

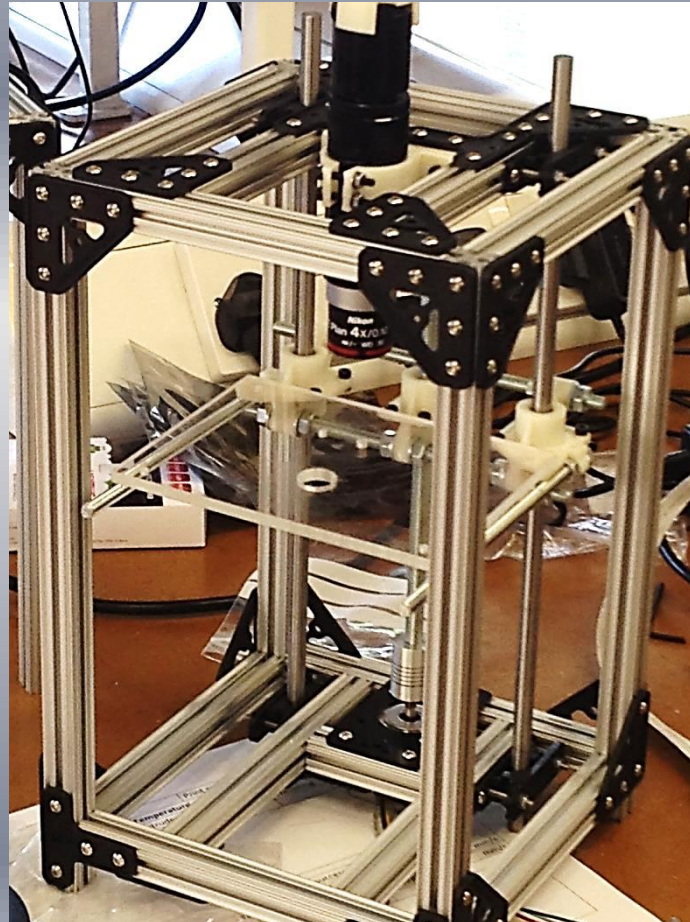
Raspberry Pi Controlled Microscope

Aims

- Open Source Microscope
- High quality images
- Modular Design
- Automation of simple tasks

Mechanical Design

Thomas Roddick

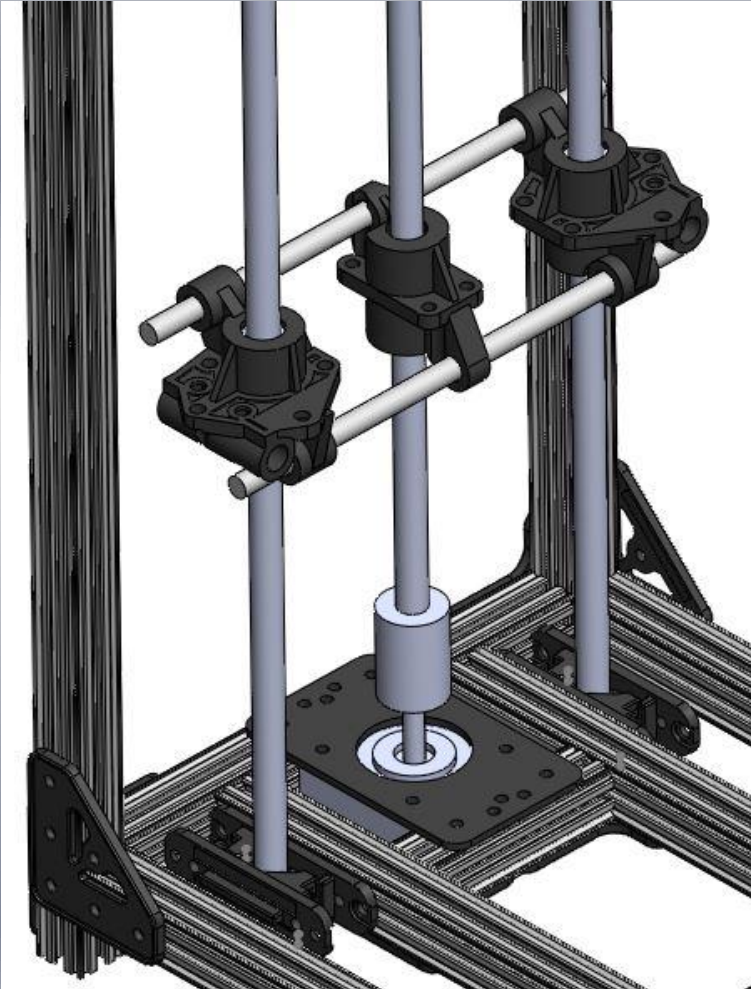


Structure

- Frame constructed from OpenBeam – an open source construction kit
- Aluminium slotted extrusion
- Brackets suitable for 3D printing – all designs available online



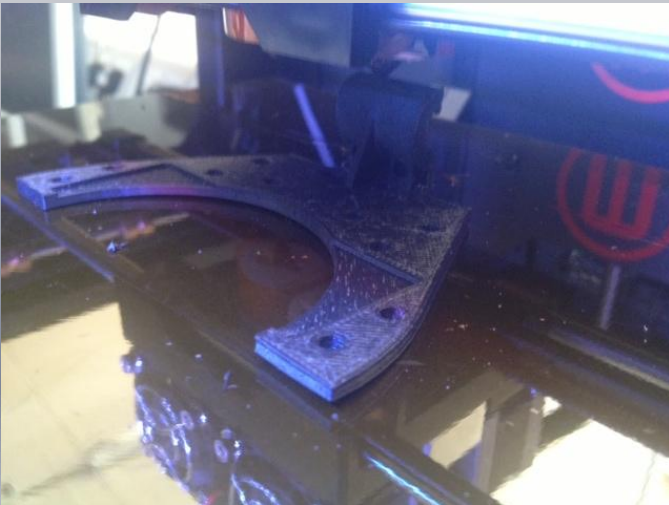
Actuation



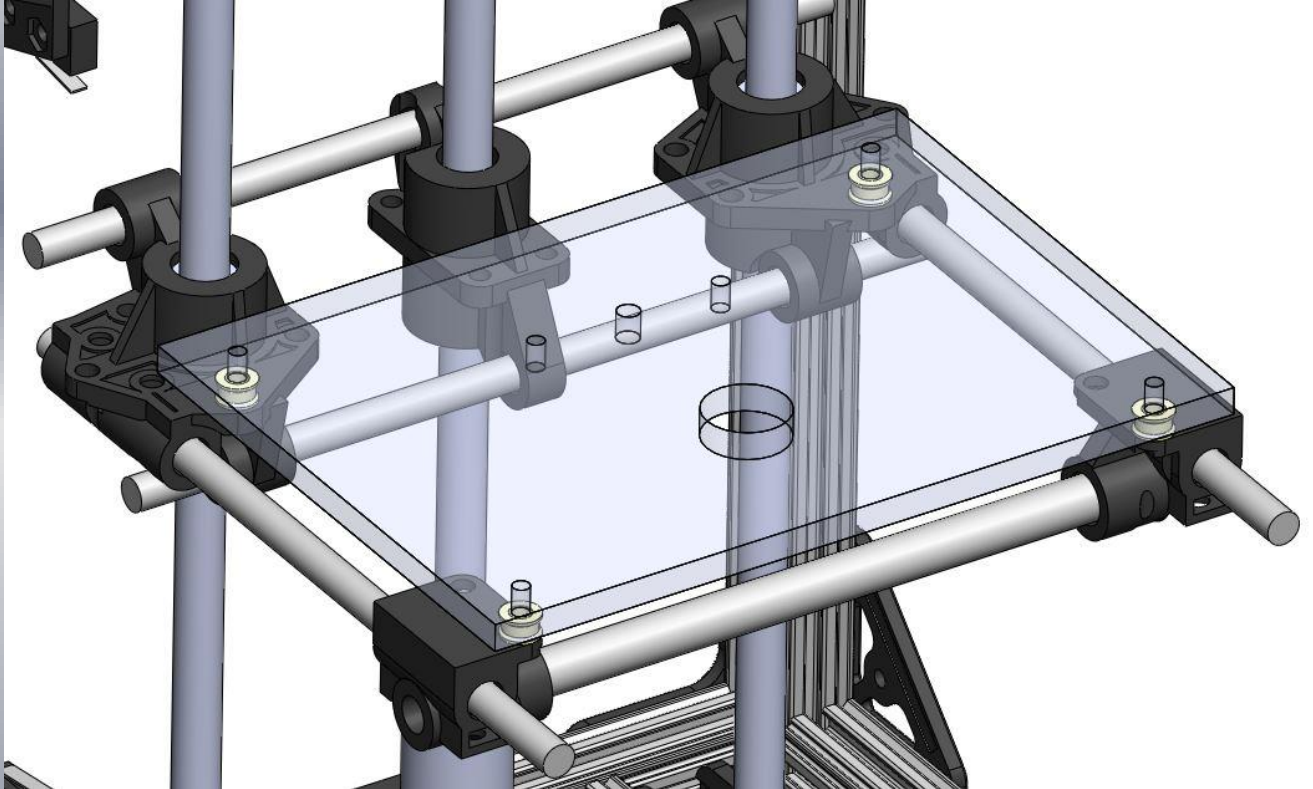
- Leadscrew driven actuation
- Inspired by RepRap 3D printers
- Bearings left and right of the screw prevent rotation
- Microswitches limit range

3D Printing

- Current version contains 15 3D printed parts,
- Printed in ABS using a MakerBot Replicator 2X printer
- All designs available on Github

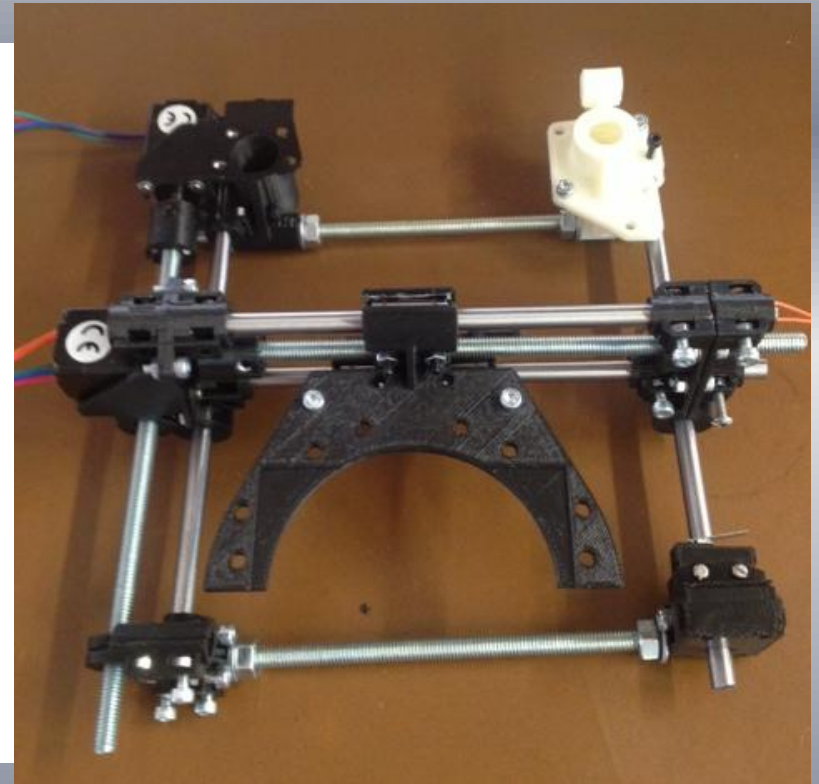
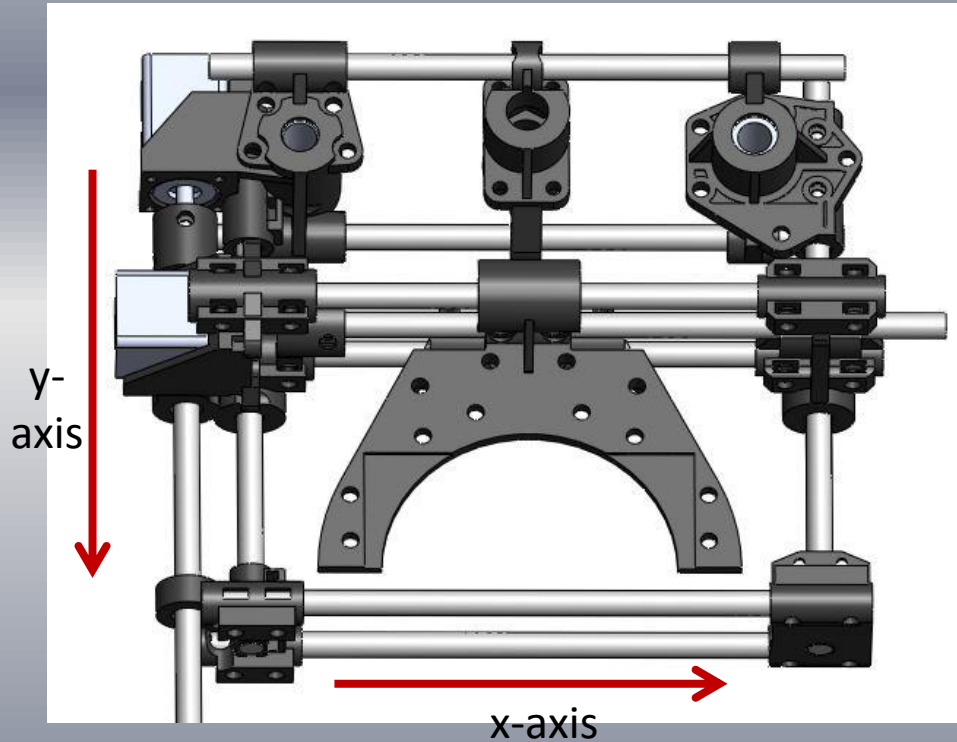


Stage



- Perspex stage supported by steel rod frame
- Designed for microscopy but suitable for other scientific applications

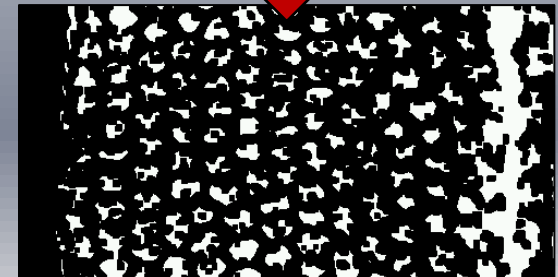
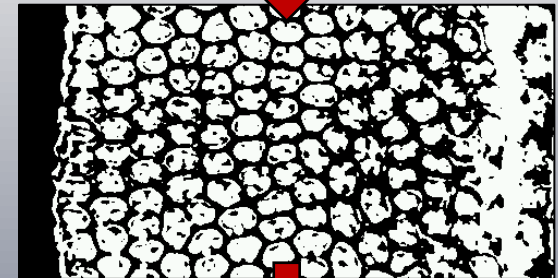
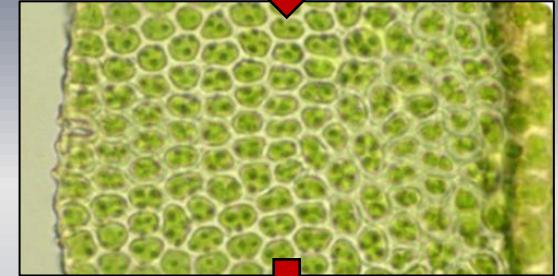
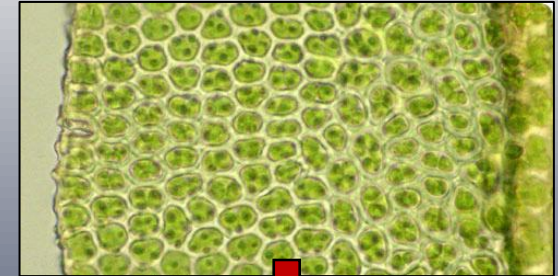
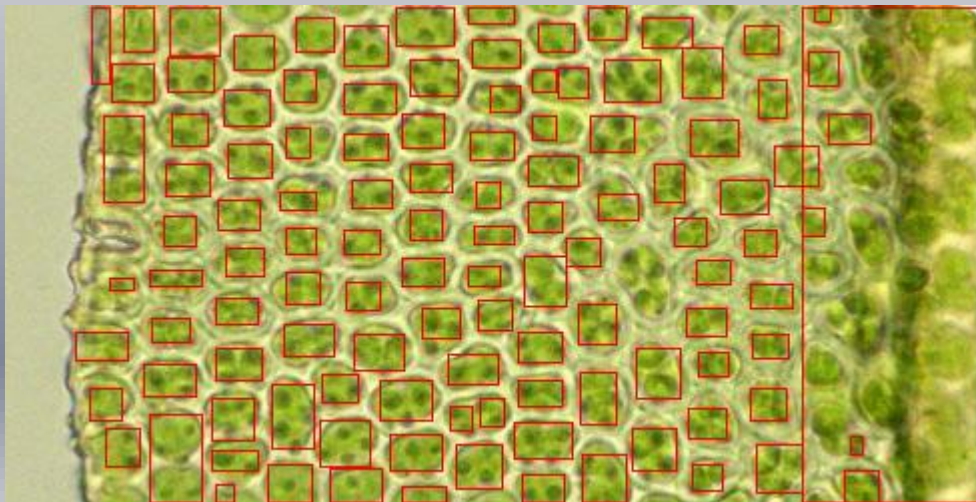
X-Y Translation



Cell Counting

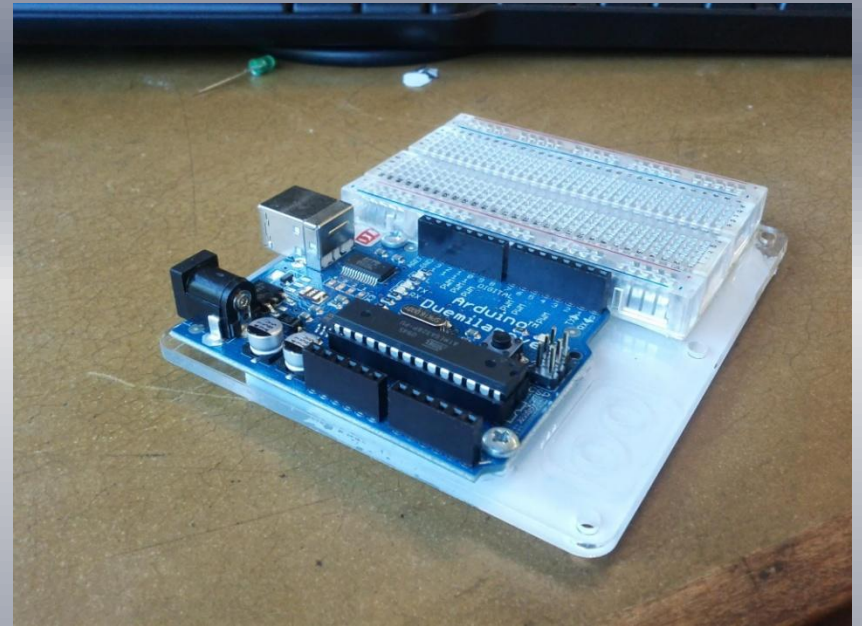
1. Gaussian blur
2. Threshold
3. Erode
4. Blob detection

Implemented using SimpleCV



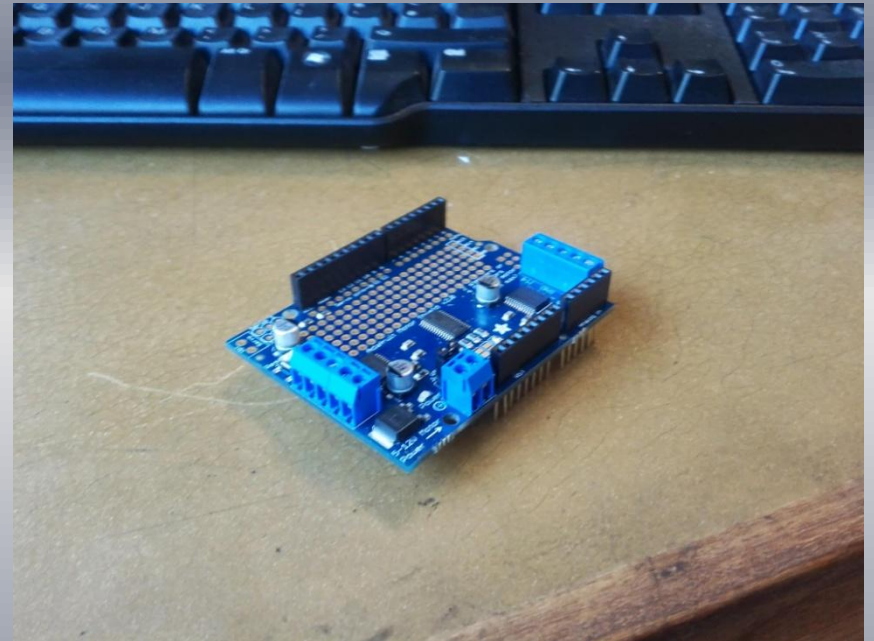
Hardware

- Raspberry Pi
- Arduino Duemilanove
- Motor Shield
- LCD Screen
- LED Ring
- 60mW White LED
- Manual Controls
- Total cost ~£100



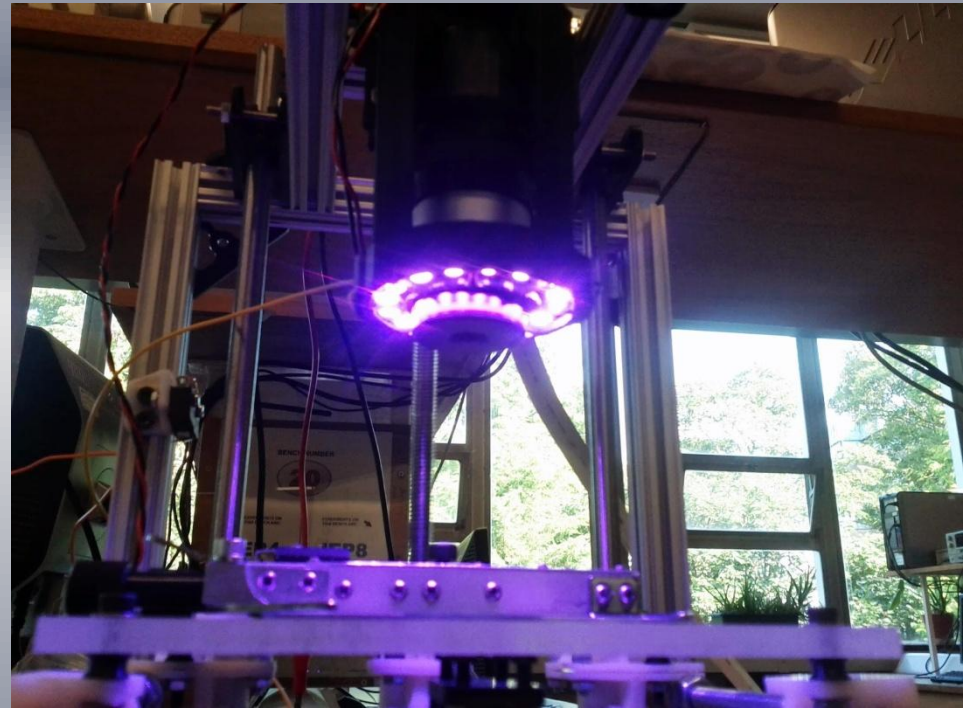
Hardware

- Raspberry Pi
- Arduino Duemilanove
- Motor Shield
- LCD Screen
- LED Ring
- 60mW White LED
- Manual Controls
- Total cost ~£100



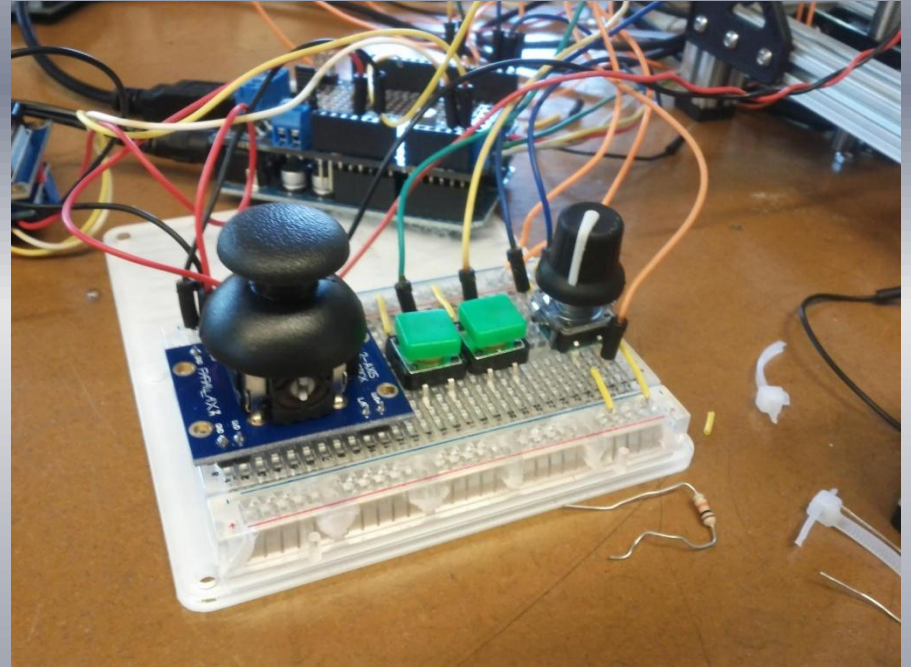
Hardware

- Raspberry Pi
- Arduino Duemilanove
- Motor Shield
- LCD Screen
- LED Ring
- 60mW White LED
- Manual Controls
- Total cost ~£100



Hardware

- Raspberry Pi
- Arduino Duemilanove
- Motor Shield
- LCD Screen
- LED Ring
- 60mW White LED
- Manual Controls
- Total cost ~£100



Software

- Arduino Sketch
- Accepts Serial Commands from Pi
- Use Serial Library/Terminal on Pi
 - Arduino IDE Serial Terminal
 - PySerial, Twisted
 - Boost ASIO
- Stepper Control
- Illumination Control

Serial Commands

- Human readable
- Format:
 - Command name
 - Argument if required, space separated
 - Terminated with newline

e.g

```
calibrate
```

```
set_ring_colour 8A2BE2
```

```
z_move_to 500
```

- Returns confirmation, value and/or error message

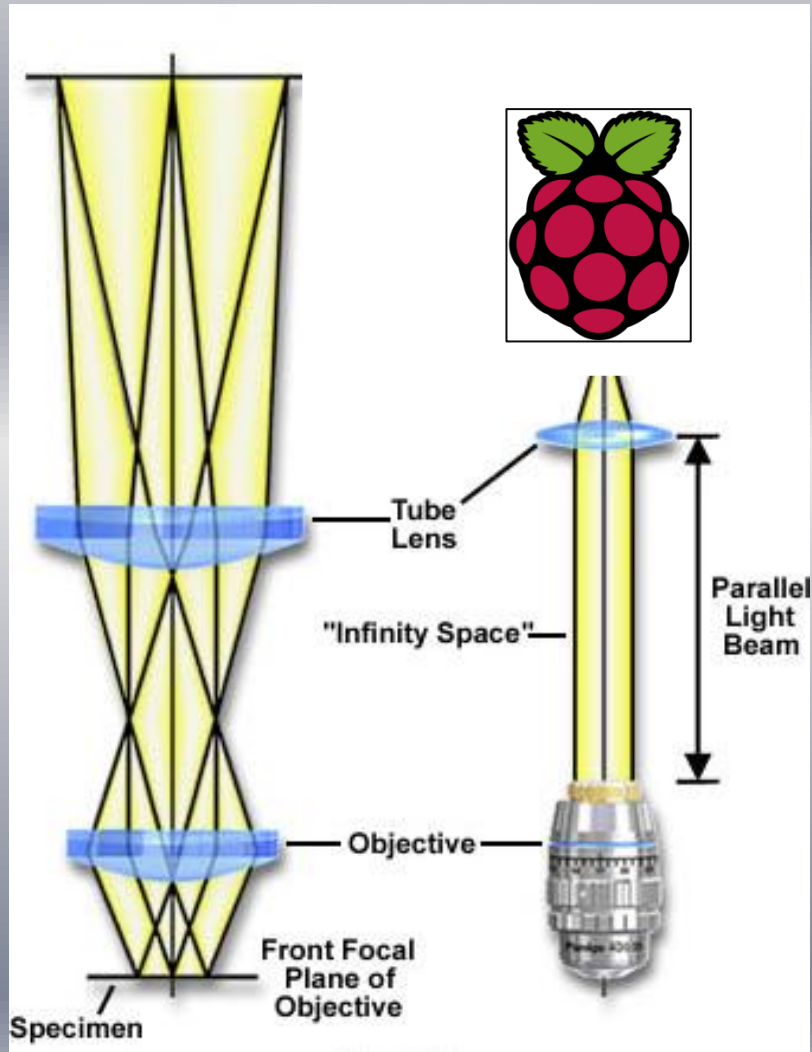
Stepper Control

- Non-blocking
- Self-calibrating
- Independent axis control
- Absolute and relative positioning
- Acceleration limits
- Serial and Manual Control

Future

- Better Pi-side software support
- GUI
- Casing
- Power Supply
- Custom PCB/Shield?

Optical system

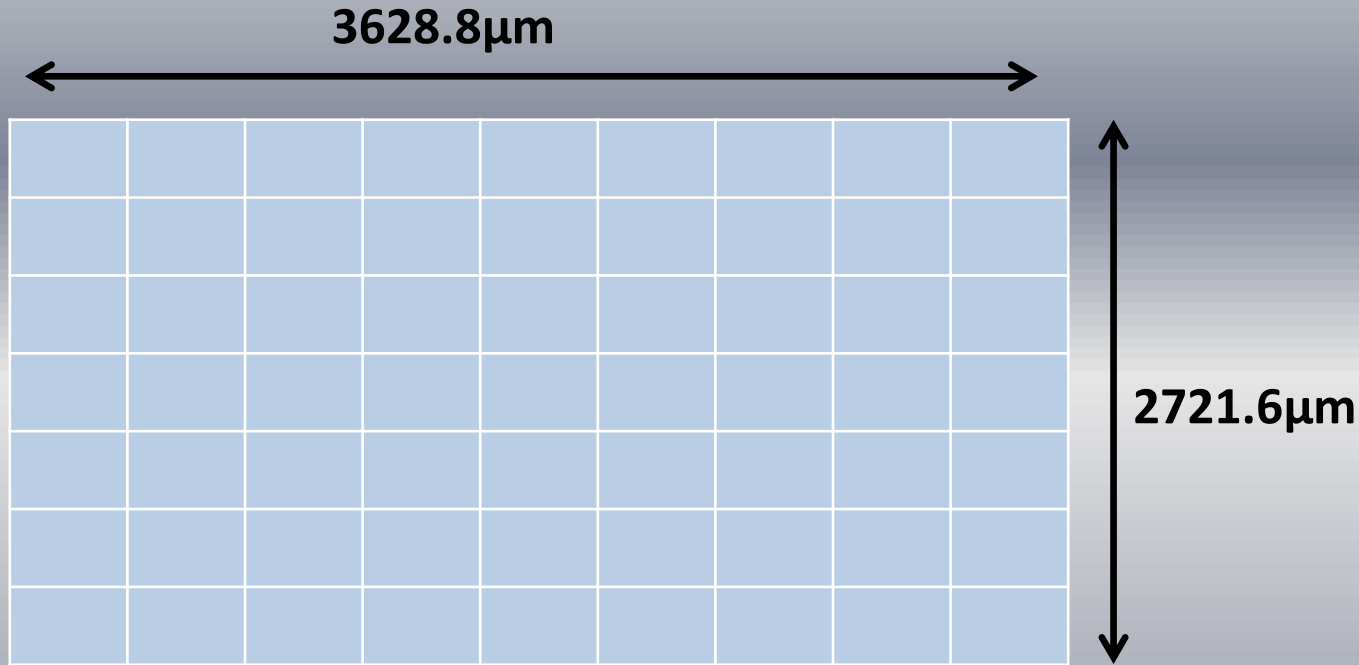


- Raspberry Pi Camera Module

- Substitute Tube Lens

- Infinity Corrected Objective

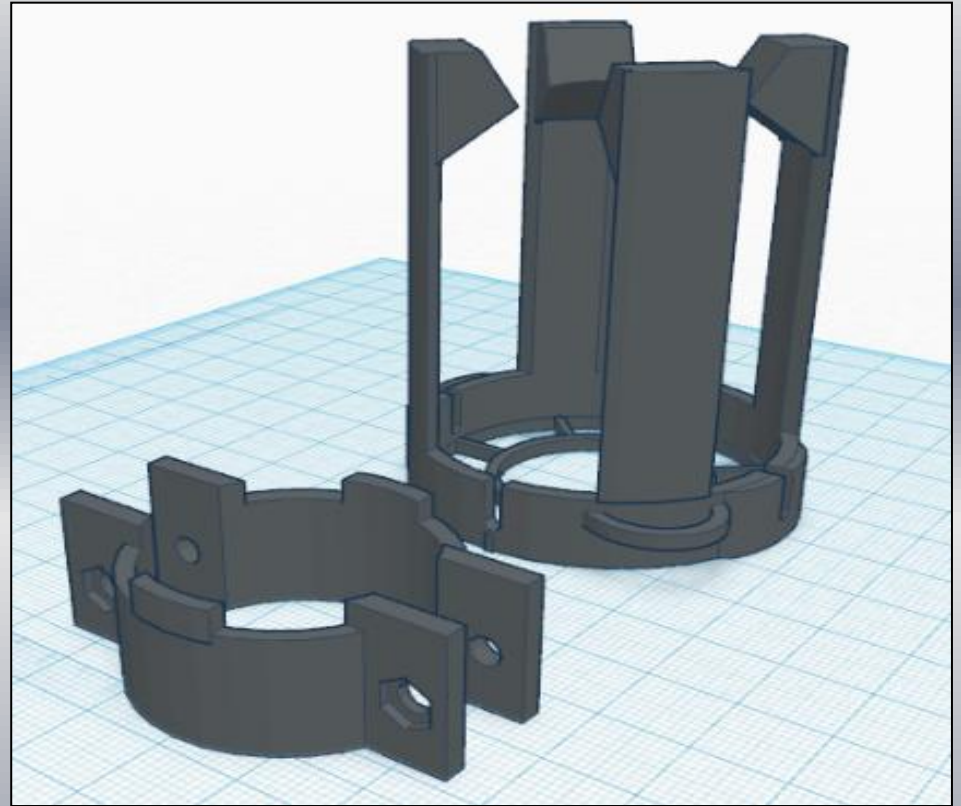
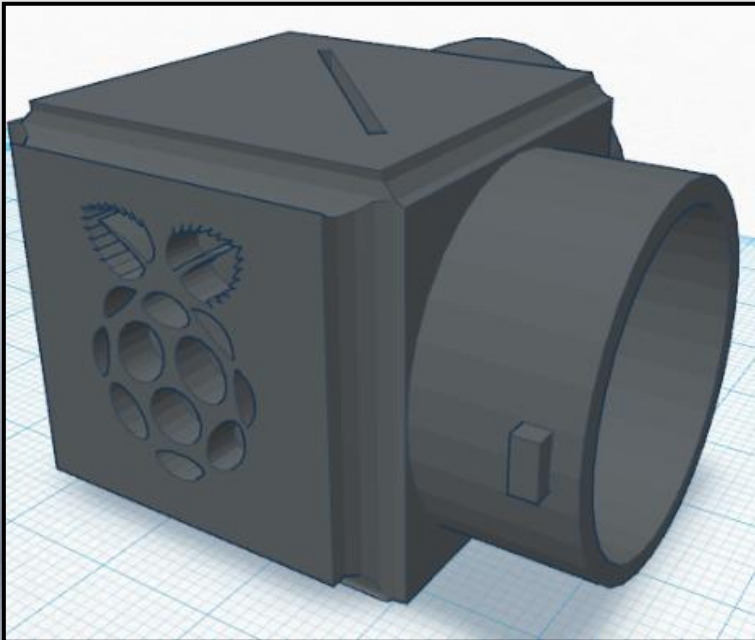
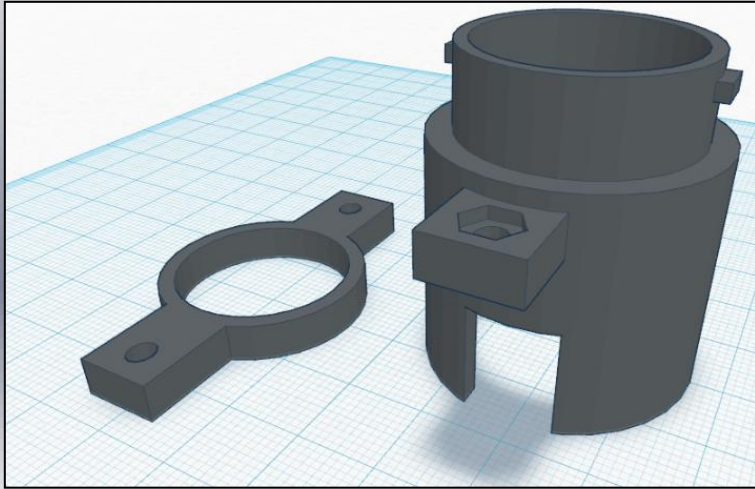
RPi Camera Module



Pixel pitch $4.1\mu\text{m}$

- **Small Sensor**
- **Suitable Depth of Field**
- **Difficulty controlling brightness etc.**

3D Printed Optics



- Cheaper than commercial tubes
- Easy to adapt

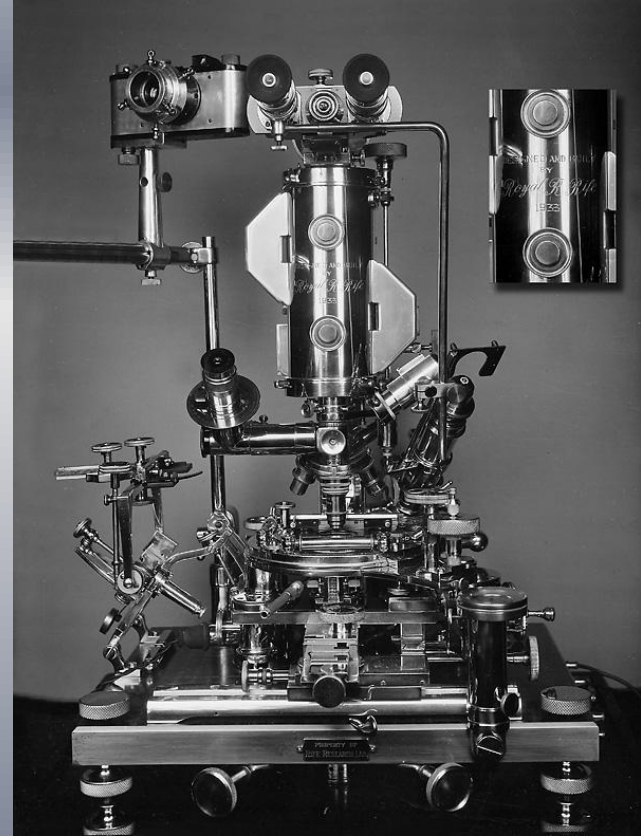
Automation

Now put together all different aspects:

- Mechanics
- Electronic control
- Optics and Imaging
- and of course, the Pi

•Open-Source project → Many possibilities

- Horizontal stage control for live sample tracking
- Focus stacking of macro images
- Extra sensors for environmental control
- And anything else you might think is useful in a microscope!



Example :: Auto-focusing

$$F = \frac{1}{WH\mu} \sum (I(x, y) - \mu)^2$$

- Using this formula to compute a 'focusing value' for an image, it is possible to auto-focus the microscope.
- Here in particular, we implement the following algorithm

Auto-focus :: algorithm

Compute focus value for
starting Image (F_{start})



Do the same for an Image
above or below (F_1)

$\xrightarrow{\text{If } F_1 > F_{start}}$

Stay there and repeat
cycle



Else

Compute focus of Image on
other side of start Image (F_2)

$\xrightarrow{\text{If } F_2 > F_{start}}$

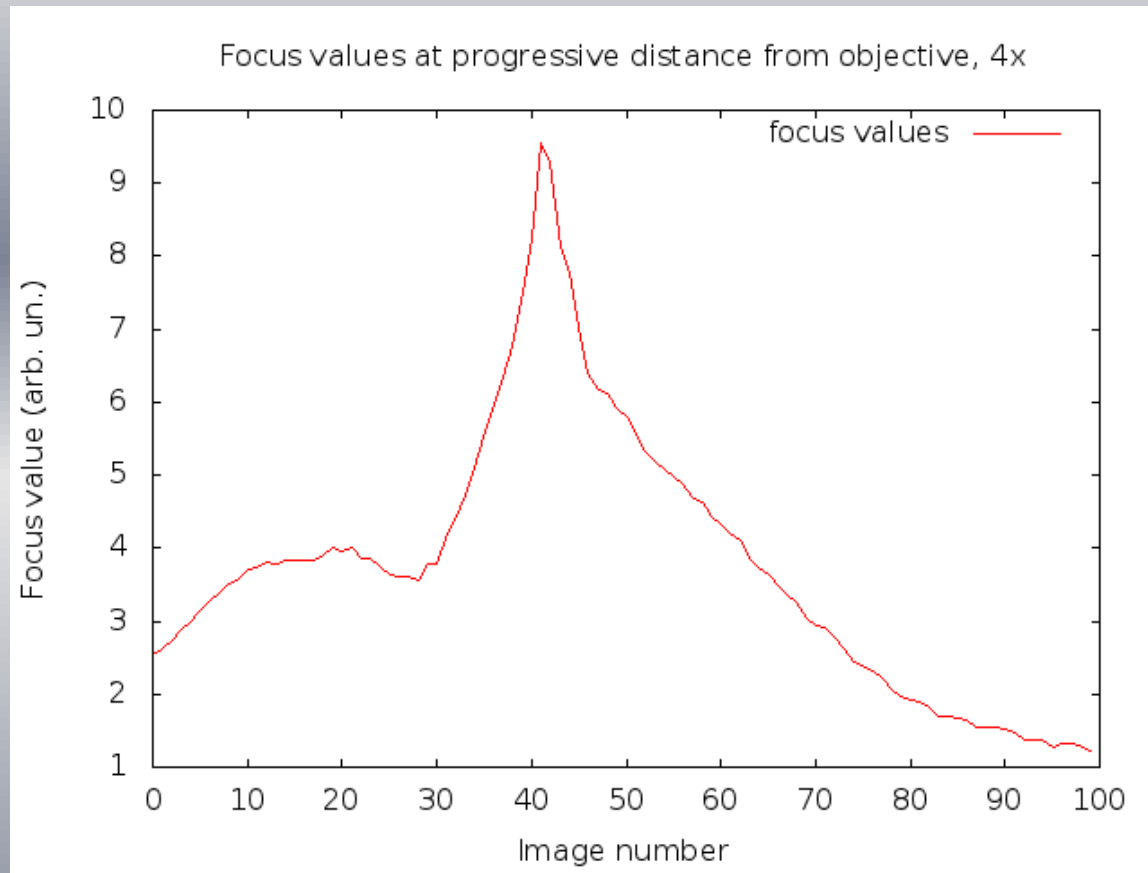
Stay there and repeat
cycle,
Halving number of steps



Else

Go back to start,
Halving number of steps

Auto-focus :: values



- For example, this is a graph of the focusing values of a sequence of images taken at equal distances, going downwards from the objective. The position corresponding to the peak is the focus point.

Auto-focus :: added_complexity

- The previous algorithm relies on the starting position being already reasonably close to the focus point. However, a more automated version is also available, that can do everything without user input.
- The possibilities for other automation software are endless, and they can all be easily implemented using Arduino commands and image processing algorithms

Auto-focus :: future_improvements

- Improvements on the efficiency of the auto-focusing algorithm will be derived primarily from:
 - Faster image acquisition (usage of MMAL library directly, rather than RaspiStill program)
 - Analysis of images loaded through MMAL in RAM memory, rather than saved on disk

Data Acquisition using the Pi

Introduction

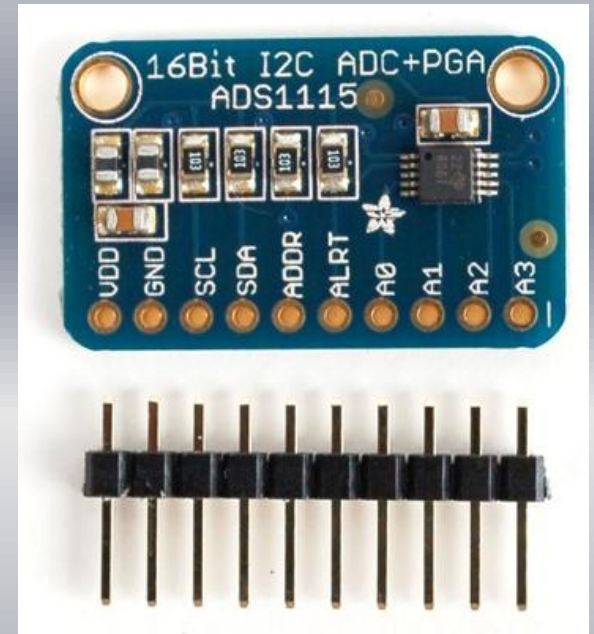
- Produced a set of tutorials based around Data Acquisition using the Pi
- All tutorials involve taking an analog input voltage, digitising it and saving the result to the Pi
- Tutorials divided into two groups:
 - Interfacing an ADC with the Pi (Ti ADS1115)
 - Interfacing the Pi with a microcontroller (Arduino Uno)

Aims

- Aimed at undergraduate level students
- With:
 - Basic programming experience
 - Little / no experience using microcontrollers
- Motivations:
 - In the future should be helpful in the microscope project
 - Many engineering students terrified by the prospect of using a microcontroller / interfacing with hardware

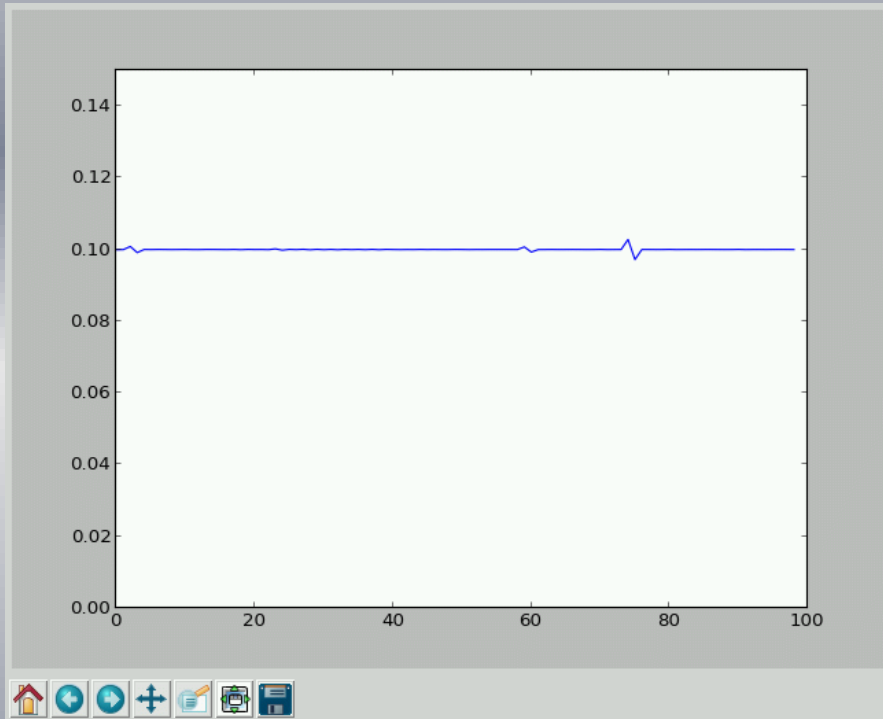
Pi + ADS1115

- Sold by Adafruit on pre-soldered PCB
- 16 bit resolution, maximum 860sps
- Interface using I2C
- Python libraries provided by Adafruit allow easy interfacing

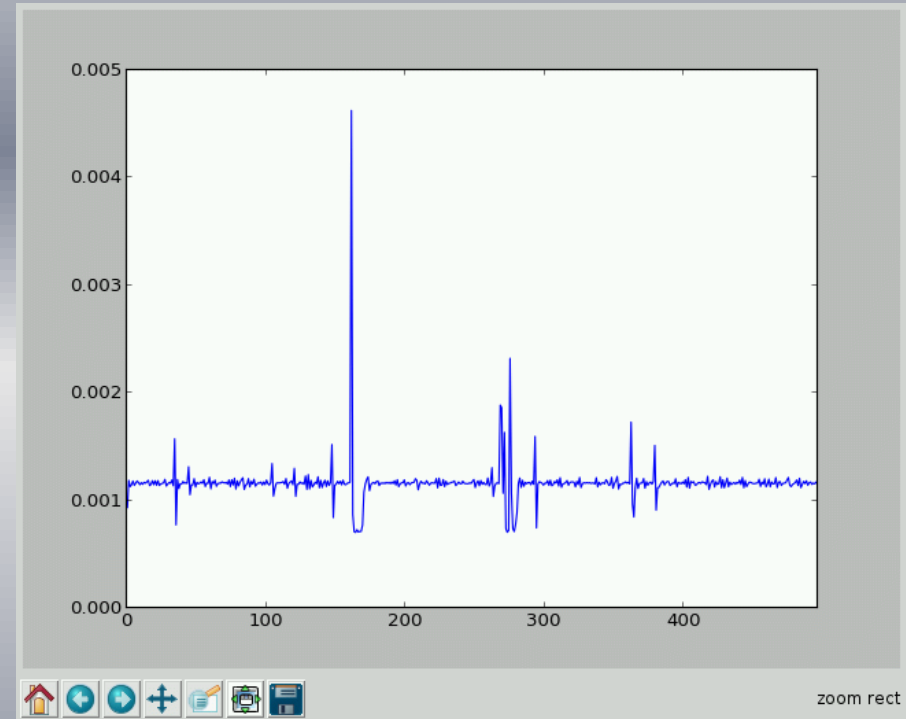


- Tutorials involve:
 - Introduction to how I2C works
 - Description and configuration of ADS1115 (using device datasheet)
 - Simple python script to log data
- Problems:
 - Raspbian not a Real Time Operating System – lots of jitter
 - Maximum sample rate 860ksps

Time Between Successive Samples



10sps



860sps

Pi + Arduino

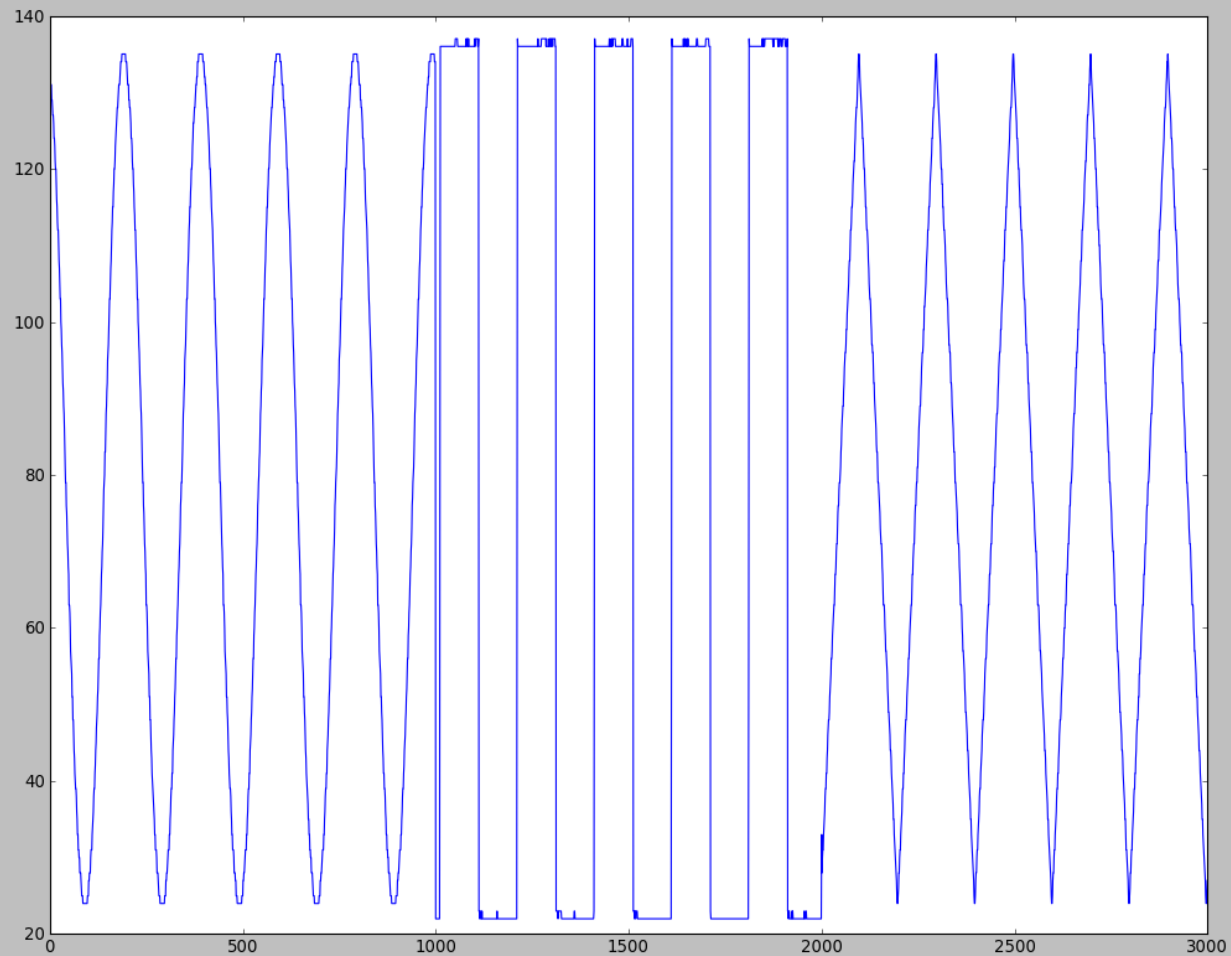
- Based around the ATmega328 microcontroller
 - Has a 10bit ADC
- Advantages:
 - Sample with very low jitter
 - Higher sample rates (10s of ksps)
- Interface options:
 - SPI
 - I2C
 - Serial

- Serial chosen for simplicity
 - Up to 50kbytes / second
 - Effective resolution of ADC drops below 10 bit above 15ksps anyway!
 - I2C and SPI require level converter
 - Allows programs to be run from a PC
- On the Pi side using python + Pyserial to communicate with the Arduino

Tutorials

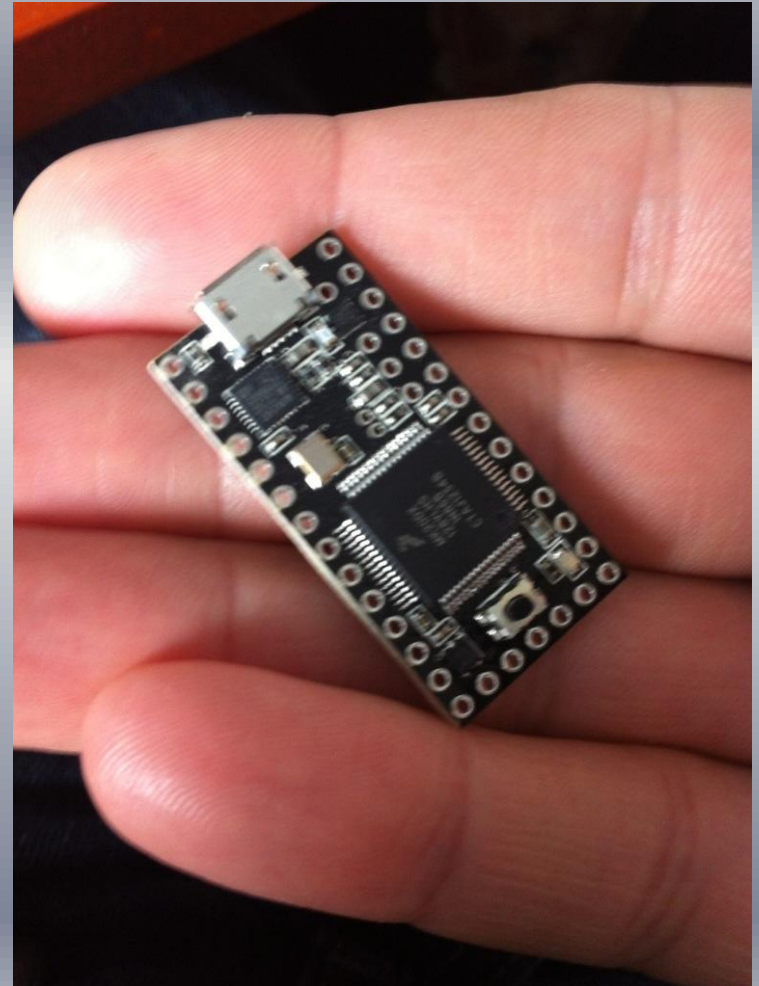
- Introduction to Arduino environment
- Serial communication between Arduino and Pi
- Interrupt Tutorials
 - External
 - Timer
 - ADC
- Data logging with Pi + Arduino
 - Continuous logging (up to 35ksps – 8 bit)
 - Burst sampling (100ksps – 8 bit)

Burst Sampling of 750Hz waveforms sampled at 100kps



Teensy 3.0

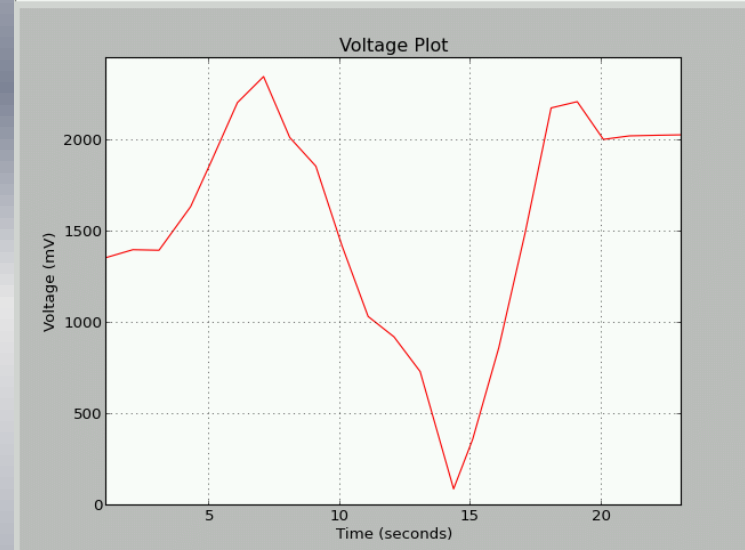
- Based around ARM Cortex M4
- Programmed from the Arduino IDE
- Serial functions work using USB protocol (12Mbit/s)
- 16 bit - 100ksps continuous sampling with Pi



3rd Year Bread Making Lab

- Existing lab involves recording a voltage output using voltage meter by hand
- Use Pi + ADS1115 to log data instead
- Gives students a digital output format

Press the "Start Sampling" button to begin data sampling. Data is sampled at 1 sample per second. The graph is updated every 10 seconds. Use the slider to adjust how many data points are shown in the graphical window. "Stop Sampling and Save Data to File" saves data to the file data.txt in csv format. To start sampling again from scratch close and then re-run the program.



Voltage reading (mV)

2024.3

Current Sampling Frequency (Hz)

1.0

Enter Sampling Frequency (Hz):

Update Sampling Frequency

Number of samples to display in plot:

500

Start Sampling

Stop Sampling and Save Data to File

Quit

Extensions

- I2C and SPI communication between Arduino and Pi
- Modify burst sampling programs to make simple digital oscilloscope
- Circuit designs
- Interrupts on the Teensy
- DMA tutorial on Teensy / Arduino DUE