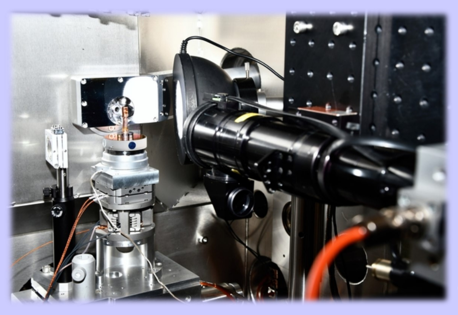
Enhancing the functionality and adding remote control capability to an I14 instrument using a Raspberry Pi

Alexander Liptak, Diamond Light Source, Harwell Science and Innovation Campus, Didcot, Oxon OX11 0DE

I. Introduction

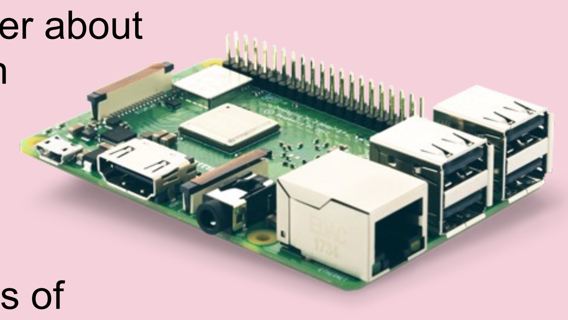
The I14 beamline uses a laser for experiments that require sample illumination. In order to operate the laser, somebody would need to physically be in the same room as the laser and manually turn it on or off as required. This simple design had many drawbacks: mainly, remote operation of the laser was not possible, and one would need to break the safety interlock at I14 in order to control it.



Overcoming this inconvenience was the first motivation for this project, an so the development of a smart networked laser controlled by a Raspberry Pi began. On top of this requirement, it soon became clear that a smart laser can do more than just switch on and off remotely. Consequently, the specification of this project was updated to include the ability to modulate the output of the laser using custom waveforms that could be generated on-demand, as well as other features like synchronizing a camera on the instrument with the laser.

II. Raspberry Pi: Server

The Raspberry Pi (RPi) is a single-board computer about the size of a credit card, chosen to act a server in this project due to its small size, lots of low-level peripherals, and performance comparable to a low-cost general-purpose PC.



Credit to Element 14
Link: http://au.element14.com/buy-raspberry-pi

The RPi runs a Python-based "server", or a series of Python scripts that allow the RPi to securely communicate over the Local Area Network (LAN) with other networked devices, as well as communicate directly with the attached laser and Arduino (discussed later). The server is composed of 5 distinct Python scripts, the combination of which allows the RPi to receive commands from any device on the LAN, carry out some action on the laser in a safe way, and return an appropriate response.

The first part is the "main server", a script in charge of listening for incoming connections, accepting a single connection at any one time, and formatting a response in a way that the "client" (or user) expects to receive it in. The second is the "parser", a script in charge of understanding what the received commands mean, executing appropriate actions, and rejecting commands that are not recognized or contain wrong parameters. The third part is the "laser controller", which is responsible for communicating with the laser over a serial interface, and carrying out tasks such as turning the laser on and off, tweaking its settings and checking its status for any errors. The fourth is the "Arduino controller" (described further in section III), which communicates with the Arduino over the serial and GPIO interfaces, in order to generate waveforms that will modulate the output of the laser. The final, and arguably most important script, is the error handler: it checks if the laser, Arduino or the server itself perform the requested actions as expected, and returns a useful message to the user instead of letting the server crash.

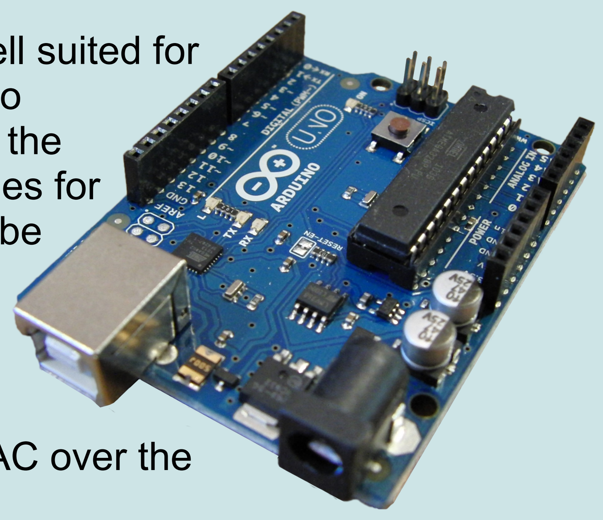
Raspberry Pi is a trademark of the Raspberry Pi Foundation

III. Arduino: Controller

The Arduino is a single-board microcontroller: a device that can execute some code, much like a computer, designed for use in embedded systems. Unlike the RPi, which runs a Linux-based operating system in addition to the Python server, the Arduino will exclusively execute only the code that it is flashed with. This means that an Arduino will, in most cases, execute code many times faster than the RPi.

Arduino is a trademark of Arduino AG

"Real-time" execution of code makes the Arduino well suited for generating high-resolution signals that will be used to modulate the laser. The Arduino communicates with the RPi over a serial interface in order to obtain the values for frequency, amplitude, and duty cycle of the wave to be generated. It also communicates with the RPi using its GPIO interface, where a combination of digital pin values corresponds to a shape of the wave that the Arduino will generate. Finally, the Arduino sends the generated wave data as a 12-bit number to a DAC over the I^2C interface (discussed further in section IV).

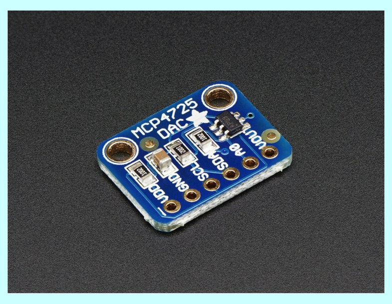


"A Arduino Uno board" by JotaCartas, obtained from Wikimedia Commons, is licensed under CC BY 2.0

Pulse Width Modulation (PWM) is a technique for mimicking analog signal by a rapidly alternating digital signal (a square wave). Arduino's support for hardware PWM is the reason why all LED and buzzer signaling in this projects is carried out on the Arduino.

IV. DAC: Modulation

A digital-to-analog converter (DAC) is a type of device that converts a digital signal into an analog signal. The MCP4725 is one of these devices, converting a 12-bit binary number into a variable voltage, which is used as a signal to modulate the laser. The DAC is updated hundreds of times per second with values that correspond to different points on a wave. The Arduino carries out a "calibration phase" on the DAC with every waveform to ensure its frequency is accurate. The calibrated wave will usually compensate for the time it takes to update the DAC by skipping a number of points on the wave.



Credit to Adafruit
Link: https://www.adafruit.com/product/935

As a safety feature, the DAC power supply is connected directly to the safety interlock of the beamline. Opening the door instantly cuts off the power to the DAC, modulating the laser amplitude to zero.

V. Conclusion

This project originated from a noticeable room for improvement of one of the components on the I14 beamline, but as it was nearing its completion, it became clear that it also opened up more room for improvement on other components across other beamlines.

The foundation for a smart networked controller that would, with little need for additional programming, enable the remote control, monitoring and automation of a component it is attached to, is something I believe many other would find useful. The outcome of this project will hopefully inspire, and possibly even guide other beamline scientists when networking their own components.

Generated with www.qrcode-monkey.com

The server in this project uses a custom network protocol that conforms to Diamond's serial protocol guidelines. The error handler allows for easy implementation of custom errors and warnings, and the parser can be edited relatively easily to define custom commands and parameters, and associate them with some corresponding functionality.

Acknowledgements

Thank you to Fernando Cacho-Nerin and Edmund Warwick for supervising me and guiding me through this project. A thank you also goes Ben Garvey for all his help with the various electronics issues and requests I had. A further thanks goes to the Diamond IT team (Richard Cooper in particular) for getting the RPi on the network. Finally, a special thanks to the Diamond Students team, without which this project would not be possible.

For more information please visit www.diamond.ac.uk
Alternatively, contact Fernando Cacho-Nerin at fernando.cacho-nerin@diamond.ac.uk or Edmund Warrick at edmund.warrick@diamond.ac.uk