMVXG,000 Initialization/Mode Control - Part A
- $PMVXG,001 Initialization/Mode Control - Part B
- $PMVXG,007 Control Port Configuration
- $PMVXG,023 Time Recovery Configuration
- $CDGPQ,YYY Query From a Remote Device / Request to Output a Sentence

Control Port Output Sentences

- $PMVXG,000 Receiver Status
- $PMVXG,021 Position, Height, Velocity
- $PMVXG,022 DOPs
- $PMVXG,030 Software Configuration
- $PMVXG,101 Control Sentence Accept/Reject
- $PMVXG,523 Time Recovery Configuration
- $PMVXG,830 Time Recovery Results

Sony

The Sony interface uses a proprietary sentence that looks like:

```
$PSNY,0,00,05,500,06,06,06,06*14

where
   PSNY
   0          Preamp (external antenna) status
              0 = Normal
              1 = Open
              2 = shorted
   00         Geodesic system (datum) 0-25, 0 = WGS84
   05         Elevation mask in degrees
   500        Speed Limit in Km
   06         PDOP limit with DGPS on
   06         HDOP limit with DGPS on
   06         PDOP limit with DGPS off
   06         HDOP limit with DGPS off
   *14        Checksum
```

GPIO

General-purpose input/output (**GPIO**) is a generic pin on an integrated circuit or computer board whose behavior—including whether it is an input or output pin—is controllable by the user at run time.

GPIO pins have no predefined purpose, and go unused by default. The idea is that sometimes a system integrator who is building a full system might need a handful of additional digital control lines—and having these available from a chip avoids having to arrange additional circuitry to provide them. For example, the Realtek ALC260 chips (audio codec) have 8 GPIO pins, which go unused by default. Some system integrators (Acer Inc. laptops) use the first GPIO (GPIO_0) on the ALC260 to turn on the amplifier for the laptop's internal speakers and external headphone jack.

I2C

I²C (**Inter-Integrated Circuit**), pronounced *I-squared-C*, is a synchronous, multi-master, multi-slave, packet switched, single-ended, serial computer bus invented in 1982 by Philips Semiconductor (now NXP Semiconductors). It is widely used for attaching lower-speed peripheral ICs to processors and microcontrollers in short-distance, intra-board communication. Alternatively I²C is spelled **I2C** (pronounced I-two-C) or **IIC** (pronounced I-I-C).

Since October 10, 2006, no licensing fees are required to implement the I²C protocol. However, fees are required to obtain I²C slave addresses allocated by NXP.

Several competitors, such as Siemens AG (later Infineon Technologies AG, now Intel mobile communications), NEC, Texas Instruments, STMicroelectronics (formerly SGS-Thomson), Motorola (later Freescale, now merged with NXP), Nordic Semiconductor and Intersil, have introduced compatible I^2C products to the market since the mid-1990s.

SMBus, defined by Intel in 1995, is a subset of I^2C, defining a stricter usage. One purpose of SMBus is to promote robustness and interoperability. Accordingly, modern I^2C systems incorporate some policies and rules from SMBus, sometimes supporting both I^2C and SMBus, requiring only minimal reconfiguration either by commanding or output pin use.

Azimuth

An **azimuth** (from the pl. form of the Arabic noun "السَّمْت" *as-samt*, meaning "the direction") is an angular measurement in a spherical coordinate system. The vector from an observer (origin) to a point of interest is projected perpendicularly onto a reference plane; the angle between the projected vector and a reference vector on the reference plane is called the azimuth.

An example of azimuth is the angular direction of a star in the sky. The star is the point of interest, the reference plane is the local horizontal area (e.g. a circular area 5 km in radius around an observer at sea level), and the reference vector points north. The azimuth is the angle between the north vector and the star's vector on the horizontal plane.

Azimuth is usually measured in degrees (°). The concept is used in navigation, astronomy, engineering, mapping, mining, and ballistics.

OLED

An **organic light-emitting diode** (**OLED**) is a light-emitting diode (LED) in which the emissive electroluminescent layer is a film of organic compound that emits light in response to an electric current. This layer of organic semiconductor is situated between two electrodes; typically, at least one of these electrodes is transparent. OLEDs are used to create digital displays in devices such as television screens, computer monitors, portable systems such as mobile phones, handheld game consoles and PDAs. A major area of research is the development of white OLED devices for use in solid-state lighting applications.

There are two main families of OLED: those based on small molecules and those employing polymers. Adding mobile ions to an OLED creates a light-emitting electrochemical cell (LEC) which has a slightly different mode of operation. An OLED display can be driven with a passive-matrix (PMOLED) or active-matrix (AMOLED) control scheme. In the PMOLED scheme, each row (and line) in the display is controlled sequentially, one by one, whereas AMOLED control uses a thin-film transistor backplane to directly access and switch each individual pixel on or off, allowing for higher resolution and larger display sizes.

An OLED display works without a backlight because it emits visible light. Thus, it can display deep black levels and can be thinner and lighter than a liquid crystal display (LCD). In low ambient light conditions (such as a dark room), an OLED screen can achieve a higher contrast ratio than an LCD, regardless of whether the LCD uses cold cathode fluorescent lamps or an LED backlight.

REFERENCES

Rutstrum, Carl, *The Wilderness Route Finder*, University of Minnesota Press (2000), ISBN 0-8166-3661-3

U.S. Army, *Advanced Map and Aerial Photograph Reading*, FM 21–26, Headquarters, War Department, Washington, D.C. (17 September 1941)

U.S. Army, *Advanced Map and Aerial Photograph Reading*, FM 21–26, Headquarters, War Department, Washington, D.C. (23 December 1944)

U.S. Army, *Map Reading and Land Navigation*, FM 21–26, Headquarters, Dept. of the Army, Washington, D.C. (7 May 1993)

Raspberry Pi For Dummies; Sean McManus and Mike Cook; 432 pages; 2013; ISBN 978-1118554210.

Getting Started with Raspberry Pi; Matt Richardson and Shawn Wallace; 176 pages; 2013; ISBN 978-1449344214.

Raspberry Pi User Guide; Eben Upton and Gareth Halfacree; 312 pages; 2014; ISBN 978-1118921661.

Hello Raspberry Pi!; Ryan Heitz; 320 pages; 2016; ISBN 978-1617292453.

Getting Started with Wolfram Language and Mathematica for Raspberry Pi; Agus Kurniawan; 73 pages; 2016; ISBN B01BON8NCI.

White, Jon, ed. (2016). Raspberry Pi - The Complete Manual (7 ed.). Imagine Publishing. p. 36. ISBN 978-1785463709.

"General Purpose Input/Output". Oracle® Java ME Embedded Developer's Guide (8 ed.). Oracle Corporation. 2014.

"GPIO - Raspberry Pi Documentation". Raspberry Pi Foundation. Retrieved 2016-11-03.

Balachandran, Sasang (2009). General Purpose Input/Output (GPIO) (PDF). Michigan State University College of Engineering.

Mastering the I²C Bus; Vincent Himpe; 248 pages; 2011; ISBN 978-0-905705-98-9.

The I2C Bus : From Theory to Practice; Dominique Paret; 314 pages; 1997; ISBN 978-0-471-96268-7.

P. Chamorro-Posada, J. Martín-Gil, P. Martín-Ramos, L.M. Navas-Gracia, *Fundamentos de la Tecnología OLED* (*Fundamentals of OLED Technology*). University of Valladolid, Spain (2008). ISBN 978-84-936644-0-4. Available online, with permission from the authors, at the webpage: https://www.scribd.com/doc/13325893/Fundamentos-de-la-Tecnologia-OLED

Kordt, Pascal; et al. (2015). "Modeling of Organic Light Emitting Diodes: From Molecular to Device Properties". Advanced Functional Materials. ***25*** *(13): 1955–1971. doi:10.1002/adfm.201403004.*

Shinar, Joseph (Ed.), *Organic Light-Emitting Devices: A Survey*. NY: Springer-Verlag (2004). ISBN 0-387-95343-4.

Hari Singh Nalwa (Ed.), *Handbook of Luminescence, Display Materials and Devices*, Volume 1–3.

American Scientific Publishers, Los Angeles (2003). ISBN 1-58883-010-1. Volume 1: Organic Light-Emitting Diodes

Hari Singh Nalwa (Ed.), *Handbook of Organic Electronics and Photonics*, Volume 1–3. American Scientific Publishers, Los Angeles (2008). ISBN 1-58883-095-0.

Müllen, Klaus (Ed.), *Organic Light Emitting Devices: Synthesis, Properties and Applications*. Wiley-VCH (2006). ISBN 3-527-31218-8

Yersin, Hartmut (Ed.), *Highly Efficient OLEDs with Phosphorescent Materials*. Wiley-VCH (2007). ISBN 3-527-40594-1

Kho, Mu-Jeong, Javed, T., Mark, R., Maier, E., and David, C. (2008) 'Final Report: OLED Solid State Lighting – Kodak European Research' MOTI (Management of Technology and Innovation) Project, Judge Business School of the University of Cambridge and Kodak European Research, Final Report presented on 4 March 2008 at Kodak European Research at Cambridge Science Park, Cambridge, UK., pages 1–12.