Single channel myoelectric control of a 3D printed transradial prosthesis

Scott A. Curline-Wandl1,2 and M. Azam Ali1*

Abstract: Patients suffering from upper limb deficiency all experience a significant reduction in their ability to interact with their environment. The primary method used to restore this lost function is the implementation of a prosthesis. However, a significant proportion of users abandon their devices—citing reasons such as cost, weight, functional control and aesthetics. This paper will discuss the use of myoelectric control technology in the control of an additive manufactured myoelectric transradial prosthesis. This prosthesis should be low cost, lightweight, easily controlled and has a morphology that mimics a human hand. As a part of this study, a single channel electromyogram controlled hand model was developed using an Arduino Uno microcontroller to control a single degree of freedom. Preliminary testing and debugging has indicated that a single input channel is adequate for relatively simple open/close protocols. However, in order to allow for the more fine control desired by patients, new preprocessing boards will need to be developed for the Arduino in order to gain access to more input channels. This will allow us to truly explore the Arduino as a budget option for implementing higher complexity control structures.

ABOUT THE AUTHORS
Scott A. Curline-Wandl has a combined Bachelor's Mechatronic Engineering/Master's Biomedical Engineering from the University of New South Wales in Sydney, Australia. He is currently working in the Australian Public Service in IT development whilst seeking funding for PhD study. His research interests in myoelectric prosthesis technology and tactile stimulation for prosthetic and virtual applications.

Dr Azam Ali has a BSc (Hons degree), MSc (1991) 1st Class in Chemistry from Jahangirnagar University (JJ), Dhnaka, Bangladesh and a PhD degree (2000) in Polymer and Materials Science from the University of Science Malaysia (USM). He is currently a senior lecturer Centre for Materials Science & Technology (CMAST), Department of Applied Sciences and specialises in biomaterials science and engineering. Azam's current research activities include material/biomaterials, understanding materials structure-functions and properties, developing and designing novel, advanced and nanostructure biomaterials, processing and manufacturing (including additive manufacturing) of medical devices and technology for health and wellbeing.

PUBLIC INTEREST STATEMENT
Patients suffering from upper limb deficiency all experience a significant reduction in their ability to interact with their environment. This inability to interact with objects within their environment has serious long term implications for the individual’s ability to participate within the workforce and in social settings.

The primary method used to restore lost limb function is through a prosthesis. However, a significant proportion of users abandon their devices—citing reasons such as cost, weight, a lack of functional control and aesthetics.

For a great many, the immense cost of a myoelectric prosthesis is a significant barrier to use of this technology. In order to address this issue of cost, this paper will describe a pilot investigation into the use of 3D printing and low cost Arduino microprocessor boards as a budget alternative to high cost devices.
1. Introduction
Patients who suffer from upper limb loss, whether their condition is congenital or the result of trauma, all have decreased abilities to interact with their environment. In order to combat this, a number of different prosthesis technologies exist with the aim of restoring lost limb function.

There are three main types of upper limb prosthesis - passive cosmesis, body powered and myoelectric devices. Cosmeses are passive, non-functional devices that aim to restore a patient’s social and psychological state by restoring the aesthetic qualities of a missing arm. In contrast, a body powered prosthesis is one controlled by the mechanical action exerted by the user via harness systems attached to various parts of their body. Finally, myoelectric devices use electromyographic (EMG) signals to interpret the user’s intention in controlling their device, and then use this information in order to control electromechanical actuators such as servo or stepper motors.

Each of these devices has a very high rejection rate amongst their users. Around 53% of cosmesis users will reject them, whereas 50% of body powered users and 39% of myoelectric users will cease use of their devices (Biddiss, Beaton, & Chau, 2007). These rates show that an unacceptable number of amputees are dissatisfied with devices currently available on the market. They cite cost, weight, comfort, functional control aesthetics and lack of sensory feedback as their primary reasons for discontinuation (Biddiss et al., 2007)—therefore it makes sense to focus on these areas when developing future devices.

Of the complaints reported in the Biddiss paper, the greatest was cost (Biddiss et al., 2007). While it depends on the healthcare service availability in their country, it is reported that patients could be paying approximately 20% of a $USD 60,000 device. By using additive manufacturing (3D printing) technology, we hope to reduce the cost of manufacture of these prostheses by allowing access to open source designs. Once the mechanical components of the device have been printed, it then becomes a matter of installing control mechanisms to allow for actuation of the device.

One attempt at a purely mechanical additive manufactured device is the Talonhand 2.7 developed by Peter Binkley (2015). This design is an open source body powered device available through the Makerbot Thingiverse, an online repository for users of 3D printers to share and collaborate their designs. The device itself is an underactuated, one degree of freedom mechanical hand designed for patients suffering from finger loss. The mechanism uses the wrist joint to control the action of five fingers in an open/close fashion.

The Talonhand is easily available to those who have access to a 3D printing service, is lightweight due to the use of ABS polymer as a printing material, and closely mimics the morphology of the human hand. Perhaps one of the most interesting things about the Talonhand is that the STL files required to print it are available for free online on the MakerBot Thingiverse. This means that someone with access to either a personal 3D printer, or access to a commercial 3D print shop can have one of these made very cheaply. If neither of these options are available, MakerBot also offers a service to have this device 3D printed and shipped to the user for approximately $55, plus shipping (Figure 1).
Despite these advantages, since it requires an intact wrist joint for actuation, this relegates the device to a fairly niche market of users. However, the incorporation of myoelectric control technology would allow a greater range of transradial amputees to use the device by replacing the action of the wrist joint with an EMG controlled electromechanical actuator, without any significant alteration to the mechanics of the device required.

In order to build upon this device whilst still working towards the ethos behind it— the concept open source technology—it is desirable to develop the device with low cost and accessibility to laypersons in mind. To that end, development of an algorithm that can be implemented using cheap, hobby microcontrollers which are fairly simple to use is an interesting concept to explore. Arduino code could be uploaded to an online repository and made freely available for download. This code would include controller software, and a Graphical User Interface (GUI) that would allow the user to tune the control algorithm to their preference. The user would then be able to buy cheap hobby electronics, upload the software themselves to the controller, and install the controller into the Talonhand themselves. Essentially, we hope to present a more affordable “budget” myoprosthesis alternative by giving the end user the means to manufacture their own devices.

This paper will describe the development of a myoelectric controller to be incorporated into the Talonhand 2.7 device, in order to provide proof of concept for its future development. A short review of the current state of myoelectric technology will be done, followed by a description of the hardware and software that we propose be added to the Talonhand. Finally, an analysis of early test results and future directions of development will be made.

1.1. Types of myoelectric controller

There are two main types of myoelectric controller that are either currently available or under development—pattern recognition controllers and non-pattern recognition controllers (Asghari Oskoei & Hu, 2007).

Pattern recognition controllers seek to interpret the user’s intended action by classifying the input of several EMG channels using a pre-trained classifier algorithm. Using training data, the controller is taught what various combinations of EMG inputs are likely to mean. From this, it will generate a control action as an output which then allows for control of joint actuators (Asghari Oskoei & Hu, 2007). Whilst these algorithms are considered state of the art for myoelectric control technology, their complexity and memory requirements often means that they demand microprocessor
technology (Asghari Oskoei & Hu, 2007). This necessity will increase the cost considerably compared to simpler control structures which can use microcontrollers.

Two ways of handling pattern recognition are Artificial Neural Networks (ANNs) and Fuzzy Logic Controllers (FLCs). ANNs have been shown to be highly successful in classifying EMG signals—their ability to represent and learn both linear and nonlinear relationships from data is extremely useful in myoelectric control (Asghari Oskoei & Hu, 2007). It has been shown that by applying ANN technology, it is possible to correctly classify four (Hudgins & Parker, 1991; Hudgins, Parker, & Scott, 1993) or six (Jingdong et al., 2006) different actions with an accuracy of 90 and 95% for those respective studies. However the implementation of ANNs is somewhat hardware limited. Although ANN libraries for microcontrollers are available (Anon, 2015), the networks possible with common microcontrollers would be relatively small, i.e. around 7 inputs and 4 outputs. To implement more complex ANN algorithms, one would need to consider more expensive microprocessor technology.

FLC have been demonstrated to have excellent tolerance to the non-repeatable nature of EMG signals (Ajiboye & Weir, 2005; Asghari Oskoei & Hu, 2007)—Ajiboye and Weir (2005) employed an FLC in the classification of surface EMG signals from the forearm, and achieved a classification rate of 94–99%. It allows for a more nuanced style of control by implementing the concept of partial truth within a set. This added nuance then opens up a very linguistic style of control, which has the added benefit of being easily accessed and understood by laypersons with no engineering training via a GUI.

Whilst they are not considered as powerful as ANNs, FLCs have the added bonus of being much more achievable by off the shelf microcontrollers. In fact, there is an embedded Fuzzy Logic Library (eFLL) written for the Arduino platform by the Robotic Research Group at the University of Piaui (Alves, 2012). Therefore, FLCs make a very attractive form of control for applications where cost is an important consideration.

Non-pattern recognition controllers, as the name would suggest, describes the rest of the control strategies available for implementation that do not seek to classify EMG signals. These systems are generally considered undesirable for systems with several degrees of freedom (Asghari Oskoei & Hu, 2007). However they are computationally inexpensive, and hence they can be cheaply implemented using off the shelf microcontrollers.

Non-pattern recognition strategies include proportional control, threshold control and finite state machines (FSMs) (Asghari Oskoei & Hu, 2007). Whereas proportional control has been used in the development of lower limb prostheses (Huang, Wensman, & Ferris, 2014), in upper limb devices it tends to be used to complement other control structures such as pattern recognisers (Fougner, Stavdahl, & Kyberd, 2014; Smith, Kuiken, & Hargrove, 2014), rather than being used as a primary control strategy.

Threshold controllers are typically used in conjunction with onset analysis techniques and FSMs (Asghari Oskoei & Hu, 2007; Carrozza, 2005). Essentially these controllers are used to detect muscle activity in an On/Off fashion by comparing signal statistics to pre-set activation thresholds. This then allows control actions to be taken-in FSMs this will usually trigger a change in state. Threshold control can also be used in onset analysis to represent muscle activation information using temporal characteristics. This makes it a necessary part of many myoelectric controllers as it allows for accurate discrimination between quiescent and active EMG signals (Asghari Oskoei & Hu, 2007; Liu et al., 2015).

FSMs are a form of computational design where the controller is considered to be an abstract machine with a finite number of discrete states. The algorithm then defines what actions the
machine undertakes in each state, and what inputs will trigger the machine to swap between its possible states. Carrozza et al. (Carrozza, 2005) developed a two channel FSM for control of a one degree of freedom transradial prosthesis. State switching was handled by threshold detection of the flexor and extensor digitorum muscle against threshold values determined statistically during a calibration phase done each time the controller is turned on (Figure 2).

During State $S_0$, the controller is doing calibration of the system based on baseline EMG activity ($V_{BL1}$ and $V_{BL2}$). Once the baseline EMG activity has been established, thresholds ($V_{TH1}$ and $V_{TH2}$) for each muscle are calculated using the following equation:

$$V_{TH1} = V_{BL1} + \Delta V_{TH1}$$

$$V_{TH2} = V_{BL2} + \Delta V_{TH2}$$

where $\Delta V_{TH1}$ and $\Delta V_{TH2}$ are preset values encoded into the algorithm.

Once this calibration is complete, the machine automatically moves to state $S_1$, where the prosthesis is set to open. In state $S_2$, the hand is being closed. One weakness of this structure noted by the Carrozza paper was that it does not allow for force control—however it was suggested by the paper that this could be remedied by only allowing the device to take action whilst it detects EMG activity, and simply holding position when activity levels are below the threshold (Carrozza, 2005).

1.2. Signal pre-processing

In practical applications, EMG signals are subject to noise and interference from sources such as mains power, movement artefacts and other bioelectric signals. Furthermore, even though EMG signals are considered to have a large amplitude compared to other electrophysiological signals, they are still much smaller than many controllers can detect. Therefore, it is necessary to do signal pre-processing on captured signals in order to remove noise and allow the signal to be sampled.

According to the literature, signal pre-processing is done primarily by active analog filters (Ajiboye & Weir, 2005; De Luca et al., 2010). The general consensus in the literature is that the EMG signal bandwidth spans from around 5–10 Hz up to approximately 400–450 Hz. Therefore bandpass filters are typically applied in this region. According to De Luca et al. the low pass corner frequency is not critical to quality filtration—placing this cut-off anywhere in the range of 400–450 Hz is justifiable (De Luca et al., 2010). The literature in general has a range of high pass cut-off frequencies between 5 and 20 Hz, however after investigation De Luca recommends 20 Hz as having the best trade-off between artefact and signal reduction for applications involving more common limb movements (De Luca et al., 2010).
However, not all EMG acquisition devices on the market filter within this bandwidth. Backyard Brains is a company based in Michigan, U.S.A., which manufactures EMG pre-processing shields designed for the Arduino Uno microcontroller board. Inspection of their analog filters shows that their filter passband is between approximately 340–1200 Hz. After asking the designers directly, this choice as apparently because when connecting to iOS devices, there is an intrinsic high pass filtration that occurs, and they wished to shift the passband away from mains noise (50 Hz). Therefore it seems apparent that the choice of acquisition and controller hardware is important to EMG filter design.

2. Hardware

There are three main components that need to be included in the Talonhand in order to achieve myoelectric control:

- A digital microcontroller
- An analog pre-processing circuit
- An electromechanical actuator

2.1. Digital microcontroller

The digital microcontroller board to be used in this device is the Arduino Uno. The board is based off of the ATMega 328p microcontroller chip, has multiple configurable digital I/O pins and has 6 analog input pins.

The main features that make Arduino based boards attractive is the open source nature of the hardware, as well as cost and availability. Since these boards are open source, there is extensive support available for them, including software libraries and hardware designed to directly interface with them.

Furthermore, the use of Arduino boards for myoelectric control has already been established within the literature. Jacob Segil reported in his PhD thesis that he successfully used an off the shelf Arduino controller to perform postural control of a myoelectric hand (Segil, 2014). This indicates that these cheap microcontrollers are capable of providing the computing power necessary for myoelectric controlled devices.

2.2. Signal acquisition and pre-processing

Signal pre-processing will be handled by a commercially available shield designed for the Arduino Uno called the Muscle SpikerShield (Backyard Brains Inc, Ann Arbor, MI). The shield is a printed circuit board designed to plug directly into Arduino Uno boards, making it ideally suited for performing signal acquisition of EMG signals using the Arduino platform. As mentioned in the Introduction, the board has a somewhat atypical passband for the filter- this has been tested and discussed in the “Preliminary Analysis of EMG Signals” section (Figure 3).
The downside of using this shield is that it can only receive a single EMG input. This essentially rules out more common control strategies that require more inputs, if one wishes to use a single shield. However for very early stage prototyping this is acceptable—a single shield can be used to confirm that the design is viable. Once viability is established, the stackable design of the shields means that more shields can be added at a later date in order to access more input channels, or a new multichannel board can be developed.

For the prototype, acquisition of EMG signals was done using noninvasive surface EMG techniques. Common biomedical surface electrodes will be placed on the forearm—these electrodes will then plug into the SpikerShield board via a three headed input cable with alligator clip connectors. For early prototyping, a single SpikerShield will be used, meaning that the device will work off a single EMG input. This has been done primarily due to budget constraints.

2.3. Electromechanical actuation
The device will be actuated by a single bipolar stepper motor installed at the wrist joint. The stepper motor was chosen over a servo motor for prototyping as it was originally presumed that grip force was less of a priority than accurate position control. The stepper motor used in this device is a 10 V, 0.5 A, 200 step motor, and is driven by an EasyDriver stepper driver (SparkFun Electronics, Niwot, Colorado).

The EasyDriver is a breakout board which implements a chopper driver which allows the Arduino to control stepper motors. The advantage of this board is that it allows for microstepping of the controlled stepper motor—meaning the stepper motor being used effectively has 1,600 steps available to it. This allows for a very precise control of position.

2.4. Power supply
The device is currently powered by two voltage regulated mains power supplies—one for the Arduino board and the other for the EasyDriver board. These power supplies are both recycled centre positive sources that connect to their respective boards via installed power plugs.

2.5. Test rig
In order to be able to test the control algorithm during its development, the hardware was assembled into the testing rig shown below Figure 4:
The EasyDriver (the red breakout board pictured above) was permanently soldered to the stepper motor inputs wires. It then had had a series of multithreaded wires soldered on to it in order to allow it to be plugged into the pins on the Arduino. These wires were connected to the Arduino via the female I/O pins on the SpikerShield.

In order to demonstrate the movement of the motor, a mock hand made of plate aluminium was attached to the motor body shown above, and the motor was then mounted such that it rotated about its own shaft.

3. Preliminary analysis of EMG signals

In order to gain more information on how the EMG signal captured by the Arduino setup behaves during different periods of muscle contraction, the Arduino/SpikerShield setup was reconfigured as a digital EMG signal acquisition card and used in a small pilot test to gather more data for inspection. This pilot test was done by attaching the electrodes to the individual's forearm in a variety of ways in order to investigate different aspects of the device's behaviour in signal capture. Signals were sampled at 500 Hz with non-overlapping windows.

The first concern was that the preprocessing circuit has a passband that is atypical compared to the literature. In order to confirm its appropriateness, filtered and raw signals were sampled and compared in both time and frequency domain. This was done by connecting the electrodes to the individual's forearm as described later in Figure 6—the positive and negative electrode were placed across the flexor digitorum muscle on the inner forearm, and the ground electrode was placed on the back of the hand. This test was done on a single subject rather than many subjects, as the emphasis of this test was to quickly verify that the filtration circuit did behave as a useful filter. The subject was asked to relax their hand, and a switch on the analog filter board was switched on and off as the Arduino recorded the signal, thereby giving us a comparison between a filtered and unfiltered EMG signal.

The left half of Figure 5 shows the raw EMG signal before filtration was turned on, whereas the right half is the signal with filtration applied. Clearly, inspection of the signal in both domains shows that significant amounts of high frequency noise is removed by the filter. Further inspection of the data presented in Annexes A and B shows that there is still a significant increase in the RMS of the filtered signal during an active clenching if the fist, and therefore we can still perform useful control actions using this somewhat abnormal bandpass filter shield.
In order to gather more information on how the EMG signal behaves during muscle contraction to inform algorithm development, filtered signals during a clench-hold-release action were recorded. Again this was done on a single healthy subject as this was a preliminary test used to orient and give direction to algorithm development. During these tests, the electrodes were again placed in a way that would capture the EMG activity in the flexor digitorum, as shown in Figure 6. The subject was asked to relax their hand, and then EMG recording was started. During recording, the subject was asked to clench their fist, hold their fist at a constant force until the EMG signal being recorded had settled, and then release their hand. This set of actions will be referred to as a clench-hold-release action. The subject was asked to perform several of these clench-hold-release actions as their flexor digitorum activity was recorded, and then these recorded actions were inspected in order to guide the development of the control algorithm. The signals captured during this test are presented in Figure 7.
Initially, it was supposed that acquisition of signals from across the flexor digitorum would be adequate for control of the device. Figure 7 shows the signal captured from 7 clench-hold-release actions in a row across the flexor digitorum. Visual inspection showed that it was very difficult to discern any apparent structure in the signal captured from this site. Whilst it is apparent that during the clenching phase of this action there is an elevated signal RMS, once the fist is maintained at a constant force value, then the signal appears to fairly quickly settle back down to RMS values which are extremely difficult to discern from a resting RMS value. It also appears that there is significant amounts of noise present when recording EMG signals from this location; it is suspected that there may be interference from some other bioelectric phenomenon here. Hence other sites were investigated as other sources of control signals.

Figure 8. Second configuration of EMG signal capture sites.

Figure 9. Time domain EMG signals captured during fist clenching action across the wrist.
The second site to be explored is depicted in Figure 8, and the EMG signals captured at this site are shown in Figure 9. At this site, the positive electrode is placed on the back of the hand, the negative electrode is placed on the inner forearm proximal to the wrist joint, and the negative electrode is placed proximal to the negative electrode on the inner forearm. This site attempts to target the muscle group that travels across the wrist joint, the idea being to look at the activity of this muscle group as a whole for signalling the opening and closing of the hand. In order to test this site, the same procedure was used as for the sampling of signals for the flexor digitorum. A single healthy subject was asked to put their hand at rest, and recording of this site was then started. Once recording was started, then the subject performed a clench-hold-release action. For the sake of consistency, the same test subject was used as the tests done on the flexor digitorum. The full set of signals captured at this site are presented in Annex A. As shown in Figure 9 and Annex A, the signals captured at this site showed a much more defined structure than those captured from the flexor digitorum. During the resting phase of the subject’s action, we see a very low signal RMS, which we can consider to be our baseline EMG signal. As the fist is actively clenched, we see a dramatic elevation of the signal RMS value. Then as a constant fist force is maintained we can see the signal settling back down to a still elevated RMS value that is still lower than an actively clenching fist. Then as the hand is relaxed, we see the RMS value settling back down to the same pre-contraction levels. It should also be noted that there appears to be far less noise present in these captured signals. This site provides a signal with a very clear structure, with obvious differences around different phases of a clench-hold-release action. Therefore, this site presented a very attractive input around which we can build a control algorithm. However, it was also noted that this myoelectric site is not very useful to transradial amputees, as it captures signals across an intact wrist joint. Therefore a slight modification to this site was investigated as shown in Figure 10:
By modifying the electrode position as shown above, we are able to generalise the device to a wider range of transradial amputees. The tests done for the previous two site were then repeated for this new site as well, using the same healthy subject as before. EMG signals captured at this site (shown below in Figure 11) also still show similar structure to those captured across the wrist, and hence this configuration of electrodes looks promising for the control of our algorithm—therefore after making this finding the development of the device was continued with this electrode placement in mind.

The full range of signals captured at this modified site are shown in Annex B. Visual inspection of this collected data suggests that these signals can be broken down into three broad phases, each of which can be characterised by a band of RMS values—“Closing”, “Holding Shut” and “Resting”. Note that these RMS values are calculated based on amplified data, and hence will vary depending on how the designer positions the potentiometer installed on the SpikerShield preprocessing board.

The closing phase describes the time when the subject is actively closing their hands, and is the greatest period of activity for this myoelectric site. It is generally characterised by time domain RMS values ranging from around 100–300 for the current potentiometer setting.

The holding phase represents the subject maintaining a constant grip on an object without changing the position of the fingers. It is characterised by mid-range RMS values that are still noticeably elevated above quiescent EMG values—typically between 30 and 90.

Finally, the resting phase represents the period when the subject is not closing their hand at all, and is characterised by very low RMS values from 0 to 25.

4. Controller software
The true power of pattern recognition controllers is in the integration of multiple inputs to generate a sophisticated control output. As the current preprocessing hardware can only filter a single channel, it was there decided that pattern recognition control would be inappropriate for this device.

Given the results of the preliminary analysis, it seemed most appropriate to develop an algorithm based on two predefined thresholds that could be used to shift the controller between three different states: “Opening”, “Holding” and “Closing”. Note that the Opening state has also been further subdivided into two substates—“Slow Opening” and “Fast Opening”.

Figure 12 depicts an FSM representing the algorithm presented in this paper. The Arduino code for this algorithm can also be found in Annex C.
$S_2$ is the Holding state—in this state the controller will only maintain the current motor position and will not move the hand. The algorithm will move into this state when the RMS of the EMG signal is between the two defined thresholds. In this state, the machine also monitors the EMG signal for threshold breaks that may trigger a change in state.

$S_3$ is the Closing state. When the controller detects an RMS greater than the upper threshold of the algorithm, it will move into this state. When in this state, the EMG signal is continuously monitored and the motor will be moved a number of steps proportional to the sampled RMS value in the closing direction. It should also be noted that an encoded, predefined maximum limit for the position prevents hyperflexion of the device. Once the motor has reached this ‘hard’ upper limit of motion, it will not move any further.

$S_4$ is the Slow Opening state. In this state the motor will start a timer, and begin to open the hand in slower increments. The purpose of this state is to allow the user some small amount of control while opening their hand, thereby allowing them to make small adjustments in their grip. In this state, a timer is maintained, and if this timer runs out then the machine will move to state $S_{1.5}$.

State $S_{1.5}$ is the Fast Opening state. In this state the motor will simply be moved straight to its fully open position rather than moving incrementally as in $S_4$. After moving to the fully open position the machine will automatically transition back to $S_4$. This state was added to the machine as it was noticed that when allowing for fine opening control as in $S_4$, the motor moves incredibly slowly and allowing the user to have the option of releasing an object very quickly was desired as well. Implementing $S_{1.5}$ allowed us to effectively have 2 modes of hand opening in order to allow these two different modes of control.

It should be noted that as with the hard upper limit placed on the closure of the hand, there was also a hard lower limit placed on the position of the hand in order to prevent hyperextension of the prosthesis.

5. Testing of device operation
Once the algorithm was created, it was uploaded onto the Arduino microcontroller with preset threshold values estimated from the initial analysis of EMG signals shown in Annex B. These preset threshold values were then manually tuned until the test subject was satisfied with their ability to control the motor and the subject’s personal responses to the controller were noted.

Testing of the operation of the hardware and algorithm was in parallel with its development. This was done so that iterative design concepts could be applied in order to ensure a rapid improvement of the prototype. By doing this, we were able to provide a preliminary proof of our concept of using cheaper hardware and 3D printing technology. However it should be noted that this means that it is still necessary to perform a larger scale trial in order to provide more concrete results.

All testing was done on a single healthy subject. By doing this, we were also able to take into account the effects of “tuning” device parameters to the individual. The device was connected to the subject as described in the preliminary analysis section, was switched on, and then the subject was asked to attempt to control it. The subject would then provide feedback on how well they felt they could control it, and was asked for any improvements that could be made to the controllability of the device. After receiving this feedback, the device would be taken and the algorithm would be further developed and tuned to the individual’s preferences. By doing this, we were able to simulate the process of customising the algorithm to the individual, with a view to eventually have a user interface (UI) that allows the user to tune the devices themselves as they see fit. The individual user’s preferences should be taken into account during tweaking of the control structure in order to maximise their chances of continued use of the device.
6. Results

6.1. Operation of hardware
During testing of the steppers, (controlling them via the serial port rather than with EMG signals) it was noted that the chopper driver chip on the stepper driver board gets extremely hot. When the chip gets overly hot, the stepper torque drops down to very low levels, meaning that overheating may cause the grip to become compromised. A heatsink mounted via thermal conducting tape was attached to the driver chip in order to resolve this issue.

Early testing of the original test rig showed that when the motor was placed in a position such that the moment generated by the weight of the mock hand was at its maximum, the holding torque of the stepper motor was insufficient to maintain position and the hand would be dropped. In hindsight, this is not surprising, as stepper motors generally cannot supply large torques.

Because of this issue, during the rest of the algorithm development stage the mock hand was removed and the algorithm was tested without the motor having to move a load. Once the algorithm was developed the motor controller setup was installed into the final Talonhand. After early testing of the motor within this mechanical environment, it quickly became apparent that the stepper motor cannot provide enough torque for this application. This further demonstrates the importance of grip torque over control accuracy. The stepper motor was therefore replaced with a high torque servomotor in order to solve this issue. With this adjustment the controller was able to actuate the hand mechanism. It should also be noted that this change solved the overheating issue by eliminating the need for a chopper driver circuit. In a commercial application, it may also be preferable to use high torque micromotors due to size constraints.

6.2. Operation of algorithm
During development, the algorithm was tested on a single non amputee subject—no large scale trials have been conducted yet. Initial testing showed that there is a significant trade-off between the fidelity of control and the speed at which the motor can respond to the user’s control actions. Essentially, this means that the user must prioritise between the controllability and response time of the device.

The device was tuned in order to give what the test subject felt was the best compromise between motor response time and finesse of control. Once this was achieved, the algorithm was deemed to be able to allow for accurate control of the position of the motor, especially after the user had time to practice the operation of the device. The user reported being able to open and close the hand in an intuitive manner and was able to maintain a stable grip very easily. This was more or less a result of the iterative development of the device done with the user’s input throughout the process.

However, it was noted by the subject that the need for active clenching of the fist in order to close the hand made the controller quite tiring to use for prolonged periods, and hence the controller would benefit from further refinement in this area. This is likely to be a simple matter of needing further tuning of pre-set variables to the individual rather than the need for drastic changes to the algorithm. It was also noted that the response time of the motor when sending an open command was slower than desirable.

Finally, it was also found that the most profound shift in the balance between response time and control fidelity when using the stepper motor was achieved by changing the microstepping value of the driver. The EasyDriver is capable of Full Stepping, Half Stepping, Quarter Stepping and 1/8th Stepping, and shifting between these modes allowed for drastic changes in accuracy and response time of the control action.

7. Discussion
Considering the options available to a single channel EMG controlled device, this controller appears to give a very good degree of control of a single degree of freedom. The user is easily able to control
the position of the device such that it would allow them to intuitively interact with their environment.

Firstly, we should note the apparent tradeoff between control fidelity and response time of the device. The user can have either an extremely fine amount of control or a highly responsive device—which may be a limitation of this kind of design. Devices with more powerful microprocessors and more advanced algorithms may be able to overcome this issue, but at a greater financial cost, and a lack of accessibility to layperson makers who are the target of this device. Hence it may be more accurate to say that there is a 3-way tradeoff between cost, controllability and response time; you can optimise two but one will suffer greatly as a result. This highlights the importance of giving the user control over their device; different users may have different design priorities which may affect their continued use of the device.

Further to this point, another important piece of information we can glean from these early tests is that the ability to tune the EMG controller to the individual is paramount. Getting the right balance between response time and control fidelity based on both individual physiological characteristics and personal preference is vital to delivering the best user experience possible.

For instance, individuals have varying mean RMS values during contraction, which will need to be accounted for. However, they may also have varied priorities of control characteristics, i.e. grip stability over response time. All of these characteristics should be considered when fitting to each individual. To that end, development should be done on the device to allow users to interface with the software uploaded to the Arduino without needing to manually change the code via the Arduino IDE.

The subject of these tests also noted that opening and closing the hand tended to require much more effort than closing an actual hand would, and hence they began to fatigue quite quickly during use. This is a concern, given that it presents significant barrier to acceptance and use of the device. This simplest way of resolving this issue would be to lower the upper RMS threshold encoded into the device. However this would then lead to a loss of grip stability, as the motor would be more likely to interpret a steady grip EMG signal as a closing signal, and therefore may tighten its grip when the user does not want it to. A more involved and expensive way to resolve the issue would be to install small pressure sensors in each finger. These pressure sensors could then be used to provide grip force feedback to the controller and hence prevent overexertion of grip force.

The effect of different multistepping levels presents an interesting option for modal control—for instance one could theoretically switch between larger and smaller step sizes in order to move the motor faster or more accurately as the situation dictates, rather than having to explicitly change the speed of the motor. However, in order to really be able to make use of this functionality it is likely that the device would require more EMG inputs to allow for accurate assessment of user intent, and therefore ensure appropriate mode switching.

The slow response time during hand opening is an unfortunate side effect of using timeouts to switch between fine control of slow and fast opening. As long as the device has only a single EMG channel available for controller input, it is difficult to see how this issue could be resolved. Hence it would be desirable to include at least one more EMG signal input into the design. This again further highlights the tradeoffs involved when trying to minimise the cost of these devices. However, it would be interesting to attempt to port a multichannel control device to the Arduino platform in order to try and circumvent this issue.

It should be noted that the options open to this device are extremely limited by the choice of pre-processing hardware. Whilst for the early establishment phases of the project a single input EMG channel is still interesting in order to study the control of a single degree of freedom, as development moves forward into more complicated devices this limitation will need to be addressed.
The other notable behaviour of the device is the overheating of the stepper driver chip. This is a natural side effect of the way chopper drivers work—they constantly have electrical current flowing through them, and hence tend to get very hot (Schmalz, 2015). In order to prevent chip damage from overheating, the chip used in this board does have an inbuilt thermal regulator which prevents current flow once its temperature reaches 165°C (Schmalz, 2015), which explains why motor torque disappears when the device overheats. The installation of a small heatsink onto the driver chip was performed, and essentially fixed the issue of stepper motor cutout. However it should be noted that this point is essentially moot as the stepper motor use in this device was incapable of providing enough torque to actuate the hand mechanism in the first place.

Finally, it should be noted that the torque that can be supplied by the motor is a significantly more important consideration than was previously anticipated. At the beginning of the design phase, accuracy of position control was prioritised over motor torque. This was an error in judgement— in terms of hardware installation is necessary to look at higher torque than accurate position control. This issue was solved by switching the stepper motors out for high torque servomotors, and adjusting the Arduino code accordingly. Once this change was made, the device was able to actuate the hand device and the user reported being able to control the hand quite well.

It is also possible to use brushed DC micromotors such as those used by Cipriani, Controzzi, and Carrozza (2011). Whilst these motors can provide large amounts of torque compared to their size, they are quite expensive. Further adding to their expense is the necessity of rotary encoders in order to use these motors for position control. Furthermore, these motors also require mechanical stops to be integrated in order to allow for stable grip maintenance. Therefore, it is preferable to first investigate the use of cheaper servomotors, in order to achieve the original goal of producing a low cost design.

Whilst this pilot study of this device does look promising, clearly more work needs to be done both in terms of device development. However it appears that it would be preferable to do more development work before the device is made available to the public. For instance, doing further exploration into the viability of the Arduino in implementing multichannel pattern recognition algorithms is recommended before making the control algorithm publicly available.

One likely additional cost to this device that has not yet been addressed is the issue for regulatory approval. As neither the Talonhand mechanism nor the Arduino, or the control algorithm presented have gone through regulatory trials as medical devices, it will be unable to be used in a clinical context until regulatory approval has been given. Currently, the schematics for the Talonhand device have been made available without regulatory approval. It is essentially up to individual user to seek out, manufacture and use the device, and code could be made available online in a similar fashion. However, regulatory costs are likely to be an issue, depending on the exact requirements of specific regulatory agencies around the world. The application of regulatory laws to open source devices does raise an interesting question— if an individual goes out and collects various pieces online to create a device themselves, are the individual pieces subject to market approval, even if they aren’t sold on the market? At any rate, there likely will need to be regulatory costs factored in to the device’s current cost, and these costs will vary depending on and the type of approvals that are required. This of course will in turn affect the viability of the device. The altruistic nature of making schematics and code freely available becomes much harder to justify if the significant cost of regulatory is applied.

Finally it is interesting to examine where this device could potentially fit in with the landscape of currently available prosthesis technologies. Clearly, this device is not as sophisticated as commercially available devices such as those offered by Otto Bock or Touch Bionics. However, the real strength of this device lies in its accessibility to people who may not have the resources to afford an expensive, commercial device. By providing people the resources to relatively easily manufacture your own prosthesis, essentially only for cost of materials to yourself, it is possible to give people with lower incomes access to these technologies.
The use of 3D printing will likely remain an interesting avenue from both a manufacturing and distribution perspective for more high-end devices as well, however. Another potential benefit of 3D printing of prosthetic devices in a commercial setting is the ability to make adjustments to a devices design. If a clinician has access to a modifiable digital schematic of a device, they can help modify and customise the device to their patient’s needs. Similar technologies have been leveraged with good success in the creation of prosthetic sockets. In this application, the patient’s amputated stump can be laser scanned, and an accurate copy of the stump can be made with an autoclaver. The autoclaved copy of the stump can then be used to vacuum form plastic material into a mold, which can be used to create the prosthesis socket. In a similar fashion, a digital schematic of a hand prosthesis design could be made available to prosthetists, and then they would be able to vary the length, end effector size, and other physical attributes before printing it. By doing this we could allow for greater customisation in prosthetic devices, which in turn should promote better uptake of upper limb prostheses. It would also allow for more rapid creation of devices for juvenile users, who tend to grow very rapidly. As a device can be quickly outgrown by a child in a growth spurt, it is clearly a benefit for a device to be able to be quickly remade to suit the user if necessary.

8. Future work

8.1. Development of hardware
Since the device is currently limited by its pre-processing hardware, the focus should then be on increasing the number of input channels available to the device. This means that the development of a new Arduino Shield designed to filter and amplify multiple EMG inputs would be a very interesting area for expansion of this project. It should be noted that the makers of the single channel shield have very recently developed a new shield that does exactly this. However, this new board incorporates components that are not strictly necessary to this project (such as an audio output), it would be interesting to develop a custom shield that incorporates a stepper motor driver and a multichannel signal preprocessor.

Considering that Arduino Uno compatible boards have six analog input pins available to them, it would seem reasonable for such a shield to have four or five EMG pre-processing channels available, leaving 1–2 pins available for other inputs such as a rotary encoder or current monitor. This could be done using any company that offers Printed Circuit Board (PCB) manufacturing and would offer a much cheaper alternative that suits this device much better than more generalist commercially available devices.

Finally, it would be desirable to install some form of sensory feedback. There are several different options to explore, including rotary encoders to allow for accurate position feedback for motor control, or the inclusion of pressure sensors in the fingers of the Talonhand. Pressure sensors would allow for signalling contact with an object that the user is trying to grip, thereby allowing for enhanced grip control. However, careful attention should be paid to the cost of additional sensors, as they could be costly- defeating the purpose of a budget device.

8.2. Future software development
Once the hardware problems described earlier are solved, it then becomes possible to explore more sophisticated control algorithms. With two or more EMG channels available, the necessity for time-out based opening of the hand disappears, since it becomes possible for control of opening and closing of the hand via comparison of the activity of different muscles in a similar fashion to the FSM presented by Carrozza et al. (2005).

With multiple channels it also then becomes interesting to look at pattern recognition based controllers- with a single input available the power of these methods isn't unlocked, but when using several channels they become very impressive. There are already prebuilt libraries available that allow for FLC (Alves, 2012) and ANN (Anon, 2015) programs to be implemented on the Arduino Uno board. Therefore it would be extremely interesting to take advantage of these available tools and see
the extent of the Arduino’s capability as a myoelectric controller. It has been shown that by incorporating a FLC based controller using four inputs (which would be easily achievable given the four channel pre-processing shield suggested earlier) was capable of classifying five different intended motions to very high degrees of accuracy (Ajiboye & Weir, 2005). Hence forward, given the availability of the eFLL on the Arduino, FLC seems like a very attractive direction for future development of this device.

Also, along similar lines to the development of pattern recognition controllers, the incorporation of feedback sensors would also require incorporation into to the algorithm. This would likely involve placing relatively simple checks on these sensors in order to ensure the motor is not forced to grip objects too strongly or to allow for recovery accurate recovery of position in the event of the stepper motor skipping a step via the use of a rotary encoder.

9. Conclusions
Whilst this single channel myoelectric controller does pose an interesting basis for future development of budget myoelectric controllers, more work needs to be done. Primarily, pre-processing hardware that can interface with cheap, open source microcontrollers in order to provide the number of EMG signals necessary in order to provide the levels of functional control and sensory feedback that amputees desire from a prosthesis must be developed. However, once this is achieved, the Arduino platform promises to be a very interesting prospect for affordable myoelectric controllers in the future.

Nomenclature

Degrees of freedom
The number of independent states that governs the state of a system.

Transradial amputation
An amputation of the forearm below the elbow.

Artificial neural network (ANN)
A family of statistical learning models inspired by biological neural networks. Well known for their application in machine learning.

Fuzzy logic control (FLC)
A form of logic that allows the truth to vary continuously between 0 and 1, thereby allowing for the concept of partial truth.

Corner frequency
The frequency at which a signal passed through a filter is attenuated by at least 3 dB. Also called cutoff frequency.

Flexor digitorum
The extrinsic flexor muscle of the fingers, primarily responsible for closing the fingers.

Fast fourier transform (FFT)
An algorithm that computes the frequency domain Discrete Fourier Transform of a sequence in a much faster and computationally efficient manner.

Acknowledgments
The authors of this paper would like to thanks Bill Ingram for his assistance in building mechanical models for use in testing motor action during the algorithm’s runtime, and Andrew Wallace for his help and advice in incorporating feedback sensors.

Authors would also like to thank Mark McDowall and the Dunedin Limb Centre for their insights into current clinical practice in supply and manufacture of prostheses for patients.

Finally, we would also like to thank the University of Otago Department of Physiology for supplying surface electrodes for use in prototyping.

Funding
The authors received no direct funding for this research.

Author details
Scott A. Curline-Wandl1,2
E-mail: s_wandl@yahoo.com.au
M. Azam Ali1
E-mail: azam.ali@otago.ac.nz

1 Bioengineering Research Team, Department of Applied Sciences, University of Otago, Dunedin, Otago, New Zealand.
2 School of Mechatronic Engineering, University of New South Wales, Sydney, NSW, Australia.

Citation information
Cite this article as: Single channel myoelectric control of a 3D printed transradial prosthesis, Scott A. Curline-Wandl & M. Azam Ali, Cogent Engineering (2016), 3: 1245541.
References

Ajiboye, A. B., & Weir, R. F. (2005). A heuristic fuzzy logic approach to EMG pattern recognition for multifunctional prosthesis control. IEEE Transactions on Neural Systems and Rehabilitation Engineering, 13, 11–291.

Alves, A. J. (2012). eFLL- A fuzzy logic library for arduino and embedded systems. Robotic Research Group, State University of Piaui. Retrieved from http://www.zerokol.com/2012/09/arduino/fuzzy-fuzzy-library-for-arduino.html

Anon. (2015). A neural network for Arduino. Retrieved August 28, 2015, from http://robotics.hobbizine.com/arduinoann.html

Asghari Oskoei, M., & Hu, H. (2007). Myoelectric control systems—A survey. Biomedical Signal Processing and Control, 2, 275–294. http://dx.doi.org/10.1016/j.bspc.2007.07.009

Biddiss, E., Beaton, D., & Chau, T. (2007). Consumer design priorities for upper limb prosthetics. Disability and Rehabilitation: Assistive Technology, 2, 346–357. http://dx.doi.org/10.1080/17483100701714733

Binkley, P. (2015). Talon Hand 2.0 (now 2.7). Retrieved August 12, 2015, from http://www.thingiverse.com/thing:229620

Carrozza, M. C., Cappiello, G., Stellin, G., Zaccone, F., Vecchi, F., Micera, S., Dario, P. (2005). On the development of a novel adaptive prosthetic hand with compliant joints: Experimental platform and EMG control. In Intelligent Robots and Systems (pp. 3951–3956). 2005 IEEE/RSJ International Conference on (IROS 2005). Pisa.

Cipriani, C., Controzzi, M., & Carrozza, M. C. (2011). The SmartHand transradial prosthesis. Journal of NeuroEngineering and Rehabilitation, 8, 29–29. http://dx.doi.org/10.1186/1743-0003-8-29

De Luca, C. J., Kuznetsov, M., & Roy, S. H. (2010). Filtering the surface EMG signal: Movement artifact and baseline noise contamination. Journal of Biomechanics, 43, 1573–1579. http://dx.doi.org/10.1016/j.jbiomech.2010.01.027

Fougner, A. L., Stavdahl, Ø., & Kyberd, P. J. (2014). System training and assessment in simultaneous proportional myoelectric prosthesis control. Journal of NeuroEngineering and Rehabilitation, 11, 75–75. http://dx.doi.org/10.1186/1743-0003-11-75

Huang, S., Wensman, J. P., & Ferris, D. P. (2014). An experimental powered lower limb prosthesis using proportional myoelectric control. Journal of Medical Devices, 8, 024501. http://dx.doi.org/10.1115/1.4026633

Hudgins, B., Parker, P., & Scott, R. N. (1991). The recognition of myoelectric patterns for prosthetic limb control. In Engineering in Medicine and Biology Society. Proceedings of the Annual International Conference of the IEEE (Vol. 13) Fredericton.

Hudgins, B., Parker, P., & Scott, R. N. (1993). A new strategy for multifunction myoelectric control. IEEE Transactions on Biomedical Engineering, 40, 82–94. http://dx.doi.org/10.1109/10.204774

Jingdong, Z., Xie, Z., Jiang, L., Cai, H., Liu, H., Hirzinger, G. (2006). A five-fingered underactuated prosthetic hand control scheme. In Biomedical Robotics and Biomechatronics. The First IEEE/RAS-EMBS International Conference on BioRob 2006, Harbin.

Liu, J., Ying, D., Rymer, W., & Zhou, P. (2015). Robust muscle activity onset detection using an unsupervised electromyogram learning framework. PLOS ONE, 10, e0127990. http://dx.doi.org/10.1371/journal.pone.0127990

Schmalz, B. (2015). Easy Driver Examples. Retrieved August 27, 2015, from http://www.schmalzhaus.com/EasyDriverExamples/EasyDriverExamples.html

Segil, J. L. (2014). Development and validation of a postural controller for advanced myoelectric prosthetic hands in Mechanical Engineering. Boulder, CO: University of Colorado at Boulder.

Smith, L. H., Kuiken, T. A., & Hargrove, L. J. (2014). Real-time simultaneous and proportional myoelectric control using intramuscular EMG. Journal of Neural Engineering, 11, 066013. http://dx.doi.org/10.1088/1741-2560/11/6/066013
ANNEX A

Results of signal capture from across the wrist joint using Arduino in time and FFT domain
ANNEX B

Results of signal capture from modified cross radial myosite using Arduino in time domain
ANNEX C

Arduino program for single channel electromyogram control of stepper motor
#include < AccelStepper.h>

#include < MultiStepper.h>
define DIR_PIN 2
define STEP_PIN 3
define MS1 4
define MS2 5
define ENABLE 6

float reading(Huang et al., 2014); //buffer for holding sample window
int window = 10; //sampling window size
float rmsReading; //Root Mean Square of our signal
float constrainedRMS; //variable for rms value constrained to expected range
int maxRMS = 500; //largest expected RMS value
int maxSteps = 600; /*maximum number of steps we want the stepper to do in a
single movement- before changing microstep this was 500*/
int maxOpenSteps = 10;
int maxPosition = 100; /*hard upper limit for motor movement- motor should not
move beyond this point*/
int minPosition = 0;
int openCount = 0;

int closeThreshold = 200; //upper rms threshold to signal closing state
int openThreshold = 40; //lower rms threshold to signal opening state

long newSteps = 0;

AccelStepper stepper(AccelStepper::DRIVER, DIR_PIN, STEP_PIN);

//setup LED pins for state visualisation
int openLed = 8;
int openLed2 = 9;
int holdLed = 10;
int holdLed2 = 11;
int closeLed = 12;
int closeLed2 = 13;

void setup() {
  Serial.begin(9600);
  /**<according to our pull out torque curve, this means the motor can supply about
  * 7 N.cm of torque at max speed
  */
  stepper.setMaxSpeed(500);
}
stepper.setAcceleration(500);

//setup LEDs for state visualisation
pinMode(openLed,OUTPUT);
pinMode(holdLed,OUTPUT);
pinMode(closeLed,OUTPUT);
pinMode(openLed2,OUTPUT);
pinMode(holdLed2,OUTPUT);
pinMode(closeLed2,OUTPUT);

//ED_v4 Step Mode Chart

// MS1 MS2 Resolution
// L L Full step (2 phase)
// H L Half step
// L H Quarter step
// H H Eighth step

//

// set microstepping mode according to chart above
pinMode(MS1,OUTPUT);
pinMode(MS2,OUTPUT);
digitalWrite(MS1,LOW);
digitalWrite(MS2,HIGH);
}

void loop() {

  //take ten readings in ~0.02 s
  for(int i = 0; i < window; i++){
    reading[i] = analogRead(A0);
delay(2);
  }

  /*calculate the rest of our signal stats*/
  for(int k = 0; k < window; k++)
  rmsReading += reading[k]*reading[k];
}
 /*for some unknown reason this squared number occasionally comes back as negative
 * despite nominally being a square number- which breaks the sqrt function being used
 * to finish the RMS calculation. In theory taking the absolute value of a number
 * which has just been squared is redundant- but this is necessary
 */
 rmsReading = sqrt(abs(rmsReading/window));

 Serial.println(rmsReading);

 if(rmsReading < openThreshold){
 //set state visualisation LEDs
 digitalWrite(openLed,HIGH);
 digitalWrite(openLed2,HIGH);
 digitalWrite(holdLed,LOW);
 digitalWrite(holdLed2,LOW);
 digitalWrite(closeLed,LOW);
 digitalWrite(closeLed2,LOW);

 //map RMS to step number space
 constrainedRMS = constrain(rmsReading,0,openThreshold);
 newSteps = map(constrainedRMS,0,openThreshold,0,maxOpenSteps);

 //send hand to open position
 stepper.move(-newSteps);
 if((stepper.targetPosition() < 0) || openCount > 3){
  stepper.moveTo(minPosition);
 }

 stepper.runToPosition();
 openCount++;

 Serial.print("Current Position: ");
 Serial.println(stepper.currentPosition());
 }
 else if((rmsReading > openThreshold) && (rmsReading < closeThreshold)){
 //don't move, just update LEDs
 digitalWrite(openLed,LOW);
 digitalWrite(openLed2,LOW);
 digitalWrite(holdLed,HIGH);
 digitalWrite(holdLed2,HIGH);
 digitalWrite(closeLed,LOW);
 digitalWrite(closeLed2,LOW);

 openCount = 0;
 }


else if(rmsReading > closeThreshold){

    //update LEDs
    digitalWrite(openLed,LOW);
    digitalWrite(openLed2,LOW);
    digitalWrite(holdLed,LOW);
    digitalWrite(holdLed2,LOW);
    digitalWrite(closeLed,HIGH);
    digitalWrite(closeLed2,HIGH);

    //do closing
    //map RMS value from RMS space to step space
    constrainedRMS = constrain(rmsReading,closeThreshold,maxRMS);
    newSteps = map(constrainedRMS,closeThreshold,maxRMS,0,maxSteps);

    //print results of this for serial debugging
    Serial.print("Steps Moved: ");
    Serial.println(newSteps);

    /*set new stepper position. restrict position to within a half revolution
     * of the stepper
     */
    stepper.move(newSteps);
    if(stepper.targetPosition() > maxPosition){
        stepper.moveTo(maxPosition);
    }

    //move stepper to new position
    stepper.runToPosition();
    openCount = 0;

    Serial.print("Current Position: ");
    Serial.println(stepper.currentPosition());
}
}
