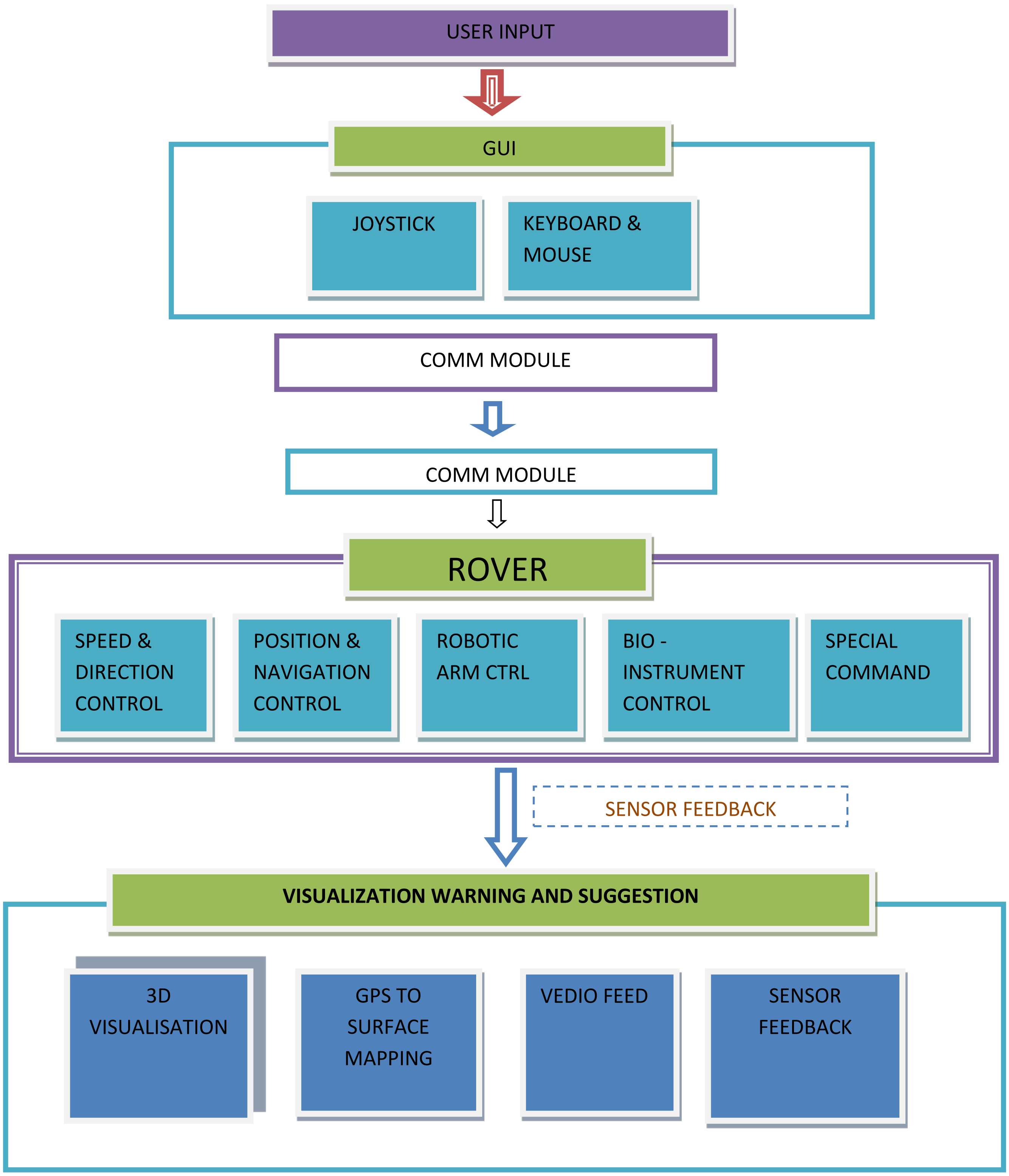
Base Station control



As you can see, each part can be a big task in itself based on how much we need and how much we want to implement

GPS NAVIGATION:

We have GPS as the only option for a world coordinate system. In order to complete any of the task trusting on GPS is must, hence we got to build a interface that intuitively lead us to wherever we want to go.

Here is what we can do about GPS interface

- The problem statement specially warns about GPS format, they have specified WGS84
- GPS WGS84 - A fully decimal format popular with GIS systems and used in higher-end GPS equipment
- Positive values occur north of the equator and east of the prime meridian to the International Date Line, while negative values occur south of the equator and west of the prime meridian
- Deg/Min/Sec: The standard minutes/seconds translation of latitude/longitude; many GPS units support this traditional format

e.g.

	WGS 84		Deg/Min/Sec		GPS	
Location	Latitude	Longitude	Latitude	Longitude	Latitude	Longitude
27th Street Garage	30.29128	-97.73858	30° 17' 28"	-97° 44' 18"	N30 17.477	W97 4.315

- Read GPS data from a microcontroller
- Interpret them and find coordinates , (Elevation is optional for now)
- Make a graphics program which can show grid coordinate and a point location on the window like this

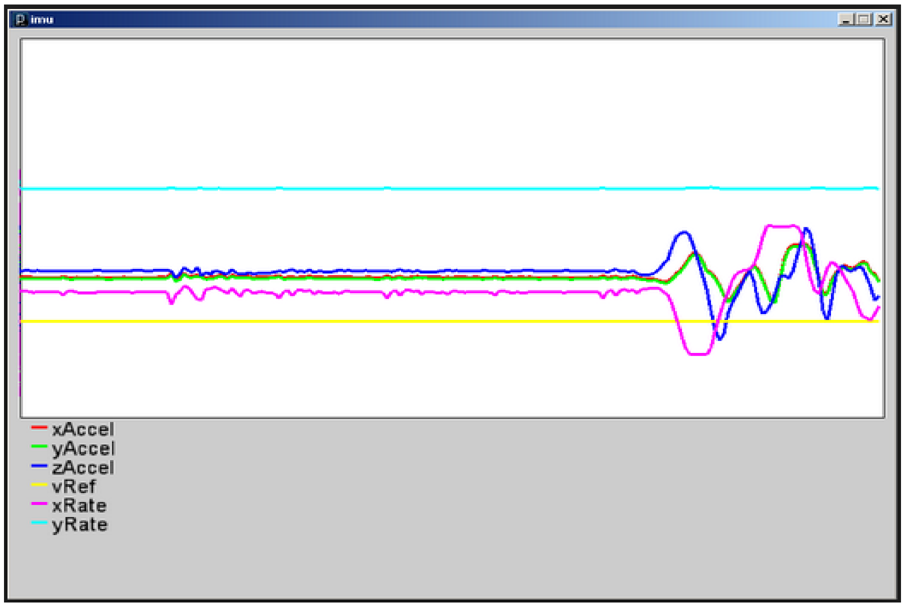
IRAQ
IRAN
AFGHANISTAN
CHINA
EGYPT
PAKISTAN
SAUDI ARABIA
OMAN
INDIA
MYANMAR
SUDAN
YEMEN
THAILAND
PHILIPPINES
AFRICA
ETHIOPIA
KENYA
DR CONGO
INDONESIA
TANZANIA
Indian Ocean
ZAMBIA

VEDIO FEED: We believe Yashwant will be our eyes.

SENSOR FEEDBACK:

- For this we have talk to MANVI (manvidhawan1993@gmail.com) and figure out what data are being produced by the IMUs and how to interpret them

- We have to plot real-time graphs like this



FOR ANY SENSOR

""""BUT UNFORTUNATELY THIS IS NOT BEING DONE YET""""

UI:

- So we are using Qt(C++) and Python
- Again Why QT:
    1. A great development environment -> Qt Creator + Qt API + Qt Documentation + Qt Community is just great to work with
    2. Qt Creator is a top class IDE
    3. A solid API evolved over years
    4. A reputation for good documentation
    5. Easy to learn and progress
    6. Open source libraries, apps & community
    7. Cross platform & multiple targets
    8. UI & backend development can be autonomous

The Original Programm was developed in QT 5.1.1 Mingw 32-Bit - http://qt-project.org/downloads

Joystick Library:

1. The Library belongs to - **Alex Diener** "adiener@sacredsoftware.net" and is Build open the Windows win-mm Library
2. It is written in C, but we have implemented it in C++ using Callback Functions provided in the Library

Serial Communication:

1. We have used "universal asynchronous receiver transmitter" (UART) communication
2. The code is modified version of the "Terminal" example in Qt Sample Project

Tile Maps:

1. Tile map- A tile engine is a computer graphics technique which generates a larger graphic from re-using a number of smaller graphics to save RAM and increase real-time rendering performance
2. A continuous image of the world at street level detail would be millions of pixels wide – much too large to download or hold in memory at once
3. In reality, web maps are made up of many small, square images called tiles. These tiles are typically 256x256 pixels and are placed side-by-side in order to create the illusion of a very large seamless image

Tiles were generated by the following software:

a. Universal Maps Downloader - http://www.allmapsoft.com/umd/

b. Gmapcatcher - https://code.google.com/p/gmapcatcher/

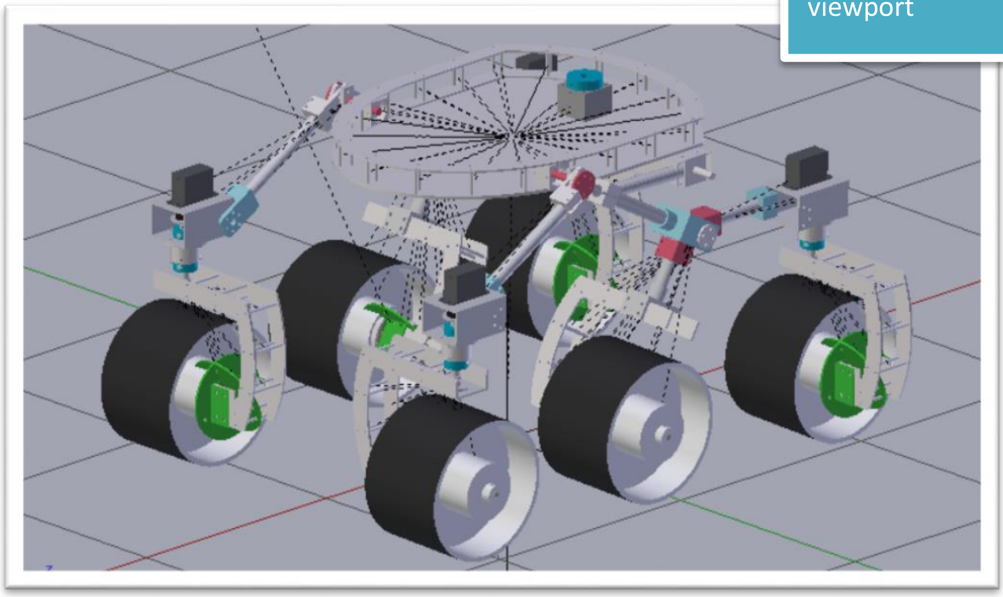
Events:

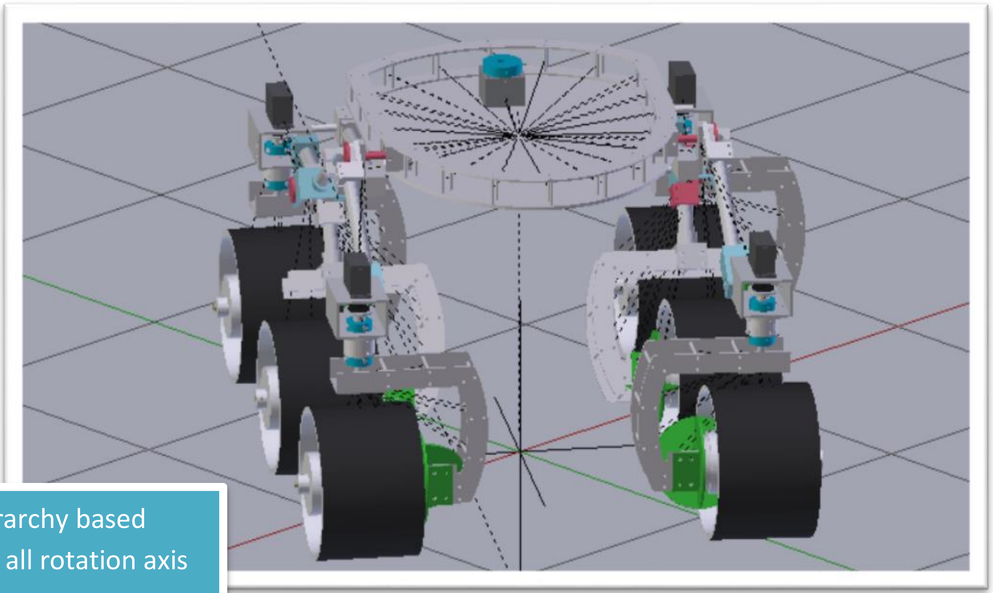
Handling window events (Like mouse events, keyboard events) – Using Qt Event Handlers

3D Visualization:

1. Blender : Used for Organizing the Rover CAD into a hierarchy model with parent child relationship
2. 3D rendering of CAD in real-time, using sensor data from Encoders and IMU
3. Although this has not been decided, we may even use a 3D rendering Library with C++ to do the same task instead of depending on Blender

A snapshot in the viewport





This is a hierarchy based model, with all rotation axis set properly

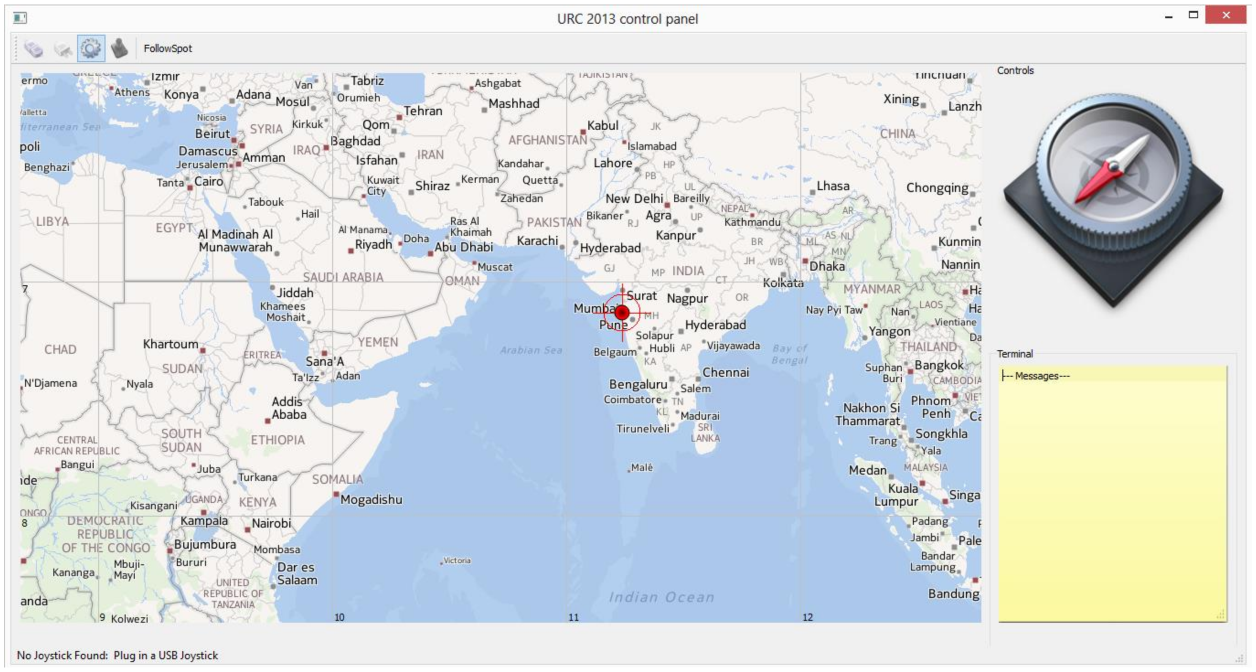
- Simulation, Animation and constraint checking has to be done.

So all these work needs to make VIRTUAL model of the rover and its component

As well as

Providing control command to the rover through communication module according to the need of the URC 2014

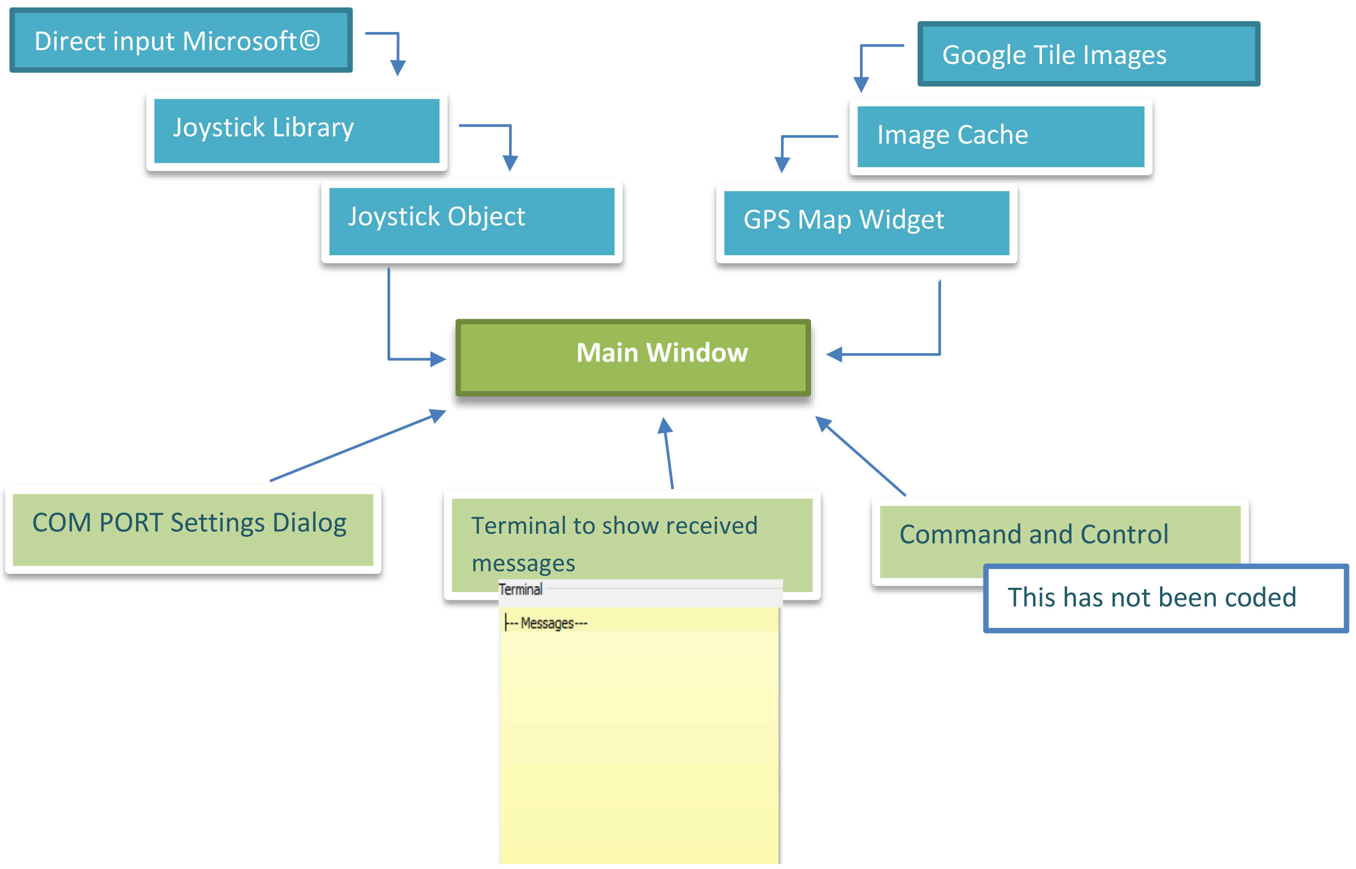
UI Coding:

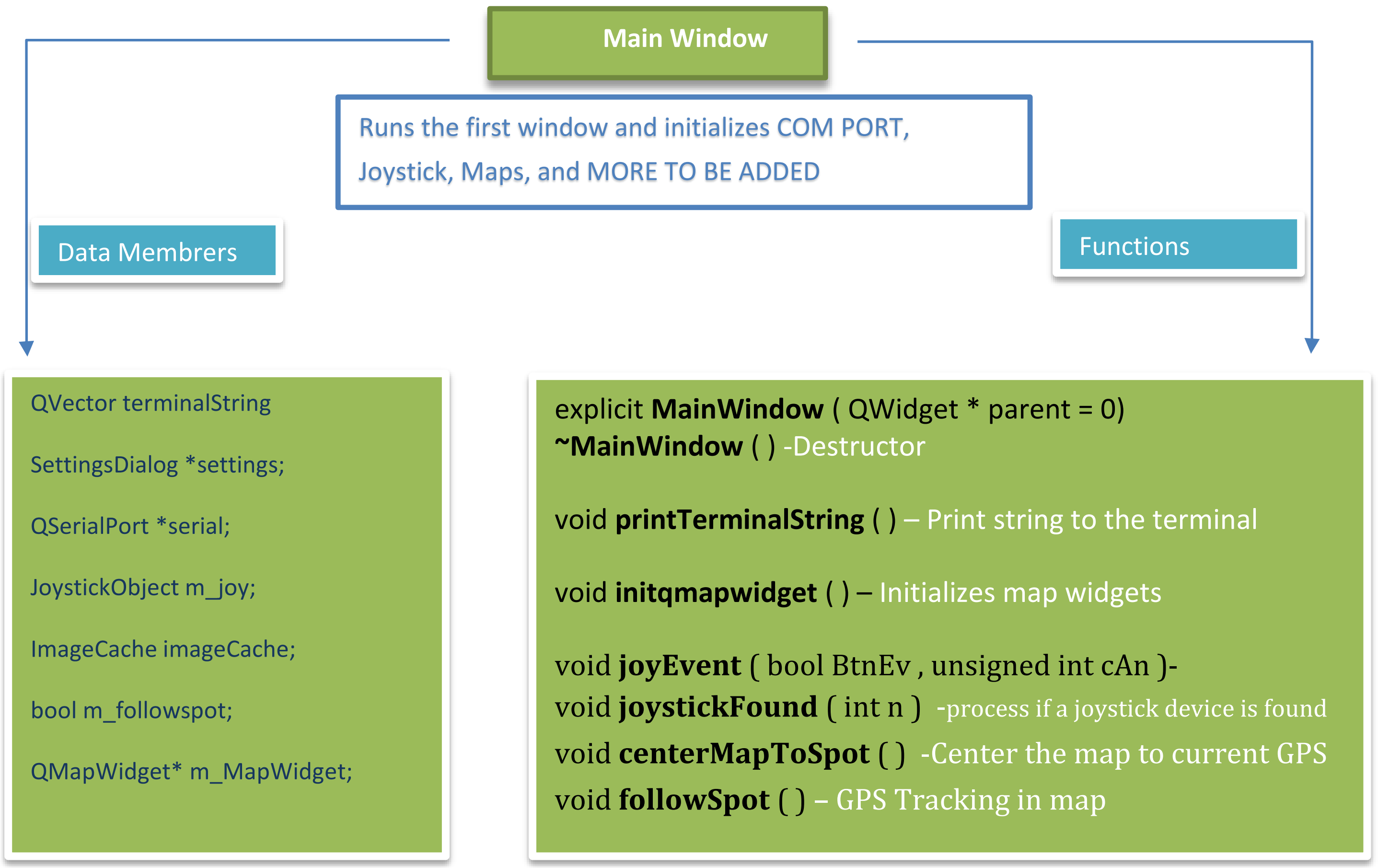


```
|-- Messages---
```

```
No Joystick Found:  Plug in a USB Joystick
```

Code Structure: Base Station Controlling UI





Runs the first window and initializes COM PORT, Joystick, Maps, and MORE TO BE ADDED

QVector terminalString

SettingsDialog *settings;

QSerialPort *serial;

JoystickObject m_joy;

ImageCache imageCache;

bool m_followspot;

QMapWidget* m_MapWidget;

explicit **MainWindow** (QWidget * parent = 0)
~MainWindow ( ) -Destructor

void **printTerminalString** ( ) – Print string to the terminal

void **initqmapwidget** ( ) – Initializes map widgets

void **joyEvent** (bool BtnEv , unsigned int cAn)-
void **joystickFound** (int n) -process if a joystick device is found
void **centerMapToSpot** ( ) -Center the map to current GPS
void **followSpot** ( ) – GPS Tracking in map

MainWindow::MainWindow(QWidget *parent) – Constructor of main window, connect to serial port, joystick, call initConnections()

MainWindow::~MainWindow() - Destructor, Delete map widgets, and whatever was created with new keyword

void MainWindow::openSerialPort(), void MainWindow::closeSerialPort() - open or close serial port

void MainWindow::writeData(const QByteArray &data) - Write data to the open serial port from the Byte buffer "data"

void MainWindow::readData() - Read all the datas available in the serial port buffer

void MainWindow::handleError(QSerialPort::SerialPortError error) – check if there was an error in reading or connecting to the serial port, if so print on status bar

void MainWindow::initActionsConnections() - connect all signas with slots (Refer QT Signal Slots mechanism

void MainWindow::joyEvent(bool BtnEv, unsigned int cAn) – slot for joystick event with arguments button event and Changed Axis No {0 – 6}

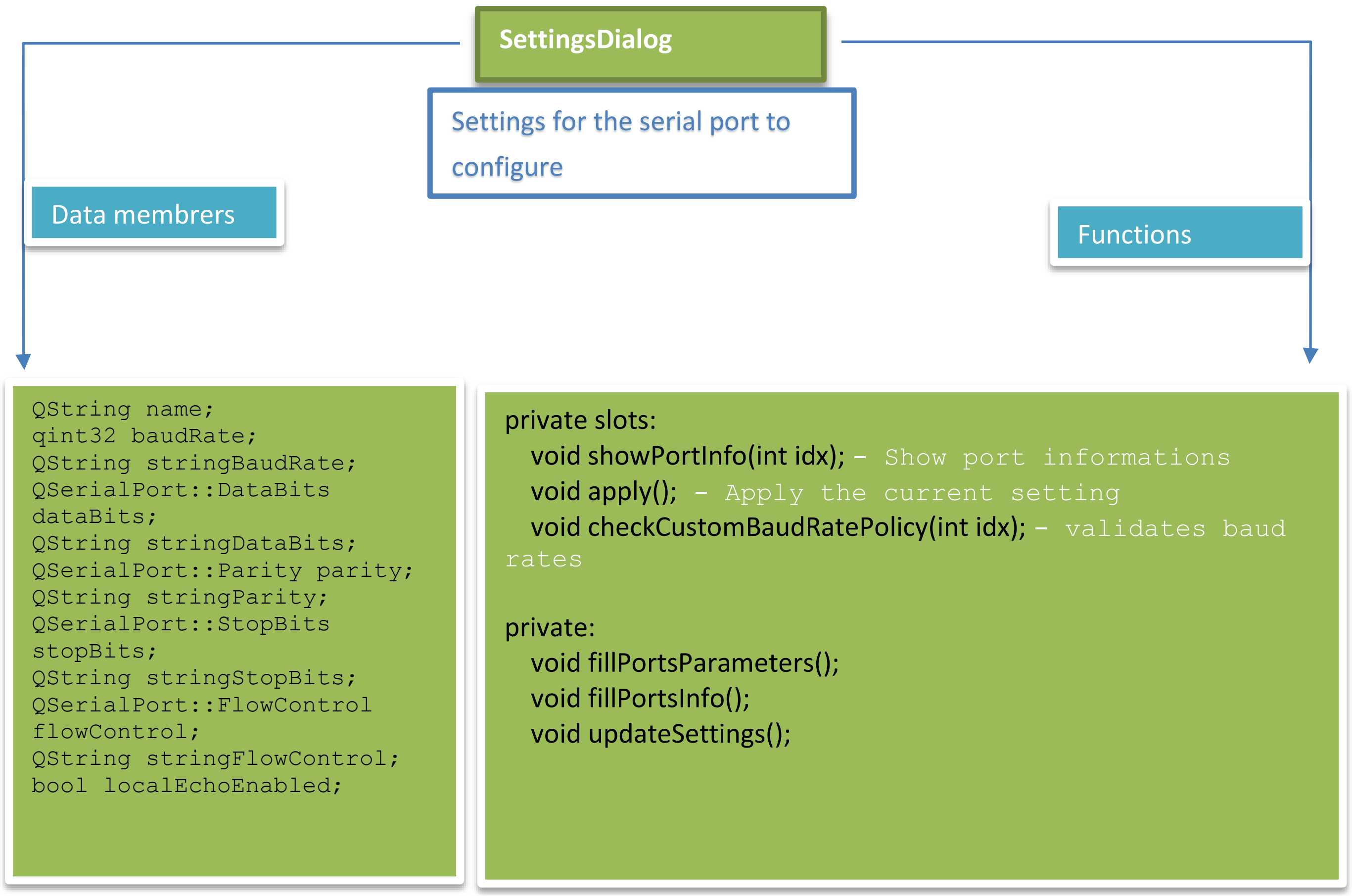
void MainWindow:: printTerminalString() - print all the string in the vector (terminalString) to the terminal

void MainWindow::initqmapwidget() - set parameters for map widget

void MainWindow::centerMapToSpot() - center the map to GPS Location

void MainWindow::followSpot() start tracking GPS marker

void MainWindow::resetAll()



SettingsDialog

Settings for the serial port to configure

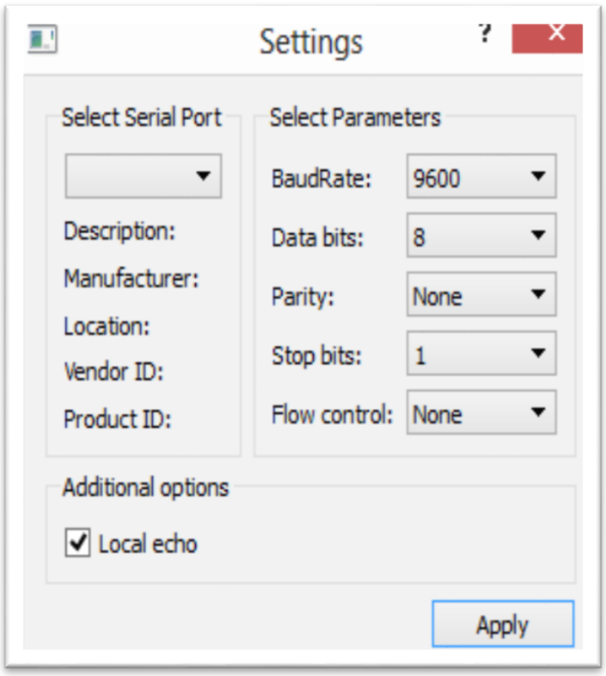
Data membrers

```
QString name;
qint32 baudRate;
QString stringBaudRate;
QSerialPort::DataBits
dataBits;
QString stringDataBits;
QSerialPort::Parity parity;
QString stringParity;
QSerialPort::StopBits
stopBits;
QString stringStopBits;
QSerialPort::FlowControl
flowControl;
QString stringFlowControl;
bool localEchoEnabled;
```

Functions

```
private slots:
    void showPortInfo(int idx); - Show port informations
    void apply(); - Apply the current setting
    void checkCustomBaudRatePolicy(int idx); - validates baud
rates

private:
    void fillPortsParameters();
    void fillPortsInfo();
    void updateSettings();
```



Settings

Select Serial Port

Description:
Manufacturer:
Location:
Vendor ID:
Product ID:

Select Parameters

BaudRate: 9600
Data bits: 8
Parity: None
Stop bits: 1
Flow control: None

Additional options

[✓] Local echo