A7 GSM/GPS & Orange Pi One Datasheet

（GSM/GPRS+GPS+AGPS）

Table of Contents

[History 3](#_Toc17386603)

[1. Idea 4](#_Toc17386604)

[2. Device Setup and Architecture 4](#_Toc17386605)

[2.1 Hardware 4](#_Toc17386606)

[2.1.1 Interfaces and Protocols 4](#_Toc17386607)

[3. Orange Pi One Config 4](#_Toc17386608)

[3.1 The Armbian Operating System 4](#_Toc17386609)

[3.2 GSM Configuration 5](#_Toc17386610)

[3.3 GPS Configuration 5](#_Toc17386611)

[3.4 Internet Configuration 6](#_Toc17386612)

[3.5 Python Service Worker 7](#_Toc17386613)

[3.6 Run on Start 7](#_Toc17386614)

[5. A7 GSM+GPRS+GPS+AGPS Module 8](#_Toc17386615)

[1. A7 GPS Module 8](#_Toc17386616)

[2. A7 Pin Descriptions 11](#_Toc17386617)

[3. PCB Dimensions 12](#_Toc17386618)

[6. A7 Schematics Diagram 14](#_Toc17386619)

[1. Schematic Diagram 14](#_Toc17386620)

[7. AT Command for A7 Module 15](#_Toc17386621)

[1. A7 GPS AT Command 15](#_Toc17386622)

[2. GPRS and GSM AT Command 15](#_Toc17386623)

[8. Orange Pi One GPIO Assignment 16](#_Toc17386624)

[1. GPIO Details 16](#_Toc17386625)

[2. PI40 Pin Description 17](#_Toc17386626)

[9. Pin Connection 18](#_Toc17386627)

[1. Pin Connection Details 18](#_Toc17386628)

[2. Data and Control Flow 18](#_Toc17386629)

[10. Serial Port Configuration 19](#_Toc17386630)

[1. Debug Port (/dev/ttyS0) 19](#_Toc17386631)

[2. Command Port (/dev/ttyS1) 19](#_Toc17386632)

[3. Data Port (/dev/ttyS2) 20](#_Toc17386633)

[4. Host Control Port (/dev/ttyS3) 20](#_Toc17386634)

[11. Orange PI one Board Initialization 20](#_Toc17386635)

[1. SD Card Preparation and Kernel Initialization 20](#_Toc17386636)

[2. Vehicle Tracker Application Run 21](#_Toc17386637)

[3. Power and Board Initialization 21](#_Toc17386638)

[12. GPS / GSM Board Functionalities 22](#_Toc17386639)

[1. GPRS Initialization 22](#_Toc17386640)

[2. GPS Initialization 22](#_Toc17386641)

[3. Get GPS Data 22](#_Toc17386642)

[4. Voice Call management 22](#_Toc17386643)

[5. SMS management 22](#_Toc17386644)

[6. Send Data to Web Server 22](#_Toc17386645)

[7. TCP Connection with Server 22](#_Toc17386646)

# History

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Modification Date | Author | Description |
| 1 | 2017-01-26 | Sanjeev Kumar | Initial Design Documents |
| 2 | 2017-02-01 | Sanjeev Kumar | Serial Port Configuration is added |
| 3 | 2017-02-01 | Sanjeev Kumar | Orange PI and A7 Board Initialization is added |
| 4 | 2019-08-30 | Sanjay Kumar | Device setup and Configuration |
| 5 |  |  |  |
| 6 |  |  |  |
| 7 |  |  |  |

# Idea

Vehicle Tracker Using GPS and GSM

# Device Setup and Architecture

## 2.1 Hardware

1. Orange Pi One and A7 module

For running the linux image and our code, we will need a micro SD Card.

### Interfaces and Protocols

Before getting started with the configuration, we have to be sure which interfaces are used to connect these parts with each other: The GSM Module is connected over the 40 Pin GPIO ( General Purpose Input Out- put) directly on the Pie. This way, it is automatically connected to the Serial UART Rx/Tx, which allows to communicate with the module directly by using the Hayes command Set (AT Commands). We won’t use these commands in the end, but they are really helpful for testing and debugging functionalities (e.g. show connected ports, send/receive sms, show signal strength, general behaviour). The GSM uses the power supply from the Pie and (if configured correctly) will start automatically whenever the Pie is started. The default serial port will be the '/dev/ttyS0' (for Orange Pi one).

Connection to the internet is established over the SIM card using the PPP protocol. The device is now basically a big, fat mobile-phone without display and keyboard.

# Orange Pi One Config

There are basically 6 steps to take care of:

1. Setup the Operating System on the Orange Pi One
2. Get The GSM Module running
3. Get The GPS running
4. Provide internet connection
5. Implement a service worker for collecting and sending coordinates
6. Run the script on startup

## 3.1 The Armbian Operating System

We are

.

## 3.2 GSM Configuration

The GSM Module should boot automatically. To test it’s functionality, we can use AT commands over the serial interface using minicom. The following command start the minicom with a specific baudrate on the ttyS0 interface:

$ ssh pi@192.168.38.99

$ sudo screen −b 9600 −D /dev/ttyS0 −o

Then, all kind of different things can be tested, e.g.:

*// send SMS to +918800213260*

AT+CMGF=1

OK

AT+CMGS="+918800213260"

> This is the text message.

+CMGS: 198

OK

*// location lat lng calculated from nearest CELL TOWERS*

AT+CIPGSMLOC=1,1

+CIPGSMLOC: 0,7.584216,47.546936,2017/11/25,14:48:49

OK

*// get IP of some website*

AT+CDNSGIP="www.google.com"

OK

+CDNSGIP: 1,"www.google.com","172.217.17.100"

## 3.3 GPS Configuration

I tried calculating the current location by triangulating over the GPRS signal strengths with the locations of the surrounding cell towers. Unfortunately, the results of these calculations were really inaccurate (+- 50m). Therefore, I decided to buy an extra GPS Receiver for more precise results.

Making the GPS packages available for the application requires the following steps:

1. Connect A7 GPS Module through serial port
2. Install GPS Daemon and Python Integration

$ sudo apt−get install gpsd gpsd−clients python−gps

$ sudo nano /etc/**default**/gpsd

Then check that the following configurations are set:

*# FILE gpsd*

*# Start the gpsd daemon automatically at boot time*

START\_DAEMON="true"

*# Use USB hotplugging to add new USB devices automatically to the daemon*

USBAUTO="true"

*# Devices gpsd should collect to at boot time.*

DEVICES="/dev/ttyS0"

*# Other options you want to pass to gpsd*

GPSD\_OPTIONS="−n"

GPSD\_SOCKET="/var/run/gpsd.sock"

To test if the GPS deamon is working, type

$ cgps

*//or*

$ gpsmon

## 3.4 Internet Configuration

To get our coordinates to the cloud, we need to setup the PPP connection.

1. Get the librarires and
2. create the web interface
3. make sure the configuration is set propperly
4. then edit the network interfaces by adding the new ppp interface

$ sudo apt−**get** install ppp screen elinks

$ cd /etc/ppp/peers/ && nano rnet

*# FILE rnet*

*# ’internet’ is the apn for sunrise connection*

connect "/usr/sbin/chat −v −f /etc/chatscripts/gprs −T internet"

*# communication port:*

/dev/ttyS0

*# baudrate*

9600

*# Assumes that your IP address is allocated dynamically by the ISP.*

noipdefault

*# Try to get the name server addresses from the ISP.*

usepeerdns

*# Use this connection as the default route to the internet .*

defaultroute

*# Makes PPPD "dial again" when the connection is lost.*

persist

*# Do not ask the remote to authenticate.*

noauth

*# FILE /etc/network/interfaces*

auto rnet

iface rnet inet ppp

provider rnet

We can now use the PPP protocol to connect to the internet. Piece of cake!

## 3.5 Python Service Worker

**while** **True**:

*# GET GPS DATA*

gpsData['lat'] = gpsd.fix.latitude

gpsData['lan'] = gpsd.fix.longitude

gpsData['time'] = gpsd.utc, ' + ', gpsd.fix.time

gpsData['alt'] = gpsd.fix.altitude

gpsData['eps'] = gpsd.fix.eps

gpsData['epx'] = gpsd.fix.epx

gpsData['epv'] = gpsd.fix.epv

gpsData['ept'] = gpsd.fix.ept

gpsData['speed'] = gpsd.fix.ept

gpsData['climb'] = gpsd.fix.speed

gpsData['track'] = gpsd.fix.climb

gpsData['mode'] = gpsd.fix.track

gpsData['sats'] = gpsd.fix.mode

trigger = float(gpsData['lat'])

**if** math.isnan(trigger):

data = {'lat': 0,'lng': 0}

**else**:

data = {'lat': gpsData['lat'],'lng': gpsData['lan']}

**try**:

**print**("publish timetoken: %d" % envelope.result.timetoken)

except Exception **as** e:

**print**("Error %s" % str(e))

time.sleep(15)

...

## 3.6 Run on Start

Almost there! The last step is to execute the worker on device start. For that we just call our script within the Pies profile configuration. This way, if the script crashes, we can just restart the device and the script will be running again. Add the following lines to the profile:

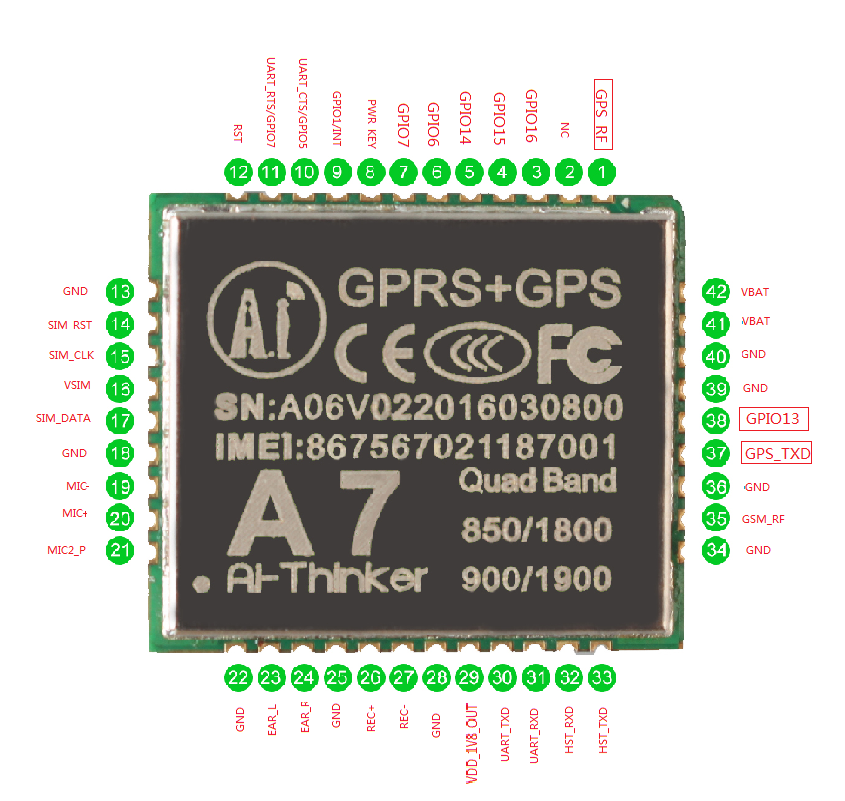
$ sudo nano /etc/ profile

*# ’/home/pi/myscript.py’ is the path to your script .*

sudo python /home/pi/myscript.py

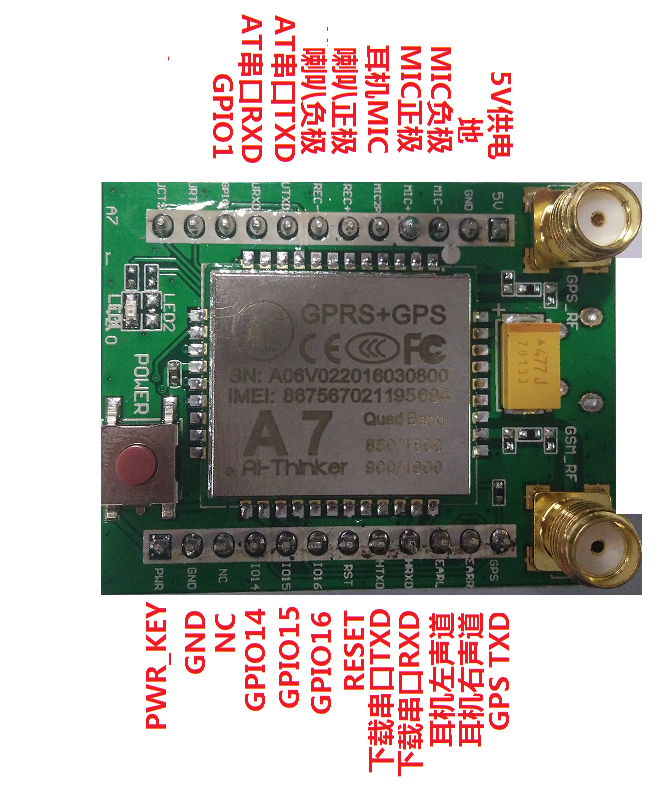
# A7 GSM+GPRS+GPS+AGPS Module

## A7 GPS Module

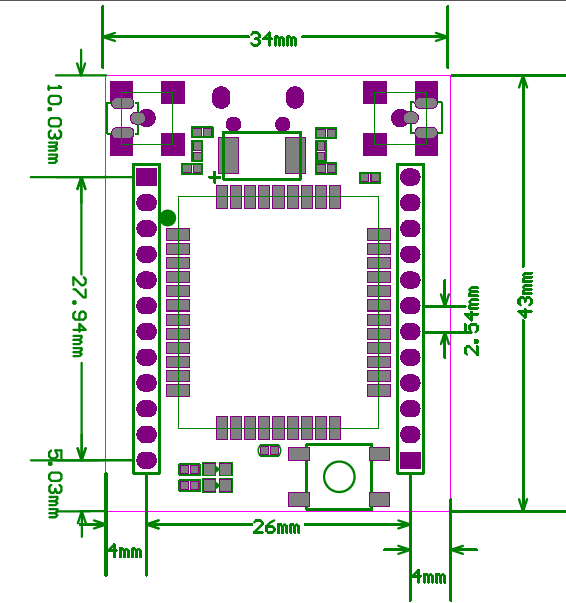


|  |  |  |
| --- | --- | --- |
| PIN No. | PIN Name | Pin Description |
| 1 | GPS\_RF | GPS RF Antenna PCB50 ohm |
| 2 | GND | Ground |
| 3 | GPIO16 | GPIO16 |
| 4 | GPIO15 | GPIO15 |
| 5 | GPIO14 | GPIO14 |
| 6 | GPIO6 | GPIO6 |
| 7 | GPIO7 | GPIO7 |
| 8 | PWR\_KEY | Supply Voltage >1.9V for more than 2 seconds |
| 9 | GPIO1/INT | Current should be less than <1mA |
| 10 | UART\_CTS/GPIO5 | AT CTS Clear to send |
| 11 | UART\_RTS/GPIO7 | AT RTS Ready to send |
| 12 | RST | RESET PIN，<0.05V,70ma |
| 13 | GND | GROUND |
| 14 | SIM\_RST | SIM ReSeT |
| 15 | SIM\_CLK | SIM CLocK |
| 16 | VSIM | SIM Voltage |
| 17 | SIM\_DATA | SIM Data |
| 18 | GND | GROUND |
| 19 | MIC- | MIC Negative pin |
| 20 | MIC+ | MIC Positive Pin |
| 21 | MIC2\_P | MIC to Power |
| 22 | GND | GROUND |
| 23 | EAR\_L | Audio Left Speaker |
| 24 | EAR\_R | Audio Right Speaker |
| 25 | GND | GROUND |
| 26 | REC+ | Recording plus |
| 27 | REC- | Recording minus |
| 28 | GND | GROUND |
| 29 | VDD\_1V8\_OUT | Voltage 1.8V output |
| 30 | UART\_TXD | AT TXD pin ，Voltage 2.8V |
| 31 | UART\_RXD | AT RXD pin ，Voltage 2.8V |
| 32 | HST\_RXD | Host RXD Pin，Voltage 2.8V |
| 33 | HST\_TXD | Host TXD Pin，Voltage 2.8V |
| 34 | GND | GROUND |
| 35 | GSM\_RF | GSM RF Antenna PCB50 ohm |
| 36 | GND | GROUND |
| 37 | GPS\_TXD | NEMA Output at GPS\_TXD at baud rate 9600 |
| 38 | GPIO13 | GPIO13 |
| 39 | GND | GROUND |
| 40 | GND | GROUND |
| 41 | VBAT | Supply Voltage 3.5V-4.2V，Current 2A |
| 42 | VBAT |

## A7 Pin Descriptions

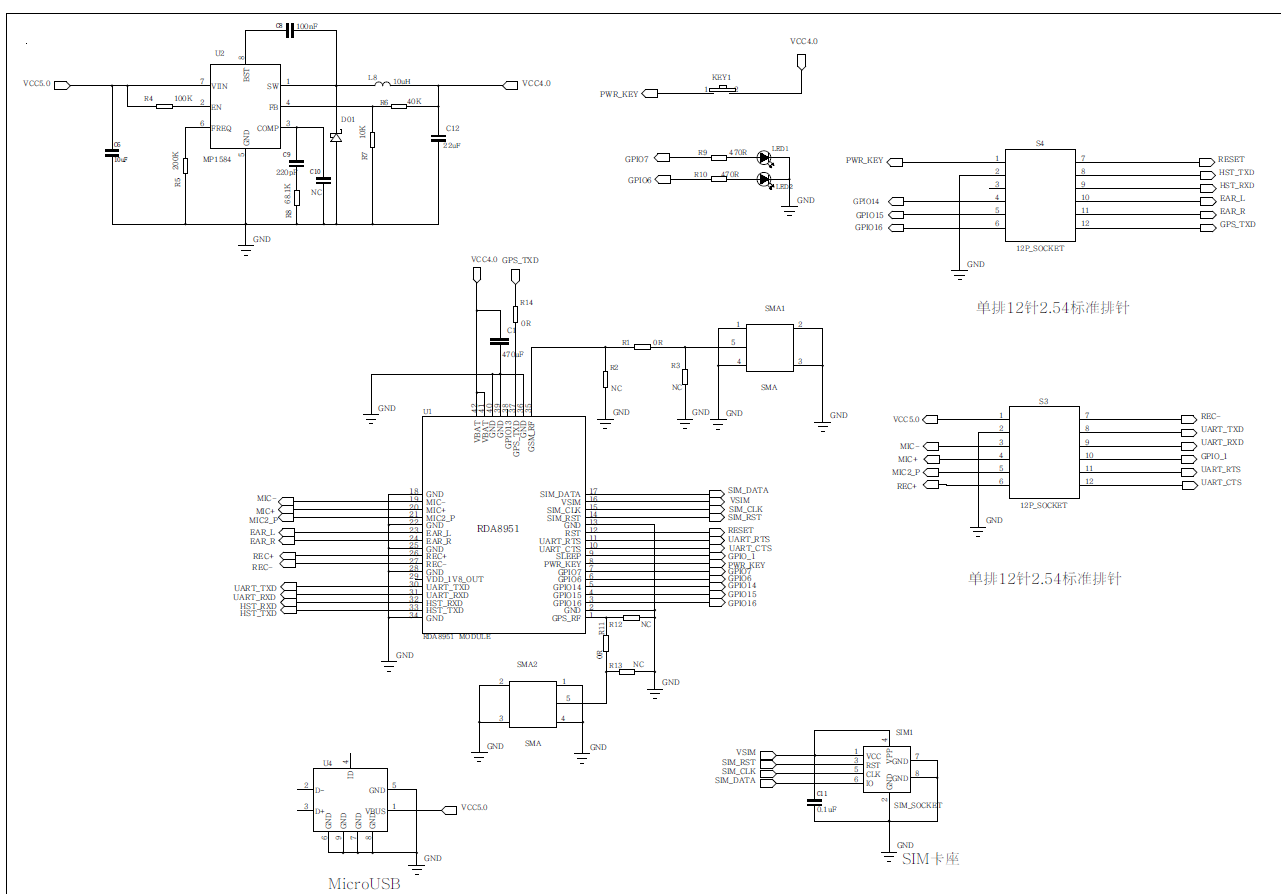


## PCB Dimensions



# A7 Schematics Diagram

## Schematic Diagram



# AT Command for A7 Module

## A7 GPS AT Command

AT+GPS=1 : POWER ON GPS

AT+GPS=0 : POWER OFF GPS

AT+AGPS=1 : POWER ON AGPS

AT+AGPS=0 POWER OFF AGPS

GPS/AGPS : NEMA Output at GPS\_TXD at baud rate 9600

AT+GPSRD=0 Stop NEM Output AT Command

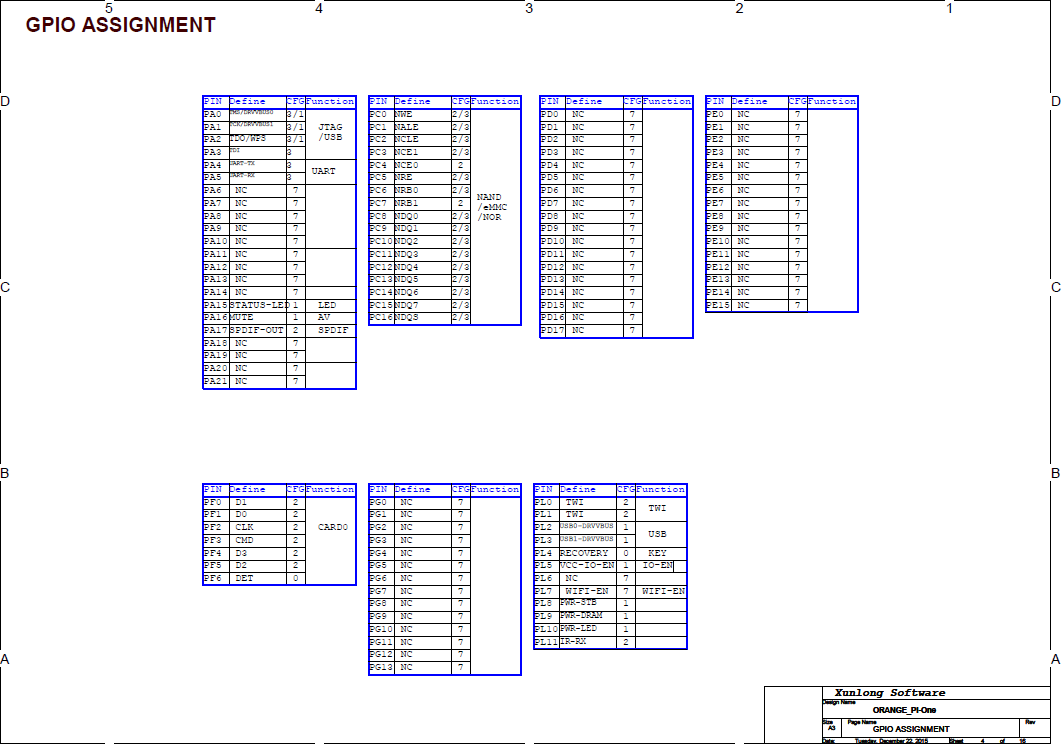
AT+GPSRD=N NEMA Output at N Seconds

## GPRS and GSM AT Command

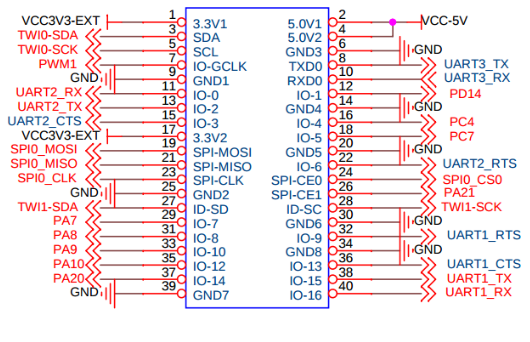
Please refer to **a7\_at instructions v1.03\_1\_.pdf Document**

# Orange Pi One GPIO Assignment

## GPIO Details



## PI40 Pin Description



# Pin Connection

## Pin Connection Details

* Pin no 2 ( Vcc 5 V ) of Orange Pi one will connect to 5V Pin of GPS Module
* Pin no 6 (GND) of Orange Pi one will connect to Ground pin of GPS Module
* Pin no 10 (Rxd) of Orange Pi one will connect to GPS\_TXD pin of GPS Module
* Pin no 29 (PA7) of Orange Pi one will connect to POWER\_KEY pin of GPS Module
* Pin no 38 ( TxD ) of Orange Pi one will connect to AT Rxd Pin of GPS Module
* Pin no 40 (RxD) of Orange Pi one will connect to AT Txd pin of GPS Module

## Data and Control Flow

* Pin no 2 ( Vcc 5 V ) and
* Pin no 6 (GND) of Orange Pi one will provide power to GPS Module
* Pin no 10 (Rxd) of Orange Pi one will receive the Data from GPS\_TXD pin of GPS Module
* Pin no 29 (PA7) of Orange Pi one will reset and start the GPS Module by giving 2 sec power
* Pin no 38 ( TxD ) and
* Pin no 40 (RxD) of Orange Pi one will control the GPS Module through AT Commands

# Serial Port Configuration

## Debug Port (/dev/ttyS0)

* This Port will be used for the Debug and Serial Print from Linux print

## Command Port (/dev/ttyS1)

* This Port will be used for send the AT Commands to GPS/GPRS Module
* This Port will be used for receive the response of AT Commands from GPS/GPRS Module

## Data Port (/dev/ttyS2)

* This Port will be used for receive the GPS Data From GPS/GPRS Module

## Host Control Port (/dev/ttyS3)

* This Port will be used for send the Host Commands to GPS/GPRS Module
* This Port will be used for receive the response of Host Commands from GPS/GPRS Module

# Orange PI one Board Initialization

## SD Card Preparation and Kernel Initialization

* Ubuntu Linux Server will be flashed in the SD Card

## Vehicle Tracker Application Run

* This Port will be used for the Debug and Serial Print from Linux print

## Power and Board Initialization

Void Init\_GPS\_GSM\_Module()

{

sunxi\_gpio\_init();

sunxi\_gpio\_set\_cfgpin(SUNXI\_GPA(17), SUNXI\_GPIO\_OUTPUT);

sunxi\_gpio\_output(SUNXI\_GPA(29), 1);

sleep(2);

sunxi\_gpio\_output(SUNXI\_GPA(29), 0);

sleep(1);

}

# GPS / GSM Board Functionalities

## GPRS Initialization

* Following Line of Code will initialize the GPRS Functionalities

## GPS Initialization

* Following Line of Code will initialize the GPRS Functionalities

## Get GPS Data

* Following Line of Code will initialize the GPS Functionalities

## Voice Call management

* Following Line of Code will initialize the GPS Functionalities

## SMS management

* Following Line of Code will initialize the GPS Functionalities

## Send Data to Web Server

* Following Line of Code will initialize the GPS Functionalities

## TCP Connection with Server