Hardware Article

Design and implementation of a low cost, modular, adaptable and open-source XYZ positioning system for neurophysiology

Thomas Campbell *, James F.X. Jones

School of Medicine, University College Dublin, Ireland

ARTICLE INFO

Article history:
Received 1 July 2019
Received in revised form 15 January 2020
Accepted 20 January 2020

Keywords:
Neurophysiology
Mechanotransduction
XYZ positioning system
Automated microscopy
Raspberry Pi
Arduino

ABSTRACT

In recent years, open-source 3D printing technologies have become increasingly applied to biological research. We have created a fully open-source, versatile and low cost XYZ positioning system using 3D printer components. As this system is controlled by a Python3 based operating system running on a Raspberry Pi 3 Model B, its behaviour can be adapted to meet multiple needs in neurophysiology. We have developed two main applications of this system. First, we have created an automated microscopy script that links seamlessly with image stitching plugins in ImageJ (Fiji) allowing the user to create high resolution montages. Second, we have created a series of movement scripts allowing the application of graded rates of stretch to muscle spindles. Here we outline the construction and implementation of this system and discuss how we have utilised this tool in our research.

© 2020 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Specifications table:

| Hardware name          | XYZ Positioning System                          |
|------------------------|-------------------------------------------------|
| Subject area           | Please select the subject area most relevant to the original community for which this hardware was developed |
|                        | • Neuroscience                                  |
|                        | • General                                       |
|                        | • Biological Sciences (e.g. Microbiology and Biochemistry) |
| Hardware type          | Imaging tools                                   |
|                        | • Mechanotransduction                           |
| Open Source License    | CC-BY-NC-SA (Creative Commons 4.0)              |
| Cost of Hardware       | $600–$1200                                      |
| Source File Repository | https://osf.io/sju9d/                           |

Abbreviations: 3D, Three Dimensional; AC, Alternating Current; CNC, Computed Numerical Code; DC, Direct Current; EMI, Electromagnetic Interference; FDM, Fused Deposition Modelling; FFF, Fused Filament Fabrication; GPIO, General-Purpose Input/Output; IDE, Integrated Developer Environment; LCD, Liquid Crystal Display; NEMA17, National Electrical Manufacturers Association (stepper motor with faceplate dimensions of 1.7 × 1.7 in.); OpenCV, Open Computer Vision; PLA, Polylactic Acid; PVA, Polyvinyl Acetate; RAMPS 1.4, Repap Arduino Mega Pololu Shield (version 1.4); SD Card, Secure Digital Card; STEM, Science, Technology, Engineering and Mathematics; STL, Stereolithography; USB, Universal Serial Bus; UTF-8, Unicode Transformation Format, 8-bit blocks; VAT, Value Added Tax.

* Corresponding author.
E-mail address: thomas.campbell@ucd.ie (T. Campbell).

https://doi.org/10.1016/j.ohx.2020.e00098
2468-0672 © 2020 The Authors. Published by Elsevier Ltd.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).
1. Hardware in context

The ability to relate coordinates to a data set can be a powerful tool in the hands of researchers and permits investigators to map parameters of interest. In the field of microscopy, this is usually carried out through the automation of stage movement which ultimately serves to move the object of interest with respect to a sensor. Such an approach has recently been described in a variety of forms [1–5] where stage movement is a recurring design principle. However, depending on the intended application, stage automation may be impractical. Small form 3D printed three-axis micromanipulators have been described [6] however such implementations naturally offer limited travel distance. Zhang et al. (2016) have previously described the creation a XYZ positioning system fashioned from a delta-form 3D printer [7]. This format overcomes the axis dimension limitations of small form micromanipulators [6] and maintains a stationary stage though at the upper extremities of its Z-axis the range of positions available to users tapers off in a conical shape.

We have created an XYZ positioning system which moves a sensor or probe with respect to an object of interest, similar to that presented by Zhang et al. (2016). However, we have constructed this device with a format resembling a gantry crane, which has permitted the creation of a system with large working dimensions (430 × 430 × 150 mm). Unlike other XYZ positioning systems, the axis dimensions of our design can easily be tailored (hardware and firmware) to suit a particular use case scenario. This system is controlled by a Raspberry Pi 3 with standardised, open-source, and affordable FFF/FDM 3D printing technologies (Arduino Mega, RAMPS 1.4 motor shield and NEMA17 bipolar stepper motors). Stepper motors perform precise rotary movements. When coupled with a timing belt or a lead screw, these precise rotations translate to accurate linear movements along an axis (most commonly X, Y and Z axes). The Arduino Mega runs open-source Marlin firmware which has been used in other automated laboratory devices [8,9] and translates G-code (the control-language for 3D printers) into precise movements along a specified axis.

Though the Arduino Mega/RAMPS 1.4 combination is capable of executing a complex series of movements, the Arduino Mega’s low processing power and memory capacity are ill suited for image capture and real time image analysis. For this reason, we have paired a Raspberry Pi 3 and an Arduino Mega in a master – slave arrangement. The Raspberry Pi 3 is a versatile and affordable microcontroller and it can easily be programmed using the Python3 programming language. Our system utilises the Raspberry Pi 3 to calculate and stream G-code movement commands to the Arduino Mega whilst at the same time performing image capture.

The use of a Raspberry Pi 3 has permitted the incorporation of The Open Computer Vision library (OpenCV) into this build which sets it apart from other XYZ positioning systems. OpenCV is a popular open-source machine learning and computer vision software library containing over 2500 optimised functions relating to machine learning and computer vision [10,11]. Our system permits users to combine these powerful features of OpenCV with automated movement. We have created one such hybrid application in the form of an automated microscopy scanning tool which we outline later in this manuscript.

Otherwise, in our investigations into mechanotransduction, the process by which a mechanical stretch is converted to an action potential, we have applied this system to an ex-vivo preparation of the rat muscle spindle and investigated the activation-thresholds (stretch distance, velocity and acceleration) for single unit spindle afferents. We believe that this system will change how mechanotransduction is investigated as it permits the assessment of activation-threshold shifting in response to a change of state i.e. temperature, pH or pharmacological agonism/antagonism. In our future work we plan to use this application to investigate mechanically sensitive neurons in the rat heart.

2. Hardware description

The frame is constructed mainly of 20 × 20 mm aluminium extrusion bolted together with brackets and 3D printed components. Three-dimensional positioning is achieved with the use of components commonly found in 3D printers (stepper motors, timing belts, lead screws, linear rods and linear rails). Resembling a gantry crane, the X-axis is driven by a timing belt whereas the Y and Z axes are each driven by two lead screws. Movement is controlled by an Arduino Mega with RAMPS 1.4 motor shield, which is commonly used to drive 3D printers. The Arduino itself is controlled by a Raspberry Pi 3. Arranging the microcontrollers in this master-slave configuration permits the automation of complex movement paradigms through the Python3 programming language. The power source for the system depends on the intended use case. For neurophysiology a linear regulated 12 V DC power supply must be used to ensure low EMI however for other applications a 12 V DC switching power supply suffices. The total weight of this system is 8.25 kg (excluding power supply unit).
3. Design files

All files are available to access through the open Science Framework.

| Design file name                          | File type                          | Open source license                                      | Location of the file   |
|-------------------------------------------|------------------------------------|-----------------------------------------------------------|------------------------|
| Fusion 360 Master File for Online Sharing | CAD file (.f3d format)             | Not open-source, but Autodesk Fusion 360 free license is available to students, educators, and hobbyists for non-profit usage | https://osf.io/sju9d/   |
| STL Files for 3D Printed Components       | ZIP folder containing Stereolithography (STL) files | NA                                                        | https://osf.io/sju9d/   |
| Marlin Firmware for XYZ Positioning System| ZIP folder containing Firmware Files | Open-source, license unspecified                         | https://osf.io/sju9d/   |
| XYZ_Operating_System.py                   | Python3 Script                     | Open-source (Python License (Python-2.0))                | https://osf.io/sju9d/   |

3.1. Fusion 360 master file for online sharing

This file includes a full 3D model of the XYZ movement system and was designed in Autodesk Fusion 360 which is available for free for non-profit use. Users can import this file into Fusion 360 and make adjustments to the system to suit their intended use case.

3.2. STL files for 3D printed components

This zipped folder contains the necessary stereolithography files for the build. Users can mass print the files by print one copy each of Plates 1 & 2. Alternately users have access to the STL file for each component should they prefer to print the parts individually.

3.3. Marlin firmware for XYZ positioning system

This zipped folder contains an adjusted version of Marlin Firmware which is appropriate for normal use with the XYZ positioning system’s hardware configuration.

3.4. XYZ_operating_system.py

This file should be placed on the desktop of the Raspberry Pi 3 and should be launched from the terminal.

4. Bill of materials

5. Build Instructions

5.1. 3D printing: calibration of 3D printer

Our 3D printer (Prusa i3 MK3, firmware 3.5.1) was calibrated for dimensional accuracy using the built-in calibrate XYZ function. Using Autodesk Fusion360, a 20 × 20 × 20 mm calibration cube was modelled (Fig. 1A) and three such cubes were subsequently printed in polylactic acid (PLA) filament (Fig. 1B). The Prusa edition of “Slic3r” (version 1.41.2) was used to slice these models and generate a G-code file. Upon completion, the X, Y and Z dimensions were measured six times for each cube using digital callipers (Fig. 1C, Mitutoyo Absolute AOS Digimatic). Calibration cubes were within margin of error (Fig. 1D) and hence the required components were printed. The settings used for these prints are outlined in Fig. 1E.

5.2. 3D printing: printing of components

For the necessary 3D printable stereolithography files, see Appendix A. As our printer had a low failure rate, we opted to print one copy of “Plate 1” and “Plate 2”. Components were sliced using the Prusa Edition of “Slic3r” (version 1.41.2) and printed using a Prusa i3MK3 (firmware version 3.5.1). All parts were printed in 1.75 mm diameter polylactic acid (PLA) fil-
5.3. Y-axis carriage assembly

Two 610 mm and two 317 mm aluminium extrusions were arranged as shown in Fig. 2A. Each of the four internal corners were reinforced with a corner brace, two 5 mm hex bolts and two T-nuts. The quadrilateral formed in Fig. 2A was further reinforced with four right-angled brackets and 5 mm hex bolts with T-nuts (five at each corner, Fig. 2B). Two motor brackets and two idler brackets were mounted on the 610 mm aluminium extrusion using two 5 mm hex bolts and T-nuts per bracket (Fig. 2C). The outer margin of each bracket was placed adjacent to the border of the 610 mm extrusion. Two 400 mm × 8 mm linear rails were mounted to the 610 mm aluminium extrusion (four 5 mm hex bolts & T-nuts per linear rail) and spaced 3 mm from the inner margin of the motor and idler brackets.

Two linear rail carriages were mounted to a Y-axis carriage bracket with six 5 mm hex bolts (Fig. 2D). T-nuts were not required for the step shown in Fig. 2D as the 5 mm hex bolts screwed directly into the linear rail carriages. In addition, two corner braces were mounted to the Y-axis carriage assembly at the positions shown (Fig. 2D). A small piece of aluminium extrusion (88 mm) was mounted to the corner braces outlined in the previous step using two T-nuts (Fig. 2E). Two corner braces were mounted on top of the 88 mm aluminium extrusion using two 5 mm hex bolts and T-nuts (Fig. 2E). These corner braces were spaced 20 mm apart. A single T-bracket was mounted to the aluminium extrusion using five 5 mm hex bolts and three T-nuts (Fig. 2E). The steps described in Fig. 2D and 2E were repeated as two Y-axis carriage assemblies were required for the build.

Both of the completed Y-axis carriages were slid onto a linear rail (Fig. 2F) orientated such that the T-brackets faced the outside of the assembly. The Y-axis end-stop bracket was mounted to the 610 mm aluminium extrusion with two 5 mm hex bolts and T-nuts and pressed against the adjacent linear rail as shown in Fig. 2F. A large limit switch was mounted to this end-stop bracket using two M3 × 16 mm hex bolts M3 nuts.

A single radial ball bearing was pressed into its corresponding impression on the idler bracket (Fig. 2F). This was repeated for the other idler bracket. Two NEMA17 bipolar stepper motors were mounted to the motor brackets using four M3 × 10 mm hex bolts (Fig. 2G). The 5 mm end of a 5 to 8 mm flexible shaft coupling was mounted to each of the motor shafts (Fig. 2G). These couplings were only advanced far enough along the motor shaft such that the first set of grub screws engaged the motor shaft when tightened.

The following steps were repeated for both the left and right side of the Y-axis assembly. A 300 × 8 mm lead screw was fed through the central aperture in the idler bracket and through the central aperture in the limb of the Y-axis carriage bracket and advanced by approximately 100 mm (Fig. 2H). An anti-backlash lead screw nut was wound onto the 8 mm lead screw and attached to the Y-axis carriage bracket using four M3 × 16 mm hex bolts and four M3 nuts (Fig. 2H). The 8 mm lead screw was advanced into the 8 mm aperture of the flexible shaft coupling and was secured by tightening the grub screws in the coupling (Fig. 2H). This completed assembly of the Y-axis (Fig. 2I).

5.4. X & Z axes assembly

One 590 mm and two 730 mm aluminium extrusions were arranged as shown in Fig. 3A. Each internal corner was reinforced using a corner brace, two 5 mm hex bolts and two T-nuts. These corners were further reinforced with a right-angled bracket, five 5 mm hex bolts and five T-nuts at each corner (Fig. 3B). Two motor brackets were mounted to the corners of the frame using three 5 mm hex bolts and three T-nuts for each motor bracket (Fig. 3C). Two NEMA17 bipolar stepper motors were mounted to the motor brackets using four M3×10 mm hex bolts for each motor (Fig. 3D). The 5 mm end of a 5 to 8 mm flexible shaft coupling was mounted to each of the motor shafts as per Fig. 3D. These couplings were advanced along the motor shaft until the first set of grub screws could engage the motor shaft when tightened.

A NEMA17 bipolar stepper motor was attached to the left XZ splitter bracket using three M3 × 16 mm hex bolts (Fig. 3E). A 2GT timing pulley was mounted on the motor shaft (Fig. 3E). The toothed portion of the timing pulley was positioned such that it was in line with the hollow chamber running through the left XZ splitter bracket. Two LM8UU linear slide bearings were inserted into the left XZ splitter, one from above and one from below (Fig. 3E). A small limit switch was mounted to the underside of the left XZ splitter using two M2 × 12 mm bolts (Fig. 3E). These bolts were tightened gently to avoid damage to the left XZ splitter. An 8 mm lead screw nut was attached to the top of the left XZ splitter using two M3 × 16 mm hex bolts and two M3 nuts. No anti-backlash mechanism was required as the Z-axis was actuated by gravity.

Two LM8UU linear slide bearings were inserted into the right XZ splitter (one in the top and one in the bottom, see Fig. 3E). An 8 mm lead screw nut was mounted to the top of the right XZ splitter using two M3 × 16 mm hex bolts and two M3 nuts (Fig. 3F). A GT2 idler pulley was inserted into the hollow channel within the right XZ splitter and secured in place using a M3 × 20 mm hex bolt and M3 nyloc nut (Fig. 3F). Two 600 × 8 mm linear rods were inserted all of the
way into the 8 mm apertures in the side of the left XZ splitter. Three LM8UU linear slide bearings were slid onto the 600 × 8 mm linear rods (two on the top and one on the bottom, Fig. 3G). The ends of the 600 × 8 mm linear rods were inserted all the way into the 8 mm apertures in the left and right XZ splitter brackets (Fig. 3G).

Two 600 × 8 mm linear rods were pressed firmly into the outer 8 mm apertures in the Z-axis motor brackets (Fig. 3H) and the assembly from Fig. 3G was slid (green arrows) onto these linear rods. An idler bracket was mounted on the 730 mm extrusion (using two 5 mm hex bolts and two T-nuts) such that its 8 mm aperture was fully pressed against the exposed end of a Z-axis linear rod (Fig. 3I). A radial ball bearing was pressed into the circular impression in the idler bracket and the assembly from Fig. 3G was slid (green arrows) through the 8 mm apertures in the side of the left XZ splitter. Three LM8UU linear slide bearings were slid onto the 600 × 8 mm linear rods (two on the top and one on the bottom, Fig. 3G). The ends of the 600 × 8 mm linear rods were inserted all the way into the 8 mm apertures in the left and right XZ splitter brackets (Fig. 3G).

Two 600 × 8 mm linear rods were pressed firmly into the outer 8 mm apertures in the Z-axis motor brackets (Fig. 3H) and the assembly from Fig. 3G was slid (green arrows) onto these linear rods. An idler bracket was mounted on the 730 mm extrusion (using two 5 mm hex bolts and two T-nuts) such that its 8 mm aperture was fully pressed against the exposed end of a Z-axis linear rod (Fig. 3I). A radial ball bearing was pressed into the circular impression in the idler bracket and the assembly from Fig. 3G was slid (green arrows) through the 8 mm apertures in the side of the left XZ splitter. Three LM8UU linear slide bearings were slid onto the 600 × 8 mm linear rods (two on the top and one on the bottom, Fig. 3G). The ends of the 600 × 8 mm linear rods were inserted all the way into the 8 mm apertures in the left and right XZ splitter brackets (Fig. 3G).

Two 600 × 8 mm linear rods were pressed firmly into the outer 8 mm apertures in the Z-axis motor brackets (Fig. 3H) and the assembly from Fig. 3G was slid (green arrows) onto these linear rods. An idler bracket was mounted on the 730 mm extrusion (using two 5 mm hex bolts and two T-nuts) such that its 8 mm aperture was fully pressed against the exposed end of a Z-axis linear rod (Fig. 3I). A radial ball bearing was pressed into the circular impression in the idler bracket and the assembly from Fig. 3G was slid (green arrows) through the 8 mm apertures in the side of the left XZ splitter. Three LM8UU linear slide bearings were slid onto the 600 × 8 mm linear rods (two on the top and one on the bottom, Fig. 3G). The ends of the 600 × 8 mm linear rods were inserted all the way into the 8 mm apertures in the left and right XZ splitter brackets (Fig. 3G).

The X-Carriage was placed firmly overlying the three LM8UU linear slide bearings and secured with six small zip ties (Fig. 3K). A 2GT timing belt was looped around the toothed 2GT timing pulley (left XZ splitter) and pulley idler (right XZ splitter). The loose ends of the timing belt were then looped around the back of the X-carriage (Fig. 3K). It is necessary that
Fig. 1. Calibration of 3D Printer. (A) Digital model of calibration cube created using Autodesk Fusion 360. (B) Completed print of three calibration cubes before their removal from the build plate. (C) Measurement of calibration cubes using a digital micrometer. (D) Dimensional accuracy of calibration cubes in X, Y and Z axes. Six measurements were taken for each axis per cube, totalling 18 measurements for X, Y and Z axes. For the X-axis, the mean was 20 mm (min 19.97 mm, maximum 20.04 mm, SD 0.02). For the Y-axis the mean measurement was 20 mm (min 19.97 mm, max 20.02 mm, SD 0.02). For the Z Axis, the mean was 20.04 mm (min 20.0 mm, max 20.07 mm, SD 0.02. (E) Details of print settings used to print the calibration cubes.

| Setting               | Value                      | Reason                                      |
|-----------------------|----------------------------|---------------------------------------------|
| Skirt                 | Enabled for first layer    | For first layer, ensures nozzle is primed   |
| Height of first layer | 200μm                      | Improves adhesion of first layer            |
| Subsequent layer height | 150μm                   | Optimal balance of print quality and print speed |
| Nozzle Temperature    | 220°C first layer, 215°C every subsequent layer | A hotter first layer permits more effective bed adhesion |
| Bed Temperature       | 60°C                       | Improves bed adhesion, particularly for objects which have a large surface area |
| Printing Travel Speed | 150mm/sec                  | Standard printing speed                     |
| First layer Speed     | 30% of normal value        | Improves adhesion of first layer            |
Fig. 2. Y-Axis Assembly. (A–I) Step by step guide to Y-axis assembly. All images generated using Autodesk Fusion 360.
the timing belt need be taut, which necessitated shortening of the belt using a scissors. A small limit switch was mounted on the Z-axis end-stop bracket using two M2x12mm bolts and two M2 nuts. This bracket was then be mounted on the top left of the Z-axis using two M5 hex bolts and two T-nuts (Fig. 3L). The X&Z axis assembly and the Y-axis assembly were carefully joined by lowering the bottom part of the 730 mm aluminium extrusion limbs into the 20 mm space on each Y-axis carriage (Fig. 3M). Each limb of the Z-axis was then secured using two 5 mm hex bolts and T-nuts on the T-bracket as well as one...
5 mm hex bolt and T-nut from their adjacent corner braces of the Y-axis carriage (Fig. 3M). This completed the assembly of the XYZ positioning system (Fig. 3N).

5.5. Axis alignment

Correct alignment of the axes is critical for normal function of the XYZ system as axis misalignment can result in the system being unable to move freely. The Y-axis linear rails were loosened, and the Y-axis lead screws wound manually until the Y-axis carriages touched the Y-axis motor brackets. For each linear rail, the 5 mm hex bolt nearest the Y-axis motor was half tightened. The Z-axis lead screws were wound manually until the upper edges of the left and right XZ splitter brackets were equidistant from the Z-axis motor brackets. With this alignment the X-axis carriage could travel along the full length of the
X-axis without resistance and this completed the X & Z axis alignment. The Y-axis lead screws were wound manually until both Y-axis carriages are almost at the end of their travel, and equidistant from their respective idler brackets. In this position all Y-axis linear rail hex bolts were fully tightened, completing alignment of the Y-axis.

5.6. Electronics and wiring

The RAMPS 1.4 shield was carefully seated on top of the Arduino Mega (Fig. 4A). Fifteen micro stepper jumpers (three per stepper motor) were installed on top of the RAMPS 1.4 shield (Fig. 4B), enabling 1/16th micro stepping mode for the X, Y and Z stepper motors. Five A4988 stepper motor drivers were installed on top of the RAMPS 1.4 shield (Fig. 4C). Correct orientation of the stepper motor drivers is necessary as incorrectly installed stepper motor drivers can cause permanent and irreversible damage to the RAMPS 1.4 shield. With the RAMPS 1.4 shield oriented as in Fig. 4C, the adjustable potentiometer for each stepper motor driver was on the right of each stepper motor driver. The LCD screen was connected to the RAMPS 1.4 motor shield by first seating the smart adapter module on the row of pins on the rear of the RAMPS 1.4 shield (Fig. 4D). Then two ribbon cables were used to connect the EXP1 & 2 ports on the smart adapter module to their corresponding ports on the rear of the LCD screen (Fig. 4D).

**Fig. 4.** Wiring of XYZ System. (A) RAMPS 1.4 shield (top) and Arduino Mega (bottom). (B) RAMPS 1.4 shield and microstepping jumpers (top). RAMPS 1.4 shield with microstepping jumper pins installed (bottom). Note, to enable 1/16 microstepping for each stepper motor, it is necessary to install three jumpers per motor as encircled. (C) A4988 stepper motor drivers shown individually (top) and installed on RAMPS 1.4 shield (bottom). (D) Connecting the LCD screen to the RAMPS 1.4 shield. First the smart adapter module is seated on the pins at the end of the RAMPS 1.4 shield. Next, EXP1 and EXP2 on the smart module should be connect to their corresponding ports on the reverse of the LCD screen. (E) The Arduino Mega and Raspberry Pi 3 can be connected over USB using a type A male to type B male connector. (F) Wiring of limit switches and stepper motors to RAMPS 1.4 shield. Note both the colour orientation for stepper motor wiring and the highlighted pins for limit switch wiring.
The Arduino Mega was connected to the Raspberry Pi 3 using a USB cable (type A-Male to B-Male). This cable was included in the Arduino Mega/RAMPS 1.4 kit listed in Table 1. The USB type B male connector was inserted into the Arduino Mega and the type A male connector into one of the USB ports on the Raspberry Pi 3 (Fig. 4E). The X, Y and Z stepper motors were wired as shown in Fig. 4F. Insulated copper wire was soldered to the limit switch electrical terminals (green circles, Fig. 4F). These wires were subsequently connected to the $X_{\text{min}}$, $Y_{\text{max}}$ and $Z_{\text{max}}$ limit switch pins on the RAMPS 1.4 shield (Fig. 4F) using female Dupont connectors (limit switch terminals can be connected to either of their respective pins on the RAMPS 1.4 shield). Care was taken when wiring power to the RAMPS 1.4 shield as incorrect polarity can permanently damage the microcontroller (see Fig. 4F). Links to more detailed information regarding the assembly of these electronic components is provided in Appendix A.

5.7. Firmware configuration

The Arduino IDE (Integrated Development Environment) is an open-source software platform developed and supported by the Arduino company. The Arduino company also designs and produces open-source hardware and have produced microcontroller boards since 2005 [12]. The Arduino IDE is used to program Arduino microcontroller boards to allow individuals in
the STEM fields to create working electronic prototypes. The Arduino IDE (version 1.8.8) was downloaded (see Appendix A) and used to edit, compile and upload firmware to our Arduino Mega microcontroller. Marlin is an open-source firmware developed for 3D printers [13] and allows the Arduino Mega to translate G-code commands to movement. Marlin firmware has been utilised in novel open-source laboratory devices such as in gamma-ray spectroscopy sampling [9] and microsyringe autosamplers [8].

5.8. Preparation of Marlin firmware using the Arduino IDE

Marlin firmware (version 1.1.9) was downloaded (see Appendix A) and edited using the Arduino IDE. From the downloaded Marlin firmware files, the file “Marlin.ino” was opened using the Arduino IDE. When Marlin firmware was opened by the Arduino IDE, a large array of tabs could be viewed along the top of the window.

The U8glib library is a graphical library required for the correct function of the LCD screen attached to the RAMPS 1.4 shield. Installation of this library was carried out using the Library Manager (Tools > Manage Libraries). Within the Library Manager window, “U8glib” was entered into the search bar and from the search results, U8glib version 1.19.1 (author Oliver) was installed.

Changes were made to the Marlin, Configuration.h and Configuration_adv.h tabs. The changes made included adjustments to numerical values and commenting/uncommenting specific lines of code to ensure that the configuration of our XYZ positioning system was correctly reflected in the firmware. Within Marlin firmware, commented lines have a “//” at the start of the line and uncommented lines do not. Lines that begin with “/” are not executed by the microcontroller as it reads the firmware. Lines that do not begin with “/” are executed by the microcontroller. The specific adjustments made to the Marlin, Configuration.h and Configuration_adv.h tabs are outlined in Table 2.

5.9. Uploading Marlin firmware to the Arduino mega

Using a type A male to type B male USB cable, the Arduino Mega was connected to a computer. The Arduino IDE application was opened and from the Tools drop-down menu, the board type was specified as “Arduino/Genuino Mega or Mega 2560”. Next, within the Tools drop-down menu, the processor was specified as “ATmega2560 (Mega 2560)” and the correct USB port was selected (as there was only one Arduino connected to the computer then was only one port listed). Verify/Compile was selected from the Sketch drop-down menu to flag any errors that may have arisen as a result of the previous changes carried out to the Marlin, Configuration.h and Configuration_adv.h tabs. Verification and compilation of the firmware took approximately one to two minutes. Once the firmware had been verified and compiled successfully, the Upload option was selected from the Sketch drop-down menu. This process also took approximately two minutes to complete. It is of critical importance not to unplug the Arduino Mega from the computer during the uploading process as a loss of power during this process can irreversibly damage the microcontroller. Once the Arduino IDE had completed the upload, it reported “Done Uploading” to the user.
Figure A: Image of the measurement setup.

Figure B: Bar chart showing the mean end stop deviation (µm) for X, Y, and Z axes. The values are as follows:
- X-Axis: 61.0
- Y-Axis: 2.1
- Z-Axis: 2.2

Figure C: Bar chart showing the mean point-to-point deviation (µm) for X, Y, and Z axes. The values are as follows:
- X-Axis: 65.3
- Y-Axis: 17.6
- Z-Axis: 6.2

Figure D: Bar charts showing the deviation (µm) for X, Y, and Z axes in Y, Z, and X movements respectively.
5.10. Initial setup of the Raspberry Pi 3 & OpenCV

Raspbian Stretch (Raspbian GNU/Linux 9.8) operating system was installed on the microSD card. Adrian Rosebrock’s guide was used to correctly install Open Computer Vision (OpenCV) on the Raspberry Pi 3 [14]. It is critically important to follow this guide exactly as failure to do so will result in the inadequate installation of OpenCV, the failure of any OpenCV functions called by Python3 scripts and ultimately impact the Automated Camera Scanning application (discussed later).

5.11. Creation of a terminal based operating system for XYZ positioning system

Python3 is an open-source and well-supported, modern programming language and runs on affordable hardware such as the Raspberry Pi 3. The Raspberry Pi 3 and Arduino Mega were paired in a master-slave type configuration. Within the Python3 environment, G-Code strings were calculated, converted to byte format and streamed to the Arduino Mega via USB connection to bring about finely controlled movements. The specifics of streaming G-code from the Raspberry Pi 3 to the Arduino Mega are further described in Appendix C. A copy of our novel operating system is freely available to download (see Appendix A) and can be viewed and edited using a Python3 script editing program such as Thonny.

6. Operation instructions

Once all hardware and software setup is complete, the file “XYZ_Operating_System.py” should be placed on the Raspberry Pi 3 desktop. The XYZ operating system should be launched by opening the Terminal and then entering the four commands listed in Fig. 5A. Each command should be entered in sequence (from top to bottom) as this ensures that the OpenCV library of functions is available for use by the XYZ operating system.

Once launched, users should follow the onscreen instructions in the terminal window to choose the application of their choice. Users can enter “1” to move the system with typed G-Code commands or “2” to enter the automated scanning section. “3”, “4” and “5” pertain to varying repeated stretch algorithms used in our mechanotransduction investigations (Fig. 5B). Once a user specifies their preferred application, further on-screen instructions guide the user. We have included three videos to demonstrate the use of this device (see Appendix D or source file repository).

[Video #1 – Launching XYZ_Operating_System]
[Video #2 – Lumbrical muscle stretching, fixed kinetic parameters]
[Video #3 – Lumbrical muscle stretching, increasing and decreasing stretch distance]

7. Validation and characterisation

The minimum step size permitted within Marlin firmware is 100 μm. In order to assess the movement accuracy of the XYZ system, both end-stop repeatability and point to point repeatability were assessed. In addition, we measured deviation attributable to the movement of other axes.

7.1. End-stop repeatability test

To assess the positional consistency of the X, Y and Z axis mechanical end-stops, a camera (UEB 1000X 8 LED 2MP USB Digital Microscope Endoscope) was mounted to the X-carriage using a custom 3D printed bracket. Using the manual G-code entry module in our terminal-based operating system, “G28X”, “G28Y” or “G28Z” were used to return each axis to their respective home position. A measurement reticle was placed under the camera while in each home position and an image of the reticle taken (Fig. 6A). This process was repeated and a series of measurement reticle images was generated for each axis. For each image, the pixels per micron value was calculated using the “Set Scale” function in Fiji (ImageJ v1.52i). This was performed in order to ensure accuracy while measuring positional deviation during testing.

The positional deviation of the system was measured by comparing the position of the measurement reticle in an image with its position in the subsequent image. The absolute values of this data set were then analysed in GraphPad Prism (Fig. 6B). The absolute value of positional deviation was calculated as it was observed that positive and negative deviation values averaged to almost zero, which did not accurately reflect this systems capability. In addition, the coefficient of error (CE) for these measurements was also calculated. Coefficient of error was calculated as the standard error of the mean for repeated measurements divided by the mean [15].

![Fig. 6. Positional Repeatability & Axis Deviation.](image)
7.2. Point to point repeatability test

This test was performed using the same apparatus as the End-Stop Repeatability test. For each axis, point A was defined as the midpoint of the axis and a measurement reticle placed under the camera at this point. Point B was defined as being 10 mm away from point A. The camera was repeatedly cycled between points A and B with an image of the measurement reticle taken each time the camera returned to point A. Positional deviation was measured in the same way as the end stop repeatability test and analysed in GraphPad Prism (Fig. 6C).

7.3. Measuring deviation attributable to movement in other axes

To measure the positional deviation of an axis attributable to movement in the other axes, a single axis was fixed whilst the other two non-fixed axes underwent a translation of 100 mm before returning to the starting position. For each axis, a series of measurement reticle images were taken and analysed as per the end-stop repeatability tests (Fig. 6D).

7.4. Results

For the end stop repeatability test, images were analysed for each axis. Unless otherwise stated, values correspond to mean ± SD. Mean deviation for the X-axis was 61.0 ± 47.3 μm (CE = 17.8%, n = 20), Y-Axis 2.1 ± 1.4 μm (CE = 14.5%, n = 21) and Z-Axis 2.2 ± 2.5 μm (CE = 25.6%, n = 20). For the point to point repeatability test, six images were analysed for each axis. Mean deviation for the X-axis was 65.3 ± 50.6 μm (CE = 17.3%, n = 20), Y-axis 17.6 ± 9.1 μm (CE = 11.5%, n = 20) and Z-axis 6.2 ± 4.2 μm (CE = 15.1%, n = 20). With the X-axis was fixed, movement in the Y-axis caused the X-axis position to deviate by 5.5 ± 5.1 μm (CE = 22.8%, n = 16) whilst movement in the Z-axis caused the X-axis position to deviate by 3.5 ± 3.2 μm (CE = 20.5%, n = 20). With the Y-axis was fixed, movement in the X-axis caused the Y-axis to deviate by 9.6 ± 8.1 μm (CE = 20.6%, n = 17) whilst movement in the Z-axis caused the Y-axis to deviate by 9.0 ± 7.5 μm (CE = 19.1%, n = 20). With the Z-axis fixed, movement in the X-axis caused the Z-axis to deviate by 11.8 ± 10.8 μm (CE = 21%, n = 20) whilst movement in the Y-axis caused the Z-axis to deviate by 11.7 ± 8.7 μm (CE = 17%, n = 20).

8. Applications of the XYZ positioning system

8.1. Automated microscopy

Included in the operating system is a module for automated camera scanning module which is accessed by entering “2” at the main menu. This module serves to capture an array of images at specific coordinates across a user defined area. The images are saved to a folder on the Raspberry Pi 3 desktop and their file names are titled as the coordinates at which the images were captured. These images can then be reconstructed to produce a single high-resolution scan of the user defined area.

8.1.1. Scanning logic

In order to define the area to be scanned, the user inputs (X₁, Y₁), (X₂, Y₂) and (X₃, Y₃): These represent three points within the build volume of the XYZ positioning system (Fig. 7A). The numerical values for these coordinates are expressed in millimetres, as this is the standard unit of measurement for the Arduino Mega/RAMPS 1.4 system. The X and Y axes limits are 430.0 and 150.0 mm respectively, and hence the user should not input values exceeding these as the system is unable to travel outside of this range. Similarly, the user should not input values less than zero. A virtual parallelogram is constructed which borders the minimum and maximum of the X and Y values of the three coordinates entered and the limbs of this parallelogram are parallel to the X and Y axes respectively (Fig. 7B). Though it is conceivably possible to define the boundaries of this parallelogram using only two points, we opted for the three point approach as we believed it to be easier to understand than the two point approach and would not require additional assumptions to be made within the XYZ operating system code to facilitate a two point approach.

A point (Xₘᵢₘᵢₗ, Yₘᵢₘᵢₗ) is calculated by calculating the minimum X value entered from X₁, X₂, and X₃ and the minimum Y value entered from Y₁, Y₂, and Y₃. From (Xₘᵢₘᵢₗ, Yₘᵢₘᵢₗ), an array of nodes is populated (Fig. 7C). Each of these nodes represents a position at which an image will be captured during the scanning process. Prior to beginning the scan, the operating system will prompt the user to enter the desired distance between adjacent X nodes and adjacent Y nodes (Fig. 7C). The scan begins at (Xₘᵢₘᵢₗ, Yₘᵢₘᵢₗ) and scans every node in a row of X values (from minimum to maximum node) before moving to the next row of nodes in (Fig. 7D). We opted not to utilise a meander scanning pattern as it was deemed to be more complicated to implement in Python3, would provide a marginal reduction in scanning time and ultimately would have no effect on output image quality. However, for time sensitive applications where a specific scanning pattern is more appropriate, users can add code to the XYZ operating system to facilitate alternate scanning patterns.
Fig. 7. Area Scanning Logic. (A) The user defines the position of three coordinates \((X_1, Y_1)\), \((X_2, Y_2)\), and \((X_3, Y_3)\). (B) A rectangular area is defined by the minimum and maximum X and Y values entered. The boundaries of this area are parallel to the X and Y axes respectively. (C) The rectangular area is populated by nodes. The first node is located at the minimum X and Y value entered \((X_{\text{min}}, Y_{\text{min}})\). Additional nodes are populated parallel to the X and Y axes based on the user defined X and Y node spacing. Where an uneven amount of nodes fit within the scanning area, no node is generated for the modulo, (e.g. if there are 17.6 nodes in X direction, there will be 17 nodes spaced as per the user defined X node spacing with the first node placed at \(X_{\text{min}}\)). (D) Scanning begins at \((X_{\text{min}}, Y_{\text{min}})\) and cycles through adjacent nodes in the X direction. Once all nodes in a given row have been scanned, the camera moves up one row and begins again at the \(X_{\text{min}}\) position. This process repeats until the final node has been scanned, after which the system returns to the home position. (E) Example of a 5 × 4 stitched image, stitched using the Grid/Collection stitching plugin (pixel coordinate method) for ImageJ developed by Preibisch et al. 2009 [16].
8.1.2. Image reconstruction in Fiji (ImageJ)

Image reconstruction was performed using a freely available stitching plugin [16] available through the Fiji version of ImageJ (version 1.52i). This plugin provides different methods to stitch images together and is accessed by selecting the Plugins drop-down menu, then Stitching, and Grid/Collection stitching. For high contrast images with easily identifiable features the plugin has little difficulty stitching images together correctly. For low contrast images with subtle features the plugin may fail to interpret how images should be appropriately stitched together. This problem can be circumvented by using the Positions from file option in the plugin (the compute overlap box should be unticked). This option requires the user to specify a directory in which the images are stored and also a directory for a file that contains information relating to the position of each image. During the scanning process, a text file, TileConfiguration.txt is generated by the XYZ operating system and stored in the same folder as the images. This text file contains the filename of every image captured and their respective coordinates expressed in pixels (which is required for the Positions from file stitching method). For a given magnification, it is necessary to first calculate the number of pixels per mm by imaging a measurement reticle in order to ensure that the pixel coordinates in the TileConfiguration.txt file are accurate.

8.2. Neurophysiology – novel tool to study mechanotransduction

Mechanotransduction is the process by which sensory neurons convert a mechanical stimulus (such as stretch) to an electrical signal (i.e. an action potential). The precise molecular mechanisms underpinning mechanotransduction have not yet been fully elucidated however considerable investigation has been done into the muscle spindle and its mechanism of action [17–19]. The muscle spindle is an afferent neuron whose sensory annulospiral endings are wrapped around intrafusal muscle fibres. Muscle spindles generate action potentials in response to mechanical stretch. As part of our ongoing investigations into the role of the epithelial sodium channel in atrial and muscle spindle mechanotransduction, we sought first to establish that the XYZ positioning system could be used to elicit mechanotransduction in the muscle spindle. A simplified schematic of this setup is presented in Fig. 8. Using this model, we successfully demonstrated muscle spindle activation thresholds for stretch distance, stretch velocity and stretch acceleration (Fig. 9). Subsequently we performed pharmacological investigations using this model of mechanotransduction and we plan to publish these results in an upcoming manuscript which focusses on our combined findings from both the atria and the muscle spindle.

The DC switching power supply listed in Table 1 was unsuitable for neurophysiological applications of the XYZ positioning system. As it was a DC switching power supply, it utilised pulse width modulation in order to maintain an output voltage of 12 V. This in turn generated electromagnetic interference (EMI) which obliterated the comparatively small electrical neural signals which were generated by muscle spindles. As such, for neurophysiological applications it was necessary to power the XYZ positioning using a regulated linear DC power supply. A custom built linear regulated DC power supply produced by Acopian Technical Company was used for this purpose (cost $575). This power supply was almost electrically silent, permitting electrophysiological investigations into muscle spindle mechanotransduction.

It was necessary to ground the XYZ positioning system in order to minimise EMI. Each of the five stepper motors (X1, Y1, Y2, Z1, Z2) were grounded by removing a bolt from the back of each motor. Thin ground wires were stripped and wound tightly each bolt before they were screwed back into the NEMA17 case. In addition, the X motor rotor, Y-axis linear rails and Arduino Mega/RAMPS 1.4 microcontroller enclosure were connected to ground.

8.2.1. Ex-vivo lumbrical/spindle preparation

In accordance with EU directive 2010/63 and local ethical approval, wistar rats were euthanised by cervical dislocation following isoflurane anaesthesia (5%). The hind feet were isolated by cutting across the talocrural joint and perfused in oxy-
Fig. 9. Stretching the muscle spindle to study mechanotransduction. (A) Afferent nerve activity from a stretched muscle spindle. Brief pulses of stretch were applied to the lumbrical every two seconds in order to elicit mechanotransduction from the muscle spindle. Each Stimulus pulse indicates the initiation of a stretch. Filtered nerve activity is represented in blue, unfiltered in green. (B) Mechanotransduction activation thresholds were assessed with gradual increments in stretch distance, speed or acceleration. For this filtered unit, activation thresholds were observed at 14.0 mms⁻¹ and 50 mms⁻². Increased stretch distance, speed or acceleration are associated with increased nerve activity (Filtered Spike Rate). (C) Overdraw of filtered nerve activity observed in (B) indicates that this was a single unit recording. All data was recorded in Spike2 (Cambridge Electronic Design). ENG, Electroneurogram. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)
generated (100% O₂) Tyrode’s solution. A 22-blade scalpel was used to make a midline incision along the plantar aspect of the foot. Skin, fascia and superficial musculature was removed. The 4th lumbrical was isolated and a glass suction electrode was used to record from its nerve. A hook was fashioned from a 23-gauge hypodermic needle and mounted to a 3D printed male luer slip fitting on a 3D printed bracket. This hook assembly was mounted to the X-axis carriage using four M3x30 hex bolts and M3 hex nuts. The hook proper was tied to the second phalanx of the fifth digit of the rat hind paw using non-sterile suture thread.

With controlled movements of the X-axis carriage, the fourth lumbrical muscle was subjected to repeat cycles stretch and relaxation in order to elicit mechanotransduction. Nerve recording was performed with a glass-suction electrode patched to the afferent ending of the nerve serving the fourth lumbrical muscle. Activation thresholds of muscle spindles in the fourth lumbrical muscle were assessed by using standard G-code commands to gradually increase and subsequently gradually decrease stretch distance, acceleration and maximum travel speed in a (see Fig. 9B).

9. Safety considerations

When wiring the DC switching power supply described in Table 1, the user should wire the live, neutral and ground wires while unit is unplugged in order to protect the user from mains voltage. In order to protect the microcontrollers, power should be switched off before plugging or unplugging any of the connectors. When powered off, the user should refrain from manually moving the X-axis carriage quickly as doing so causes induces a current through the motor circuits and may damage the microcontrollers. It is safe to manually move the X-axis carriage slowly. During testing of the positioning system, care should be taken to safely cable-manage all wiring such that they are not damaged during normal operation of the XYZ positioning system.

10. Discussion

The steps of hardware assembly and software modification required to recreate the XYZ positioning system have been outlined. Considerable preparation was required to acquire all necessary components prior to commencing the assembly.

---

| Tab and Entry | Note |
|---------------|------|
| Marlin.h      |      |
# include <U8glib.h> | This entry should be added to the top line of the Marlin.h tab |

| Configuration.h |
|-----------------|
| Endstop Settings Section | |
# define USE_XMIN_PLUG | Remove the “//” to uncomment this line |
# define USE_YMAX_PLUG | Remove the “//” to uncomment this line |
| Movement Settings Section | |
# define DEFAULT_AXIS_STEPS_PER_UNIT (80, 400, 400, 400) | |
# define DEFAULT_MAX_FEEDRATE (100, 2, 5, 5) | |
# define DEFAULT_MAX_ACCELERATION (100, 100, 100, 100) | |
# define DEFAULT_XJERK 0.3 | |
# define DEFAULT_YJERK 0.3 | |
# define DEFAULT_ZJERK 0.3 | |

| Homing Section | |
# define X_HOME_DIR –1 | |
# define Y_HOME_DIR 1 | |
# define Z_HOME_DIR 1 | |

| Machine Dimensions Section | |
# define X_BED_SIZE 430 | |
# define Y_BED_SIZE 150 | |
# define X_MIN_POS 0 | |
# define Y_MIN_POS 0 | |
# define Z_MIN_POS 0 | |
# define Z_MAX_POS 430 | |

| LCD and SD Support Section | |
# define SDSUPPORT Remove the “//” to uncomment this line |
# define REVERSE_ENCODER_DIRECTION Remove the “//” to uncomment this line |

| LCD/Controller Selection (Graphical LCDs) Section | |
# define REPRAP_DISCOUNT_FULL_GRAPHIC_SMART_CONTROLLER Remove the “//” to uncomment this line |

| Configuration_adv.h | |
# define Y_DUAL_STEPPER_DRIVERS Remove the “//” to uncomment this line |

---

Table 2
Modifications to Marlin Firmware. All of the above modifications are required for correct functioning of the XYZ positioning system. Changes included the inclusion of the U8glib graphical library, adjustments to numerical values and the uncommenting of code lines. Calculations pertaining to steps per mm are available in Appendix B.
It is estimated that a complete build requires approximately 10 h, similar to what is required when constructing an FFF/FDM printer from a kit such as those sold by Prusa.

For the majority of use cases, the entry cost of this system is attractively low at $670.24 and comparable to other open source XYZ positioning systems such as the delta format system ($500) described by Zhang et al. (2016) which cost or multi-axis stress sensing system ($600) described by Agcayazi et al. (2018). We believe that an exact replica of our system could realistically be constructed for $600 because of the lower VAT rates in the United States versus Ireland. Further savings could be made by reducing the size of the axes. The strength of our system is that it can be easily adjusted by users to suit any intended use – all design files have been provided and the kinetic properties of this system can easily be adjusted in firmware.

It should also be considered that our device incorporates a Raspberry Pi 3 and the OpenCV library which makes it unique compared to other open source positioning systems. Users can easily add additional functionality to the Python3 based operating system without affecting existing functionality. The inclusion of OpenCV in this system provides users with thousands of image processing tools and an option to incorporate artificial intelligence and machine learning into their use case. We have developed an automated scanning utility to fit our research needs and we believe that other teams will similarly develop novel imaging utilities.

For electrically sensitive applications such as electrophysiology, the entry cost rises given the necessity of acquiring an electrically silent power supply. Our Acopian power supply brought our total system cost to $1357.09. We believe that savings can be made in opting for a smaller power supply. Our Acopian power supply was rated for a maximum of 7.2 amps however the peak amperage draw from our system motors is approximately 2.8 amps ([5 motors × 0.4 mA/phase × 1.4 (peak current)]. A power supply rated for 3.5 amps should provide adequate operational headroom. In respect of cost it must also be stated that the prices included in Table 1 and for the Acopian power supply are inclusive of 23% VAT which is appropriate for the Republic of Ireland. Given the lower VAT rates in the United States we are confident that the neurophysiology variant of this system could be constructed for approximately $1200.

For microscopy it is plausible that users could simply modify an entry level 3D printer (approximately $150–200) however the XYZ positioning system has been designed with electrophysiological applications in mind. The XYZ positioning system's axes are large, permitting stepper motors to be positioned far from the recording area (which minimises EMI). Entry level 3D printers usually have small axes and thus the close proximity of their stepper motors to the recording area would generate EMI and prevent nerve recording. Entry level 3D printers usually have small print beds that are driven by timing belts (usually the Y-axis). It would not be possible to utilise movement along this axis during electrophysiological recording as the vibrations generated by the moving build plate would interfere with the comparatively small nerve signals. In addition, entry level 3D printers are usually powered by low cost pulse width modulation style power supplies that generate large amounts of EMI and thus prevent the recording of comparatively small nerve signals. Inevitably, it would be necessary to purchase of a suitable linear regulated DC power supply as we have done. Either for microscopy or electrophysiology, a considerable amount of time would be required to customise and redesign an entry level 3D printer and we believe that readers would be better served to iterate on the XYZ positioning system rather than beginning anew with an entry level 3D printer.

The point to point deviation for each axis was 65.3 μm (X-Axis), 17.6 μm (Y-Axis) and 6.2 μm (Z-axis). The achieved positional repeatability values are a magnitude greater than that described by Campbell, Eifert et al. (2014) who in their description of “Openstage” have claimed submicron positioning repeatability of 0.1–1 μm [2]. This should not be regarded as a direct comparison however, given that this submicron repeatability was achieved with stage automation rather than sensor automation. To achieve such low repeatability values requires important mechanical compromises. The maximum reported operational travel speed of Openstage is reported as 0.75 mms⁻¹ whereas the XYZ system travels operates at 100 mms⁻¹ (X-axis), 2 mms⁻¹ (Y-axis) and 5 mms⁻¹ (Z-axis). Such travel speeds were required as part of the mechanotransduction experiments and hence improved positional repeatability was not pursued. However, the repeatability of the XYZ system could further be improved by opting for stepper motors with smaller step angles (0.9° as opposed to 1.8°), gearing each motor down to increase the number of steps required per revolution, and swapping to stepper driver modules with finer resolution (1/512th vs. 1/16th microstepping). These mechanical compromises would reduce motor torque and reduce travel speeds which for the large XYZ system axes (430 mm × 150 mm × 430 mm) would greatly reduce the rapidity of the system. In addition, the X-Axis could be configured to be driven by a lead or ball screw instead of a timing belt which would significantly improve its positional repeatability. Movement in each axis was observed to cause deviation in the position of the other two axes (Fig. 6D). The magnitude of these deviations is likely attributable to the magnitude of the translation utilised in these tests. Specifically, in our tests, we applied large translations of 100 mm and subsequently returned to the starting position before reimagining the measurement reticle. These unintended deviations are likely attributable to a combination of our device’s large volume and the size of the size of the translation tested. We hypothesise that smaller iterations of this device would be more rigid and thus be less affected by this form of positional inaccuracy. Future iterations of this device could also utilise a greater number of CNC milled steel parts for the frame which would improve the structure’s rigidity.

The main limitations of the XYZ positioning system are mechanical in nature. In our implementation, the X & Z axis assembly is tall and heavy and as such we opted to reduce the Y and Z axis travel speeds to 2 mms⁻¹ and 5 mms⁻¹ respectively. This reduction in speed preserves positional integrity of the system by reducing the likelihood of stepper motors stepping erroneously. However, the assembly can be adjusted to the desired specific use case and a simple reduction the size of the Z-axis would greatly reduce its inertia and permit positional accuracy at greater travel speeds. In our case, it was nec-
ecessary to have a tall Z-axis to permit the use of a dissection microscope during our electrophysiology experiments. In addition, the X-axis repeatability is an order of magnitude greater than that of the Y and Z axes. Applications requiring more precise repeatability should utilise Y-axis rather than X-axis mediated translation whereas applications requiring fast translation should utilise the X-axis. Though the axes and frame are rigid enough to support payloads over a kilogram, we would recommend payloads under 1 kg to ensure movement accuracy, particularly if considerable movement will be performed along the X-axis which travels at higher speeds than the Y and Z axes. Another potential limitation to consider is that the default minimum step size permitted by Marlin firmware is 100 μm. We believe that the firmware can be modified to permit single microsteps which with our hardware configuration would correspond to 12.5 μm resolution in the X-axis and 2.5 μm resolution in the Y and Z axes. Further gearing could be applied to achieve submicron resolution. This level of resolution was far beyond the scope of what was required for our intended use of this device and hence we did not explore this further. The Raspberry Pi camera is currently not supported by the system though support for this could be added with additional code to the XYZ operating system. The inclusion of a joystick or otherwise more ergonomic controller was not possible within the timeframe of this project. A joystick would most easily be implemented via the many spare GPIO pins on the Raspberry Pi 3 and additional code would be required in the XYZ operating system. We have provided a useful link in Appendix A should wish to pursue this further.

In summary, we have designed, constructed, and programmed an adaptable XYZ positioning system. We have utilised this tool during neurophysiological investigations into mechanotransduction in the rat. In addition, we have incorporated the OpenCV library into the build, and subsequently developed an automated microscopy scanning tool. All components and software utilised were open-source, free to access or available at low cost. Given the ease with which these components can be accessed and the potential that such a system offers, it is believed that other research groups may find this system an attractive and useful experimental tool.

11. Human and Animal Rights

The work on mechanotransduction described in our application section was performed in accordance with EU directive 2010/63 and local ethical approval (AREC-15-36-Jones).

Declaration of interest

None.

Funding

This work was funded by the School of Medicine, University College Dublin.

Appendix A: Links to software and Author files

Software

Arduino IDE https://www.arduino.cc/en/main/software
Autodesk Fusion360 https://www.autodesk.com/products/fusion-360/overview
ImageJ (Fiji) https://fiji.sc/#download
Marlin Firmware http://marlinfw.org/meta/download/
OpenCV https://opencv.org/
Slic3r (Prusa Edition) https://www.prusa3d.com/drivers/
Thonny https://thonny.org/

Author Files

- Author files are available to access at: https://osf.io/sju9d/

Files available include:

- Autodesk Fusion 360 Master File
- 3D Printable STL Files
- Modified version of Marlin Firmware version 1.1.9
- Operating System for XYZ Positioning System

Links to more detailed information regarding electronics assembly
Appendix B: Calculations pertaining to steps per mm for X, Y and Z axes

X Axis: one full turn of the stepper motor’s rotor causes twenty teeth of the timing belt to pass over the timing belt pulley. Each tooth has a spacing of 2 mm and a microstep corresponds to 1/16th of a full step. Thus, it follows:

\[ \text{distance travelled in 1 revolution} = \text{length of 20 teeth} \]
\[ = (20) \times (2\text{mm}) \]
\[ = 40\text{mm} \]

\[ \text{distance travelled during 200 full steps} = 40\text{mm} \]
\[ \text{distance travelled during 3200 microsteps} = 40\text{mm} \]
\[ \text{distance travelled during 80 microsteps} = 1\text{mm} \]

\[. . . 80 \text{ microsteps per mm} \]

Y & Z Axes: both the Y and Z axes are driven by lead screws that have a lead of 8 mm. This means that in one full rotation of the stepper motor’s rotor the lead screw nut advance by a distance of 8 mm.

\[ 200 \text{ full steps} = 8\text{mm} \]
\[ 3200 \text{ microsteps} = 8\text{mm} \]
\[ 400 \text{ microstep} = 1\text{mm} \]

\[. . . 400 \text{ microsteps per mm} \]

Thus, it is necessary that in the Marlin firmware, steps per mm for the X, Y and Z axes are 80, 400 and 400 respectively.

Appendix C: Principles of streaming G-Code from Raspberry Pi 3 to Arduino

For bilateral communication between the microcontrollers (Raspberry Pi 3 & Arduino Mega), it is necessary that the baud rate of both microcontrollers is equal. Baud rate is defined as the rate of data transmission in a system and is measured in bits per second. By default, the baud rate in Marlin firmware (version 1.1.9) is 250,000 bits per second and as such we chose to mirror this with our Raspberry Pi 3 by including the following code at the start of our Python3 operating system script:

```
import serial
ser.baudrate(250000)
```

Marlin firmware can interpret and action G-Code commands received over serial connection only if they are in byte format. Typed user inputs are categorised as strings (i.e. not in byte format) and as such are not interpreted by Marlin 1.1.9. In addition to this, Marlin 1.1.9 requires the inclusion of “\n” at the end of a G-Code command in order to actuate a submitted line of G-Code (the \n command is equivalent to the pressing of the enter key). A simple concatenation of a user’s typed G-Code with the “\n” string, followed by a conversion to byte format allows Marlin 1.1.9 to read a user’s G-Code command and carry out the corresponding action. The encoding for these bytes must be in UTF-8 format (Unicode Transformation Format). This is an encoding format which uses 8-bit blocks to encode individual characters of text. An example of the required Python3 code to convert a user typed string to the format which can be read by the Arduino is as follows:

```
UserTypedInputString = input(""")
ArduinoEnterKey = str("\n")
CombinedString = (UserTypedInputString + ArduinoEnterKey)
CombinedBytes = bytes(CombinedString, 'UTF-8')
```
With a user's input converted to the appropriate format (as described above), it can be sent to the Arduino using the following code:

```python
ser.write(CombinedBytes)
```

With everything combined, the combined code should read as follows:

```python
import serial
ser.baudrate = 250000
UserTypedInputString = input('"
ArduinoEnterKey = str(''
"
CombinedString = (UserTypedInputString + ArduinoEnterKey)
CombinedBytes = bytes(CombinedString, 'UTF-8')
ser.write(CombinedBytes)
```

Appendix D. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.ohx.2020.e00098.

References

[1] A. Courtney et al, The flexiscope: a low cost, flexible, convertible, and modular microscope with automated scanning and micromanipulation, bioRxiv (2018) 442210.
[2] R.A. Campbell, R.W. Eifert, G.C. Turner, Openstage: a low-cost motorized microscope stage with sub-micron positioning accuracy, PLoS ONE 9 (2) (2014).
[3] J.P. Sharkey et al, A one-piece 3D printed flexure translation stage for open-source microscopy, Rev. Sci. Instrum. 87 (2) (2016) 025104.
[4] A.M. Chagas et al, The €100 lab: A 3D-printable open-source platform for fluorescence microscopy, optogenetics, and accurate temperature control during behaviour of zebrafish, Drosophila, and Caenorhabditis elegans, PLoS Biol. 15 (7) (2017) e2002702.
[5] T. Agcaayazi et al, Multi-axis stress sensor characterization and testing platform, HardwareX 5 (2019) e00048.
[6] T. Baden et al, Open Labware: 3-D printing your own lab equipment, PLoS Biol. 13 (3) (2015) e1002086.
[7] C. Zhang, B. Wijnen, J.M. Pearce, Open-source 3-D platform for low-cost scientific instrument ecosystem, J. Lab. Autom. 21 (4) (2016) 517–525.
[8] M.C. Carvalho, R.H. Murray, Osmar, the open-source microsyringe autosampler, HardwareX 3 (2018) 10–38.
[9] M.C. Carvalho, C.J. Sanders, C. Holloway, Auto-HPGe, an autosampler for gamma-ray spectroscopy using high-purity germanium (HPGe) detectors and heavy shields, HardwareX 4 (2018) e00040.
[10] A. Kaehler, G. Bradski, Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library, O’Reilly Media, Inc., 2016.
[11] OpenCV. OpenCV Website Homepage (About Tab). Information about OpenCV listed on OpenCV's homepage. Available from: https://opencv.org/about.html, [Webpage] 2019 (cited 2019 09/04/2019).
[12] Arduino. Arduino “About Us” Webpage. Arduino “About Us” webpage. Available from: https://www.arduino.cc/en/Main/AboutUs, [Webpage] 2019 (cited 2019 20/03/2019).
[13] Marlin. Marlin “About Marlin” Webpage. “About Marlin” section of Marlin website. Available from: http://marlinfw.org/docs/basics/introduction.html, 2019 (cited 2019 20/03/2019).
[14] A. Rosebrock, Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi. Step by step guide on how to install OpenCV 3 on Raspbian Stretch. Written on September 4th 2017. Available from: https://www.pyimagesearch.com/2017/09/04/raspbian-stretch-install-opencv-3-python-on-your-raspberry-pi/, [Webpage] 2017 (cited 2019 20/02/2019).
[15] Bioscience, M. Stereology.info - Understanding the Purpose of the Coefficient of Error. Available from: http://www.stereology.info/coefficient-of-error/, [Webpage] 2020 (cited 2020 13/01/2020).
[16] S. Preibisch, S. Saalfeld, P. Tomancak, Globally optimal stitching of tiled 3D microscopic image acquisitions, Bioinformatics 25 (11) (2009) 1463–1465.
[17] A. Simon et al, Amloride-sensitive channels are a major contributor to mechanotransduction in mammalian muscle spindles, J. Physiol. 588 (Pt 1) (2010) 171–185.
[18] G.S. Bewick et al, Autogenic modulation of mecanoreceptor excitability by glutamate release from synaptic-like vesicles: evidence from the rat muscle spindle primary sensory ending, J. Physiol. 562 (2) (2005) 381–394.
[19] R.W. Banks et al, Evidence for activity-dependent modulation of sensory-terminal excitability in spindles by glutamate release from synaptic-like vesicles, in: Sensorimotor Control of Movement and Posture, Springer, 2002, pp. 13–18.