A7 GSM, Distance Sensor & Orange Pi One Datasheet

（GSM/GPRS+ Distance Sensors）

Table of Contents

[History 2](#_Toc30418625)

[1. Idea 3](#_Toc30418626)

[2. Device Setup and Architecture 3](#_Toc30418627)

[2.1 Hardware 3](#_Toc30418628)

[2.1.1 Interfaces and Protocols 3](#_Toc30418629)

[3. Orange Pi One Config 4](#_Toc30418630)

[3.1 The Armbian Operating System 4](#_Toc30418631)

[3.2 GSM Configuration 4](#_Toc30418632)

[3.3 Distance Server Configuration 5](#_Toc30418633)

[3.4 Internet Configuration 6](#_Toc30418634)

[3.5 Python Service Worker 7](#_Toc30418635)

[3.6 Run on Start 7](#_Toc30418636)

[5. A7 GSM+GPRS+GPS+AGPS Module 8](#_Toc30418637)

[1. A7 GPS Module 8](#_Toc30418638)

[2. A7 Pin Descriptions 11](#_Toc30418639)

[3. PCB Dimensions 12](#_Toc30418640)

[6. AT Command for A7 Module 14](#_Toc30418641)

[1. GPRS and GSM AT Command 14](#_Toc30418642)

[2. PI40 Pin Description 14](#_Toc30418643)

[7. Pin Connection 15](#_Toc30418644)

[1. Data and Control Flow 15](#_Toc30418645)

[8. Serial Port Configuration 16](#_Toc30418646)

[1. Debug Port (/dev/ttyS0) 16](#_Toc30418647)

[2. Command Port (/dev/ttyS1) 16](#_Toc30418648)

[9. GPIO Pin Configuration 16](#_Toc30418649)

[10. Orange PI one Board Initialization 17](#_Toc30418650)

[1. SD Card Preparation and Kernel Initialization 17](#_Toc30418651)

[2. Distance Tracker Application Run 17](#_Toc30418652)

[3. Power and Board Initialization 17](#_Toc30418653)

# 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 | 2020-01-20 | Sanjeev kumar | Distance Sensor is integerated |
| 6 |  |  |  |
| 7 |  |  |  |

# Idea

Distance Tracker Using Distance Sensor 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 Distance running
4. Provide internet connection
5. Implement a service worker for collecting and sending Distance Measurements
6. Run the script on start up

## 3.1 The Armbian Operating System

We are having standard Armbian Images

.

## 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 Distance Server Configuration

print "Distance measurement in progress"

gpio.init()

TRIG1 = port.PA21

ECHO1 = port.PC3

TRIG2 = port.PA19

ECHO2 = port.PC2

gpio.setcfg(TRIG1, gpio.OUTPUT)

gpio.setcfg(ECHO1, gpio.INPUT)

gpio.setcfg(TRIG2, gpio.OUTPUT)

gpio.setcfg(ECHO2, gpio.INPUT)

gpio.output(TRIG1, 0)

gpio.output(TRIG2, 0)

print "Waiting for sensor1 to settle"

gpio.output(TRIG1, 0)

time.sleep(1)

gpio.output(TRIG1, 1)

time.sleep(0.00001)

gpio.output(TRIG1, 0)

while gpio.input(ECHO1) == 0:

pulse\_start = time.time()

while gpio.input(ECHO1) == 1:

pulse\_end = time.time()

pulse\_duration = pulse\_end - pulse\_start

distance1 = pulse\_duration \* 17150

distance1 = round(distance, 2)

print("Distance1: " + str(distance) + "cm")

print "Waiting for Sensor2 To Settle"

gpio.output(TRIG2, 0)

time.sleep(1)

gpio.output(TRIG2, 1)

time.sleep(0.00001)

gpio.output(TRIG2, 0)

while gpio.input(ECHO2) == 0:

pulse\_start = time.time()

while gpio.input(ECHO2) == 1:

pulse\_end = time.time()

pulse\_duration = pulse\_end - pulse\_start

distance2 = pulse\_duration \* 17150

distance2 = round(distance, 2)

print("Distance2: " + str(distance) + "cm")

## 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 DISTANCE DATA*

...

## 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:

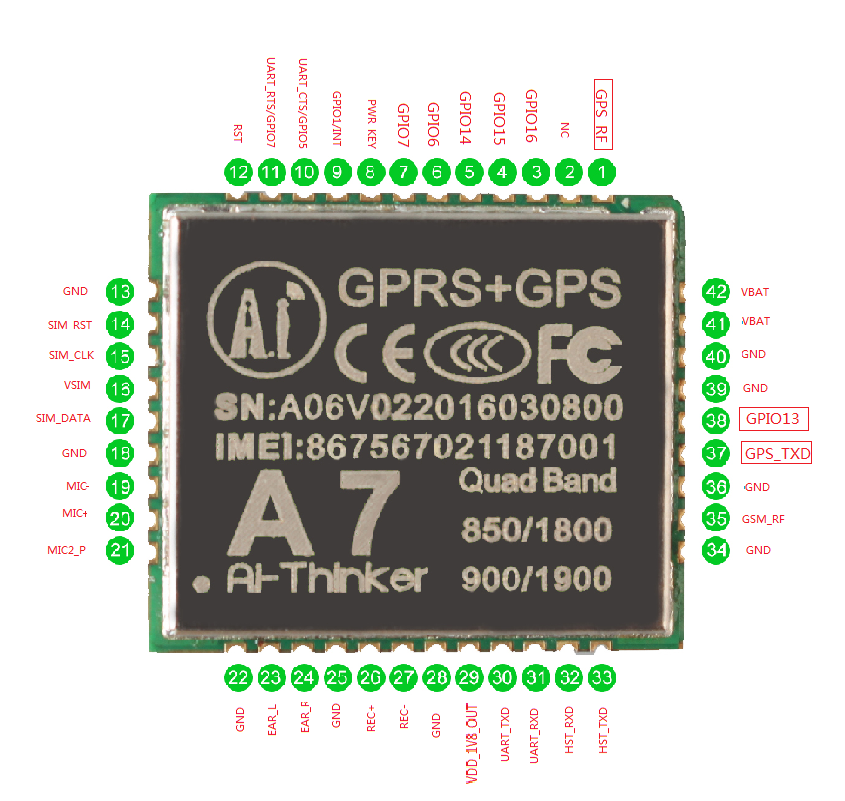
$ sudo nano /etc/ profile

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

sudo python /home/pi/myscript.py

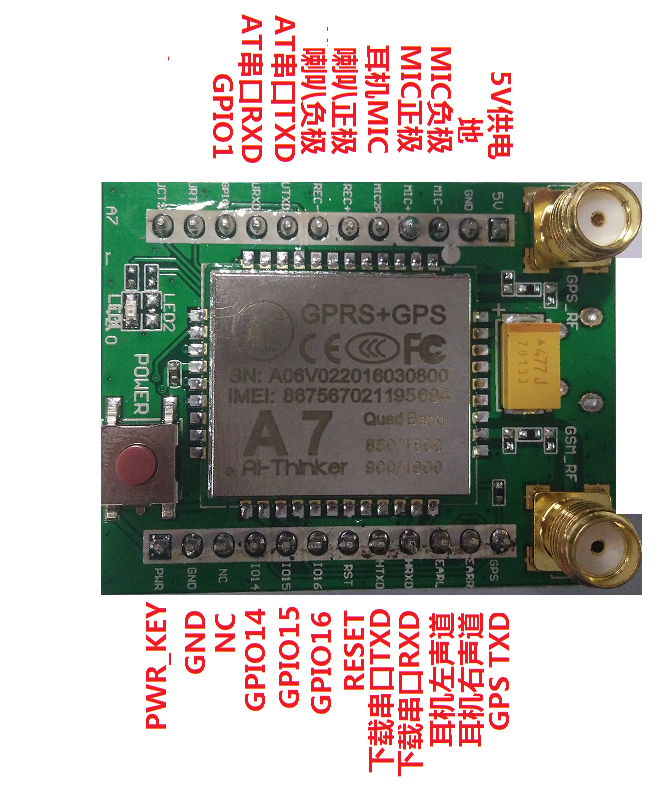
# A7 GSM+GPRS+GPS+AGPS Module

## A7 GSM Module

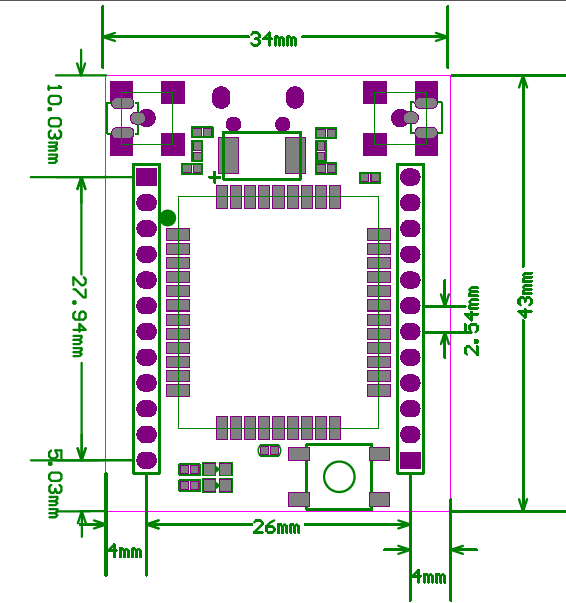


|  |  |  |
| --- | --- | --- |
| PIN No. | PIN Name | Pin Description |
| 1 | GPS\_RF | GPS RF Antenna PCB50 ohm |
| 2 | GND | Ground |
| 3 | GPIO16 | GPIO16 |
| 4 | GPIO15 | GPIO15 |
| 5 | GPIO14 | GPIO14 |
| 6 | GPIO6 | GPIO6 |
| 7 | GPIO7 | GPIO7 |
| 8 | PWR\_KEY | Supply Voltage >1.9V for more than 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

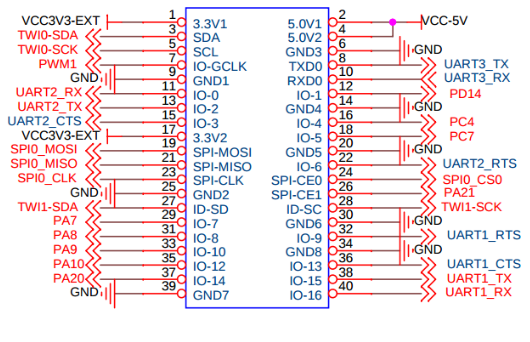


# AT Command for A7 Module

## GPRS and GSM AT Command

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

## PI40 Pin Description



# Pin Connection

* Pin no 2 ( Vcc 5 V ) of Orange Pi one will connect to 5V Pin of GSM Module
* Pin no 6 (GND) of Orange Pi one will connect to Ground pin of GSM Module
* Pin no 2 ( Vcc 5 V ) of Orange Pi one will connect to 5V Pin of First Distance Sensors
* Pin no 6 (GND) of Orange Pi one will connect to Ground pin of First Distance Sensors
* Pin no 2 ( Vcc 5 V ) of Orange Pi one will connect to 5V Pin of Second Distance Sensors
* Pin no 6 (GND) of Orange Pi one will connect to Ground pin of Second Distance Sensors
* Pin no 29 (PA7) of Orange Pi one will connect to POWER\_KEY pin of GSM Module
* Pin no 38 ( TxD ) of Orange Pi one will connect to AT Rxd Pin of GSM Module
* Pin no 40 (RxD) of Orange Pi one will connect to AT Txd pin of GSM Module
* Gpio Pin no (port.PA21) of Orange Pi one will connect TRIG1 Pin of First Distance Sensors
* Gpio Pin no (port.PC3) of Orange Pi one will connect to ECHO1 pin of First Distance Sensors
* Gpio Pin no (port.PA19) of Orange Pi one will connect TRIG2 Pin of First Distance Sensors
* Gpio Pin no (port.PC2) of Orange Pi one will connect to ECHO2 pin of First Distance Sensors

## Data and Control Flow

Orange Pi one will provide power to GSM Module

* Pin no 2 ( Vcc 5 V ) and
* Pin no 6 (GND)
* Pin no 29 (PA7) of Orange Pi one will reset and start the GSM Module by giving 2 sec power

Orange Pi one will control the GSM Module through AT Commands

* Pin no 38 ( TxD ) will used to send the AT command to GSM Module
* Pin no 40 (RxD) will used to receive the response of AT Commands

Orange Pi one will control the Distance Sensor Module through ECHO and TRIG Command

* Pin no ?? (PA21) will used to send the TRIG1 command to Distance Sensor Module
* Pin no ?? (PC3) will used to receive the ECHO1 from sensor Modules

Orange Pi one will control the Distance Sensor Module through ECHO and TRIG Command

* Pin no ?? (PA19) will used to send the TRIG2 command to Distance Sensor Module
* Pin no ?? (PC2) will used to receive the ECHO2 from sensor Modules

# 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

# GPIO Pin Configuration

Connect with first distance sensor

* TRIG1 = port.PA21
* ECHO1 = port.PC3

Connect with second distance sensor

* TRIG2 = port.PA19
* ECHO2 = port.PC

# Orange PI one Board Initialization

## SD Card Preparation and Kernel Initialization

* Ubuntu Linux Server will be flashed in the SD Card

## Distance 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);

}

# References on Distance Sensors

# Ultrasonic Sensor HC-SR04 Sample Python Code for Orange Pi PC Plus v1.1

The code in this repository is similar to the one at [ModMyPi](https://www.modmypi.com/blog/hc-sr04-ultrasonic-range-sensor-on-the-raspberry-pi) however this code works on the Orange Pi PC Plus v1.1 (using a Raspbian Image).

I used the following library to make GPIO pins work on my board:

[Orange Pi PC GPIO PyH3](https://github.com/duxingkei33/orangepi\_PC\_gpio\_pyH3)

Pay attention how much voltage you send to your input pin! The ultrasonic sensor sends 5V. Use resistors to make it around 3V.

Some images about the setup:

![alt tag](https://raw.githubusercontent.com/balazspekar/ultrasonic-sensor-orange-pi-pc-plus/master/1.jpg)

![alt tag](https://raw.githubusercontent.com/balazspekar/ultrasonic-sensor-orange-pi-pc-plus/master/2.jpg)

![alt tag](https://raw.githubusercontent.com/balazspekar/ultrasonic-sensor-orange-pi-pc-plus/master/3.jpg)

Orange Pi Pc Plus v1.1 PIN Layout:

![alt tag](https://raw.githubusercontent.com/balazspekar/ultrasonic-sensor-orange-pi-pc-plus/master/4.jpg)

https://www.rhydolabz.com/wiki/?p=16325