Scientific Data Acquisition on a Raspberry Pi

Benjamin Seeley[1,2], James Mudd[1], Mark Basham[1]

[1] Diamond Light Source, Oxfordshire, UK

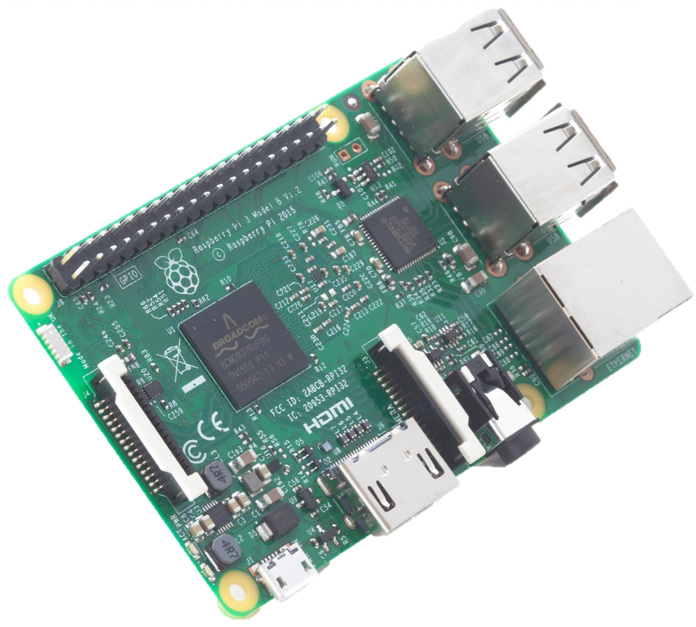
[2] School of Science and Technology, Nottingham Trent University, Nottinghamshire, UK

Introduction

The Generic Data Acquisition (GDA) package is used across all the Diamond Beamlines. It's developed in house and is designed to be easily customised to the users needs with interchangeable configurations for individual beamlines sitting on top of a common core infrastructure. Whilst the separation between the core and the configurations can blur in places, the main software architecture means that the core infrastructure is a robust, customisable platform which should run on any device capable of running the Java Virtual Machine (JVM).

A Raspberry Pi is a low cost, credit card sizes computer produced by the Raspberry Pi Foundation. Having just passed the 10 million unit sales mark, it's the best selling British computer ever made. This wide spread popularity, large supporting community, and low barrier to entry make it an ideal test bed for running GDA in a low power, low performance environment. It is also exceptionally fully featured machine considering the cost, with a 1.3GHz Quad Core ARM processor, 1GB of RAM, and built in Wi-Fi. Much like the servers running GDA, the Raspberry Pi runs a variant of Linux which in turn is capable of running java.

By creating a version of GDA which runs on the Raspberry Pi, we create not only an excellent tool for outreach, but also an easy to set up, low cost platform for other facilities to use to evaluate GDA as a tool for their own use.



Installing the system

As one of the primary use cases of the system is as an easy to set up evaluation tool, it's especially important that the system is easy to install. To that end, the initial setup process has been automated in so far as can be reasonably achieved. There are however some manual steps still required.

The first of these steps is to create an SD card with an install of the latest version of the Raspbian Lite Linux operating system which is available from the Raspberry Pi website. Having connected to the system, the user needs to enter the in-built configuration tool to enable the camera and i2c interfaces. They can perform this using the following commands:

```
sudo raspi-config
(Go to option 6 to enable the camera)
(Option 9 then Option A6 to enable the i2c)
Choose finish to restart the Pi
```

Once these configuration changes have been made, it becomes a case of running one final command to start the automated installer:

```
curl -s -L opengda.org/getRpiServer | bash
```

This will install everything you need to get started with GDA on a Raspberry Pi.

An example of the images captured as part of a scan and the subsequent 3D reconstruction.

Running the system

As a core part of developing the Raspberry Pi implementation of GDA was the creation of a stand-alone hardware server which can run in parallel to GDA on the Pi to allow access to the very versatile GPIO interface.

As the hardware server is stand alone, it can also be integrated into existing GDA deployments on beamlines with the same scannable interface as any other device. This hardware server is further extended by the creation of an interfaces for the Raspberry Pi Camera module and for Arduino's connected via i2c.

In order to run GDA, the Hardware Server and a Client you can simply connect an Ethernet cable to the Pi and a Laptop. On the laptop open 3 ssh connections to pi@raspberrypi.local and follow these instructions to get it started:

1. Using the first connection, enter "sudo su" to enter root mode and from there run ./starthardware – this will start the hardware server and will output a short message before sitting waiting for GDA to connect.
2. Using the second connection just enter ./startgda which will commence the start up of GDA – You'll know it's completed when the first connection starts outputting lines about creating systems
3. Once the two components are running, use the third connection to run "telnet localhost 9999" which will start a client connection ready to use. Test it's working with a pos command.

For more detailed documentation, sample data as well as the source code, scan the QR code to be taken to the project's Git repository

For more information please visit www.diamond.ac.uk or contact Ben Seeley at benjamin.seeley97@gmail.com, James Mudd at james.mudd@diamond.ac.uk, or Mark Basham at mark.basham@diamond.ac.uk