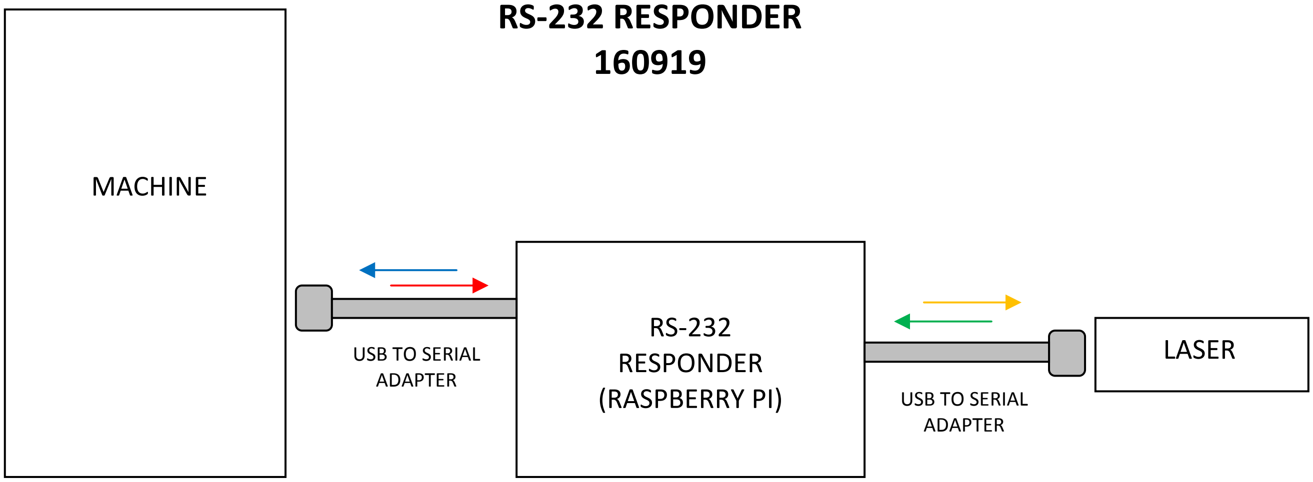
RS-232 RESPONDER
160919



SCOPE

The project goal is to write a software application that will run on a Raspberry Pi that functions as an RS-232 responder between an industrial machine and the DPSS Laser (see above diagram). In general, the machine will send queries and commands that need to be interpreted by the software application. Some queries/commands will require the application to translate and forward the query/command to another port that is connected to the DPSS Laser. The laser will then respond to the translated command. This response will need to be translated and potentially forwarded to the machine. Not all queries/commands will need to be translated and forwarded. Some will simply require known responses.

CRITICAL REQUIREMENTS

- Application need to run on Raspberry Pi—Development environment open.
- Qty 2 USB-to-Serial hardware connections need to be made to Raspberry Pi. Hardware for this is IOMEGA Model GUC232A USB to Serial RS-232 adapter. Will be provided.
- Application should boot automatically after Raspbian OS loads, and should run indefinitely.
- Must be able to boot and run without a keyboard, mouse, or monitor connected.

MATERIALS PROVIDED

- Raspberry Pi kit
- Case for Raspberry Pi
- Power supply for Raspberry Pi
- Qty 2 IOGEAR Model GUC232A USB to Serial adapters
- Logic and translation table for queries and commands
- Serial connection protocol parameters

DELIVERABLES AND TIMELINE

- Functioning application loaded on Raspberry Pi—Required within 4 days of receipt of all above materials.
- Rework of application after functional test
- Source code for application—Required within 3 days of successful functional test

- Return of all development tools upon completion of project or October 15, 2016, whichever comes first.

RS-232 Responder Project

DPSS LASERS INC.

Contents:

1. Source Code:

```
##This is a serial interpreter script - used to establish communication
##between the DPSS Laser and the third party Machine

##Ver 1.57 (Final)

import serial, re, time  ##Import serial, string and time packages

ser = serial.Serial(port = '/dev/ttyUSB0', baudrate = 9600, parity =
serial.PARITY_NONE, stopbits = serial.STOPBITS_ONE, bytesize =
serial.EIGHTBITS, timeout = None) # Define serial port for Machine
ser2 = serial.Serial(port = '/dev/ttyUSB1', baudrate = 9600, parity =
serial.PARITY_NONE, stopbits = serial.STOPBITS_ONE, bytesize =
serial.EIGHTBITS, timeout = None) # Define serial port for Laser


print("connected to: " + ser.portstr)
print("connected to: " + ser2.portstr)

count =1

while True:
    line = ser.read()                   ##Reads command from Machine
    time.sleep(0.02)                ##Waits 0.02s - to ensure command is
received
    bytesToRead = ser.inWaiting()   ##Number of characters in queue to be
read
    line += ser.read(bytesToRead)
    print (str(count) + str(': ') + line)   ##Prints to terminal what the
script is receiving
    count = count+1


##Script 'interprets' the commands and forwards to laser/machine as per
specifications

    if line == "?C1;" or line == "?c1;":
        ser2.write("?C1\r\n")
        line2 = ser2.readline() ##line2 = ser2.read(7)
##      if line2 != None and re.match('\d.\d\d\D\r\n', line2):
        line3 = line2.replace("\r\n",";")
```

```
            ser.write(line3)

        elif line == "?C1S;" or line == "?c1s;":
            ser2.write("?CS1\r\n")
            line2 = ser2.readline() ##line2 = ser2.read(7)
##          if line2 != None and re.match('\d.\d\d\D\r\n', line2):
            line3 = line2.replace("\r\n",";")
            ser.write(line3)

        elif line == "?C1S;" or line == "?c1s;":
            ser2.write("?CS1\r\n")
            line2 = ser2.readline() ##line2 = ser2.read(7)
##          if line2 != None and re.match('\d.\d\d\D\r\n', line2):
            line3 = line2.replace("\r\n",";")
            ser.write(line3)

        elif line == "?G;" or line == "?g;":
            ser2.write("?G\r\n")
            line2 = ser2.readline() ##line2 = ser2.read(3)
            line3 = line2.replace("\r\n",";")
            ser.write(line3)

        elif line == "?Q;" or line == "?q;":
            ser2.write("?Q\r\n")
            line2 = ser2.readline()
            list1 = list(line2)
            list1.remove('\r')
            list1.remove('\n')
            list1[2] = '000'
            line2 = "".join(list1)
            line2 += ';'
            ser.write(line2)

        elif line == "?STB;" or line == "?stb;":
            ser.write("10.00A;")
##          ser.write('\r\n')

        elif line == "?SUP;" or line == "?sup;":
            ser.write("1;")
##          ser.write('\r\n')

        elif line == "?T1;" or line == "?t1;":
            ser2.write("?T4\r\n")
            line2 = ser2.readline()
            fline = line2.replace('\xb0', "")
            fline2 = fline.replace('\r\n',";")
            ser.write(fline2)

        elif line == "?T3;" or line == "?t3;":
            ser.write("2000;")
##          ser.write('\r\n')

        elif line == "?X61;" or line == "?x61;":
            ser2.write("?HOURS\r\n")
            line2 = ser2.readline()
            fline = line2.replace("\r\n",";")
            fline2 = fline.replace(" ","")
```

```
            fline3 = fline2.replace("o","")
            fline4  = fline3.replace("u","")
##          print fline4
            ser.write(fline4)

        elif line == "D0;" or line == "d0;":
            ser2.write("LASER=0\r\n")
            line2 = ser2.readline() ##line2 = ser2.read(4)
            if line2 == "OK\r\n":
                ser.write(";")

        elif line == "D1;" or line == "d1;":
            ser2.write("LASER=1\r\n")
            line2 = ser2.readline()  ##line2 = ser2.read(4)
            if line2 == "OK\r\n":
                ser.write(";")

        elif line != None and re.match('\D\d:\d\d\d\d;', line):
            ser.write(';')

        else:
            pass

    ser.close()         ##Close port
```

2. Raspberry Pi 3 OS:
   (from *https://www.raspberrypi.org/blog/raspbian-jessie-is-here/*)

Raspbian is an unofficial port of Debian Wheezy armhf with compilation settings adjusted to produce optimized "hard float" code that will run on the Raspberry Pi. This provides significantly faster performance for applications that make heavy use of floating point arithmetic operations. All other applications will also gain some performance through the use of advanced instructions of the ARMv6 CPU in Raspberry Pi.

The Raspbian operating system is based on Debian Linux, and the different versions of Debian are named after characters from the "Toy Story" films. Recent versions of Raspbian have been based on Debian Wheezy (the penguin who's lost his squeaker in "Toy Story 2"), but Raspbian has now been updated to the new stable version of Debian, which is called Jessie.
The Raspberry Pi 3 microprocessor is running the Raspbian Jesse OS.

Look and feel
The first thing anyone starting the new Jessie image from scratch will notice is that the default behaviour is to boot straight to the desktop GUI, not to the Linux command line. This was a decision taken because this is the expected behaviour for all modern computers; the default interface for a personal computer in 2015 is a desktop GUI, not just text on a screen. It is still

possible to set the Pi to boot to the command line for people who prefer that – just toggle the relevant setting in the Raspberry Pi Configuration application described below.
When the desktop launches, you might notice some slight tweaks to the appearance of things like menus, check boxes and radio buttons. This is because the appearance of Raspbian is now based on version 3 of GTK+, the user interface toolkit used for the LXDE desktop environment. The older version 2 of GTK+ is slowly being replaced with version 3 in many applications, so this change was inevitable at some point – the new appearance isn't a huge change, but does look slightly more modern. Many of the applications in Raspbian are still using GTK+ version 2, but the PiX theme for GTK+2 has been changed to bring it into line with that for GTK+3.
You'll notice on the menu bar that there is now an eject icon at the top right – this is a new plug-in that allows USB drives and the like to be safely ejected without the risk of losing data. It's slightly risky to just pull out a USB drive, particularly if you have just copied a file to it, as the system manages the write to a drive in the background, and the write takes a finite amount of time. If you pull the drive out before the write has finished, you'll corrupt the file and lose data – clicking the eject icon and then selecting the drive to remove waits for any pending writes to complete and then prompts that it is safe to remove the drive.

Office applications

One of our main aims with regard to Raspberry Pi is not just to make it a great cheap computer for education, but also to make it a great cheap computer in its own right. To this end, we want to make it possible to use a Pi to do the sort of things you'd do on a Mac or a PC, so we're including some more applications that we think people will find useful. In this release, we have added the LibreOffice suite and Claws Mail.

LibreOffice is a full-featured office suite which is compatible with Microsoft Office files – it includes a word processor, spreadsheet, presentation graphics, vector drawing and database programs, all of which should feel familiar to anyone used to using Office. It has had some optimisation for Pi, and runs pretty well, particularly on Pi 2.
Claws Mail is an email client for those of us who are old-fashioned enough to prefer not to do email through a browser – it supports all common email protocols, and offers all the functionality of a standalone mail client like Windows Mail or Thunderbird.

Java tools

There are also two new applications in the Programming category – these are two new environments for writing Java applications, called BlueJ and Greenfoot (from the University of Kent and Oracle). If you're interested in learning Java, or already a Java programmer, have a look at them. There are some sample projects for both in the /home/pi/Documents directory.

Settings and configuration

There are a couple of new settings dialogs in this release, found under the Preferences entry in the main menu. The first is Raspberry Pi Configuration – this is a GUI version of the old raspi-config command-line application, which provides all the same functionality in a nicer interface. (The old raspi-config is still on the system and can be accessed from the command line by typing "sudo raspi-config", but it shouldn't be necessary to do so any more.)

The new Raspberry Pi Configuration allows you to enable and disable interfaces, tweak performance and configure internationalisation options, such as timezone and keyboard. It also allows some more control over boot options than was available in the past, with the option to automatically log in as the “pi” user available when booting to both CLI and desktop.

There is a new keyboard setting dialog, accessed from the Localisation tab, but hopefully many people won’t need this – the system will detect some common keyboards sold for use with Pi and set up the GUI keyboard driver correctly. If that doesn’t happen, it’s now easy to choose the right country and keyboard type in this dialog.

The other new setting dialog is the Main Menu Editor. This is a Pi version of a menu editor called Alacarte, written in Python – this should make it easier for people to add or remove items to the main menu. (And, by popular demand, the Other menu is back on the system – but it will now only appear if applications are installed that don’t appear in any other categories…)

Updated applications

There are updates to several of the applications that used to come with Raspbian. There are new versions of Scratch, Sonic Pi, and the Epiphany web browser; none of these have changed fundamentally in operation, but they all include bug fixes and performance improvements.
Support has been added for some of the new Pi peripherals that have been released recently, including the Sense HAT as used in Astro Pi – this is now supported under Scratch and Python.
Python users used to have to launch Python with sudo in order to allow access to the GPIO lines – Python can now access GPIOs as a standard user. Also for Python, the Pygame Zero game environment is installed by default – have a look atpygame-zero.readthedocs.org for information on what it can do.
One final small thing – if you want to get a screenshot of your Pi, just press the Print Screen button on your keyboard. A PNG file will be put in your home directory, thanks to the (slightly strangely named) scrot utility.

Where can I get it?

This is a major version upgrade – due to the large number of changes to the underlying operating system, we strongly recommend using Jessie from a clean image, so you’ll need to download a new Jessie image from the downloads pageon our site. (Some people have had problems extracting the zip files, as the large size of the image file causes zip to use a different format internally. They can be successfully unzipped with 7-Zip on Windows and The Unarchiver on Mac – both are free applications.)
Starting with a clean image is the recommended way to move to Jessie. If you really need to update a Wheezy image, we have tried an unsupported upgrade path which is documented on the forums here. This has been shown to work on a vanilla Wheezy image, but we can’t predict what effect it may have on any packages or data that you have installed, so this is very much at your own risk. Feel free to add your experiences and improvements to the upgrade process to the forum so others can benefit.

3. Location of the Software on the Raspberry Pi 3:

The serial165.py file, along with the launcher script which enables the script to execute at boot can be found in the folder ‘dpsslasers’ which is in the home directory of the pi.

To access the file via the Linux Terminal:

- Open the terminal console.
- Enter ‘ cd / ‘. This will take you to the home directory.
- Enter ‘ cd dpsslasers/ ‘. This will take you to the Dpsslasers directory.
- Enter ‘ ls ‘. This will list the contents of the dpsslasers directory.
- The contents of the directory will be listed, which will include the files: serial165.py and launcher.sh
- The script may be executed by giving the command ‘ sudo python serial165.py ‘ (Please note that since the script is set to execute automatically at boot, relaunching the script via this command will execute it twice and may cause undesirable behavior.)
- The script may be edited, using the Gedit editor by giving the command ‘ sudo gedit serial165.py &’. (Please note that making any changes to the serial165.py directly will cause the execution of the updated file when the system boots the next time.)
- To check if the system is currently executing the script, enter the following command: ‘ ps aux | grep python ‘ and press enter. This should return the filename ‘serial 157.py’ which means the script is currently running.
- To stop execution of the script, enter the following command: ‘ sudo pkill -f serial157.py’. This will immediately stop execution of the script. To check if the script has actually stopped execution you may enter the ‘ ps aux | grep python ‘ again and it will not show serial157.py as running.

4. Accessing and modifying the source code:

The script has been written using Python 2.7. The libraries used are: Pyserial, Time, Re. All libraries being used have been already added to Python in the Raspberry Pi 3.

Code has been commented to give user an idea of what the commands are doing. For more information, please refer the following resources:

https://pythonhosted.org/pyserial/
https://docs.python.org/2/tutorial/index.html
https://docs.python.org/2/tutorial/controlflow.html

To access the file via the Linux Terminal:

- Open the terminal console.
- Enter ‘ cd / ‘. This will take you to the home directory.
- Enter ‘ cd dpsslasers/ ‘. This will take you to the Dpsslasers directory.

- Enter ‘ ls ‘. This will list the contents of the dpsslasers directory.
- The contents of the directory will be listed, which will include the files: serial157.py and launcher.sh
- The script may be executed by giving the command ‘ python serial157.py ‘ (Please note that since the script is set to execute automatically at boot, relaunching the script via this command will execute it twice and may cause undesirable behavior.)
- The script may be edited, using the Gedit editor by giving the command ‘ gedit serial157.py &’. (Please note that making any changes to the serial157.py file directly will cause the execution of the updated file when the system boots the next time.)
- To check if the system is currently executing the script, enter the following command: ‘ ps aux | grep python ‘ and press enter. This should return the filename ‘serial 157.py’ which means the script is currently running.
- To stop execution of the script, enter the following command: ‘ sudo pkill -f serial157.py’. This will immediately stop execution of the script. To check if the script has actually stopped execution you may enter the ‘ ps aux | grep python ‘ again and it will not show serial157.py as running.

5. Auto boot instructions:

The script is currently set to auto execute using the built in crontab utility in the Linux OS. Following is the sequence of steps that was followed to achieve this:

- Navigate to dpsslasers directory by giving the command ‘ cd dpsslasers/ ‘
- Create a .sh (Shell script) file and name it ‘launcher.sh’ by giving the command ‘ gedit launcher.sh &’
- Type the following in the launcher.sh editor window:

  ```
  cd /
  cd home/pi/dpsslasers
  sudo python serial157.py
  cd /
  ```

  and save the file.

- Now, this is made into an executable file by giving the ‘chmod 755 launcher.sh’ command in the terminal. (Please make sure you are still in the dpsslasers directory when giving this command.)
- You may test if our shell script is working by giving the command: ‘sudo sh launcher.sh’ in the terminal window. This should run the python script.

- To log any errors during the crontab execution process, a directory for logs is made by giving the following commands:
  ‘ cd /’
  ‘mkdir logs’
- Now, we add our launcher.sh file to the crontab utility. To do this, enter: ‘sudo crontab -e’. This will open the crontab window. Navigate to the end of the window and enter the line: @reboot sh /home/pi/bbt/launcher.sh >/home/pi/logs/cronlog 2>&
- Reboot the system to ensure it works by giving the command: ‘ sudo reboot’
- If it does not work, you can always navigate to the logs directory created earlier and see any errors that you might have by giving the following commands:
  cd logs
  cat cronlog