A cheap Bluetooth serial port for your Raspberry Pi

March 24 2013

Posted by Miguel Grinberg under Raspberry Pi.

While working on my Arduino based robot vehicle project the idea came to me that the Bluetooth module that I'm using as a remote control receiver for my robot would make a great addition to my Raspberry Pi.

In this article I describe how to enable the Raspberry Pi's serial port to talk to other devices over Bluetooth using this module.

Required hardware

The nice aspect of this project is that it requires very little hardware, most of the action happens in the software side. But of course we do need some hardware, which I list below:

- A Raspberry Pi.

  If you have been living under a rock or in another planet for the last year or two and never heard of the Raspberry Pi then you should know that this is a credit card sized Linux computer created in the UK that sells for just $35 USD. I've got mine through Element 14.

- A Bluetooth slave module

  This is a small and inexpensive Bluetooth to serial adapter that is mainly targeted to Arduino users. I've got mine from the Virtuabotix store at Amazon.com for $14.95. The Chinese electronics stores on Ebay sell it for less, but of course you have to wait longer to get it.

- Four female-to-female breadboard wires

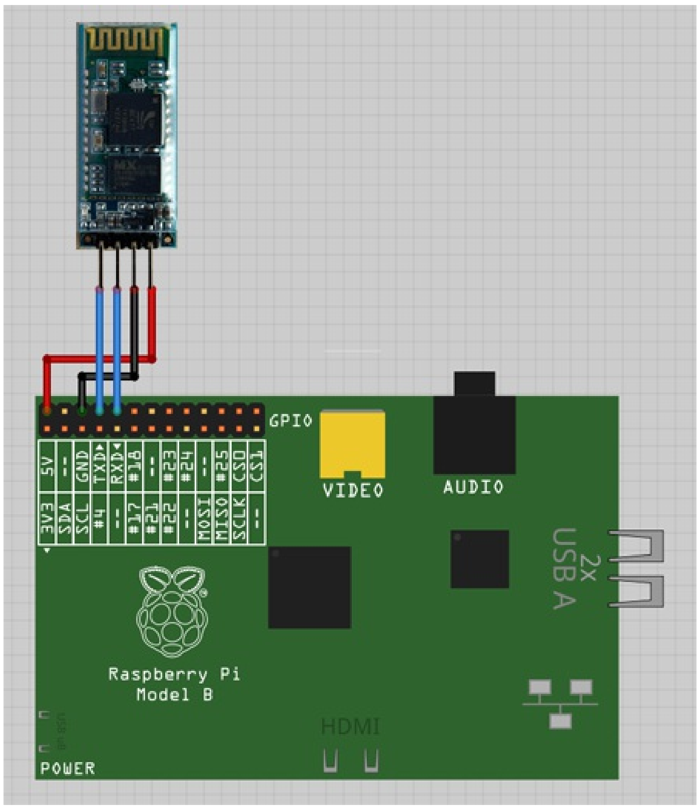
  If you buy the Bluetooth module from the Virtuabotix store at Amazon (link above) then you get the wires included with it. If you are buying from another place you may want to ask if wires are included or sold separately.

- A computer or device with a Bluetooth terminal software

To establish a communication with the Raspberry Pi over Bluetooth you need another device that can speak Bluetooth. If your computer has a Bluetooth adapter then you just need to find a terminal software that you can use to send and receive data, like HyperTerminal on Windows, or `screen` on OS X and Linux. A computer is not the only choice, though. For example, I will use my Android cell phone with the free BlueTerm app installed.

Wiring

The wiring is very simple and is better explained with a diagram:



If you prefer to see it in table format, here are the four connections that you need to make:

RPi GPIO pin	BT module pin
5V (Pin #2)	VCC
GND (Pin #6)	GND
TXD (Pin #8)	RXD
RXD (Pin #10)	TXD

Note that the `TXD` and `RXD` connections between the RPi and the Bluetooth module are crossed, this is what makes one end receive what the other end sends.

Raspberry Pi configuration

I'm going to assume you are running a recent release of Raspbian on your Raspberry Pi. If you are running another OS then you will need to find out how the changes below are done in your system.

By default the Raspberry Pi is configured to write boot time messages to the serial port, and also to start a login console on it. Unfortunately, the default baud rate that the RPi uses for its serial port is 115200 bps, while the

Bluetooth module comes preconfigured from factory to 9600 bps.

It is easier to configure the RPi to use 9600 bps so we will try that first. There are two config files that need to be updated.

File `/boot/cmdline.txt` contains the kernel options that are used to boot the system. In my Raspbian based system this file contains the following options:

```
dwc_otg.lpm_enable=0 console=ttyAMA0,115200 kgdboc=ttyAMA0,115200 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait
```

The interesting options are `console` and `kgdboc`, because these configure the serial port device `/dev/ttyAMA0` to 115200 bps. You need to change these two configurations to 9600 bps. After you make these changes the file should read:

```
dwc_otg.lpm_enable=0 console=ttyAMA0,9600 kgdboc=ttyAMA0,9600 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait
```

The second configuration file is `/etc/inittab`. Inside this file you have to locate the following line:

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

This tells the system to start a terminal on the serial port, and again it uses 115200 bps to configure the port. You have to change this line to use 9600 bps:

```
T0:23:respawn:/sbin/getty -L ttyAMA0 9600 vt100
```

Remeber that these are configuration files so they are not writable to the default `pi` user, to edit these files you have to use `sudo`. As far as text editors, Raspbian provides two, `vi` and `pico`. If you don't know either then you will probably be more confortable using `pico`. For example, to edit `cmdline.txt` with `pico` you would run the following command:

```
$ sudo pico /boot/cmdline.txt
```

The system will ask for your password and after that you will be able to make changes to this file. I recommend that you save original copies of these config files in case you make a mistake.

With those changes made the RPi is configured to talk to the Bluetooth module. If you now power up your Raspberry Pi you will notice that the LED in the Bluetooth module blinks rapidly. This is the sign that the Bluetooth module is ready and waiting to be paired with another device.

Connecting from a Bluetooth terminal

Now leave the RPi running with the Bluetooth module in its blinking state and go to the Bluetooth enabled computer or smartphone that you will connect to it. Your device should now find the Bluetooth module with the name `linvor` when you set it to discover devices.

If you are using an Android device with BlueTerm then start the app and from the app menu select "Connect device". Android does the baud selection automatically so you don't have to configure it. From a terminal software running in a computer it is likely that you will need to configure the speed, number of data bits per character, parity, and number of stop bits per character. The values you need to use are:

- Speed: 9600 bps
- Data bits: 8 bits
- Parity: None
- Stop bits: 1 bit

The Bluetooth module comes preconfigured with a PIN number. To complete the connection your computer or smartphone will ask you to enter this PIN. The factory default PIN is `1234`.

The LED in the Bluetooth module will now stop blinking and remain lit, indicating that it has made a connection.

And here comes the fun part. You need to reboot the Raspberry Pi so that the new serial port settings take effect. To reboot the RPi run the following command in a local or network shell:

```
$ sudo reboot
```

Now watch the Bluetooth terminal on your PC or smartphone while the Pi reboots. Boot messages should be appearing on your terminal, and as soon as the RPi is up you should get a login prompt there as well.

You can now login from your Bluetooth terminal and use the command line prompt as you normally would over a local or network shell.

Below is a short video that shows my Raspberry Pi connected to my Android smartphone:

Direct serial port access

While you can't deny there is a coolness factor in being able to run a Raspberry Pi from a smartphone, let's be honest, that isn't extremely useful. But having a wireless serial connection into the RPi can be useful for regular applications, for example for debugging or for remote control.

To be able to use the serial port from a user application running in the Raspberry Pi we first need to tell the system to not use it as a console. To do this we need to go back to the two configuration files we touched before. I once again recommend that you back up your config files before changing them, in case things don't work out well and you need to revert the changes.

In file `/boot/cmdline.txt` we are going to remove the two serial references by taking away the first `console` and `kgdboc` sections:

```
dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait
```

And in file `/etc/inittab` we will comment out the serial console task:

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

After making these changes reboot the system so that they take effect. With these changes the RPi will not use the serial port at all, so the port will be free to any application that wants to use it as `/dev/ttyAMA0`.

If you want to play with sending and receiving data through the serial port then you can install `minicom` into your RPi using the following command:

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

And then you can open a terminal on the serial port with this command:

```
$ minicom -b 9600 -o -D /dev/ttyAMA0
```

Anything you type inside the `minicom` terminal will be sent to the serial port, and anything the other side sends will be displayed.

Configuring the Bluetooth module

The Bluetooth module comes preconfigured from the factory with a set of defaults, which are:

- Baud rate: 9600
- Bluetooth ID: linvor
- PIN: 1234

But these values can be changed by sending special commands through the communication channel. Now that the Raspberry Pi is connected to the module we can try this.

These special commands that configure the Bluetooth module can be sent from a connected remote device, or they can be sent from the local system, in this case the Raspberry Pi. Since the Raspberry Pi is the star of the article I'm going to also use it to do the configuration.

Here is a quick summary of the most useful configuration commands:

Command	Description
AT	Test the communication with the Bluetooth module.
AT+VERSION	Report the module's version number
AT+NAME*name*	Change the Bluetooth ID to *name* (maximum 20 characters).
AT+PIN*nnnn*	Change the PIN to *nnnn*, which must be a four digit number.
AT+BAUD*n*	Change the baud rate. Use one of the following values for *n*: 2: 2400bps 3: 4800bps 4: 9600bps 5: 19200bps 6: 38400bps 7: 57600bps 8: 115200bps

One tricky aspect of sending these commands is that the Bluetooth module has a very short timeout, so all the characters in a command must be entered really quickly. The safest way to get the entire command in time is to type it in a text editor window and then use copy/paste to send it really fast.

If your Raspberry Pi session is local, then there is no clipboard support unless you work inside the X environment, so you will need to run `startx` and then open a Terminal window. If you are using a remote shell over the network then you can use the clipboard in your host operating system.

To send the commands we can use `minicom`. So let's fire it up one more time:

```
$ minicom -b 9600 -o -D /dev/ttyAMA0
```

Note that it isn't necessary to have a connection to do this, so you can do this while the Bluetooth module is in its blinking state.

You now need to use any method to get the string `AT` into the clipboard, and then hit paste inside the minicom window to send the command to the module.

When you send `AT` the module should respond with this in the `minicom` window:

```
OK
```

If you get this response then you know that everything is all right. If you don't get a response then for some reason the Raspberry Pi is unable to communicate with the Bluetooth module.

When I send `AT+VERSION` to the module I get the following response:

```
OKlinvorV1.5
```

To change the baud rate to the fastest rate of 115200 we need to issue the command `AT+BAUD8`, and the module will respond with:

```
OK115200
```

Because the baud rate was changed now the communication will break, and we will need to exit and restart minicom with the updated speed:

```
$ minicom -b 115200 -o -D /dev/ttyAMA0
```

To change the name of the module to MyBT we must issue the command `AT+NAMEMyBT`, and the module will respond with:

```
OKsetname
```

And to change the PIN to 4321 the command is `AT+PIN4321` and the response from the Bluetooth module will be:

```
OKsetPIN
```

Note that when the name and/or the PIN change the module requires a power cycle for the changes to take effect.

Also don't forget that if you change the baud rate and later want to reestablish the serial console you will need to change the two config files to reflect the new baud rate you have selected in the Bluetooth module.

Conclusion

I hope this article gives you some ideas to bring new life to the serial port in your Raspberry Pi. If you have any questions feel free to ask below in the comments!

Miguel

72 comments

#1 Fooky Barrall said 5 years ago

What kind of security it provided to the password set in plain text over the bluetooth link? Could a bluetooth sniffer decode the characters as they were sent?

#2 Miguel Grinberg said 5 years ago

@Fooky: I'm not an expert on this, but my understanding is that there is no encryption for the PIN. Bluetooth has a pretty short range though, the sniffer would need to be located pretty close to intercept the communication.

#3 maihoaomv said 5 years ago

would this also work with an ipad? if so any suggestions as to how i would go about setting up the ipad to talk to the bluetooth?

#4 Miguel Grinberg said 5 years ago

@maihoaomv: I haven't tried this myself, but I believe you should be able to connect using nBlueTerm, a free BlueTooth terminal app for iOS (https://itunes.apple.com/app/id516628472).

#5 Josh said 5 years ago

Could this be used to connect bluetooth keyboard/mouse to the pi?

#6 Miguel Grinberg said 5 years ago

@Josh: in theory yes, but you would need a device driver on the Pi that reads the serial port and translates the keyboard/mouse commands to the proper actions. I do not think this driver exists right now.

#7 David Cain said 5 years ago

Any way to get the Raspberry Pi to stop saying "Uncompressing Linux... done, booting the kernel." on the serial port after I've taken out the boot messages and the login prompt? I'm afraid it's going to confuse the device I plan to hook up.

#8 Miguel Grinberg said 5 years ago

@David: I didn't even notice this message still goes to the serial port. Search in all files in /etc and subfolders for the string "ttyAMA0", maybe there is something still referencing the serial port? Another thing to try could be to configure the console to go to /dev/null or anything other than /dev/ttyAMA0.

#9 Jhonmont said 5 years ago

Hi, great guide, just come up with this trying to figure if i would be able to use the serial console from the raspberry to an android phone via cable. This sounds better. Just a quick question, my situation is:
Want to use the raspberry pi as SD Cards backup device while traveling (because its cheap and tinny).
will be running mount/copy/move/delete commands to do that over the console.
Will this be enough to run those commands over my phone via BT?
Thanks again.

#10 Miguel Grinberg said 5 years ago

@Jhonmont: the SD card is not used as secondary storage, it is the main storage, that's where the root file system is. You will not be able to remove the boot SD card to mount another one.

#11 Jhonmont said 5 years ago

Miguel,
Sorry for the confusion, i did mean putting an extra SD card reader via an USB Port. this is the one i would use for copying/backup the SD cards. not touching the Pi main SD slot.
Its possible?
thanks

#12 Miguel Grinberg said 5 years ago

@Jhonmont: Ah, that makes more sense :)
Assuming the Pi can recognize your USB card reader I don't see any problems with your plan. Of course you will also need network access so that you can write the data from the SD cards somewhere.
If I had to do this myself I would not worry about bluetooth console access, I would just have a script that runs after boot that mounts the SD card and backs it up unattended.

#13 taher said 5 years ago

i want to do some controlling using bluetooth. can anyone here tell me how can i compare bluetooth data in linux terminal

#14 Ian said 5 years ago

Fantastic! Thank you so much for your hard work.

#15 Ian said 5 years ago

I put a link to your genius on my forum, hope that's ok.
http://btinterface.com/BTInterface/forum/viewtopic.php?f=4&t=101

#16 Jped said 5 years ago

Hey, Great tutorial. THis helped me alot! I am trying to connect an arduino to the raspberry pi through bluetooth. I have successfully got them to communicate with minicomm. However, I can not seem to figure out how to read the serial port into a python script. I have tried to use pyserial, but I do not get any communication.

#17 Miguel Grinberg said 5 years ago

@Jped: have you seen this tutorial? http://www.doctormonk.com/2012/04/raspberry-pi-and-arduino.html

#18 Jped said 5 years ago

@Miguel, I tried this.. did not work..

#19 Alex said 5 years ago

Hi,
Thanks for the tutorial. I have a question regarding the use of a master Bluetooth module in place of the slave module. would it be possible to use a master module (which is what I have at this time) and for it to communicate to an android device? (e.g. nexus 7)

Thanks,
Alex

#20 Miguel Grinberg said 5 years ago

@Alex: My understanding is that phones are typically masters, so you need a slave connected to the Arduino.

#21 Joe Guerra said 5 years ago

I burned out my raspberry pi (in 5 minutes). Using a 700ma supply, I had usb keyboard / mouse & Livnor bluetooth module hooked up to the GPIO pins.

I got my credit back, RMAing the unit.

Just afraid to attempt the same hookup.

#22 Ajith said 5 years ago

Hi, Great reading this. I would like to ask if I can use the same Bluetooth module to pair with another device which is not a PC. I want to use the serial port to send commands to my Vehicle OBD Adapater. How I can pair this device with that?

#23 Miguel Grinberg said 5 years ago

@Ajith: you can pair this Bluetooth module with any BT master, but not with another BT slave. There is a BT master module that you can get, which should enable you to connect the Pi to a device that has a BT slave.

#24 Craig Swanson said 5 years ago

How can I send a python executable file from my Windows 7 laptop to my Raspberry Pi using bluetooth? I have the bluetooth connections established right now. I want to be able to create Python code on my laptop, then transfer it via bluetooth to the Pi and then have the Pi execute the code.
Craig

#25 Miguel Grinberg said 5 years ago

@Craig: I showed in this article how I had a terminal in my cell phone controlling the Pi. That's what you need but instead of your cell phone you use your computer. If you can login to your Pi from your computer you can (for example) copy/paste your scripts and then run them.

«« « » »»

Leave a Comment

Name

URL

Email

Comment

Captcha

reCAPTCHA

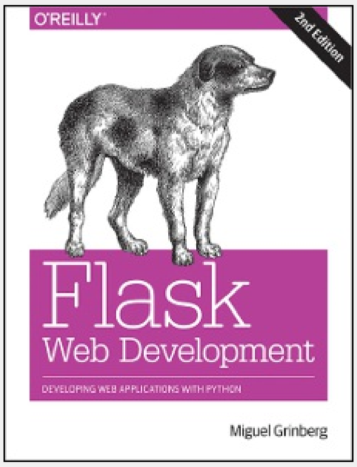
Please upgrade to a supported browser
to get a reCAPTCHA challenge.

Why is this happening to me?

Submit

Flask Web Development, 2nd Edition

If you want to learn modern web development techniques with Python and Flask, you may find the second edition of my O'Reilly book useful:



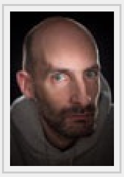
About Miguel

Welcome to my blog!

I'm a software engineer, photographer and filmmaker, currently living in Dublin, Ireland..

You can also find me on Facebook, Google+, LinkedIn, Github and Twitter.

Thank you for visiting!



Categories

- AWS (1)
- Arduino (7)
- Authentication (6)
- Blog (1)
- C++ (5)
- Cloud (6)
- Database (11)
- Docker (1)
- Filmmaking (6)
- Flask (78)

- Games (1)
- HTML5 (1)
- Heroku (1)
- JavaScript (8)
- Microservices (2)
- Movie Reviews (5)
- Netflix (5)
- Node.js (1)
- OpenStack (1)
- Personal (2)
- Photography (7)
- Product Reviews (2)
- Programming (95)
- Project Management (1)
- Python (90)
- REST (6)
- Rackspace (1)
- Raspberry Pi (7)
- Robotics (6)
- Security (10)
- Video (5)
- Webcast (3)
- Windows (1)