Development of a Dual-Actuator Wave Robot - DAWR

Philip Bailey*, Arjun Patel†, Kaustubh Jalundhwala‡, Damon Roberts§, and Luke von Freeden¶
Department of Electronics and Computer Science, University of Southampton
Email: *pab1g14@soton.ac.uk, †ap5g14@soton.ac.uk, ‡ksj1g14@soton.ac.uk, §dr5g14@soton.ac.uk,
¶lcvf1g14@soton.ac.uk

***Abstract*—This report describes the design and build of a remotely controlled robot which takes advantage of a novel locomotion technique inspired by eels. The robot is designed to be capable of traversing complex terrain using a minimal number of drive motors to create wave like motion. The robot developed successfully demonstrates the design concept. Due to budget, time, and manufacturing limitations the robot remains too unreliable for deployment.**

I. Introduction

Zarrouk et al. [1] developed a mechanism which is able to create wave like motion using a single actuator to propel a robot in the $10\,\mathrm{cm}$ scale. The design uses a 3D printed spiral and segments to create a snake like body. Passive wheels were used for smooth motion and steering [1]. The motivation for the use of a single actuator was to allow their design to ultimately be miniaturised to the $1\,\mathrm{mm}$ scale for use in invasive medicine.

We have extended their concept in a different direction. We have created a purely wave-propelled robot for use on terrain where wheels are unsuitable whilst minimising actuator count and therefore cost. We use two wave sections which can move relative to one another providing steering, and a rearing motion to traverse obstacles.

A. Biological Inspiration

Snakes and eels both use wave like motion and provide part of our biological inspiration. The motion of eels is termed "anguilloform". In anguilloform motion a third or more of the body is undulated to give forward propulsion [2]. In water, the body must form at minimum, one complete wave cycle so that lateral forces are balanced. Due to this force balance, anguilloform motion is effective in any plane. This fact was used within this project to support the approach taken to oscillate the body vertically rather than laterally as snakes do [3]. Eels have been known to move across land for brief periods using motion similar to that used in water: demonstrating that an amphibious robot could be created that uses the same movement on land and in water.

The hydrodynamics of aguilliform motion is not fully understood but is commonly explained by considering elements as shown in Figure 1. It can be seen here that at each point in time there are two main elements that are responsible for generating the forwards motion. When on land, these elements are the portions of the eel that are touching the ground at any one moment. This movement can be compared with the locomotion of snakes. The movement of eels through water creates very little wake, as the wave propagates along the body of the eel it creates areas of high and low pressure which are crucial for its movement through water [5]. Figure 2 shows a 2D image of the resulting vortices created in an eels motion. This pattern is indicative of high efficiency [6].

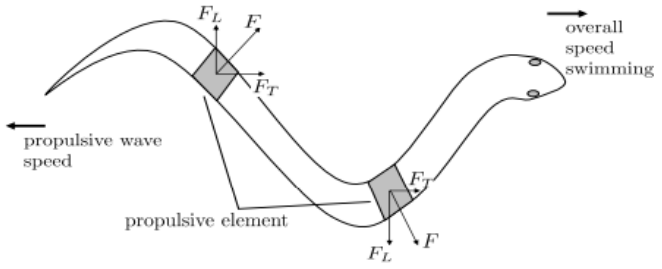


Fig. 1: Propulsion elements at a single moment in an eel's motion in water [4].

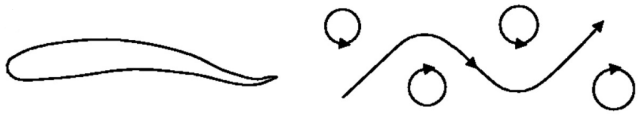


Fig. 2: Wake of an eel's movements displaying a double row of single vortices [6].

There are many important parameters to consider when imitating the slithering of an eel, complex kinematics are necessary to fully describe the undulation used by snakes and eels. This motion, however, can be simplified while maintaining efficiency. To slither efficiently, a snake reduces its contact with the ground as much a possible; they lift their weight to assist their motion. A two dimensional sinusoid is responsible for the large lateral movement of the body but another wave travels in the vertical plane along the body as it reduces contact with the ground [7]. The interface with the ground is also a key aspect of the locomotion. The benefits of snake scales, which only slide easily in one direction, can be incorporated into a robot designed to operate in an environment that includes water as well as land. An eel cannot scale steep inclines, whilst snakes use their scales to assist them in gripping the surface beneath them [7].

The efficiency of undulatory motion is a huge benefit that can be exploited with development of anguilloform robots. The range of surfaces and inclines that can be tackled by such a robot would be a key unique feature. Particularly, this type of robot would fit within a swamp like environment perfectly, as it would be able to cross areas of water and land with no change to its motion.

II. Body

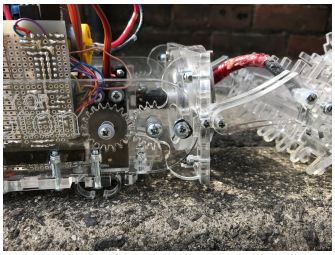
The body is constructed primarily from laser-cut acrylic. It has three main components; a rotational stage which facilitates the yawing of the rear tail, a pitching stage which facilitates the lifting of the front tail and the main body of the DAWR which supports both tails and the electronics.

A. Rear Tail

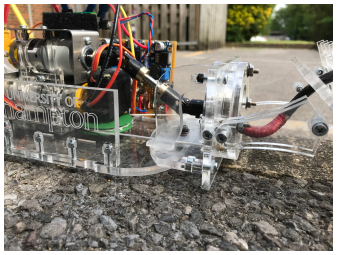
The rear tail acts like the rudder on a boat; when rotated to the left the DAWR yaws right and vice-versa. This rotational stage is actuated by a single servo which uses a string and pulley-wheel mechanism. The stage is attached to the main body by a single bolt with a bearing to ease actuation; rotation is limited by circular slot and a bolt.

B. Front Tail

The front tail allows the robot to cross obstructive terrain and mount obstacles. Actuation of the pitching stage is facilitated by a pair of servos which use a set of laser-cut acrylic gears with a 1:1 ratio, to rotate the head-piece. The gears allow the servos to be mounted far enough from the front plate to allow sufficient rotation. A servo is used either side to minimise torsional stress across the front plate and ensure smooth rotation of the pitching stage.



(a) The laser cut gears used to lift the front tail of the DAWR .



(b) The rear tail, notice the string connecting the servo and the tail piece.

Fig. 3: Rotary stages

C. Main body

The main body supports the two drive motors, three servos, battery and electronics and is made from laser-cut acrylic. The two drive motors are stacked, one atop the other, to reduce the length of the main body. They are held together using a laser-cut acrylic structure bolted to the base plate. With an increased budget it would be possible to further reduce the length of the main body by replacing the brushed DC gearbox motors with a power dense alternative, such as brushless DC motors or an active differential planetary gearbox. This would also allow reduction in the height of the robot; if the height of the main body was less than that of the tail it could be possible for the DAWR to function if it were upside down.

III. The screw

The screw is a three-dimensional helix with a pitch of 100mm and a length of 225mm. The screw was 3D-printed out of a special polylactic acid (PLA) filament impregnated with 15% by weight of short carbon fibres. The printed cross-section of the screw is 8mm and the inner diameter of the helix is 100mm. Each screw is supported by a large bearing which is mounted at the extremes of the main body; in addition to mechanical support, the bearing ensures smooth rotation for efficient power transfer. The front screw is connected to the drive motor by a flexible coupling which allows for efficient power transfer at any angle up to 45^o; greater angles are achievable though undesirable. The centre of flexion of the coupling is aligned with the centre of rotation of the gear mechanism of the pitching stage, which is essential for adequate function. The rear screw is connected to its drive motor using two flexible couplings. This is necessary to allow the axis of the motor to be translated with respect to the screws. As with the front screw the centre of flexion of the screw-end coupling is aligned with the centre of rotation of the yawing stage.

The 3D printable carbon fibre material was not provided for free, unlike the PLA and acrylic, thus only limited experimentation with the material could be conducted. Before commissioning the production of the screw in carbon fibre, a number of PLA models were printed to test the function and geometry. Unfortunately, one week before the deadline the rear screw was damaged; as with all 3D printing the inter-layer adhesion is a major point of weakness and it was this that lead to the screws failure. In order to print a screw which was long enough for the design it had to be printed lying down, as such there are multiple sections of the screw where the cross-section between layers is small, and thus weak. To help reinforce the screw, polyester thread was wound around sections of the screw and soaked in epoxy-resin.

IV. The Tail

A. Aims

The tail must polarise the motion of of the screw in order to move in a wave motion. To create the flexible tail, it is split into multiple repeating sections along the whole length of the screw. Increasing the number of pieces makes the motion more fluid and also will impart a more uniform force onto the ground. To create the wave motion from the helix, at each point along the screw the pieces should be free to move vertically, however they should be locked from moving left and right. This will result in the constrained pieces following a transverse wave path as the screw moves. Figure 4 shows a cross section and side view of the screw and a single tail piece.

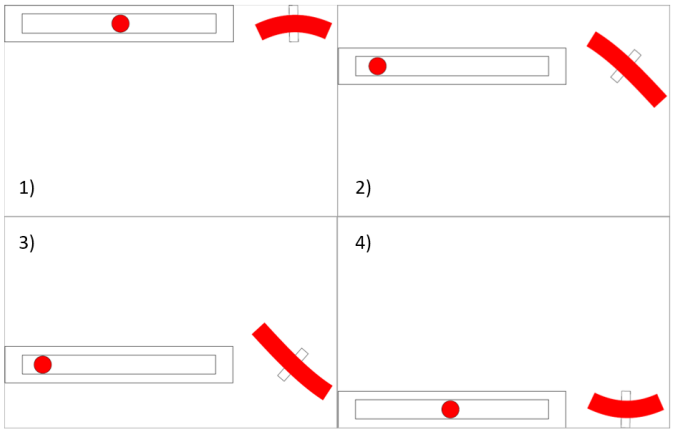
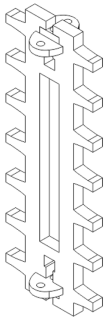
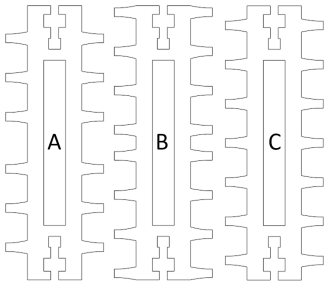


Fig. 4: Cross section and side view of screw with a single tail piece showing its motion through half a rotation of the screw (tail piece is white and screw in red).



(a) Design assembly of an individual section of the tail.



(b) Three different section designs used along the tail in order A-B-C.

B. Design and Build

The tail was chosen to be made from acrylic as prototypes can quickly and easily be made through laser cutting plus it allows for strong precise pieces. Due to some of the limitations of laser cutting, novel methods were used to connect the sections. The sections were designed in SOLIDWORKS, where the designs could easily be exported and printed. As the laser cutter can only cut in a single plane each connecting section was made from five separate pieces, one main section and four connecting pieces. An example of an assembled section is shown in Figure 5a.

The main pieces are made from 5mm acrylic so they do not flex and snap under the force imparted onto them. The connectors are made from 3mm acrylic as the extra strength is not needed and saves space. The screw runs through the centre slot, which is the same width as the diameter of the helix so that the screw can freely move horizontally. The slot is slightly taller than the swept circle of the screw allowing the sections to be pushed vertically whilst rotating as it moves. The connectors are attached to the main body using M3 bolts and nuts locked into place by the acrylic. The sections are connected together using a loose nut and bolt so they can rotate freely. The length of the connectors are tuned to give an optimal relationship between the spacing of the sections and angle they can rotate without clashing. The sections only need to rotate enough to follow the line of the helix closely to create a more fluid movement and provide a consistent driving force.

Tall main pieces create a larger moment between the edge in contact with the ground and the centre of rotation, thus providing more force. Taller pieces also provide more ground clearance for the robot. However, taller pieces have less space to rotate around the screw. To combat this, three different tail sections were designed which can overlap without clashing, shown in figure 5b. This has the benefits of increased height but can still rotate to a sharp enough angle to move along the screw. Small curves are present on the edges of the meshing legs to increase the strength, as testing showed that sharp 90 degree corners made the legs prone to snapping off. The tail is built with a repeating A-B-C pattern thus at all points the three pieces are able to mesh.

Acrylic is a relatively low friction material, and laser cutting creates sharp 90 degree corners which means there would be a small, low friction contact point between the tail pieces and a hard, smooth ground surface. To improve the robot's traction small dots of glue were added to each of the legs. This increases the friction so the legs do not slip. Additionally, the inside edges of the sections in contact with the screw were sanded down. This reduces 'grating' along the screw as it is not perfectly smooth due to the printing method. The sanding prevents the edges from jamming against the screw and lowers the maximum torque requirements of the motor. Finally, the tail is connected to the body by long connector arms which increases the distance from the first tail section to the body allowing the section to move freely. These are also designed in such a way to prevent the piece from over rotating.

V. Electronic Design

The robot contains two DC motors to drive the main screws. There are two servos to pitch the front screw, and a single servo to yaw the back screw (1501MG, Power HD). The control signals are provided using a Raspberry Pi.

A. Power

The robot is powered using a a 4-cell 16.8 V, 2.2Ah lithium-polymer battery. The battery is fitted with a battery alarm which provides an audible output in case of under-voltage. The DC motors are powered directly from the battery. The other components are powered from the battery via pre-built, adjustable voltage buck converters (LM2596 DC to DC High Efficiency Voltage Regulator, Valeford). Three buck converters are used: 1) powering the raspberry pi at 5.2 V, 1 A; 2) powering the two pitch servos at 6 V, 1.9 A; 3) powering the single yaw servo at 6 V, 1.5 A. The servos are split between two buck converters to ensure that the converter's 3 A current limit is not exceeded. The buck converters were purchased as a pack making it cheaper to power the servos using two separate buck converters than one with a higher current capacity.

B. Actuator control

The motors are controlled using an integrated motor driver chip (TLE2062, Infineon), as shown in Figure 6. The chip

provides a full H-bridge allowing for forward and backward drive. PWM gate drives are used to control speed. The motor drive signals are generated using hardware timers on the raspberry pi. As a result, the PWM drive signals persist through an operating system (OS) crash. To ensure that the robot fails safe, the motor driver outputs from the raspberry pi are ANDed with the output of a re-triggerable monostable. If the monostable is in it's stable state the input to the motor driver is logic 0. The monostable is held in it's unstable state using a software toggled output from the Raspberry Pi. If the OS crashes, the pin ceases to be toggled.

The servos use a standard hobby servo interface, a $50\,\mathrm{Hz}$ PWM signal. The angle of the servos is changed from $-90°$ to $90°$ by varying the pulse widths from $1\,\mathrm{ms}$ to $2\,\mathrm{ms}$.

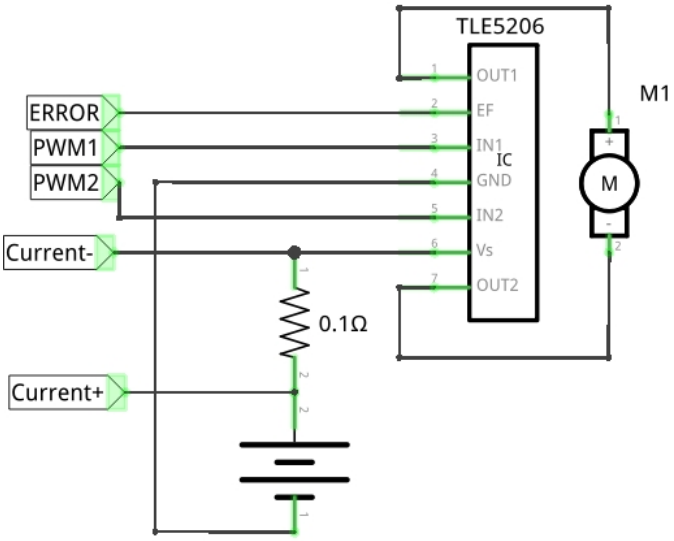


Fig. 6: A single motor drive circuit. Logic 1 on the input pin results in the corresponding output pin being connected to V_s. The outputs `Current+` and `Current-` are connected to a current sensing chip as in Figure 7.

C. Current Monitoring

The current draw by the motors and servos is monitored. Monitoring the current gives feedback about the amount of torque that the actuators are applying, enabling the controller to limit the torque to meet the mechanical constraints. $0.1\,\Omega$ resistors are placed in shunt at the output of the buck converters powering the servos and between the battery and each motor driver, giving 4 in total. The voltage across each shunt resistor monitored using dedicated current monitoring ICs (INA219, Texas Instruments). The current flowing in the resistor is derived from the voltage reading and the current readings are communicated via an I^2C interface. The bus voltage and power can also be monitored by the ICs. Figure 7 shows an example of how the current sensing chips are used.

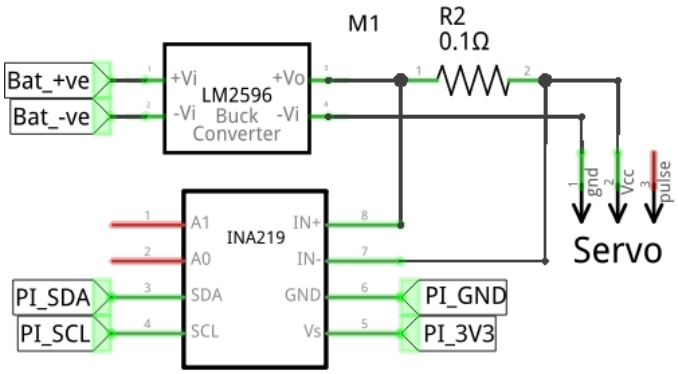


Fig. 7: The buck converter and current sensing circuit used for the servos. Two servos are used for pitch and are powered in parallel. The pins A1 and A0 are used to set the INA219's I^2C address. The servo's pulse pin is connected directly to a GPIO pin on the Raspberry Pi.

VI. The Software

The control of the DAWR is via a PlayStation® controller which interfaces with a Raspberry Pi over Bluetooth. Automation of the robot was not attempted as not enough sensing equipment could be installed with the budget constraints that were set.

A. Overall Architecture

The software is hosted on a Raspberry Pi (Raspberry Pi 3 Model B V1.2, Raspberry Pi Foundation). The majority of the programming is done in Python, with some HTML and JavaScript. The motors are controlled using a Sony Dualshock 3 PS3 controller, connected to the Pi via Bluetooth. Button inputs from the controller vary the PWM outputs on various GPIO pins using a script that runs in the background on the Pi.

Current monitoring is employed as a proxy measurement for the motor torque. The Pi is configured as a Wi-Fi access point and a simple Python micro-server package, *Flask*, was used to build a web page which displays various diagnostics from the robot, such as battery charge levels and motor currents, but can also be used for control of the robot if desired. A *SocketIO* based back-end allows near real-time updating of values asynchronously to the rest of the code without reloading the actual page.

B. Control

The Pi, using the *pigpio* [8] library, is capable of outputting separate hardware timed PWM signals on each of its GPIO pins. The PWM values are set via a daemon, meaning they can be updated from multiple programs at the same time, with the queuing of instructions handled by the Linux kernel. To receive input from the Dualshock 3 (DS3) controller, an open source driver *Sixad* [9] is used. This allows the Pi to pair with the controller as if it were a PS3. The library allows the controller to be read as if it was a standard HID (human interface device) controller. To convert the button inputs to a

useful form, a custom wrapper named *sixAxis* [10] has been used. This wrapper uses *Pygame* [11], a library of standard game related functions, to map the control inputs to a simple function one can call. *Pygame* implements an event queue for button inputs, meaning multiple button presses can be stored and computed in order. This, combined with the *pigpio* daemon allows multiple PWM values to be changed near simultaneously.

C. Internal Monitoring

The motors being used are powerful enough to snap the acrylic body, and there is a chance the screw could also be damaged if it jams. Therefore, current monitoring of the two main drive motors and the pitch control servos was implemented. The torque being applied by the motor is equivalent to the current, and hence can be used as a proxy for the actual torque. An *INA219* I^2C current monitor chip was used, as Python libraries were readily available, and I^2C simplifies the addressing of each chip. A simple supervisor script monitors the sensors. If one reads above the set threshold, power is cut to all motors temporarily. If this happens multiple times, the robot will shut down, and a manual reboot will be required.

D. External Monitoring

The web-page hosted on the Pi allows the user to monitor the robot with a phone, tablet, or any WiFi enabled device.

1) Wireless Access Point: The WiFi chip on the Pi is set to run in host mode, meaning it broadcasts a network that one can connect to. This is an ideal setup for a headless device, as anyone can connect to the network, and then either use SSH or VNC (remote desktop software) to modify scripts, debug issues, or monitor connected devices. Two standard Linux packages, *hostapd* and *dnsmasq* are used. *Hostapd* manages the hardware, and security of the access point, and *dnsmasq* manages the IP address setup for connected devices. Several networking routing rules were then put in place, which tells the Pi to route incoming traffic through it to its actual internet connection, which allows the Pi, and any connected device to access the internet via the second WiFi chip installed on the Pi.

2) Flask Server: To display data from the current sensors, and as an alternate way to control the robot a simple web-server setup was implemented. *Flask* provides a high level Python based web development environment, which is compatible with *Websockets* [12]. Together, a simple, asynchronously updating web-page can be created. This means the data on the web-page can update itself, without the client needing to reload the entire web-page. Using Pythons *threading* module, a background worker is used to collect data from the file written to by the current sensors, and send this data to the web-page. As it is setup as a separate thread the updating of the web-page doesn't become blocking, meaning the controller and sensors are prioritised.

VII. Discussion

The DAWR was designed to further explore the use of multiple wave actuators as a mode of locomotion. The aim was to produce a robot which is able to cross complex terrain, such as marshland or swamp, where other robots would falter.

The 3D printed carbon/PLA screws are not as strong as the team had first hoped; this appears not to be an issue with the material, per say, but a more fundamental issue with 3D printing. The majority of failures which occurred in the screw were due to layer separation; a result of limited adhesion between layers. Zarrouk et al also 3D printed their screw, however had a smaller robot overall and thus likely had far lower torsional forces. If the DAWR were to be remade on the same scale a sand-cast, aluminium screw might be a better option. The laser-cut acrylic is also very brittle, failures with these parts were mainly caused by the roughness of the 3D printed screw, though the pieces are likely too fragile to be used for traversing rough terrain. The design is not waterproof, and extensive development to achieve this.

VIII. Conclusion

Fundamentally the mechanism worked and the DAWR has shown that it is able to produce directional force sufficient to propel itself around and mount obstacles. The combination of a yawing and pitching stage seems to be sufficient to clear navigate across terrain. Several improvements have been outlined within the body of this report, with further funding and more development an improved design could be created that fulfils all the requirements of the system.

References

[1] D. Zarrouk, M. Mann, N. Degani, T. Yehuda, N. Jarbi, and A. Hess, "Single actuator wave-like robot (SAW): design, modeling, and experiments," *Bioinspiration & Biomimetics*, vol. 11, no. 4, July 2016.

[2] E. D. Tytell, "The hydrodynamics of eel swimming: I. Wake structure," *Journal of Experimental Biology*, vol. 207, no. 11, pp. 1825–1841, 2004. [Online]. Available: http://jeb.biologists.org/cgi/doi/10.1242/jeb.00968

[3] M. Sfakiotakis, D. M. Lane, and J. B. C. Davies, "Review of fish swimming modes for aquatic locomotion," *IEEE Journal of Oceanic Engineering*, vol. 24, no. 2, pp. 237–252, 1999.

[4] M. El Rafei, M. Alamir, N. Marchand, M. Porez, and F. Boyer, "Motion control of a three-dimensional eel-like robot without pectoral fins," *IFAC Proceedings Volumes (IFAC-PapersOnline)*, vol. 17, no. 1 PART 1, pp. 750–755, 2008.

[5] P. Cocozza, "Harder, better, faster, slipperier: why humans should swim like eels," 2015. [Online]. Available: https://www.theguardian.com/lifeandstyle/shortcuts/2015/nov/04/learn-to-swim-like-an-eel-its-way-faster-than-crawl

[6] U. K. Müller, J. Smit, E. J. Stamhuis, and J. J. Videler, "How the body contributes to the wake in undulatory fish swimming: flow fields of a swimming eel (Anguilla anguilla)." *The Journal of experimental biology*, vol. 204, no. Pt 16, pp. 2751–2762, 2001.

[7] H. Marvi, "the Role of Functional Surfaces in the Locomotion of Snakes the Role of Functional Surfaces in the," *Georgia Institute of Technology*, no. August, 2013.

[8] Joan, "pigpio library," 2018. [Online]. Available: http://abyz.me.uk/rpi/pigpio/

[9] F. Coelho, "sixad," *GitHub Repository*, 2016. [Online]. Available: https://github.com/gizmo98/sixad

[10] Spydee, "sixaxis," *GitHub Repository*, 2014. [Online]. Available: https://github.com/spydeee/sixAxis

[11] P. Shinners, "Pygame," 2011. [Online]. Available: http://pygame.org/

[12] K. F. Bates, "async_flask_2," *GitHub Repository*, 2017. [Online]. Available: https://github.com/Chainfrog-dev/async_flask_2