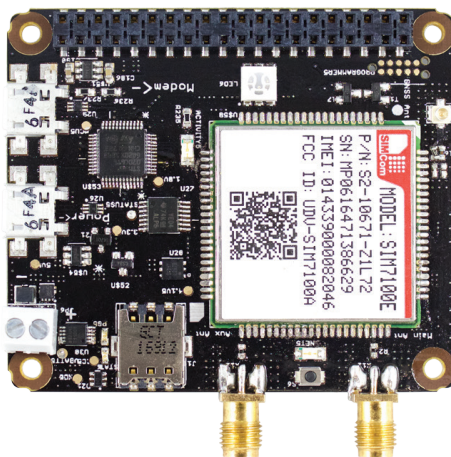




IoT Bit

INTERNET OF THINGS MODULES IN BIT'S



IoT Bit 4G v1.6
User Guide

IoT Bit 4G

Specifications

Modem	(100 mbps down/ 50 mbps up) Excellent for large downloads and video streaming.
Processor	Arm Cortex M0
Dimensions	65 x 65 x 11mm
Power	Operates at 3.3V, Micro USB socket 5V1, 3.6 – 4.2V LiPo battery
Power Consumption	Idle ~25mA Modem on ~100mA Sending SMS ~200mA Making Call ~250mA Transmitting Data ~400mA
Regions	America, Asia, Europe

Connectors:

Micro USB	Micro USB 2 x USB 2.0 Connector
Sim Card	Micro Sim
GPIO Connector	40-pin 2.54 mm (100 mil) expansion header: 2x20 strip
Battery Terminal	2-pin battery terminal
GPS	U.FL Connector
Antenna	SMA Female

HAT Features

- 4G Mobile Data for the Raspberry Pi
- Single terminal command set-up
- Modular GPS antenna support
- Compatible with Sensly HAT
- Supports any micro sim
- Real Time Clock
- High-efficiency power regulation
- Compatible with solar panels & battery packs

Key Applications

- Mobile data hotspot
- Media Streaming
- Industrial/Home automation
- Print server
- GPS Tracking
- Wireless access point
- Environmental sensing/monitoring (e.g. weather station)
- IoT applications
- Robotics
- Server/cloud server
- Security monitoring
- Gaming

IoT Bit 4G

Summary

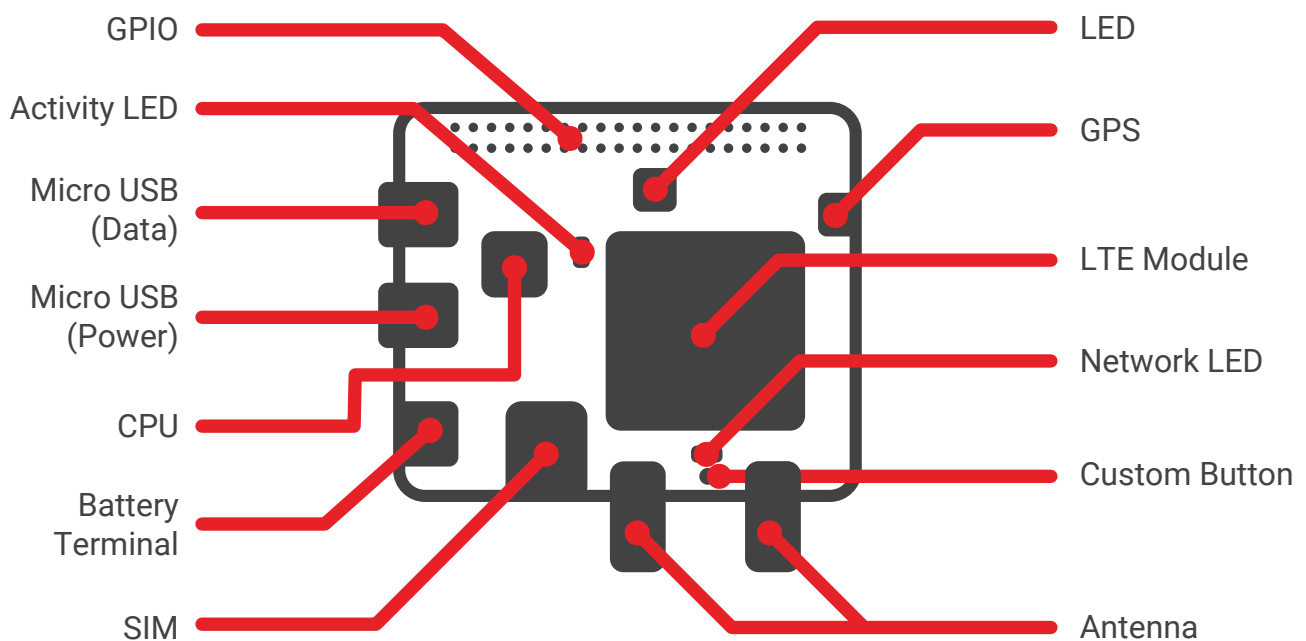
Product Name

IoT Bit 4G

Product Description

IoT Bit is proud to present the 4G Development board, a 4G HAT for the Raspberry Pi which provides mobile data for the Raspberry Pi and other mini computer. Our intelligent HAT module provides your Raspberry Pi with mobile data, GPS positioning information and battery support. This is the perfect module for hackers, scientists, and creators as it gives your Pi powerful connectivity wherever you are. Simply plug our module into your Raspberry Pi and start playing.

Board Layout:



IoT Bit 4G

LTE Region

Module / Region	LTE Bands	3G/HSPA+ Bands	GSM/ GPRS/EDGE Bands	GNSS Technology	Certification	Network lock
SIM7100A (North America, South America)	B2, B4, B5, B17	B2, B5	No GSM Support	GPS, GLONASS	FCC, AT&T, IC, PTCRB	No
SIM7100E (Europe)	B1, B3, B7, B8, B20, B38, B40	B1, B8	900/1800 MHz	GPS, GLONASS	CE	No

Figure Index

Figure 1 - **IoT Bit kit**
Figure 2 - **IoT Bit kit connected**
Figure 3 - **IoTBit dimensions**
Figure 4 - **40-pin header**
Figure 5 - **Power USB**
Figure 6 - **Battery port**
Figure 7 - **Modem USB**
Figure 8 - **Sim Card positio**
Figure 9 - **Sim Card orientation**
Figure 10 - **Simcom module**
Figure 11 - **Botton**
Figure 12 - **RGB Led**
Figure 13 - **Antennas**

Figure 14 - **GPS**
Figure 15 - **UART enable config file**
Figure 16 - **UART enable cmdline file before**
Figure 17 - **UART enable cmdline file after**
Figure 18 - **UART modem testing**
Figure 19 - **Firmware port**
Figure 20 - **Firmware Upload**
Figure 21 - **Testing the drivers**
Figure 22 - **Installation files**
Figure 23 - **Wvdial interface**
Figure 24 - **Interfaces file**
Figure 25 - **IoT Bit GUI**
Figure 26 - **rc.local file**

1.Introduction

- 1.1 Safety information
 - 1.2 Legal information
-

2. Overview

- 2.1 Features
 - 2.2 View of IoT Bit board
 - 2.3 Interface overview
-

3. Interface Application

- 3.1 Power supply
 - 3.2 Battery
 - 3.3 USB
 - 3.4 SIM card
 - 3.5 Simcom Module
 - 3.6 UART
 - 3.7 Buttons
 - 3.8 LEDs
 - 3.9 Antenna connection
-

4. Operation Procedures

- 4.1 Firmware upgrade
 - 4.2 Setting up the IoT Bit to work with Raspberry Pi
 - 4.2.1 Easy setup
 - 4.2.2 Test drivers
 - 4.2.3 Old version
 - 4.2.4 Raspbian
 - 4.2.5 Enable UART
 - 4.2.6 Minicom
 - 4.2.7 Network Manager GNOME
 - 4.2.8 Set up wvdial
 - 4.2.9 Setting up the network interface to use the modem automatically
 - 4.2.10 GUI
 - 4.3 GPS
 - 4.4 Windows drivers
-

5. Troubleshooting

1. Introduction

This document contains the information for the IoT Bit (version number). It describes the process, safety information and different components for the IoT Bit kit.

1.1 Safety Information

Do not expose to water or high moisture environments.
The board should be placed on a flat, stable, non-conductive surface.
Minimize the handling of the board when connected.
Do not operate the device next to flammable gases or fumes.
Do not operate in any kind of aircraft. Make sure device is switched off for flight duration.
RF interference can occur if it is used next to a TV, radios, computers or other electronic equipment.
Do not operate in environments where the network is restricted such as hospitals or emergency call centres.
The incorrect connection of the 40-header pin may result in damage to the device.
Do not place next to a heat source.

1.2 Legal Information

Do not use these products for any illegal activities.
Warranty and legal documentation is available to view at:

Altitude.tech/altitude-tech-terms-and-conditions/

COPYRIGHT

THE INFORMATION CONTAINED HERE IS PROPRIETARY TECHNICAL INFORMATION OF ALTERED CARBON LTD. TRANSMITTING, REPRODUCTION, DISSEMINATION AND EDITING OF THIS DOCUMENT AS WELL AS UTILIZATION OF THE CONTENT ARE FORBIDDEN WITHOUT PERMISSION. OFFENDERS WILL BE HELD LIABLE FOR PAYMENT OF DAMAGES. ALL RIGHTS ARE RESERVED IN THE EVENT OF A PATENT GRANT OR REGISTRATION OF A UTILITY MODEL OR DESIGN.

2. Overview

2.1 Features

- Supports any micro SIM. Slot your SIM card in and get going.
 - 4G Mobile Data for the Raspberry Pi and other mini computers.
 - Easy set-up with our software to streamline communication between IoT Bit and Raspberry Pi.
 - Trigger events with text messages.
 - External antennas for improved signal reception.
 - Battery support for remote applications.
 - Solar panel and battery pack support.
 - Real-Time clock.
 - High-efficiency power regulation.
 - Compatible with solar panels & battery packs
-

2.2 View of IoT Bit board

Accessories include:

- 40-Pin Header
- USB
- Antenna
- GPS



Figure 1 - IoT Bit Kit

Assembly:

IoT Bit with all the accessories plugged in.

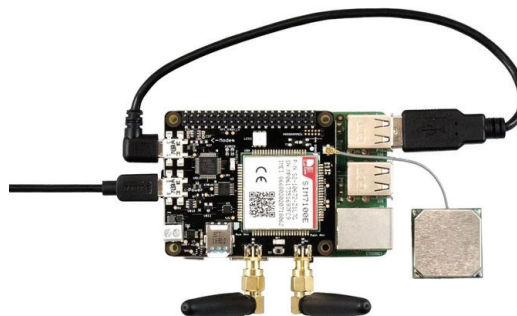


Figure 2 - IoT Bit Kit Connected

2.3 Interface Overview

Physical Specification:

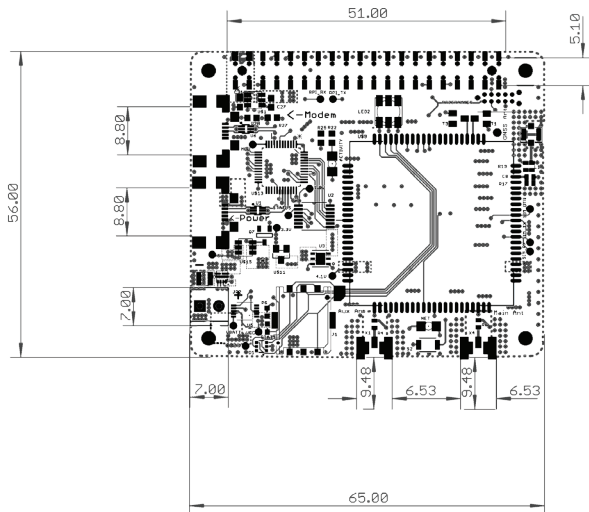


Figure 3 - IoT Bit Dimensions

GPIO:

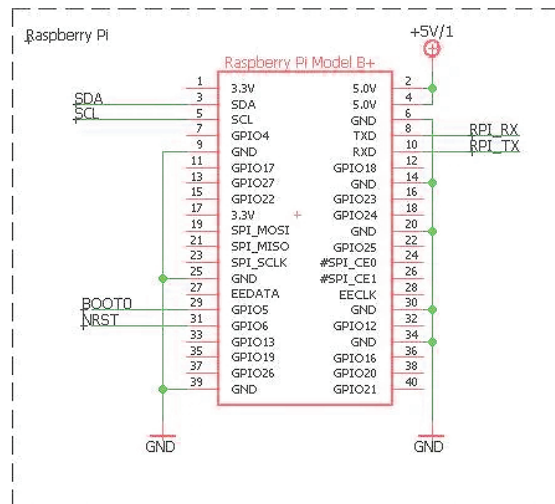


Figure 4 - IoT Bit GPIO 40-Pin Header

Pin 2 - +5V
Pin 4 - +5V

Pin 3 - SDA
Pin 5 - SCL

Pin 8 - RX
Pin 10 - TX

Pin 29 - BOOT0
Pin 31 - NRST

RGB LED

Triple colour LED, red, green and blue

Activity LED

Is reserved for future use.
Currently used for in-house debugging

Network LED

Blue colour LED

Power LED

Green colour LED

Charge LED

Blue colour LED

Micro USB (Data)

USB used to communicate with the modem of the Simcom chip

Micro USB (Power)

USB power input

MCU

Cortex M0 microprocessor
STM32F070CBT6

Mobile Data Module

Simcom chip version:
- SIM7100X (Region specific) **4G**

Battery Terminal

Port to connect a battery to make a power-independant system

SIM

IoT Bit support any 4G data nano SIM card

Custom Button

Push button

Antenna MAIN

The main antenna used for receiving and sending data

Antenna AUX

Auxillary antenna used for receiving

GPS

GPS antenna used for geographical location

3. Interface Application

3.1 Power supply

The USB type 2.0 labelled POWER is used for power IoT Bit. The voltage input should be 5v with a minimum of 2A current for the IoT Bit. If the Raspberry Pi is powered via the IoT Bit, the input current should be at least 2.5A.



Figure 5 - Power USB

3.2 Battery

The battery should be a Li-Po with 3.7v and a minimum of 2 MAh capacity. The input voltage shouldn't be higher than 4.2v. The battery charging circuit can be damaged if it exceeds this and will void your warranty.

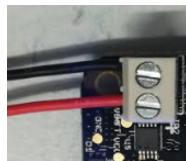


Figure 6 - Battery Port

3.3 USB (modem)

USB type 2.0 labelled MODEM is used to connect the modem, it transmits data through the cable to the Raspberry Pi or a Windows system. This USB interface gives you access to a few different ports on the modem.



Figure 7 - Modem USB

List of ports accessible via the USB:

- Diagnostics port for output developing messages
- NMEA port - outputs GPS information
- AT command port - for sending and receiving AT commands
- Modem port for PPP protocol
- USB-Audio port - use to transmit Audio signals

3.4 Sim card

The SIM card port connects with the Simcom module.
The Simcom module uses the SIM card with the different functions:



SMS



Phone calls



Internet access

Sim card orientation:

The orientation of the sim card is essential for the IoT Bit to operate. The chip of the SIM card should be facing down and the orientation of the card should be as shown in the figure above, the SIM card needs to be fully inserted.

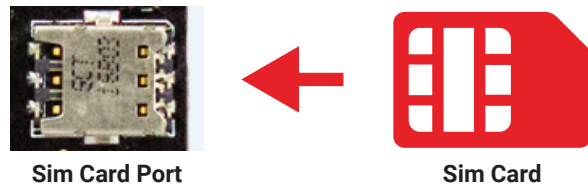


Figure 8 - Sim Card Position



Figure 9 - Sim Card Orientation

3.5 Simcom Module

This is the 4G/LTE/3G/2G Module used for mobile data communication on the IoTBit.



Figure 10 - Simcom Module

The datasheet can be found in the link below:

https://simcom.ee/documents/SIM7100E/SIM7100A%20SPEC_V1503.pdf

3.6 UART

A communication channel between the Raspberry Pi and the processor to access different functions:

- Power Management
- Communication between the Simcom chip and the processor

3.7 Buttons

Reserved for future use.
At the moment, this is used for in-house development only.

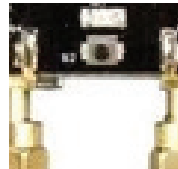


Figure 11 - Button

3.8 LEDs

RGB LED:

This will turn Red at the start, meaning that the board is not connected to the modem USB, also can indicate if the SIM card is not in the slot. When the RGB is blue, the USB modem is ready to connect and pair with any other device via the USB modem. This LED will have more functions with future versions.

White - System booting up, power has been connected.

Red - System failure, going into reset mode.

Blue - Initial testing for the system.

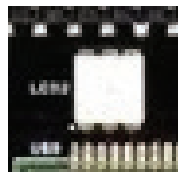


Figure 12 - RGB LED

Activity LED:

Used for the initialisation of the modem, if it remains on the modem is hanging.
Currently is used for in-house debugging.

Network LED:

When it turns on it shows that the IoT Bit is reading the SIM card. When this LED is blinking, it means that the signal is locked to the provider of the SIM card. If there is any internet data available on the SIM card, the LED will start blinking even more frequently, which means it should be able to transfer data.

Power LED:

This LED should be on when the IoT Bit is connected via USB.
When the IoT Bit is powered via battery, this LED will be off.

Charge LED:

LED is on when the IoT Bit is connected via USB. When the IoT Bit is powered via battery, this LED will be off. This LED also works to indicate when the battery is fully charged. If the IoT Bit is connected via USB and the battery is connected too, once the battery is charged this LED will turn off.

3.9 Antenna Connections

Antenna Ports:

The IoT Bit needs the antennas to be able to pick up the signal from the SIM card provider. In the store, we facilitate a range of antennas for the needs of the project in hand. The configuration of setting up the antennas is essential for the IoT Bit to perform correctly. To get the most of the data signal, both antennas should be connected the main and the auxiliary antenna ports.

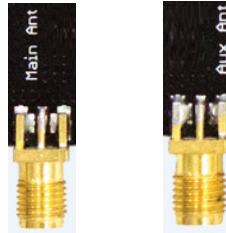


Figure 13 - Antennas

GPS Antenna Port:

Each IoT Bit comes always with a GPS antenna that connects to this port. The GPS antenna does not need to be connected for the IoT Bit operate.

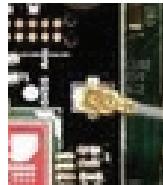


Figure 14 - GPS

4. Operation Procedures

4.1 Firmware Upgrade

Development of IoT Bit is an ongoing process and updates of the firmware will be announced on our website.

http://download.altitude.tech/Firmware_Updates/IoTBit

Before you start you will need to enable the **UART** peripheral.
This allows the IoT Bit to be updated via the Raspberry Pi.

4.1.1 Enable UART

It is recommended that at this point you backup the system.

Open terminal and enter:

\$ sudo nano /boot/config.txt

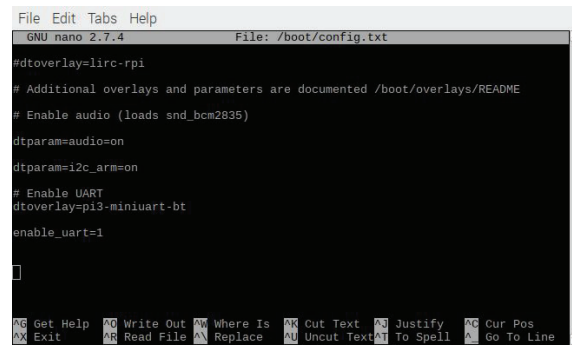
Add the following line to **/boot/config.txt**

enable_uart=1
dtoverlay=pi-miniuart-bt

Save the file using **CTRL-X** then **Y** and **ENTER**.

Next open up the **cmdline.txt** file using
the following command.

\$ sudo nano /boot/cmdline.txt



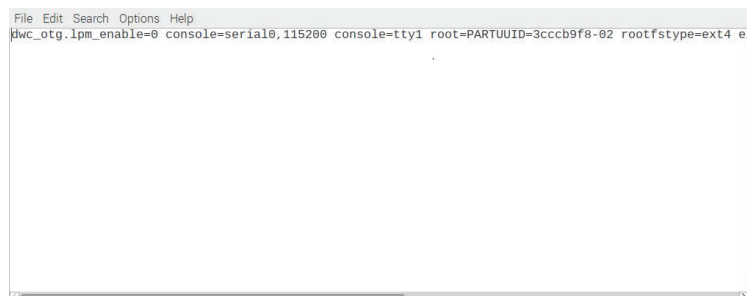
```
File Edit Tabs Help
GNU nano 2.7.4 File: /boot/config.txt

#dtoverlay=lirc-rpi
# Additional overlays and parameters are documented /boot/overlays/README
# Enable audio (loads snd_bcm2835)
dtparam=audio=on
dtparam=i2c_arm=on
# Enable UART
dtoverlay=pi3-miniuart-bt
enable_uart=1
```

Figure 15 - UART Enable Config File

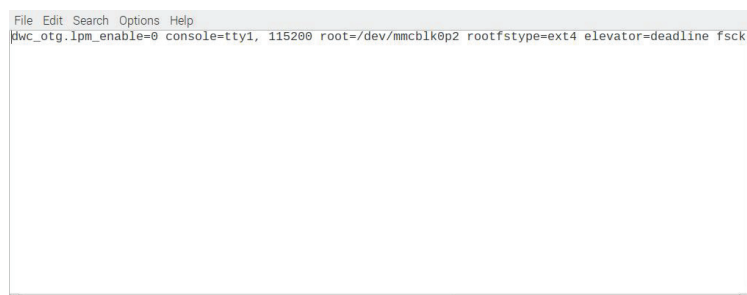
Modify the file so the contents from the line below replace the second line:

1. dwc_otg.lpm_enable=0 **console=serial0,115200 console=tty1**
root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait
2. dwc_otg.lpm_enable=0 **console=tty1** root=/dev/mmcblk0p2 rootfstype=ext4
elevator=deadline rootwait



```
File Edit Search Options Help
dwc_otg.lpm_enable=0 console=serial0,115200 console=tty1 root=PARTUUID=3cccb9f8-02 rootfstype=ext4 elevator=deadline rootwait
```

Figure 16 - UART Enable Cmdline File BEFORE



```
File Edit Search Options Help
dwc_otg.lpm_enable=0 console=tty1, 115200 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline fsck
```

Figure 17- UART Enable Cmdline File AFTER

\$ sudo reboot

To test this use:

```
$ sudo apt-get install minicom
$ minicom -D /dev/serial0
```

Depending on you version of Raspbian, your **UART port** will either be **ttyAMA0**, **ttyS0** or **serial0**. If you are using Raspbian Stretch it will be **serial0**.

You should now see '**Modem Ready**' continuously repeated.

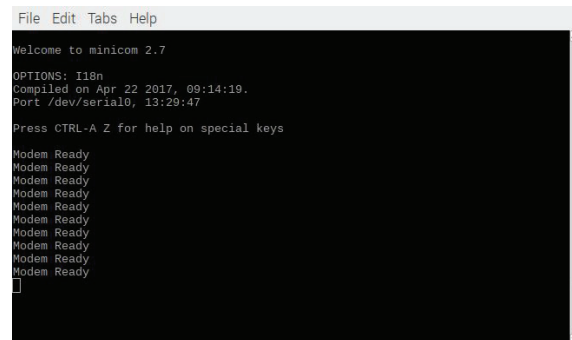


Figure 18 - UART Modem Testing

In order to run the firmware update, it is required to download the latest version of the firmware:

\$ http://download.altitude.tech/Firmware_Updates/IoTBit

Once **UART** has been enabled, we are ready to upload the new firmware.

ATTENTION! It is important at this stage, before continuing, to power the IoT Bit and the Raspberry Pi independently to do this step. Execute the python script as shown:

\$ sudo python (name of the file)

The script will ask you for the serial port if you are using Raspbian stretch and above you can input **'/dev/serial0'**.

If the upload is successful, the program will write and read from multiple memory addresses then say **'verification ok'**.

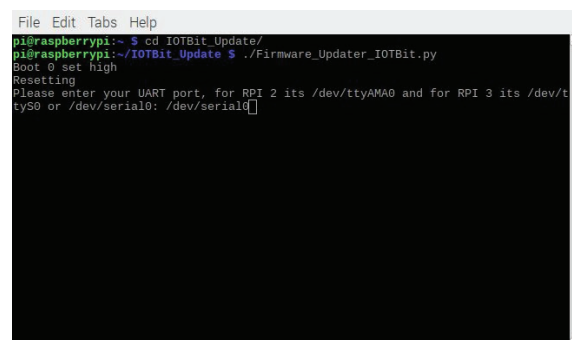


Figure 19 - Firmware Port

Once the upload of the firmware is complete, **restart the system.**

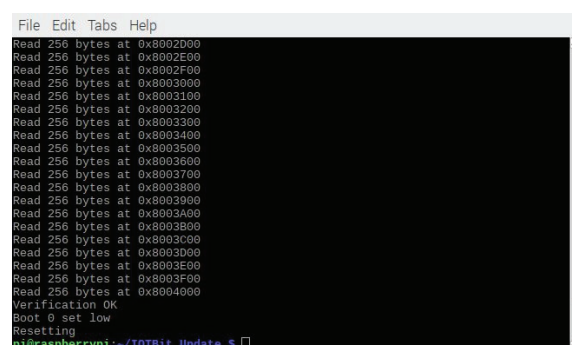


Figure 20 - Firmware Upload

4.2 Setting up the IoT Bit to work with Raspberry Pi

There are multiple ways to get this up and running. The easiest setup is to download the latest image version in our store. This is an image with all the following features already pre-installed. The other option is to install the drivers manually and only enable features you want.

The other option is to install minicom manually and use just the features you want in your system.

4.2.1 Easy setup

Download the disk image and flash it using Win32 Disk Imager or equivalent. You will need an SD card that is **8GB** or bigger. This is an image with all the following features already pre-installed.

Download the latest version below:

[http://download.altitude.tech/IOTBit%20\(Formerly%20Pianywhere\)/Dev/18-04-17/IOTBitQMLinterface-18-04-17.zip](http://download.altitude.tech/IOTBit%20(Formerly%20Pianywhere)/Dev/18-04-17/IOTBitQMLinterface-18-04-17.zip)

To use insert the SD card into the Raspberry Pi then turn on the IoT Bit by plugging in the power cable to the USB port labelled POWER. Once connected, the IoT Bit will power the Raspberry Pi.

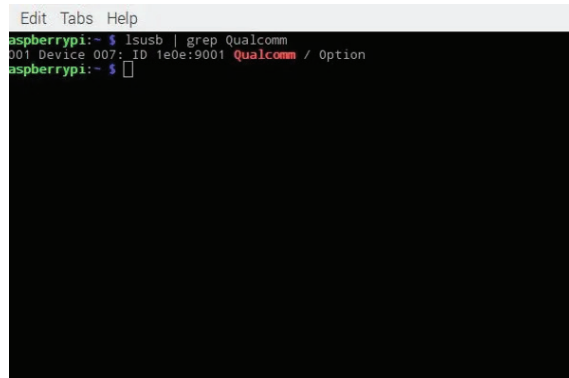
4.2.2 Test Drivers

To test if the drivers are installed, run the following command.

```
$ lsusb | grep Qualcomm
```

The output will show a Qualcomm device is connected if the drivers are installed.

If you do not see a **Qualcomm** device, this means that you are probably using a version of Raspbian **older** than Raspbian stretch. In this case please follow the guide below for instructions on how to manually install the driver.



```
Edit  Tabs  Help
raspberrypi:~ $ lsusb | grep Qualcomm
001 Device 007: ID 1e0e:9001 Qualcomm / Option
raspberrypi:~ $
```

Figure 21 - Testing the Drivers

4.2.3 OLD version

For version 4.4.1 of Raspbian and earlier follow these steps:

Before starting make sure that your Raspberry pi's date and time is correct and you are connected to the internet then open the terminal and type:

```
$ git clone https://github.com/Altitude-Tech/IOTBit_Install
$ cd IOTBit_Install/
$ chmod u+x ./IOTBit_Install.sh
$ sudo ./IOTBit_Install.sh
```

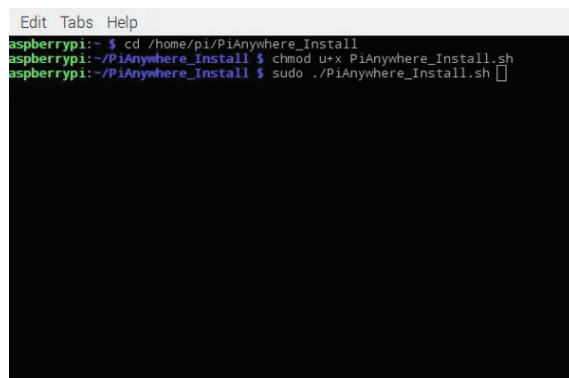
This should take around **30 minutes** to complete as installing the kernel headers takes a long time. Once this is completed and your IoT Bit is connected via **USB** to the Raspberry Pi, you should be able to run:

```
$ lsusb | grep Qualcomm
```

Then you should see a Qualcomm device connected. As a final check you can also run:

```
$ ls /dev/ttyUSB*
```

Finally, there should be 5 usb devices connected. These are the virtual com ports for the modem.



```
Edit  Tabs  Help
raspberrypi:~ $ cd /home/pi/PiAnywhere_Install
raspberrypi:~/PiAnywhere_Install $ chmod u+x PiAnywhere_Install.sh
raspberrypi:~/PiAnywhere_Install $ sudo ./PiAnywhere_Install.sh
```

Figure 22 - Installation Files

4.2.4 For Raspbian versions 4.4.2 and higher

Download the latest version of Raspbian as it contains all the necessary drivers to use the IoT Bit. If you already have an image and the objective is just to incorporate IoT Bit into your project jump section 4.2.5 i.e (Minicom)

<https://www.raspberrypi.org/downloads/raspbian/>

Once the image is operating in the Raspberry Pi, it needs to be updated and upgraded using the following commands:

```
$ sudo apt-get update
```

Next, do not have the IoT Bit plugged into the Raspberry Pi for this step. Upgrade all your installed packages to their latest versions with the command:

```
$ sudo apt-get dist-upgrade
```

4.2.5 Minicom

To install minicom type the following command in command prompt:

```
$ sudo apt-get install minicom
```

Access the minicom terminal using:

```
$ minicom -D /PortLocation
```

For the USB bus, **PortLocation** will typically be **/dev/ttyUSB2** to send AT commands to the modem.

In order to see the minicom options press **CTRL+A** and then **Z**, this will show all the different options that can be used. The local echo will need to be enabled. This is done by pressing **CTRL+A** then **E**.

Now you can manually send AT commands to the modem. The AT command reference is a comprehensive guide on all possible commands the Simcom module accepts. **This can be downloaded below:**

<https://goo.gl/joHphC>

4.2.8 Network Manager GNOME

Network Manager GNOME will provide an easy to use GUI to enable you to connect to the internet. In this step, we will show you how to get the IoT Bit connected to the internet using this method.

The first thing is to download Network Manager GNOME by using the following command in the terminal:

```
$ sudo apt-get install network-manager-gnome
```

Then reboot the Raspberry Pi.

```
$ reboot
```

After reboot you should see the Network Manager icon on the top right of the desktop.

Right click and select edit connections, then click on **"Add"**.

Select **mobile broadband** and click on **"create"**, then select **"Any device"** then click **"Next"**.

Select your country and click **"Next"**, then choose the network provider of the SIM card company and click **"Next"**.

Select your data plan and APN and click **"Next"**, then check everything is correct and click **"Apply"**. After this you can save your network and the internet should be ready to use.

4.2.9 Setting up wvdial

This step will cover using wvdial with the IoT Bit instead of Network Manager. The reason for using wvdial instead of Network Manager is when you want to be able to switch between AT command mode and internet mode easily. As once the network is configured via Network Manager, it will have removed access to the ATPort available via the USB.

Now we need to configure the **wvdial.conf** file to enable IoT Bit to connect to the internet using your SIM card. You will need the USSD code used to fetch the SIM cards registered number.

The default is ***99#** and should be the same for most SIM cards or you can just use the SIM cards' phone number if you can't find the USSD code. Just remember that you will have to edit the file to change the number if you change the SIM. We will need to open the file found in the **iotbit_Install** folder called **wvdial.conf** using a text editor of your choice.

```
$ sudo nano /home/pi/iotbit_Install/wvdial.conf
```

Then add to the file your USSD code or phone number in the **'Phone ='** field. Save and exit the file by pressing **CTRL+X** then **Y**.

We now need to back up the **old wvdial.conf** file and move your **modified wvdial.conf** to the correct place for the system to find it.

```
$ sudo mv /etc/wvdial.conf /etc/wvdial.conf.bak
$ sudo mv /home/pi/iotbit_Install/wvdial.conf /etc/wvdial.conf
$ sudo wvdial
```

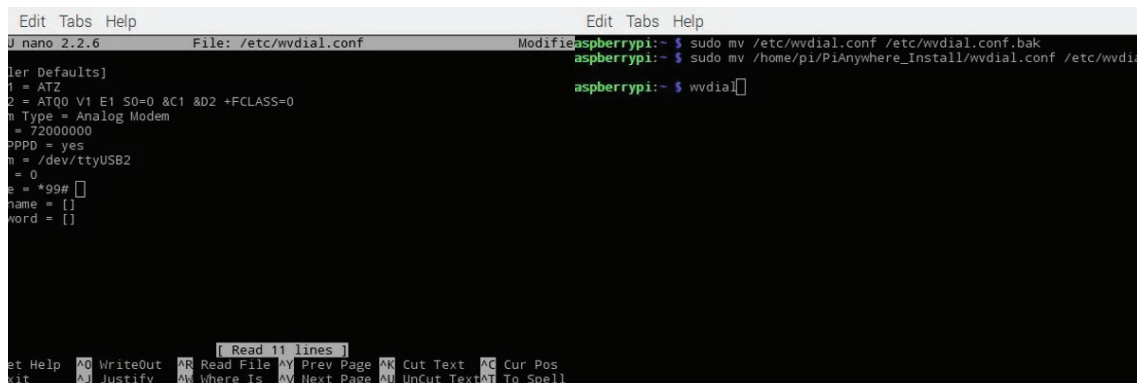


Figure 23 - Wvdial Interface

This last step takes around **30 seconds**.

If successful, you should be connected to the internet with your Raspberry Pi.

4.2.10 Set wvdial to use modem automatically

To enable the Raspberry Pi to connect IoT Bit to the internet using the mobile network automatically, we use the following steps.

First, open the file called interfaces using:

```
$ sudo nano /etc/network/interfaces
```

Then add the following lines at the bottom of the file:

```
$ auto ppp0
$ iface ppp0 inet wvdial
```

Then reboot the Pi, ensuring that the IoT Bit is on and the Modem USB port is connected to the Pi.

```
$ sudo reboot
```

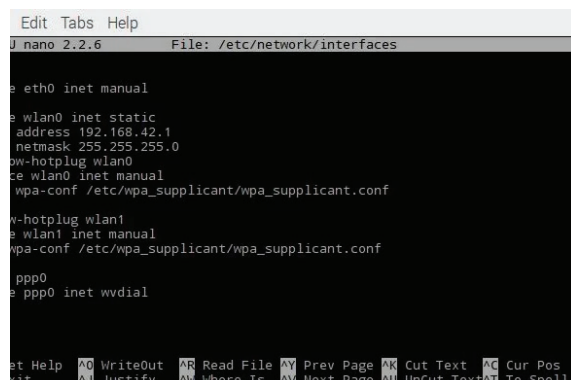


Figure 24 - Interfaces File

If successful when the Raspberry Pi boots you should be able to connect to the internet using your IoT Bit.

4.2.11 GUI

Version 1 GUI functions:

- Send SMS
- Delete SMS
- Hang Up
- Signal Quality
- Display SMS
- Make Call
- Set GPIO to High / Low

These functions simplify communicating with the IoT Bit modem and by using them in an automated script, we'll be able to control hardware attached to the Raspberry Pi's GPIO pins by simply sending an SMS to the IoT Bit. This can be useful in many projects, where you want to turn on or off the devices from a remote location, it can be done with nearly any device, as long as properly set up.

The first thing to do is to run in the Terminal:

```
$ sudo apt-get update  
$ sudo apt-get upgrade
```

This will ensure you have everything up to date and all packages installed. The main module we used in this guide is **PyQt5** for **Python 2.7** which should be installed by using the following command:

```
$ sudo apt-get install python-pyqt5
```

After the PyQt5 is downloaded, you will need "**IoT_Bit_library.py**", "**IoT_Bit_GUI_V1.py**" and "**popup.py**" scripts we have made, to download them go to the GitHub link:

https://github.com/Altitude-Tech/IOTBit_Functions_GUI

After downloading them, save them in the **"/home/pi"** directory.

Before running it we need to give it executable permissions by opening and writing on the terminal:

```
$ sudo chmod +x IoT_Bit_GUI_V1.py
```

To run the GUI script just input into the terminal:

```
$ ./IoT_Bit_GUI_V1.py
```

It's recommended to use **Python 2.7** since this GUI and the library has been created using it. If you want to use any versions above or under **Python 2.7** you will have to modify the code and it might not work optimally.

Instructions:

To **Send an SMS**, you must first enter the phone number in the first entry box. Then enter your text message in the second entry box right next to the send SMS button and click on **send**. If successful, a message should show up on the **Display** box to confirm that it was sent.

To **Display SMS**, click the "**Display All**" to see all messages, 1 to see the first SMS, 2 to see the second SMS, and so on. After inputting the number click the "**Display SMS**" button and the message will show on the Display box below.

To **Delete the SMS**, click the "**Delete All**" button to delete all SMS. To delete a specific SMS select "**Delete Specific SMS**" from the drop-down menu and click "**Delete SMS**" button, a pop up will open for you to input the SMS you wish to delete. After typing the SMS number, press "**OK**" to delete that specific SMS and close the box if you don't wish to delete any other specific SMS. (Remember the SMS you wish to delete refers to the index number, next to the SMS when you click "**Display All**" and not the order in which you see them in the display window). Then to delete all SMS with the status "**Received Read**", select "**Delete Read**" from the drop-down menu and click "**Delete SMS**". To delete all SMS with the status "**Received Read**" and "**Stored Sent**", select "**Delete Read & Sent**" from the drop-down menu and click "**Delete SMS**". To delete all SMS with the status "**Received Read**", "**Stored Sent**" and "**Stored Unsent**" you need to select "**Delete Read, Sent & Unsent**" from the drop-down menu and click "**Delete SMS**".

To **Make a Call** input the phone number into the same entry box as the phone for SMS and click on "**Make Call**", to **Hang Up** just click on the "**Hangup**" button once.

To **Check Signal** you only have to click on the "**Signal Quality**" button and a message will show up on the display box, there are only five options "**Poor Signal**", "**OK Signal**", "**Good Signal**", "**Exceptional Signal**" and "**No Connection**".

Finally you have the **Set GPIO to HIGH/LOW**. This function will set the chosen GPIO pin to either low or high depending on the last text message sent, for example: "**PIN26H**" will set pin 26 to high and "**PIN26L**" will set it to low after the button is clicked. We have done this for pins 26, 19 and 13, other GPIO pins can be added just make sure they are not being used by the IoT Bit.

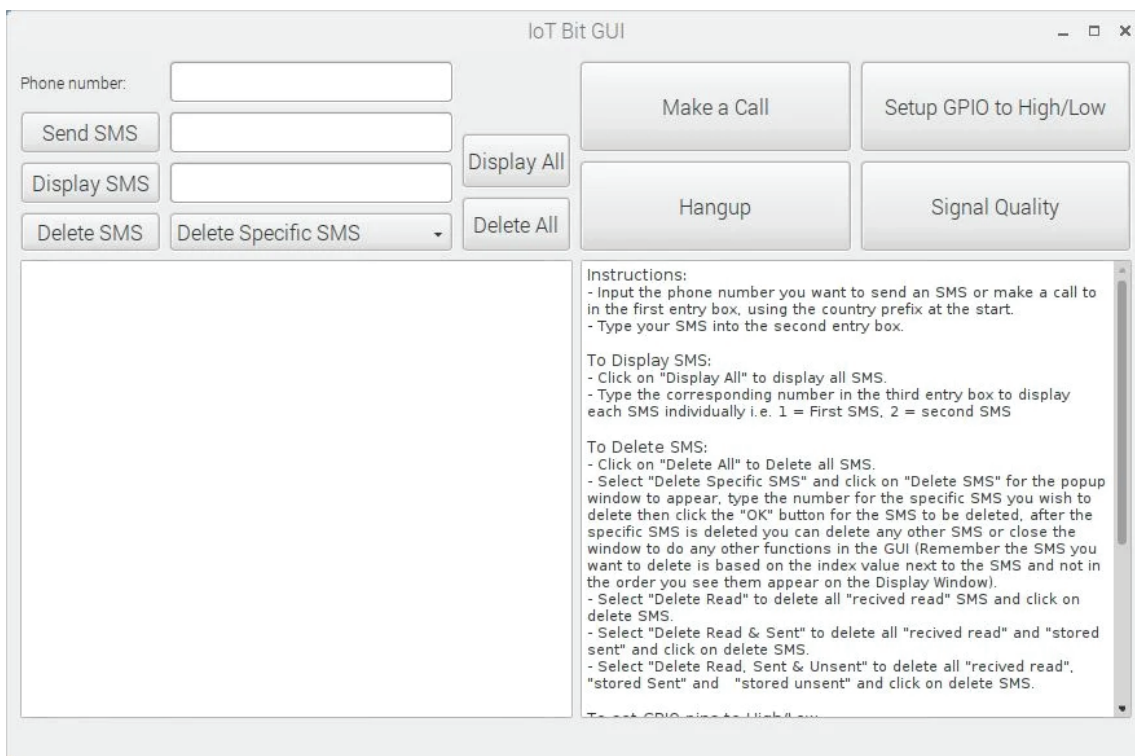


Figure 25 - IoT Bit GUI

4.3 Auto_Script.py Script

To set up the GPIO functions to be running indefinitely you need to download the "**Auto_Bash.sh**" and "**Auto_Script.py**" from this GitHub link and save them in your **"/home/pi"** folder:

https://github.com/Altitude-Tech/IOTBit_Functions_GUI

Then to make the bash script executable do:

\$ chmod +x Auto_Bash.sh

After saving them to your pi and making the bash script executable, all you need to do is open terminal and write on the command line:

\$ sudo nano /etc/rc.local

Then at the end just before the "exit 0" input:

bash /home/pi/Auto_Bash.sh

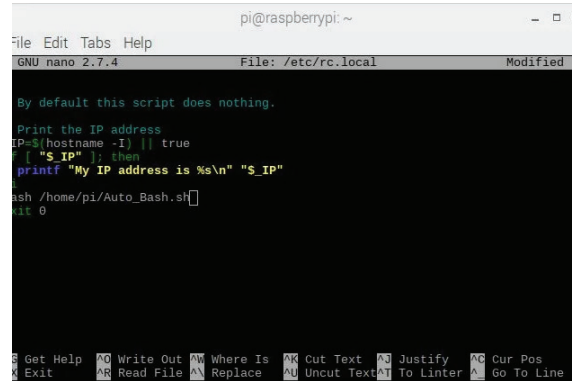


Figure 26 - rc.local file

Reboot your Pi to start up the script.

You are all done **congratulations**.

The script should be running and whenever you send a message to any of the pins that are set up they will go to high or low respectively. Keep in mind you can always change the keyword to set the pin to high or low.

List of SMS Commands:

- | | | |
|----------|----------|----------|
| • PIN26H | • PIN19H | • PIN13H |
| • PIN26L | • PIN19L | • PIN13L |

Remember that you can change the SMS commands to any word you wish and add any GPIO pins not used by the Raspberry Pi. Just go into the "**Auto_Script.py**" and look for the above commands and replicate them for other pins.

4.4 GPS

For enabling the GPS information an AT command is required. Type in the minicom terminal:

AT+CGPS=1

It will show a message of **OK!**. This means the GPS is enabled and ready to use. It typically takes around a minute to get the first lock. To pull out the GPS coordinates, use the following AT command in minicom:

AT+CGPSINFO

4.5 Windows Drivers

IoT Bit can be used on a windows system. All you have to do is download these drivers and install them:

<http://download.altitude.tech/IoT%20Bit%20Windows%20Drivers/>

5. Troubleshooting

Q: Which countries will IoT Bit work?

A: IoT Bit will work in any country with either 4g, LTE, 3g or GSM networks, we work out which one you will need based on your shipping address so please ensure this is correct.

Q: Which IOS system does these products support?

A: Windows, Raspbian, and Ubuntu.

Q: How do I power the IoT Bit and Raspberry Pi?

A: When using a Raspberry Pi with the IoT Bit HAT, power the IoT Bit from the wall using a Raspberry Pi charger.

Q: What is the internet speed of the IoT Bit?

A: The datasheet states that can go up to 100MB, depending on the signal reception of the antenna.

Q: Do you have the dimensions for this product?

A: The board size is 65mm by 56.5mm then the antenna that the supply comes out of is 23mm of the board. So all together is 87mm by 78mm.