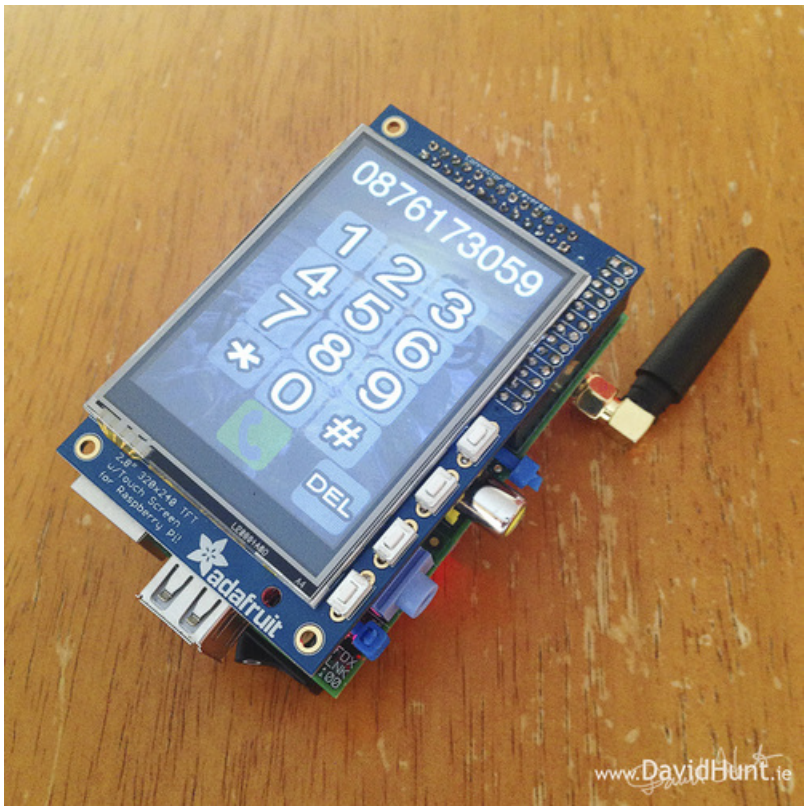
adafruit learning system

PiPhone - A Raspberry Pi based Cellphone

Created by David Hunt

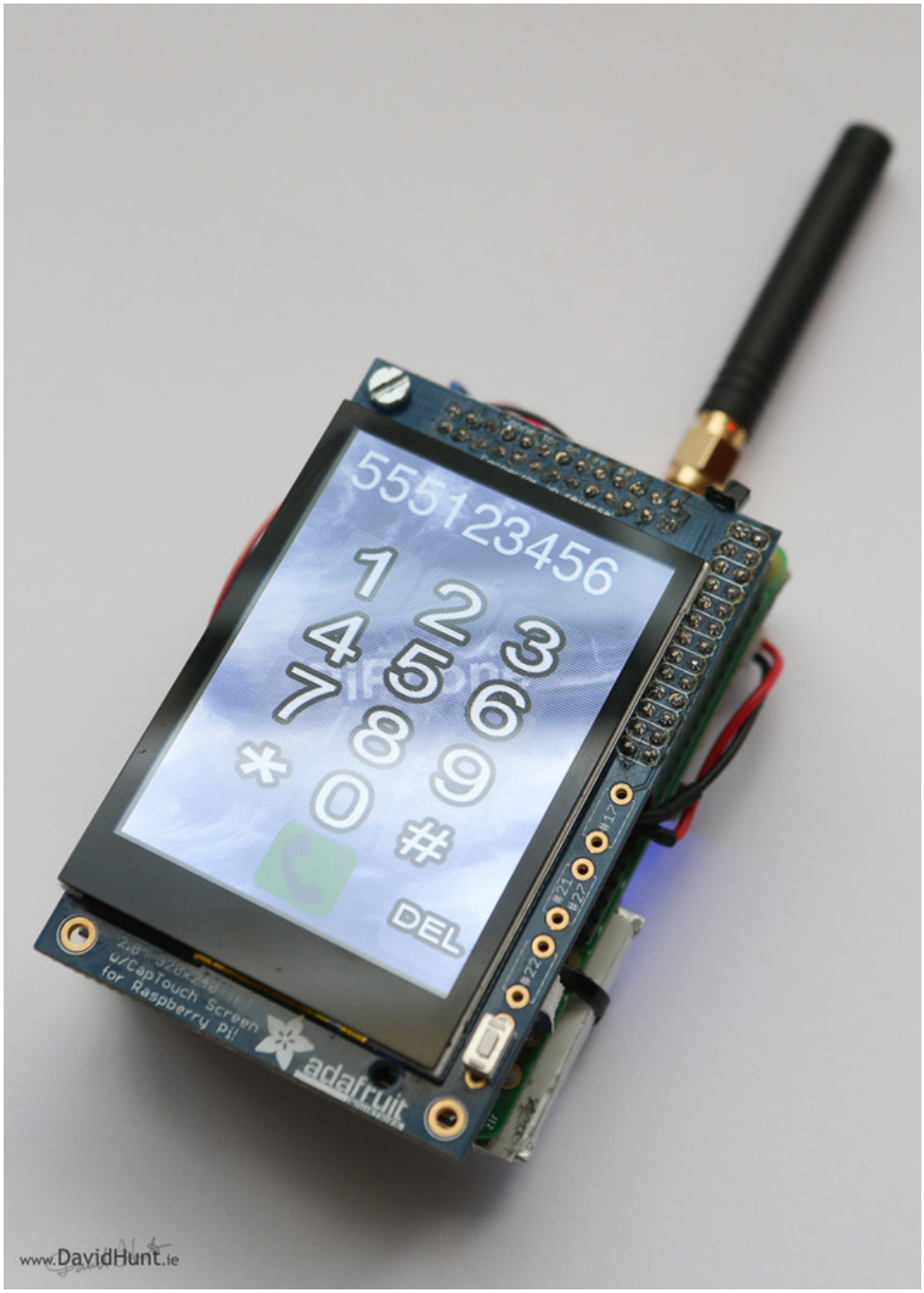


Last updated on 2018-08-22 03:43:47 PM UTC

Guide Contents

Overview

Ever wanted to build you own cellphone? Well now you can with this guide that uses a Raspberry Pi, PiTFT, and a FONA to make a functional cellphone that you can call your friends with!



Things You'll Need:

- **Raspberry Pi computer**, the **Model B (http://adafru.it/998)** or **Model B+ (https://adafru.it/dH0)** is probably easier to set up and get going, but of you are comfortable with the **Model A (http://adafru.it/1344)**, **then that will work**

fine. You don't need any USB ports, and once set up, you don't need the Model B (http://adafru.it/998)'s ethernet port. The Model A (http://adafru.it/1344) save you on power, allowing you to have more talk time :) The **Model B+ (http://adafru.it/1914)** will also work, if you're ok with the PiTFT being slightly offset from the Pi, as the GPIO pins are in a slightly different place. It's also worth noting that the Model B+ uses quite a bit less power than the Model B, so it's probably better suited to a portable application like the PiPhone.

- **Resistive 2.8" PiTFT (http://adafru.it/1601)** or **Capacitive 2.8" PiTFT (https://adafru.it/e9Y)** - 2.8" TFT touchscreen for Raspberry Pi. The capacitive version looks nicer and has a glass top that does not require pressing down, but it's more expensive!
- **SD memory card (http://adafru.it/102),** 4GB or larger.
- **1200mAh Lithium Polymer battery (http://adafru.it/258)**
- **DC-DC converter (http://adafru.it/1903)**
- **FONA (http://adafru.it/1963)** + **Antenna (http://adafru.it/1859)**
- **"Hands-free" Headphones with Microphone (http://adafru.it/1966)**
- You can use a **DC-DC converter (http://adafru.it/1903)** to give us 5V for the Raspberry Pi. Or just go with a **big USB battery pack such as this one (http://adafru.it/1566)**, but it won't look as tidy.

In some situations a **USB to TTL Serial Cable (http://adafru.it/954)** may be the preferred way to log in and configure the Raspberry Pi, if a spare keyboard and monitor are unavailable.

Some additional parts, tools and skills are also required: soldering iron and solder for putting the connectors on the PiTFT display, and connecting the DC-DC converter to the Pi; some means of holding all the pieces together — could be as simple as a few rubber bands, to a drilled-out plastic electronics enclosure, to an elaborate custom 3D-printed case. This all depends on your available resources. Read through to see what's involved in the project and come up with ideas along the way.

For connectors an wiring, the 26-way pin header is supplied with the PiTFT, but for ease of wiring I'd suggest soldering the needed wires directly rather than using the pin header, as you'll need to source a connector to go on to it. But if you do have the relevant connector and crimpt tool, thats all good too.

Check out David Hunt's blog (https://adafru.it/e9Z) for more about this project and other great projects!

If you want to see the original PiPhone in action, check out Dave's video. It uses an older GSM card, but you'll get the idea.

Pi Setup

To ensure that all the software interdependencies can work, it's easiest to start with a clean installation.

So, we need to set up the following:

- Fresh install of Raspbian
- PiTFT Setup
- Python development libraries
- Wiring Pi library
- Wiring Pi Python wrapper
- PiPhone python script

Raspbian Setup

This project is easiest to start with a pre-prepared rasbian image from Adadfriut that's got all the setup done for the PiTFT touchscreen, saving a lot of time. There's also a script for Pi's that already are up-and-running.

There are two versions of the 2.8" PiTFT, the resistive and the capacitive. You can find links to the images and/or setup scripts here:

- Capacitive PiTFT Setup guide (https://adafru.it/l6d)
- Resistive PiTFT Setup guide (https://adafru.it/dDK)

Once the PiTFT is running, we first need to change the orientation of the screen, as we want the keypad to appear in portrait mode. We can do this by editing the adafruit modules config file:

```
sudo nano /etc/modprobe.d/adafruit.conf
```

Change the 'rotate' parameter for the fbtft_device to zero, save the file and exit the editor.

```
options fbtft_device name=adafruitct28 rotate=0 frequency=32000000
```

Connect a monitor and keyboard (or use a USB-to-serial console cable), power the Raspberry Pi from a USB phone charger or powered hub, and work through the usual first-time boot configuration.

The following options are **required**:

- *Expand Filesystem*

The following are very useful and **recommended:**

- Under *Internationalization Options,* select *Change Timezone* and *Change Keyboard Layout* to match your region.

The following are **optional:**

- Under *Advanced Options,* select *Hostname* to give this Pi a unique name (such as "picam") to distinguish it from other Raspberry Pi's on the network.
- Under *Advanced Options,* select *SSH* to enable command line access from the network (helpful for further configuration and troubleshooting).

- Other settings can be configured to your liking.

The following should **not** be used:

- *Overclock.* This is a portable, battery-operated project and an overclocked Pi will draw more current. Overclocked systems are also more likely to corrupt the SD card filesystem. Do not enable this option.

Once the basic system configuration is done, you can also set up wireless networking if you plan on using this capability. This guide may be of assistance (https://adafru.it/aUB).

PiTFT Setup

Once the Pi is fully configured and on the network, work through the relevant PiTFT tutorial, one for the Resistive version and one for the Capacitive version.

For the Resistive Version: Adafruit PiTFT — 2.8" Touchscreen Display for Raspberry Pi (https://adafru.it/d4W)

For the Capacitive Version: Adafruit 2.8" PiTFT - Capacitive Touch (https://adafru.it/jsf)

You may want to read up about the PiTFT and stuff you can do with it. If you're using the ready-to-go image then a cursory review on how to install is handy. Once you have a Pi that boots to a login prompt on the 2.8" TFT, you're in good shape.

You'll also need to calibrate the touch screen!

Likewise, the optional tactile buttons on the PiTFT are not required for this project. You can install the buttons for other things if you like, but the PiPhone software is entirely touchscreen-based.

PiPhone Software Setup

Now we get the PiPhone python script and icons.

```
wget https://github.com/climberhunt/Piphone/archive/master.zip
unzip master.zip
```

Now give it a try. The software must be run as root (using the sudo command) in order to access the TFT display

```
cd PiPhone-master
sudo python piphone.py
```

If all goes well, after a few seconds' initialization you should see the Lapse Pi splash screen , Followed a couple of seconds later by the time-lapse information.

If this *doesn't* happen, an error message should give some sort of troubleshooting guidance; missing library or driver, etc. This is why we recommend working through the TFT tutorial first.

Once youve got that working, have the Pi boot straight into the PiPhone software by editing**/etc/rc.local** and adding the following lines before **exit 0**

```
# Start PiPhone software at boot
cd /home/pi/PiPhone-master
python piphone.py
```

Next time you reboot you should see the text console and then it will start the PiPhone software.

Serial Port Setup

There is almost zero setup in the serial port, we just have to make it available for comminucating with the FONA. This involvles simply disabling the getty that's running by default, which normally allows us to log into the Raspberry Pi over the serial port. Since we now want to connect the FONA to this serial port, we don't want a login prompt to be sent to the FONA.

Open /etc/inittab file

```
sudo nano /etc/inittab
```

and cange the line that contains the getty for /dev/AMA0 by putting a # infront of it:

```
#T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

I'd suggest not deleting the line, as you may want to comment in back in later.

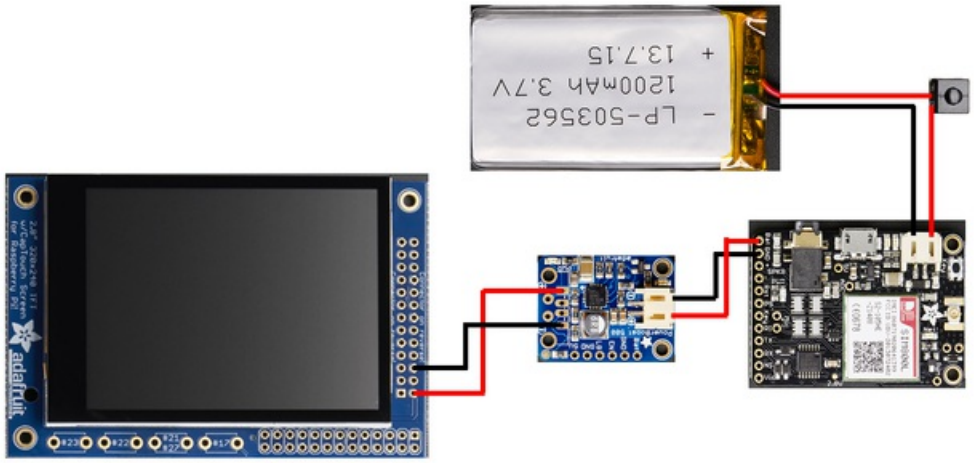
Hardware

Wiring

There are two main sections to the wiring setup, so they're shown in seperate diagrams below. First, we'll start with the Power.

Since the Fona has fantastic power management, we'll make use of that to manage the battery, recharging, etc. The only tricky bit is to get the 5V needed for the Raspberry Pi and the PiTFT, so we'll use a powerboost DC-DC converter for that which will covert the ~3.8 volts from the FONA up to 5V.

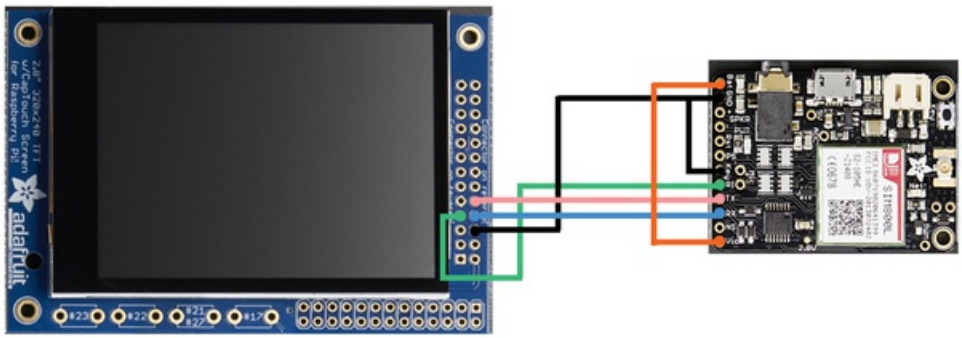
Here's the diaigram of the power wiring:



We put a swtich on the lipo battery so we can power everything on and off from one switch. The battery terminals on the pin heeader are then fed into the 500mA boost DC-DC converyer, which outputs 5V to the PiTFT. We're feeding it into the optional header, as the main header on the piTFT is connected to the Raspberry Pi, so the power feeds through the PiTFT and on to the Raspberry Pi.

For the connections from the FONA to the DC-DC converter and from there to the Raspberry Pi, you can either solder directly onto the boards, or use the 2.5mm pitch pin header and get some suitable connectors to go onto those. Alternatively, you could solder the wires directly into the holes on the PCBs, but that means that you can't take it apart as easily, but it's quicker to get up and running. :)

Next we'll look at how to wire the Fona, including how to connect it back to the Raspberry Pi UART port. Here's the diagram:



There are a few things worth noting:

1. The Vin pin is connected to Vbat, to set the logic level converter. Without this the Fona won't do much. See the **pinout info of the Fona (https://adafru.it/yJc)**.
2. The Key pin is connected to ground, this means that the Fona will ALWAYS be pwered on when there's powere coming into the battery port. This is done so that we can use one switch to powere everything up and down. The downside is that the unit must be powered on during charge.
3. THe Ring Indicator wire is optional. The current PiPhone software does not use it, so it's only really there for when the RI signal is handled in software in the future.

After those items of note, there's really just GND, Rx and TX wires.

All connections are made to the optional header on the PiTFT, as the main header on the Raspberry is taken up with the PiTFT header.

So that's it. As you can see it's really just soldering a few wires into the correct place! :)

Assembly

Here's an image of a suggested assembly layout, but it's up to yourself how you want your own custom PiPhone to look:

I put a double layer layer of duct-tape on the back of the Raspberry Pi, the used velcro to put on the DC-DC, and a bolt for the FONA. For the LiPo battery I used a cable-tie to tie it to the Raspberry Pi.

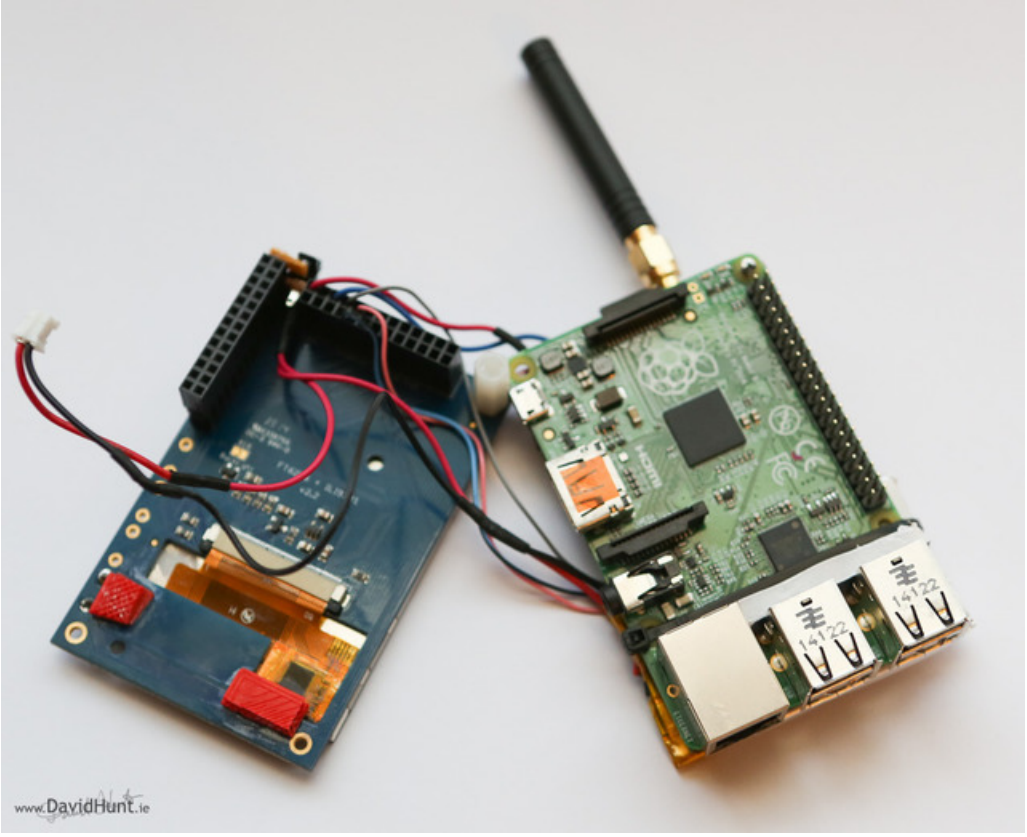
Here's a few more suggestions:

You could use a sheet of acrylic/perspex and mount the Raspberry Pi on one side, and the Battery/FONA/DC-DC converter on the other.

Depending on the size of your battery, you could mount the battery between the PiTFT and the Raspberry Pi. You might even be able to fit the FONA in there, but be very careful that there are no shorts!

And if you're an expert maker, you might even try to take some of the connectors off the Raspberry Pi to slim the whole thing down.

Why not top it all off with a 3D printed case! I'd love do see what you do here! :)



There's also something else you could try. Since the FONA has a microphone input and a speaker output, you could try connecting up an **Electret Mic (https://adafru.it/dDa)** and a **Mini Metal Speaker (https://adafru.it/dDb)** so that you don't need headphones!

Code Walkthrough

In this section we'll briefly touch on the contents of piphone.py. We won't go through all the code, just talk about each section and what it does. You can open up piphone.py here (https://adafru.it/ea3) and maybe have it open as you read through this page.

Imports

In this section of code, we import all the python libraries needed to make our script functional. They're all important but the main library is pygame. PyGame is an SDL (Simple DirectMedia Layer (https://adafru.it/eHP) wrapper that gives python access to all the wonderful functionality in SDL, giving us access to audio, keyboard, mouse, touch-screens and video hardware. It's because of these libraries, plus the thousands of lines of code in the other included libraries that allows us to implement this application in under 500 lines of python.

UI Classes

These are Phil Burgess' classes for the user interface. They define the Icon class and the Button class, which takes care of our getting our icon images onto the screen.

UI Callbacks

Once a particular button is pressed, a callback function is called. This then allows the application to do what's relevant to that particular button. For example when we press the green dial button, a function is called that tells the FONA to dial the number on the screen.

Global Stuff

Next we have a bunch of global variables which are needed throughout the application.

The global section also has the "buttons" array, which defines all our screens. If you want to customise the app for your own use, this is the first place you go to start laying out your own screens. Each button has an x and y co-ordinate for where it should go, and a width and height for where it's touch sensitive. For each button there as also a callback function specified, which gets called when it's pressed.

Assorted Utility Functions

All we have in here is a couple of functions to load an save the settings, if needed.

Initialization

Here's where we get into the core of the application. We first initialize everything, start pygame, init the screen, touch-screen, load the icons and buttons.

Main Loop

An finally we're into the main loop. This is a small loop that gets a touch input, and depending on which screen is being shown, and which button is pressed, calls it's callback. Then we take care of drawing the number on the screeen as it's

typed and we finally update the frame-buffer before going around the main loop again.

When the dial (or hangup) icons are pressed, a function is called to tell the FONA to dial or to hang up.

User Guide

Operation

So now you've everything built, and you want to make your first call. Well, it's easy. Power on the PiPhone, and wait until you're presented with the numeric keypad.

Insert your **"Hands-free" Headphones with Microphone (http://adafru.it/1966)** into the 3.5mm jack connector on the FONA.

Dial your number and press the green 'dial' icon, and pretty soon you should be able to talk to your friends using a cellphone you built yourself!

Debugging

If you have problems making calls, here are a few things to check.

First, make sure that the red LED on the FONA near the SIM800 module is blinking once every two seconds. This means the FONA is correctly associated to the cellular network. If it's blinking faster, then it's not associated, and will not be able to make calls.

The next thing to check is that the Raspberry Pi is talking correctly to the FONA.

This can be checked by logging into the Raspberry Pi and attempting to send a few commands to the FONA using a serial communications application. The easiest app to check this is minicom. So log into the Raspberry Pi and type:

```
sudo minicom -D /dev/ttyAMA0 -b 115200
```

You should then be presented with the following:

```
Welcome to minicom 2.6.1

OPTIONS: I18n
Compiled on Apr 28 2012, 19:24:31.
Port /dev/ttyAMA0

Press CTRL-A Z for help on special keys
```

All you need to do to confirm that the serial communications to the FONA are OK is to issue a few 'AT' commands. Type 'AT' followed by the return key, and you should see 'OK' coming back from the FONA.

```
AT
OK
AT
OK
```

If you don't see the OK (or zero) coming back, then check you'r wiring. Swapping the Tx and Rx wires on on side is a very common issue.