Python Smart Terminal

Contents [hide]

Goal

[edit]

This program is up and running. Makeing improvements from time to time for my projects. Code at GitHub, see [Code at GitHub]

SmartTerminal an open source, programmable rs232 smart terminal in particular for use with microcontrollers like the arduino. I think it is documented well enough so people can relatively easily extend and adapt the program. It easy to download and use even for those without a desire to dive into the code. I assume some knowledge of Python, and a Python Environment to run it in. Was Python 2.7 now upgraded to Python 3.6 and a bunch of other enhancements. See features below.

See the graphical user interface here (with screen shot): Smart Terminal GUI

This is an article started by Russ Hensel, see "http://www.opencircuits.com/index.php?title=Russ_hensel#About My Articles" About My Articles for a bit of info.

Who should use this program and How:

Who	What	How
Person with little programming experience, no interest in Python. Looking for download and install.	Probably should use another program.	Not well suited to use this, I do not plan to fix this.
Programming microcontrollers in text based language, willing to install or has Python installed.	Run the basic program. Customize in fairly minor ways.	Edit the parameter file to configure the terminal. Could use processing extensions written by others.
Modest Python experience	Modify all over the place, monitor equipment, save data to database	Program should be well documented with an API, sample extensions. Code changes localized (now) to one class plus the parameter file.

What/Why

[edit]

Almost all my Arduino and other microcontroller projects use serial communications for a least debugging. And in many cases I pair the Arduino with a Raspberry Pi for a very flexible system. A good serial monitor, or terminal program is useful in this sort of project. Because I wanted features not in most terminals and the ability to program it myself, I started one from scratch using Python.

The program is also designed to be a stand alone data logging, environmental monitoring program. In this case it will kick off and run without an operator. Data can be access over the web. I will largely leave a discussion of these features to another page, but probably about half the code is devoted to these features.

Currently the terminal waits for a carriage return (or line fead) until it displays (or make available for processing) the received string. This make sense for my application, it help the processing. This may not work best for you, let me know.

So what are the features?

- Free open source
- Runs across OSs Linux (inc **Raspberry Pi**), Mac or Windows
- Python
- "RS232" all standard baud rates etc.
- Easily adjusted serial communications parameters
- Multiple, preloaded with data entry fields.
- Clear (erase) receive area for removal of clutter.
- Copy all or part of receive area.
- Programmable
- I have extensions to work with specific arduino programs. These: Run a clock, Monitor a Green House, Monitor a root cellar, Monitor a Well.
- Database Interface (now MYSQL can connect across the network)
- Email interface.
- Easily Modified or Replaceable GUI
- Parameter file for wide range of modifications of program behavior.
- Easy to run multiple instances with different parameters.
- Uses standard Python logging class.
- Includes some advanced data logging features in addition to database data logging.
- For more on the features see SmartTerminal Features

Limits:

- Polls the comm port at 10 to 100 hz. So full lines (ending with carriage return <cr>) cannot come in too fast. This is theory I have not pushed the terminal.
- One instance of the program per com port at a time, but easy to run multiple instances.

What would you like to see in the program or documentation? Email me.

A much earlier version of this terminal was described in my instructable [Python Terminal for Cheap Pi Arduino Connection] It may have some information that is useful, but the program has grown a lot since then.

Installation

[edit]

This program is intended for those who at least occasionally develop in Python. I expect that they already run some things in Python and will just add this as an additional project. There is no install program you just download the files, place where you keep your projects and run. A bit more later in this section.

My Environment/Your Environment

[edit]

The program has a better chance of running if your environment is not too much different from mine. The most important is that it is Python 2.7 or compatible. **No now Pyton 3.6**

Before you begin to install you should know a bit about the environment that I have used to build, test and run the terminal. If your environment differs too much you may have trouble getting it to run.

I run Python mostly using the install that comes with Anaconda Spyder and often use the IDE it installs. This is not necessary, it is just an nice install that downloads a lot of stuff that technical folks find useful. I have use conda and pip to add to this install and do not know offhand all that is in it. You can look at the include statements to get some idea of what you might need to add. Or you can just keep running it and add the packages it complains about.

Download

[edit]

Code at GitHub, see [Code at GitHub] (it is Python and you can run directly from the source) Email me if you have issues (use this link User:Russ_hensel). You will get a zip file, unzip it and you should get:

```
.... whatever --|
                |-- SmartTerminal
```

Put them in your system making "....whatever" anything convenient for your Python (that is move the files to where you keep your Python source).

Note that there may be a certain amount of left over, dead code, in the directory I am cleaning out bit by bit, someday it may be nice and neat. For now if you want to tinker look at the design info below first.

Run

[edit]

Run it until it stops complaining about dependencies (in the console), after that (and perhaps even before) the GUI should come up. You are installed.

I have run the program on both Windows 10 and Rasperian on a RPi. It should work in most OS's. Let me know about issues.

Some parts of the program think that mySql is available. It should run fine without it (there will be a message or two in the console), untill you try to use the database then it is not so gracious. Straight ahead use as a terminal does not use the database.

Configure to Run

[edit]

Basic

[edit]

Basic configuration of comm parameters like port, baud rate .... is all done in a file called parameters.py. It seemed easier to simply use a Python text file instead of some other format like an ini file. Pretty much all the file does is set instance variables in itself. It is used by the program controller (smart_terminal.SmartTerminal) to create an instance of Parameters and then the values can be used. Save the original (parameters.py, maybe I will include a backup maybe not) in case you mess it up too much.

I have made yet another pass to clean up and comment the code in parameters. Let me know if you have issues. You should understand some values are being phased out but may still have some implementation and some may be coming in and have little or no implementation. The comments should let you identify these situations.

Parameters starts out with some "meta" parameters. These are defined early in the creation of the objects and may effect other values. In any case you can always define a value twice, the last one always wins. The most important meta parameter is mode, you should not change it from self.mode = "Terminal" unless you understand the implications or do not mind going on a ride.

All the com port values are defined in pretty much one place, find it (say search on "baud") an change it to what you need.

Command Line Arguments

[edit]

If you run with the command line parameters=paramaters_b then after the regular parameters file runs, then the system looks for parameters_b.py and uses that to override values that you might want to tweak (or completely redo. There are two examples in the directory follow the pattern in them and you should be fine).

This can be especially useful if you want to run two copies connected to different ports and possibly running in different modes. In this case it is also nice to change (its in parameters) the **icon** and **color** for each instance of the program. You can write or use the little bat file to start them (although this leaves a dos console hanging around) Command line arguments can also be placed in shortcuts. In either case they may take some tweaking to run in/from your file locations.

The above mostly applies to Windows, but the program run fine with Linux (including the Raspberry Pi) and I suppose the Mac. Of course the .bat file and shortcuts will not work, but similar facilities exist in the other OSs.

Running It

[edit]

When you run it it should open a windows a lot like the picture Smart Terminal GUI. Errors may show up in your Python console or the log file (look in parameters.py for the name of the log file, typically self.pylogging_fn = "smart_terminal.py_log"). The most likely errors will point to missing Python modules like pyserial. You should install with pip (or conda if using Spyder). Let me know how it goes.

Normally the terminal does not open the com port until you press the <Open> button. The parameters are displayed in the GUI if you do not like them (for example when you press the <open> button the port open status changes to "open failed", not the desired "open") you can shutdown, edit parameters.py and restart. There is a simpler way.

- First configure parameters.py to know the name of a text editor on your system. For mine this is one of:

```
self.ex_editor   =  r"leafpad"    # linux and pi
self.ex_editor   =  r"D:\apps\Notepad++\notepad++.exe" # for windows.
```

It is set up to auto switch between the two os to make copying the whole program back and forth between the windows and linux a bit easier.

- Now when you run it the button <Edit Parms> should let you edit the parameters.py file. Edit it and save.
- Hit the <Restart> button. In a flash the program should restart with the new parameters.

..... more here soon .....

Notes on the Code

[edit]

In some ways I am proud of the code, it was sometimes a slog, and I had to learn a lot of Python to get it going. On the other hand it kind of sucks, it has lots of dead code, ass backwards was of doing things, poor naming..... There is also a good bit of code on features that I am adding, but that is unfinished. Until I loose interest in it it will probably improve. However, adding features is more part of the life of a programmer than polishing old features; making something better that seems good enough is not always the priority it should be. If you do not like it, mostly keep it to yourself unless it is accompanied by an offer to improve it. I do not need ideas, I need time. That said if you think you have a helpful comment contact me, my page will tell you how: User:Russ_hensel

Design

[edit]

see: Python Smart Terminal Technical

Customizing/Extending

[edit]

Simple customization may be done simply by changing the parameter file, see: Smart Terminal Parameter Examples If you want to add code that can be done by messing with any of the source code (not recommended) or by programming an extension, see: Python Smart Terminal Technical and Writing You Own Extensions to SmartTerminal

Additional Info

[edit]

- Click on the category smart terminal below (and perhaps the others as well)
- For a graphing (plotting) application that is a companion to this see Python Smart Terminal Graph
- Writing You Own Extensions to SmartTerminal
- Debugging the Smart Terminal
- Smart Terminal as Smart Data Logger obsolete, gone, may come back.
- Smart Terminal GUI
- GreenHouse Monitor Program
- Smart Terminal Parameter Examples
- [https://github.com/russ-hensel/python_smart_terminal python_smart_terminal at git hub]
- Python-Terminal-for-Cheap-Pi-Arduino-Connection/ Info on much earlier version may or may not be useful.

Categories: Python Projects | Python | Arduino/RaspberryPi | SmartTerminal | Serial Communications