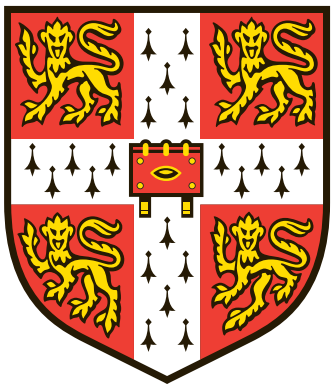
An Investigation into the Design and Construction of a Live Cell Imaging System

Fergus Riche

Magdalene College



Mini Research Project Report

Centre for Doctoral Training
in Sensor Technologies and Applications

Department of Chemical Engineering & Biotechnology
University of Cambridge

Supervisors: Dr. A Kabla, Dr. R Bowman

21 March 2016

EPSRC Centre for Doctoral Training in
Sensor Technologies & Applications

The work described in this report is the result of my own research, unaided except as specifically acknowledged in the text, and it does not contain material that has already been used to any substantial extent for a comparable purpose. This report contains 12 pages and 6096 words (excluding this page and the appendices).

Signed: _____ Date: _____

(Student)

I confirm that I have cleared the laboratory space I have used for the work described in this report, to the satisfaction of the supervisor of this mini project. All samples have been properly and safely stored or disposed of according to University guidance.

Signed: _____ Date: _____

(Student)

I confirm that the student above has cleared the laboratory space used in this project to my satisfaction.

Signed: _____ Date: _____

(Mini project supervisor)

Contents

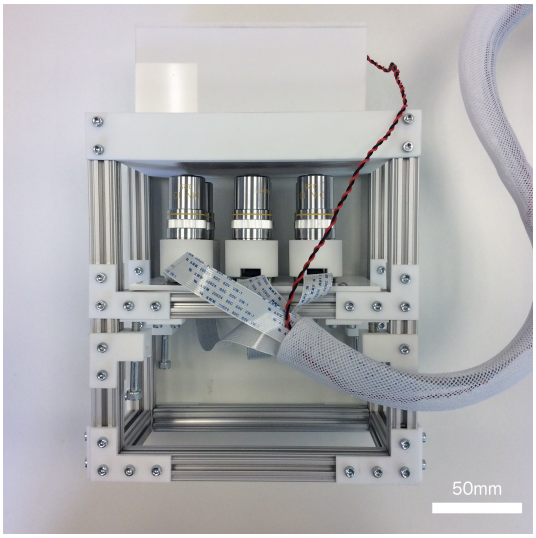


Figure 1: The previous state of the art for low cost open source multiplexed live cell imaging, as developed by the author, the "MultiPi". Whilst the device was able to gather high volumes of concurrent data from a single experimental run, it was not able to focus the cameras independently, and required a large control module, placed outside the incubator.

1 Abstract

The application of engineering science to biological development is a promising new field for biologists and engineers alike. Many phenomena in embryology, such as the development of coordinated networks of beating heart cells in cardiogenesis, and the wound healing response in brain tissue[1] appear to be closely related to the mechanical properties of the tissues involved[2, 3, 4], signaling the cells' passive as well as active involvement in the constitution of their solid environment. In order to further this field by developing devices to enable the widespread imaging of live cells, advantage must be taken of the recent availability of low cost image sensors and processors, as well as developments in rapid prototyping additive manufacture techniques, which now enable low cost peer to peer sharing of mechanical designs and devices.

This report describes the design and construction of a contribution (see figure 2) to the state of the art in low cost automated live cell imaging, in two principal areas; a combination of existing technologies to facilitate the easy and useful arrangement of a multiplexed imaging system, and the development of a new device to enable this imaging system to be automatically controlled. The benefits of using such a system are explained and shown to extend far beyond a simple cost and resource saving. In addition, the design is described in detail to highlight notable features, and the results of tests reported to confirm that it is satisfies the specification demanded of it.

2 Introduction

Live Cell Imaging (LCI) is the study of live biological specimens, usually by optical methods[5], at magnifications large enough to clearly examine individual cells. Such techniques have been in use for over seventy years, being first developed around a traditional light microscope using the most advanced image capture technology at the time- silver halide film. Despite great advances in all of the fields related to LCI since the first experiments were performed[6], very little has changed in the way that most commercial LCI systems are designed; typically being based around a bench-top light microscope, and adding in the additional functionalities of cell environmental control and image capture by means of external devices. As a result, these systems often become vastly overspecified and inflexible, presenting many research labs with a significant barrier to entry in the study of cellular development.

LCI is a useful technique for a variety of fields, one of the most promising of which is synthetic biology. This relatively new discipline, arising out

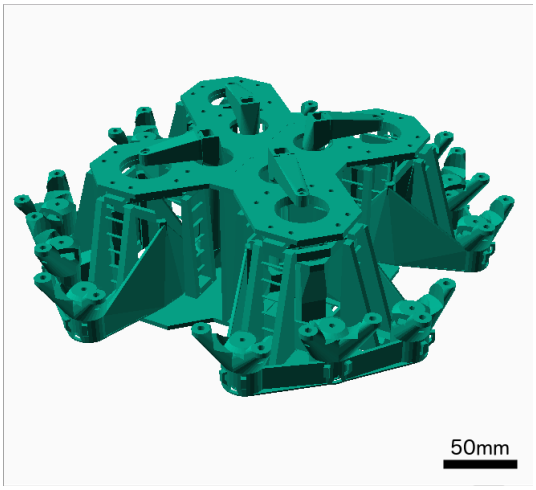


Figure 2: A computer render of complete imaging platform, suitable for imaging up to four specimens simultaneously, as described in this report. This design brings applies the mechanical advantages of low cost flexure mechanisms to the principle of multiplexed imaging to achieve flexible high throughput dynamic imaging of evolving specimens.

of the study of molecular biology, seeks to apply the principles of engineering design (see figure 3) to biology[7], in order to achieve a comparable level of technological progress using living matter, as has been achieved with traditional engineering materials, such as metal, wood and concrete. LCI is useful in applying two of these principles; characterisation, where optical assays are frequently used to examine the time response of a system, and abstraction, where little is known about the extent to which biological systems can be thought of in terms of traditional engineering multi-input multi-output systems, and optical methods appear to be a promising avenue of discovery due to the large volume of spatio-temporal data that they can generate[8, 9]. In addition to research, LCI has many applications in industry, not least in assisted fertility, where optical observation of *In Vitro* Fertilised eggs is one of the few methods of selection for implantation allowed by the legislation[10]. As the market for fertility treatment is competitive, it is desirable for clinics to be able to select the most viable eggs for implantation in order to ensure a successful pregnancy in as few attempts as possible. As a result, assisted fertility clinics are one of the most active areas for the development of systems to image the very earliest stages of development[11].

In both industry and research, equipment and resource (time, space, personnel) costs can dominate the landscape of methodology and instrumentation choices. The open source technology movement, which has made significant contributions to overcoming these costs in the software domain by means of allowing flexibility in how contributions are made and encouraging greater usage of scientific and engineering principles in evaluating technology, is now expanding to hardware, where it has already made contributions to the field of LCI (see figures 1, 4). Figure 4 shows the openflexure microscope[12], a 3D printed XYZ positioning stage and imaging system based on the Raspberry Pi, capable of a range of motion of $\pm 5mm$ with a resolution of $\sim 70nm$ and a repeatability of $\sim 1\%$ over its entire range. This report describes a design based on combining this stage with a camera multiplexer (see figure 5) to solve the problem of low cost, high throughput imaging, as well as the design of a motor drive, making low cost closed loop control of such a system possible for the first time.

3 Specification of Design Criteria

The current state of the art of open source live cell imaging, as described in § 2 facilitates either high data output through the multiplexing of compact imaging units, as with the MultiPi, or the precise control of sample position and focus when setting up an experiment, as with the openflexure stage. A significant bottleneck in the current mode of multiplexing is that a Raspberry Pi unit is required to drive each camera that is used, driving up the cost, size and computational complexity of any such device. Similarly, the only easy and repeatable way of translating the openflexure stage during a measurement is to use a microcontroller such as the Arduino with a corresponding motor drive board. Whilst this solution is simple and economical enough for a single stage, it quickly becomes vastly more expensive and electrically complex for a system that has more than one stage.

In order to mark a shift in the capabilities of open source LCI technology, the proposed design would have to allow multiplexed positional control with minimal hardware redundancy, which is to say that there should not be a significant amount of hardware that is not being used to its full potential in the design. In doing so, the design should address the problem of multiplexed positional control as well as that of multiplexed image collection by offering a simple scalable design structure (see figures 6 and 7), along with a simple interface that does not require the user to know about the details of the multiplexing mechanism. The number of units within a multiplexed cluster was chosen to be four, on the basis of the symmetries of the openflexure stage which naturally suit it to arrangement in a square configuration.

Having establish that units were to be multiplexed in groups of four, the minimum number of axes specified for independent simultaneous control was therefore 12, though the capability to exceed this minimum was determined to be desirable,

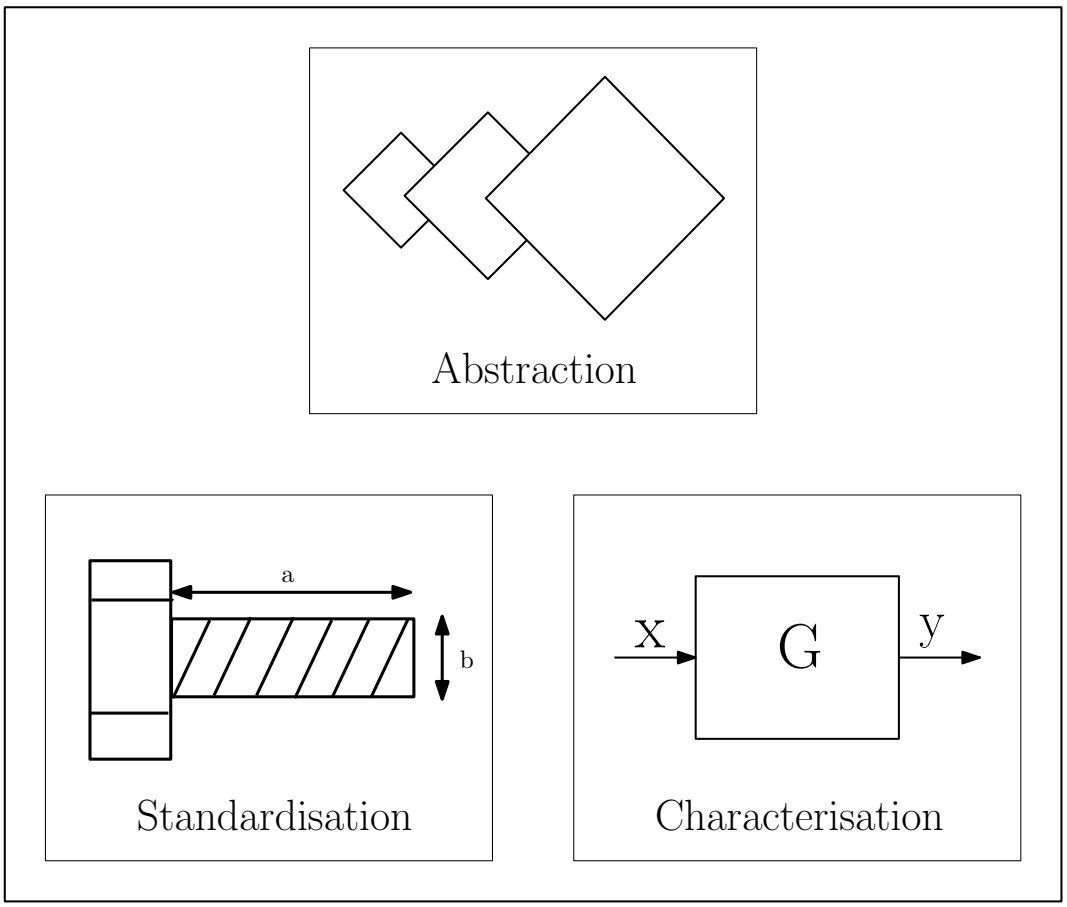


Figure 3: Engineering design epistemology can be viewed in terms of three principle components; abstraction, standardisation and characterisation. Standardisation refers to the inherent homogeneity of the most basic components of a system, characterisation to knowledge of the way components transfer energy from input to output, and abstraction to the ability to compose higher level components from basic components, which inherit the properties of standardisation and characterisation from their constituent components.

should it be possible. In addition to this minimum, the motor drive design was required to allow the possibility of easy path setting and following, in order to facilitate the implementation of scanning and tracking applications in future developments. In order to allow future developers to test additions to the system, and in light of the computational complexity of the operations needed to follow paths specified by the user, the motor control units were specified as programable microcontrollers, communicating via I2C, a simple protocol designed for inter-chip communication that uses only two wires and has a high data transfer rate. An additional benefit of the I2C protocol is that as long as the devices to be added to the bus have been pre-assigned unique addresses, the devices can be added without any other configuration steps being taken, allowing daisy chaining of the devices to expand multiplexing capabilities as necessary.

Unlike the motor controllers, which were to be designed essentially *ab initio*, a camera control and multiplexing unit is contingent on the Raspberry Pi at one end, and the camera module at the other. In order to avoid the complexities associated with the camera communication protocol, which is not currently made publicly available, two possible solutions were considered. The first of these was to use the Raspberry Pi Compute module, which offers access to two camera ports, and is amenable to embedded applications, simplifying communication between the two modules that would need to be used for a four-way cluster. This was, however, felt to violate the principle of minimal hardware redundancy, and so a second avenue of enquiry was established, revealing the IVMECH camera multiplexer[13] shown in figure 5, which was then specified as the camera controller. Whilst this component of the design is not open source, it was determined that the benefits of using it to demonstrate the principle of operation of the new device outweighed the downsides of using a closed technology to do so.

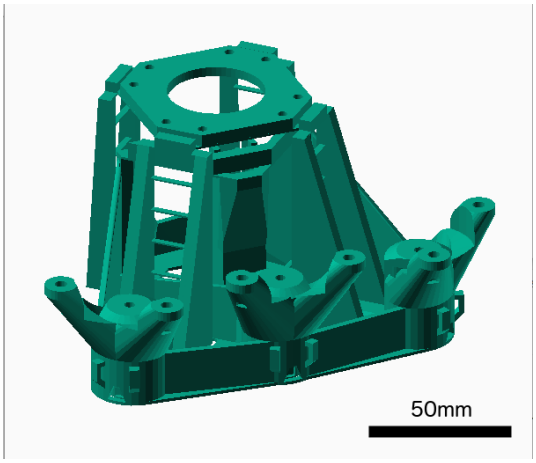


Figure 4: A computer render of the openflexure microscope, a precision XYZ stage, designed to print on low cost filament deposition type 3D printers in one piece, requiring minimal additional hardware for operation. This image shows the device in the large stage configuration, suitable for use with a microscope objective. Relative motion of the stage and the base is accomplished with the use of flexure hinges, which are well suited to production on filament deposition 3D printers.

4 Design Description and Justification

4.1 Mechanical

The challenge of the mechanical design, as outlined in § 3, was to integrate four of the openflexure stages into one unit, in order to allow imaging of more than one area of a culture vessel at one time. This integration is necessary due to the operation of the stage; as it accomplishes relative motion of the sample and the camera by moving the top of the stage, on which the sample rests, relative to the base, two stages cannot independently move a sample that spans both of them in their native configuration. The design solution to this was to fix the top of the stage, where the sample rests, and allow the base to move, thus moving the camera along with it. Figure 8 shows this design, which occupies the least amount of space possible in the plane, leaving only enough room around the stage to allow a full range of independent motion. This satisfies the specification by integrating the stages in a compact manner without any redundant hardware, as well as being a low cost solution that requires only sheet material and fasteners to implement. It should be noted that whilst the design shown in figure 8 features sample illumination originating from the centre of the platform, alternative designs that allow centrally placed samples are easily made possible by using alternative external sample illuminators and modifications of the openflexure stage which do not feature the illumination column at the rear of the stage, as shown in figure 4.

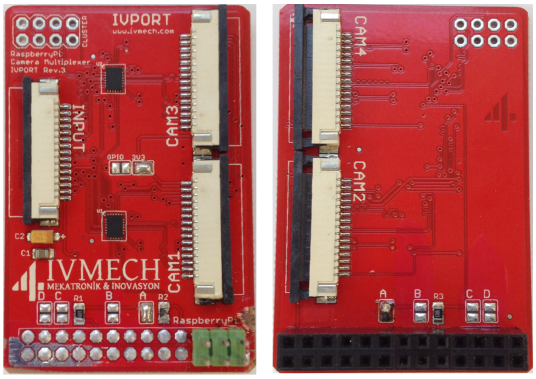


Figure 5: The IVMECH Raspberry Pi Camera Multiplexer. (Left) Top face, showing the camera cable connectors, and modified to accept a header allowing access to the Raspberry Pi's GPIO pins when installed. (Right) bottom face, showing the camera cable connectors.

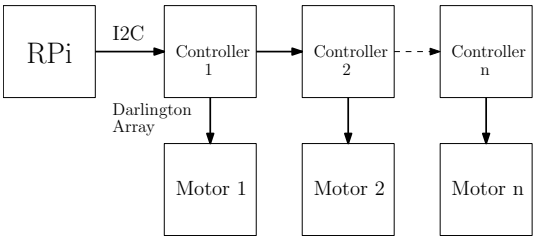


Figure 6: A solution neutral control diagram for the motor controller boards. The I2C bus allows communication between as many as 128 individual devices on the bus. Using a darlington array for the motor drive allows an independent choice of microcontroller and motor driver, as most microcontrollers can comfortably drive a darlington pair.

4.2 Electrical

In order to fulfill the requirements that the motor control boards be able to control at least twelve axes simultaneously, whilst being reprogrammable and amenable to scaling in the number of axes controlled, it was determined that a microcontroller should be used to act as the central control unit of the board, rather than a specialised motor driver chip. This decision was made on the basis that whilst a specialised chip would provide an initial benefit in terms of a reduction of development costs, the significantly greater flexibility that a microcontroller would offer would quickly become apparent in any situation where the board had to be adapted or improved.

The chip chosen for this application was the ATtiny 841, one of the newest parts offered by At-

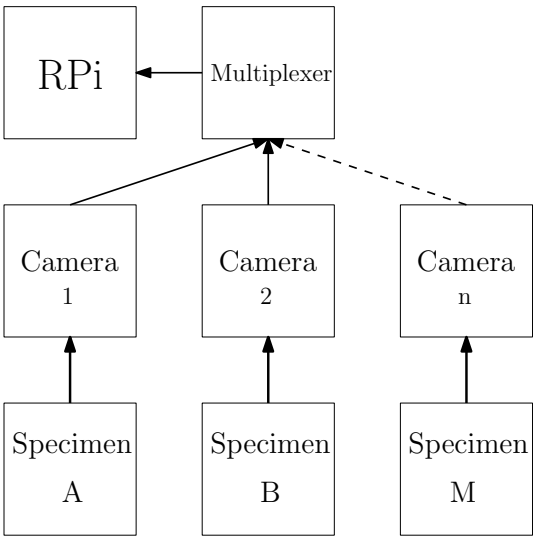


Figure 7: A solution neutral control diagram for the camera controller. Note that unlike the scheme for the motors, the cameras are specified to be controlled by a single control unit, which collates their input and appears to the Pi as a single camera.

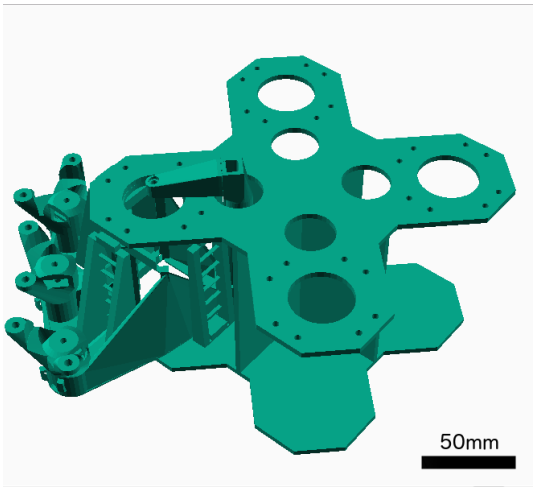


Figure 8: A computer render of the scope integration platform, as designed by the author, complete with an openflexure stage in position. This design is maximally compact for the number of stages it accommodates, whilst allowing for a full range of translation from each stage.

mel in its ATtiny line. This chip combines a large 8KB onboard memory with hardware support for the I2C communication protocol chosen for the design, as well as a calibrated internal oscillator, and 12 general purpose I/O lines. This combination of features is especially suitable for this application, as it both increases the reliability of the I2C communication line, and requires the use of very few external components. Figure 9 shows the schematic of the design used in the production and testing run, showing the microcontroller as well as the motor driver chip. The driver chip chosen for the design was the ULN2003, a commonly available chip that allows the low current outputs of the ATtiny 841 to drive the motor phases, which in addition to requiring around 200mA per phase, is an inductive load, meaning that it generates a voltage that opposes the change of current through the phases. This means that when a phase is switched off suddenly, as occurs in a pulsing regime such as that used to drive stepper motors, it can generate an extremely large voltage that might damage the microcontroller if it were to be directly connected to the motor. In order to prevent this "back EMF" from damaging the circuit, the ULN2003 incorporates a "flywheel diode", a diode that passes current generated through the back EMF through a low resistance line safely to ground, thereby contributing significantly to the long-term reliability of the motor drive circuit.

In addition to the motor driver chip, each MCU is outfitted with two indicator LEDs to assist in visual inspection and debugging of the circuit, in order to remove the immediate necessity to fully instrument the circuit in testing, which can act as a barrier to use and development; both in terms of the cost of instrumentation and the expertise needed to analyse the result. An In Service Programming header (ISP-2 in the figure) was also included for each MCU to simplify and save time programming the chips, a process which is undertaken frequently during development.

The other significant element of the electronic design was the interface between boards, and between the first board and the Raspberry Pi (see figure 10). Whilst the interface between boards could be quite simple, consisting only of the power and signal lines, the interface from the boards to the Pi required more care in its implementation. This is because the Pi operates at 3.3V, whilst the motor controller boards operate at 5.0V. This mismatch is bridged by the PCA9306D, a bidirectional level shifter chip designed for I2C communication between systems on different voltage levels. As all the boards were designed to be the same to reduce tooling and assembly costs, as well as complexity in the design and use of the system, the boards feature this chip whether or not they are connected to a Raspberry Pi. As it is not recommended to leave unconnected inputs floating on an active chip, as they would be on all boards not connected to the Raspberry Pi, a switch was implemented on the enable pin of the chip in order to render the chip inactive when the inputs were not connected.

The elements of this design were combined on a single printed circuit board, each of which is able to control a single 3 axis stage and can occupy any point in the chain connected to the Raspberry Pi. This modular aspect of the design satisfies the principle of minimal hardware redundancy as well as allowing scope for expansion, as any implementation

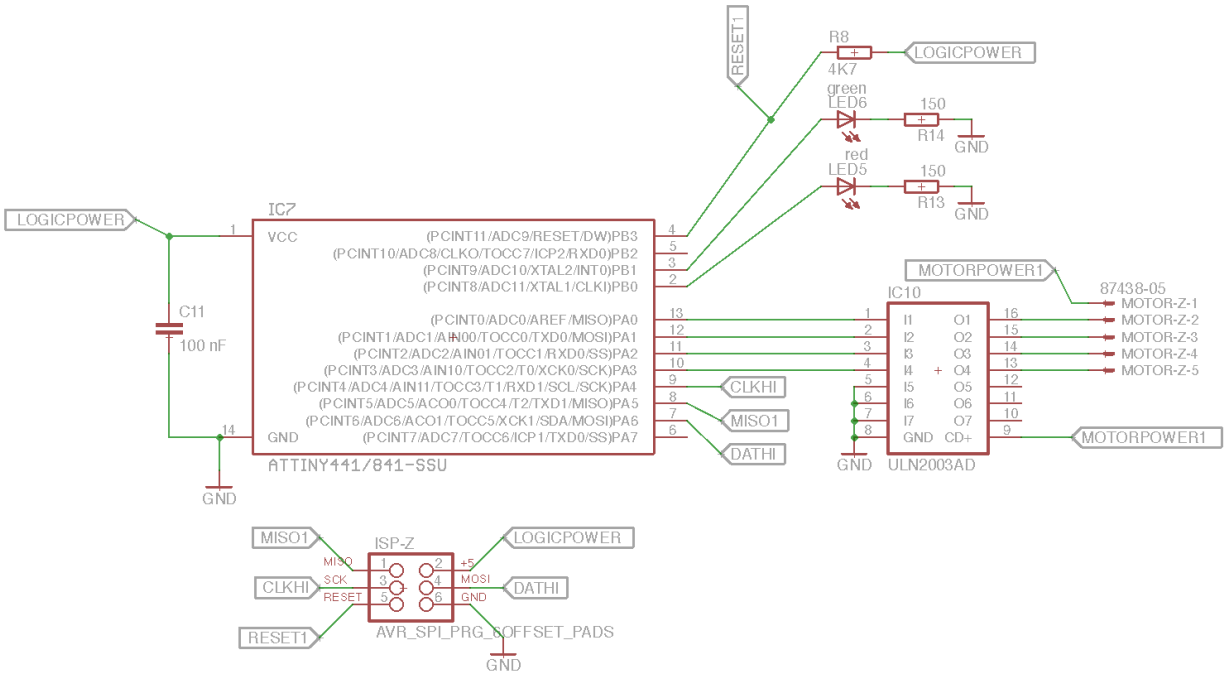


Figure 9: The MicroController Unit and power drive sections of the motor driver board. This unit is repeated three times on a board, once for each axis. Notable features of the MCU include the smoothing capacitor across the power input and the pullup resistor connected to the reset pin, which both act to increase the reliability of the circuit. The power drive (Right, IC10) is a simple integrated circuit that allows the low current microcontroller to switch the high current, inductive load of the motor.

of the design is able to use the minimum number of control boards necessary to drive its stages.

4.3 Software

As described in § 4.2, the control of the motor drive was implemented using a programmable microcontroller, which allowed flexibility in how the communication and execution of instructions was performed. This section presents an outline of the control flow implemented in the device, and § A.1 gives a full listing of the code used.

One important feature of the I2C communication protocol is that devices can either be set in Master or Slave mode. The difference between these modes is that Slave devices may only issue data on request from the master, which means that a slave device cannot signal to a master when it has completed an instruction. The effect this feature had on the design of the motor control software is that in order to comply with the design requirement that the motor controllers facilitate the implementation of path following, the controllers needed to be able to make many small consecutive moves seamlessly, without having to wait for the commands to be sent. As I2C prevents the Slaves from independently signaling that they require the next move instruction, a queue system was implemented to allow the Master to calculate large sections of the path in advance, and send the instructions to the Slaves for execution, one after another. The overall control flow in pseudocode is therefore as follows;

```
if (not receiving instructions){
        if(queue has stored instructions){
                execute the instructions
                                        }
        else{
        idle until new instructions
        are received
            }
                                }

elseif (receiving instructions){
        store instructions in queue
        return to execution
                                }
```

An additional peculiarity of I2C is that it specifies no data transfer protocol; whilst the first byte transmitted is always the address of the Slave that the Master wishes to transmit to, the bytes after the address byte are free to be anything that the master wishes to transmit. A data transfer protocol (see § A.1, A.3) was therefore implemented to allow the transmission of 16 bit signed integers from the Raspberry Pi to the motor control boards, which allows the motor to be moved up to 8 turns in a clockwise or anti clockwise direction in steps of 0.088 degrees, in one command. As the maximum effective speed of the motors was found to be 10

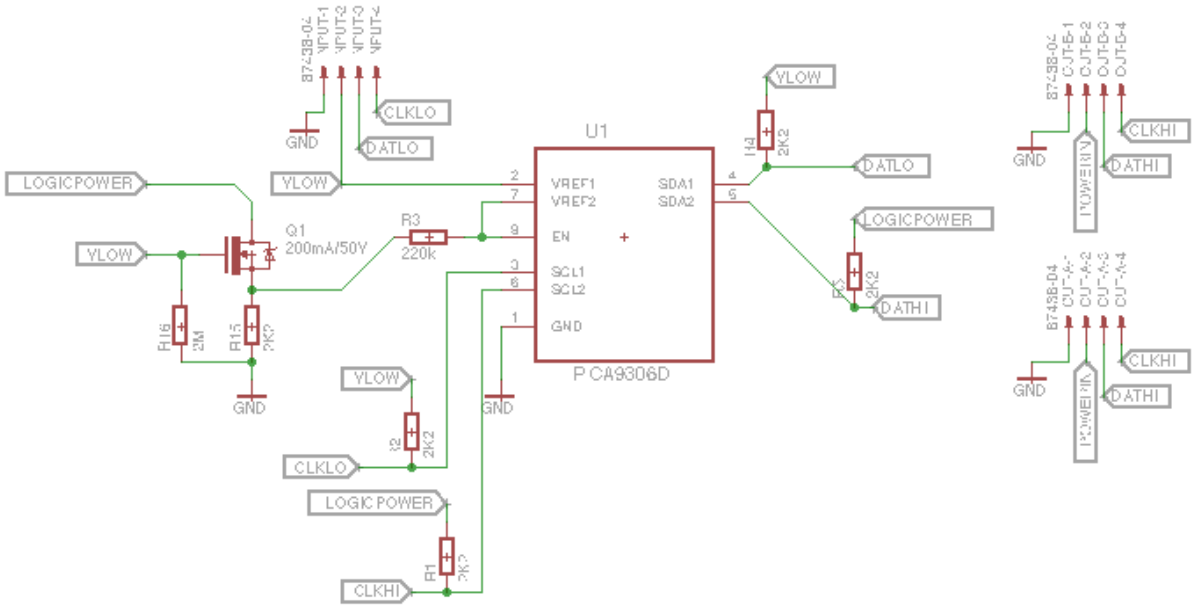


Figure 10: The level shifter and board to wire interfaces of the motor driver board. This shifter is required for I2C communication from the 3.3V Raspberry Pi to the 5.0V motor driver board. The transistor Q1 was implemented into the design in order to ensure the shifter would only activate when connected to a Raspberry Pi, thereby eliminating the chance of the I2C bus malfunctioning due to floating inputs from the boards not connected to the Raspberry Pi.

RPM, this was deemed to be an acceptable range of motion for one command to achieve.

5 Manufacture and Methods

In order for an open source hardware design to spread effectively without commercialisation, it must be easy to reproduce. Furthermore, it must be cheap and preferably rapid to reproduce, in order for potential users or developers to consider adopting it. In light of these considerations, it was determined that in addition to adhering to the design criteria for operation given in § 3, all the materials and methods required to assemble a completed device should be obtainable and achievable by any reasonably well equipped or funded laboratory.

Whilst this requirement could be interpreted to mean that the custom components would need to be producible from raw material by a prospective user, this is not necessarily so; the openflexure stage, for example, can be produced in-house on a range of low cost (<1000 GBP) 3D printers, but it can also easily be produced, albeit at a higher cost per device, by one of the many 3D printing bureaus that now exist to rent time or space on their machines. Instead, this requirement was interpreted to mean that all the materials required to assemble a fully functioning device should be widely available for order, such that a user with minimal production or tooling capacity could still easily produce a fully featured device. A Bill of Materials, as well as an estimated cost of the final design is given in § B.

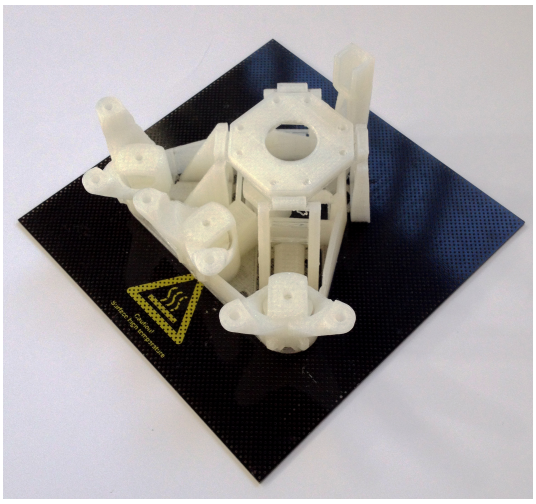


Figure 11: The openflexure stages used in the design were printed in-house on RS IdeaWerk Pro machines, which retail for 590 GBP at the time of publication. No alteration to the design or the printer was required for successful prints, demonstrating that the stage was found to be truly low cost, in both financial terms and resource consumption.

Manufacture of a complete device consists of three distinct components; the openflexure stage, which must be 3D printed in plastic (see figure 11),

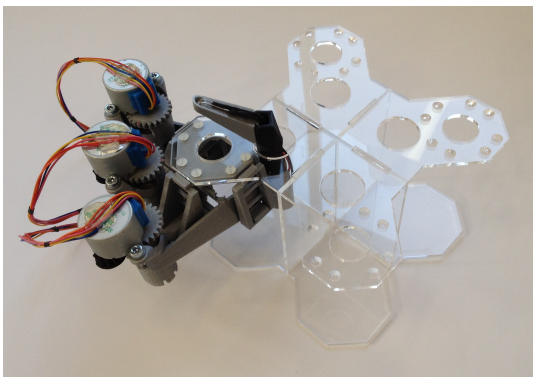


Figure 12: The physical realisation of the design shown in figure 8, outfitted with motors to drive the stage automatically. Note that this realisation uses the "small stage" version of the openflexure stage, as opposed to the "large stage" version shown in the render.

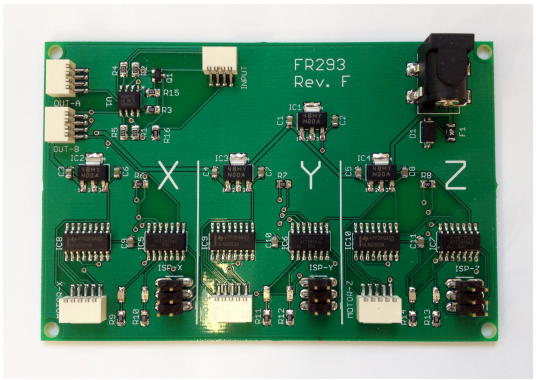


Figure 13: The motor controller boards were produced by a contractor, and soldered both by hand and using the reflow protocol described in §C.2. Little prior experience of soldering was found to be necessary in the assembly of the boards using the reflow protocol, which produced no discernible faults on any of the boards tested.

the microscope multiplexing stand, which can be cut from 3mm thick sheet material of any type and assembled by hand with acrylic adhesive, and the motor controller board (see figure 13), which can be produced by any board manufacturing house, and assembled with components either by the board manufacturer, or by the user.

After the manufacture of the device components, the board can be programmed with the code given in § A.1, using the protocol described in § C.1, which requires only an Arduino UNO and jumper cables. After this, the motor control boards and the raspberry Pi can be connected as shown in figure 14, and the motors attached to the boards. The device is then ready for operation or alteration according to the user's wishes.

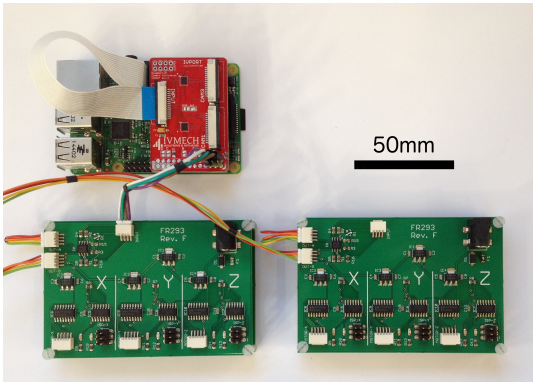


Figure 14: The complete integrated electronic array, featuring a 4-way camera multiplexer, and 4 3-axis motor driver units. The driver units here are shown stacked for compactness, with each 2 unit stack requiring only 100x66x25mm of space.

6 Testing and Discussion

A number of tests were performed in order to determine that the design satisfied the requirements set out for it, as well as to examine the performance of the design to determine areas for improvement in future iterations. The first test performed was to examine the motor controller output to ensure that the output waveform was of the expected form. Figure 15 shows the result of this test, which confirmed that the controller was indeed functioning, though some distortions from the expected square wave were observed. The first of these is seen on the trailing edge of each pulse, where the phase is energised. This spiking behavior can be attributed to the inductance of the motor phases, which is not suppressed by the flywheel diode when the phase is activated, only when it is deactivated. This is confirmed by noting that a comparable spike is expected on the leading edge of the pulse, when the phase is deactivated. Inspection of the plot reveals that whilst a spike is present, it is indeed suppressed by the flywheel diode. The second distortion is observed at the bottom of the waves, which are at around 0.8V rather than at 0V as might be expected. This can be explained in terms of the diode voltage drop within the motor driver chip itself, which is expected to be around 0.7V as the driver circuit is composed of Bipolar Junction Transistors which are fully saturated when the input is held high. As neither of these distortions were found to have any significant effect on the operation of the motor, it was determined that the behavior of the design was satisfactory.

Figure 16 shows a typical response of a single motor phase attached to a control axis on a motor control board to commands sent from the Raspberry Pi. It was found that the time period between receiving the commands and executing them varied from $\sim 0ms$ to $100ms$ on each attempt, the cause of

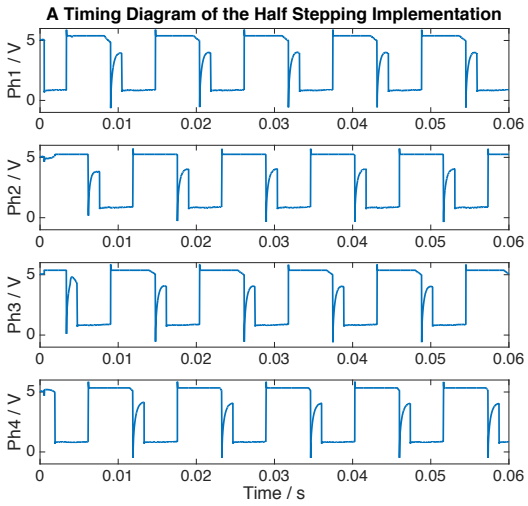


Figure 15: This plot shows all four phases of a motor attached to the motor driver board, running the half stepping implementation referenced in § C.1. The important feature of the plot is that the motor phase voltage waveforms maintain a constant phase relationship. Another notable feature is the distortion of the waveform on the step edge, which shows the effect of the inductance of the motor on the changing current demand of the power driver chip.

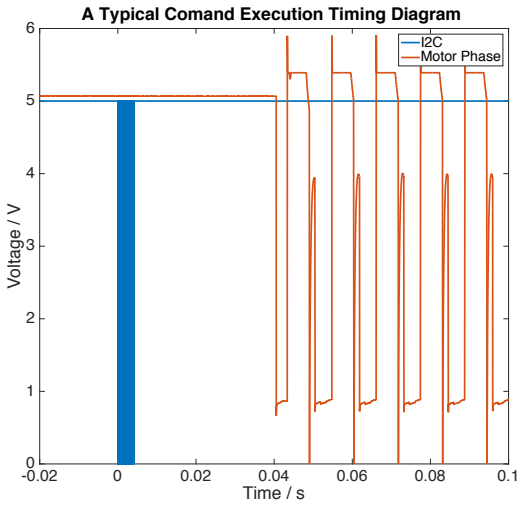


Figure 16: The time between the i2C communication, shown here as a dense cloud of points near $t = 0$, and the activation of a single motor phase, shown as a periodic waveform, is typically around 50ms for the code given in § C.1

which was determined to be the period of delay inserted into the operation loop if the microcontroller found itself in a state where it was receiving instructions, but had no instructions in its queue. This explanation was confirmed by sending new commands to a microcontroller which was already in the process of executing commands in its queue, where it was found that no gap was discernible when switching from old to new commands, as no delay routine would have been run.

In order to examine this delay effect further, the response of the first phase of three motors attached to consecutively addressed microcontrollers, to a series of I2C commands requiring each microcontroller in the sequence to execute the same instructions was measured, and the results are shown in figure 17. This result shows that the microcontrollers appear to respond in series, which is expected as the commands for the microcontrollers were sent sequentially, but also that in addition to the initial delay, there was a subsequent delay from one microcontroller to another, much greater than the delay between the I2C instructions. The most likely explanation for this behavior is slight differences in the frequencies of the internal oscillators of the microcontrollers, which would cause the execution of internal commands in each microcontroller to proceed at differing rates, including the delay function that was determined to cause the initial interval between the reception of commands and their execution. As in the case of the distortions within the waveform, it was determined that whilst this behavior was unexpected, it was not detrimental to the operation of the device, and furthermore that it could be diminished by manipulating the value of the delay constant in the code to arbitrarily low levels.

In addition to the tests performed on a single board to determine if the motor control boards functioned as intended; allowing reliable control from the Raspberry Pi, a test was performed to examine the requirement that the communication bus would extend reliably over all boards that were connected to the chain. It was found in this test that the initial design was not able to support communication over more than two boards at a time, due to unexpected interaction between the I2C level converter chip and the circuit used to activate the chip when the Raspberry Pi was connected. This interaction caused a voltage drop large enough to disable the chip completely when more than two boards were connected, preventing any commands from the Raspberry Pi from reaching the microcontrollers on the motor control boards. This fault was resolved by leaving all of the level converters permanently activated, which contrary to expectations, did not appear to cause any unreliability on the operation of the signal line. In order to ensure totally reliable operation of the system in the future, the design will be amended to ensure that the observed behavior of the activation circuit matches its intended operation.

As the design features the openflexure stage attached not by its base as originally designed, but by the top of the stage to the multiplexing stand, a

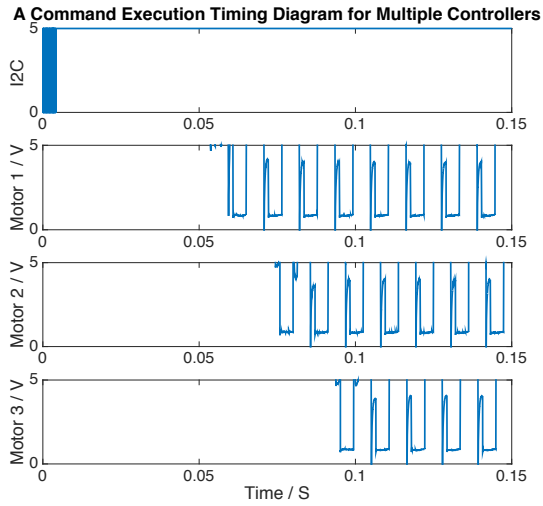


Figure 17: This timing diagram is an extension of that shown in figure 16, showing the response times of three consecutively addressed MCUs to an I2C command. Note that the MCUs respond in sequence, as expected, and that there is a delay of approximately 0.01s between each of their responses.

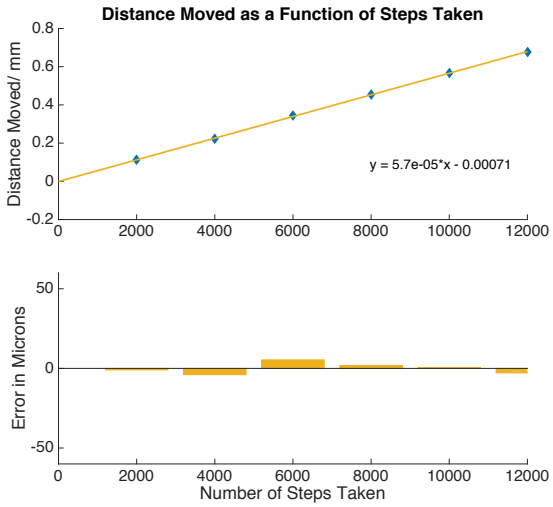


Figure 18: A plot of the distance moved by the stage as a function of motor steps taken, as measured by the relative motion of an optical calibration standard.

test was carried out to ensure that the mechanical performance of the stage remained relatively unaffected in the design. Figure 18 shows the result of translating the stage via the motor control on the X axis over a span of ˜1mm in order to check for linearity of motion, as well as to determine the coefficient relating steps of the motor to the distance moved by the stage. This figure shows that whilst the motion of the stage does show some observable error in the middle of its range, this error does not appear to be systematic, or large enough to warrant concern for live cell imaging applications, where a deviation of camera position of $<$10 microns over a range of 1mm is negligible. It was therefore concluded that whilst the mechanical performance of the stage suffered from the stage being mounted in an unconventional manner, the extent of this degradation was not serious enough to warrant modification of the stage or stand.

Further to the tests performed on the electrical and mechanical properties of the system, the optical performance of the system was measured in order to compare it to the performance of the MultiPi system shown in figure 1, as well as the benchmark system, a commercial inverted microscope used in the Cambridge University Engineering Department Biolab. Three parameters of interest were measured from the system, the magnification, spatial frequency response and resolution. The magnification of this system, which uses an inverted Raspberry Pi lens, was calculated as ˜11x (see figure 19), in comparison with ˜4x and 10x, offered by the MultiPi and commercial systems re-

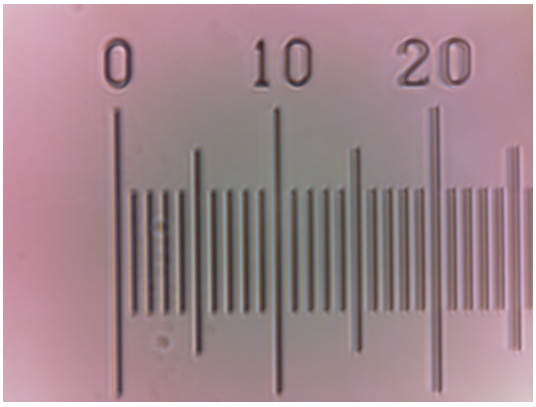


Figure 19: An image of the scale bar used in the calibration of the imaging system, which has units of tens of microns. The observed magnification of the system is 11.06.

spectively. Figure 20 shows the modulation transfer functions[14] of all three systems, calculated by taking the Fourier transform of the an optical impulse passed through the system, estimated from the derivative of an optical step as shown in figure 21. These measurements can be combined to form an estimate of the resolution of the system, which is given below where Δx denotes the minimum resolvable feature size, p denotes the linear pixel size, f_{min} denotes the location of the first minimum of the Modulation Transfer Function and M the magnification;

$$\Delta x \simeq \frac{p}{f_{min} M} = 2.55 \mu m$$

This figure is comparable to, though slightly larger than, the original figure of 2 microns estimated by the designers of the optical pathway.

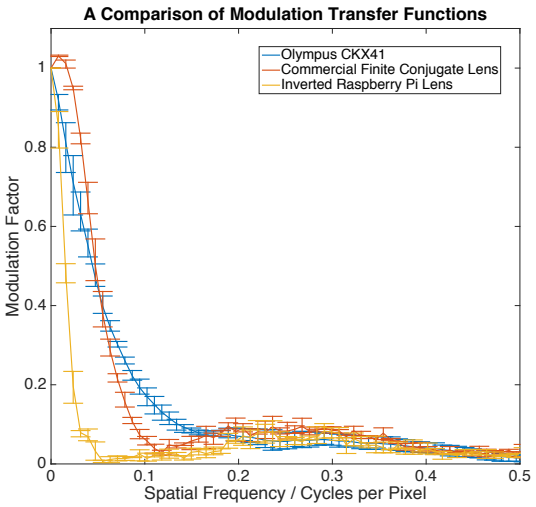


Figure 20: A comparison of the Modulation Transfer Function of various imaging systems. Shown here are the inverted Raspberry Pi lens design tested in this report, a commercial finite conjugate lens adapted for use in the system shown in figure 1, and a commercial Olympus system currently in use for live cell imaging in the Cambridge University Engineering Department Biolab. The MTF is a measure of spatial frequency response of an optical system, which shows the degree to which high frequency components such as small details and sharp edges are preserved after being transmitted through the system.

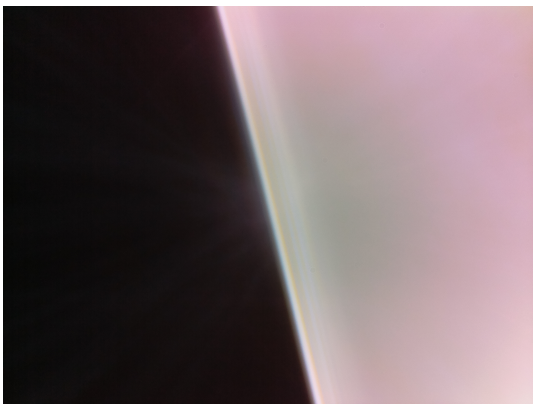


Figure 21: An image of an optical "step function", as used in the calculation of the resolution of the inverted Raspberry Pi lens imaging system. The step is formed by focussing on the polished straight edge of a silicon wafer, which allows the measurement of the spatial frequency response. It can be seen from the ringing observable around the edge that the optical system is lacking in high frequency support.

7 Conclusion and Further Work

In conclusion, a device has been designed, constructed and tested, which acts to bridge the gap between the latest advancements in low cost open source live cell imaging. By integrating a solution to the problem of mechanical sample positioning to a multiplexed design in a robustly scalable and cost efficient manner, the device adds significant value to the current propositions for users in terms of basic capability, ease of use and throughput of experiments. As part of this design, a novel contribution to the problem of multiplexed motor control has been presented in the form of the stackable motor control board, which allows direct control of the sample position and focus on a large number of stages from a single Raspberry Pi board. Alongside the control hardware, a basic framework for the communication and execution of commands has been presented, which has been designed to allow easy implementation of path following and closed loop control routines at the level of the Raspberry Pi control unit. In addition to being designed for functionality, the device has been designed with manufacturability and low resource consumption in mind, requiring only a low cost 3D printing unit, basic soldering equipment and parts which can be freely purchased or ordered in its production. As a consequence of this, a single module of the design has been shown to cost approximately half of the previous open source high throughput solution, the MultiPi, whilst offering vastly improved performance in terms of independent control over the images captured, ease of use, maintainability of the system, and scalability to implementations involving many units of the system for users requiring high amounts of parallel data.

The device has been shown to work as expected, being able to maintain standards of open loop stage positioning accuracy and resolution far in excess of that required to perform live cell imaging experiments, with the potential for the accuracy and repeatability to be improved with the addition of closed loop control, which would act to eliminate the effect of creep and backlash in the mechanical elements of the stages. Furthermore, the optical performance of the system has been examined in comparison to similar live cell imaging systems, in order to assist further decisions regarding the suitability of the current pathway for imaging specimens of interest, as well as to inform future pathway designs with a quantitative baseline of performance.

Whilst the design presented here has been shown to satisfy the requirements set out for it in terms of basic functionality, hardware efficiency and easy manufacturability, further work will be required to develop the control framework to such a stage that would facilitate the automatic discovery of sites of interest in live cell imaging applications. This step will require the application of a machine learning based computer vision system to the stage con-

trol framework in order to automatically direct the stages to explore the range of features available to them in the sample and identify the most interesting for further observation, thus allowing the user to perform many studies automatically.

8 Acknowledgements

I would like to thank Dr Alexandre Kabla of the Cambridge University Engineering Department and Dr Richard Bowman of the Cambridge University Department of Physics for their help in the development of this design, as well as Dr Richard Roebuck, Mr. Mark Huntsman and Mr. Adam Grieg of CUED for their assistance and advice in its production.

References

[1] Franze, K. *et al.* Neurite branch retraction is caused by a threshold-dependent mechanical impact. *Biophysical Journal* **97**, 1883–1890 (2009).

[2] Tambe, D. T. *et al.* Collective cell guidance by cooperative intercellular forces. *Nature Materials* **10**, 469–475 (2011).

[3] Blanchard, G. B. *et al.* Tissue tectonics: morphogenetic strain rates, cell shape change and intercalation. *Nature Methods* **6**, 458–464 (2009).

[4] Mayer, M., Depken, M., Bois, J. S., Jülicher, F. & Grill, S. W. Anisotropies in cortical tension reveal the physical basis of polarizing cortical flows. *Nature* **467**, 617–621 (2010).

[5] Stephens, D. J. & Allan, V. J. Light Microscopy Techniques for Live Cell Imaging. *Science* **300**, 82–86 (2003). URL `http://www.sciencemag.org/content/300/5616/82`.

[6] Papkovsky, D. B. (ed.) *Live Cell Imaging*, vol. 591 of *Methods in Molecular Biology* (Humana Press, Totowa, NJ, 2010). URL `http://link.springer.com/10.1007/978-1-60761-404-3`.

[7] Wellhausen, R. & Oye, K. A. Intellectual Property and the Commons in Synthetic Biology: Strategies to Facilitate an Emerging Technology. In *2007 Atlanta Conference on Science, Technology and Innovation Policy*, 1–2 (2007).

[8] Sommer, C. & Gerlich, D. W. Machine learning in cell biology - teaching computers to recognize phenotypes. *J Cell Sci* **126**, 5529–5539 (2013). URL `http://jcs.biologists.org/content/126/24/5529`.

[9] Doxzen, K. *et al.* Guidance of collective cell migration by substrate geometry. *Integrative Biology* **5**, 1026–1035 (2013). URL `http://pubs.rsc.org/en/content/articlelanding/201`

[10] Human Fertilisation and Embryology Authority, S. a. I. D. HFEA Code of Practice: Guidance Note 10. Embryo testing and sex selection (version 4.0). URL `http://www.hfea.gov.uk/496.html`.

[11] Wong, C., Chen, A. A., Behr, B. & Shen, S. Time-lapse microscopy and image analysis in basic and clinical embryo development research. *Reproductive BioMedicine Online* **26**, 120–129 (2013). URL `http://www.sciencedirect.com/science/article/pii/`

[12] Sharkey, J. P., Foo, D. C. W., Kabla, A., Baumberg, J. J. & Bowman, R. W. A one-piece 3d printed flexure translation stage for open-source microscopy. *Review of Scientific Instruments* **87**, 025104 (2016). URL `http://scitation.aip.org/content/aip/journal/rsi/`

[13] ivmech/ivport. URL `https://github.com/ivmech/ivport`.

[14] Burns, P. D. Slanted-edge MTF for digital camera and scanner analysis. In *Is and Ts Pics Conference*, 135–138 (SOCIETY FOR IMAGING SCIENCE & TECHNOLOGY, 2000). URL `http://www.losburns.com/imaging/pbpubs/26pics2000`

A Device Code

The code listed first is that used to program the motor controller boards, as described in §C.1. The second listing is a python script that can be run on the Raspberry Pi to check functionality of the boards once connected, and the third listing is a python script that can be run to allow the user to specify both the board and motion desired.

A.1 Motor Controller Board Code

```
//////Beta version of motor controller code, released 09/04/16////////
//////Author: Fergus Riche, fr293@cam.ac.uk//////////////////////////
// QueueArray creates a FIFO queue, StepperF is my modified version of the
//bundled stepper library that deals with half stepping and turns off all
//phases when motor is not moving to save power, WireS is the i2c library
//for the attiny 841
#include <QueueArray.h> //http://playground.arduino.cc/Code/QueueArray
#include <StepperF.h>
#include <WireS.h>     //https://github.com/orangkucing/WireS

// led is an indicator led attached to physical pin 2 of the attiny 841 SOIC
//define your own slave address for each motor axis with I2C_SLAVE_ADDR
//the motor has 2048 full steps per revolution; i.e. 4096 half steps
#define led                 0
#define I2C_SLAVE_ADDR      8
#define STEPS            4096
// create a queue of ints to act as the command buffer
// create a stepper attached to physical pins 10, 11, 12, 13
QueueArray <int> buffer;
Stepper stepper(STEPS, 7, 8, 9, 10);

//create variables for the current task, and for any values recieved over
//i2c
int currenttask = 0;
int receivedValue = 0;

// enable the led as an output, blink it at startup to show that everything
//is ok
//begin the i2c communication and define the action to be performed on
// reciept of data as an interupt routine
//set the speed of the stepper in rpm
void setup()
  {
  pinMode(led,OUTPUT);
  Blink(led);
  Wire.begin(I2C_SLAVE_ADDR);
  Wire.onReceive(transfer);
  stepper.setSpeed(10);
  }

// the loop runs ad infinitum and checks the status of the buffer
//if it contains commands, take the first one out of the queue and make it
//the current task for the stepper
//if not, do nothing for 0.1s
void loop()
   {
  if (!buffer.isEmpty())
       {
currenttask = buffer.dequeue();
    stepper.step(currenttask);
```

```
        }
    else
        {
54  delay(100);
        }
 }
// this function is called as an interrupt on reciept of a correct address
58 //byte on the i2c line
// bytes arriving in twos are converted into signed 16 bit ints and added
//to the queue, the check is there to prevent misalignment of bytes as a
//result of a misread
62 void transfer(size_t c)
  {
     if (Wire.available() == 2)
       {
66   int receivedValue = Wire.read() << 8 | Wire.read();
   buffer.enqueue(receivedValue);
        }
  }
70 // this function is there to make blinking a pin easier for debug purposes
//it contains delays, and cannot be used inside any other interrupts
void Blink(int pin)
  {
74   for(int i = 0; i <3; i++)
        {
     digitalWrite(pin,HIGH);
     delay (100);
78   digitalWrite(pin,LOW);
     delay (100);
        }
  }
```

A.2 Raspberry Pi Test Code

```
#########Test version of motor control software, released 09/04/16#######
#########Author: Fergus Riche, fr293@cam.ac.uk#################################
# smbus is the library that lets us create instances of i2c busses
4 import smbus

# the processor chip provides one accesible i2c bus, so create an instance
    using
#that and connect SDA to GPIO pin 3, and SCL to GPIO pin 5
bus = smbus.SMBus(1)
9
# write some bytes to every multiple of 8 from 1-12
#choosing these adresses is advantageous because they convert to hexadecimal
    well
#which makes human readout of i2c possible
# sending (10,10) corresponds to a request for 1290 steps- approximately 1/4
14 #of a revolution
for x in range(1,13):
        bus.write_byte_data(8*x,10,10)
```

A.3 Motor Drive Code

```
#########Useable version of motor control software, released 09/04/16#######
#########Author: Fergus Riche, fr293@cam.ac.uk#################################
# Run as > python motor_drive.py 'module_number' 'x_move' 'y_move' 'z_move'
4 # smbus is the library that lets us create instances of i2c busses
```

```
# sys lets us bring in command line arguments
import smbus
import sys

# select the module you require with the first argument
module = int(sys.argv[1])

# the address numbering scheme must be sequential multiples of 8
#i.e. module 1 has x at 8, y at 16 and z at 24
addr = 8 + ((module - 1) * 24)

# read in the arguments here and pass them to varables for convenience
x = int(sys.argv[2])
y = int(sys.argv[3])
z = int(sys.argv[4])

# the processor chip provides one accesible i2c bus, so create an instance
    using
#that and connect SDA to GPIO pin 3, and SCL to GPIO pin 5
bus = smbus.SMBus(1)

# as there is no data transfer protocol specified by I2C, we can use our own
#the bitwise operations split the ints specified in the argument into the two
#bytes of a signed 16 bit number for each axis
bus.write_byte_data(addr, x >> 8, x & 255)
bus.write_byte_data(addr + 8, y >> 8, y & 255)
bus.write_byte_data(addr + 16, z >> 8, z & 255)
```

B Device Cost Sheet

Costs given here are approximate, based solely on the costs incurred by the author during development, and do not take into account tooling costs or labor, though they do include consumable costs. As all materials were purchased at current market prices for small orders, it is expected that these figures represent an upper bound on the material cost of the devices which could be reduced in a production environment.

Item	Quantity Required	Cost/Item (GBP)
Motor Controller PCB	4	35.74
Motor Controller Components	4	30.62
Raspberry Pi 2 B+	1	34.01
Raspberry Pi Camera Module	4	22.2
IVMECH Camera Multiplexer	1	60.47
Camera Extension Cables	4	5.12
Stepper Motor x3	4	7.47
Microscope Multiplexer Stand	1	4.00
OpenFlexure Microscope	4	5.00
Complete System	N/A	**523.08**

C Protocols

C.1 To Program an ATtiny 841 Using an Arduino Uno

This protocol describes a method for programming the ATtiny 841 Atmel AVR microcontroller using an Arduino UNO and Arduino code. It can easily be extended to any other model in the ATtiny range, provided that the required board managers exist.

Materials Required: 1 Arduino UNO, 1 ATtiny 841 (pins broken out), 1 breadboard (if using), 6 jumper leads, 1 internet connected computer with the Arduino IDE version $\geqslant$1.6.7 installed.

1. Open the Arduino IDE preferences pane, and insert the URL of the board manager(s) you wish to use, which are required to compile and upload code to the ATtiny chips, in the *Additional Boards Manager*

URLs field. Multiple URLs must be separated by commas. As of 09/04/16, the following URLs have working managers for the ATtiny 841. (https://raw.githubusercontent.com/damellis/attiny/ide-1.6.x-boards-manager/package_damellis_attiny_index.json, http://drazzy.com/package_drazzy.com_index.json)

2. Open the *Boards Manager* pane from *Tools, Board* and scroll to the bottom of the list to find the newly available boards to install.

3. Select the Arduino Uno from *Tools, Board* and open the ArduinoISP example. Upload it to the UNO and insert a 10μF capacitor across the reset and ground pins of the UNO, making sure that the negative terminal of the capacitor is connected to the ground pin, if the capacitor is polarised. The UNO will now act as a programmer for the target chip.

4. In the *Tools* menu, set *Board* to ATtinyx41, *B.O.D* to B.O.D Disabled, *Chip* to ATtiny841, *clock* to 8MHz Internal, and *Programmer* to Arduino as ISP.

5. Connect the pins of the chips as shown in table 1

6. In the *Tools* menu, click *Burn Bootloader* and wait for confirmation of sucess from the IDE output.

7. Write and upload the sketch to the ATtiny chip in the usual manner. Note that to prepare the IDE for programming a different board, step 4 must be repeated, to match the identity of the new board.

Arduino Pin	ATtiny 841 SOIC14 Physical Pin	ICSP Pin Designation
GND	14	Ground
5V	1	Vcc
10	4	Reset
11	7	MOSI
12	8	MISO
13	9	SCK

Table 1: A connection table for programming ATtiny series microcontrollers with an Arduino UNO

C.2 To reflow solder using hot plates

This protocol describes a method for reflow soldering of SMD components on small to medium sized (0-100mm square) PCBs. It is particularly suited for boards containing plastic connectors that can be damaged in infra-red lamp based reflow ovens, as well as boards that are not soldering properly in reflow ovens, as the dominant mode of heat transfer is conduction, and the reflow process can easily be observed as it progresses.

Materials Required: 1 PCB, 1 set of components to solder, solder paste, 2 hotplates capable of maintaining temperatures of $50°C \leqslant T \leqslant 400°C$, 1 steel or aluminium plate large enough to support your largest PCB with a border of 20mm surrounding the PCB (the thickness should be between 1 and 2 mm), 2 pairs of spade tipped tweezers, 1 pair safety glasses, 1 pair heat resistant gloves.

1. Set one hotplate next to the other and ensure the tops are clean and free from distortion. Turn one on to $100°C$, and the other to $350°C$. Allow ample time for them to come to temperature

2. Apply solder paste using either a stencil or an applicator syringe. A stencil is recommended due to the higher precision of paste deposition it offers, as well as the superior evenness of the applied paste thickness.

3. Place components on the board, noting that slight errors in the positions of the components will generally be corrected by the surface tension of the solder at reflow.

4. Wearing the safety gloves and glasses, position the metal plate on the $100°C$ plate, and allow it to come to temperature. Note that thermal distortion of the plate may initially occur, which can be corrected by pressing down on the corners of the plate with the tweezers.

5. Once the plate has come to temperature, place the PCB on it and wait for up to two minutes for it to heat up. During this time, thermal distortions can be corrected by pushing gently on the corners of the PCB, in the same manner as step 4. The solder paste may change slightly in appearance, and the flux may activate.

6. Once the PCB has come to temperature on the $100°C$ metal plate, transfer the PCB on the metal plate to the $350°C$ hotplate. Correct thermal distortions as before, and observe the pads of the PCB carefully. The solder paste should begin to melt within 2 minutes, and be fully molten within 3. Any pads that do not reflow within 3 minutes can be corrected by hand using a rework station. Take care also that the silkscreen is not scorched before the PCB reflows; if this occurs, then a redesign of the board may be necessary.

7. Transfer the PCB and the metal plate back to the $100°C$ hotplate for approximately one minute and then to a wire rack. This step is to avoid thermal shock to the components. Once the PCB has cooled to around $50°C$, the metal plate can be safely extracted.

8. Repeat steps 1-7 with all further PCBs to be soldered, then turn off the hotplates and allow them to cool. Wash the PCBs with isopropanol and allow to dry.

D Risk Assessment Review

Table 2 shows the risk assessment performed for the project. It lists the hazards identified during the course of the project, as well as the control measures taken to minimise their associated risks. The safety procedures listed here were followed rigorously and no accidents or injuries were recorded during the course of the project. It can therefore be concluded that the risk assessment procedure provided a fair and accurate understanding of the hazards involved in the project, and that the measures taken to mitigate these hazards were sufficient.

Hazard	Effect	Control Measures	Residual Risk
$220°C$ extrusion nozzle and $100°C$ bed on 3D printers	Burns	Use heat resistant gloves when touching the machine in operation. Do not touch the extrusion nozzle directly	Low
Class IV laser in Laser cutter	Burns, damage to eyes if viewed directly	Interlocks are present on the cutter to prevent operation of the laser when the guards are raised	Low
Particulate matter formed in laser cutter when cutting acrylic	Unknown, possible damage to respiratory system upon prolonged inhalation	Machines are vented to particulate filters during operation	Low
$350°C$ and $100°C$ hotplates used in reflow soldering	Burns	Use heat resistant gloves whenever the hotplates are turned on. Only handle materials at high temperature with tweezers	Low
Splatters of solder occurring during hand or reflow soldering	Damage to eyes	Use safety glasses to ensure eyes are well protected during soldering	Low
Isopropanol	Irritation of skin, respiratory system upon prolonged exposure	Use only in a well ventilated environment, wash hands after use	Low
Extru-Fix Acrylic Adhesive	Poisoning if swallowed	Use only in a well ventilated environment, wash hands after use and do not eat or drink whilst using the substance	Low

Table 2: The risk assessment for the project.