Developing 3D Fishtail Testing Platform for AUVs

CS39440 Major Project Report

Author: Petter Wessel Kokkim (pwk1@aber.ac.uk)

Supervisor: Dr. Otar Akanyeti (ota1@aber.ac.uk)

9th May 2022

Version 2.0 (Release)

This report is submitted as partial fulfilment of a BSc degree in Artificial Intelligence And Robotics (GH76)

Department of Computer Science

Aberystwyth University

Aberystwyth

Ceredigion

SY23 3DB

Wales, UK

Declaration of originality

I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the regulations on Unacceptable Academic Practice from the University's Academic Registry (AR) and the relevant sections of the current Student Handbook of the Department of Computer Science.
- In submitting this work, I understand and agree to abide by the University's regulations governing these issues.

Name Petter W. Kokkim

Date 09.05.2022

Consent to share this work

By including my name below, I hereby agree to this project's report and technical work being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Name Petter W. Kokkim

Date 09.05.2022

Acknowledgements

I am grateful to my supervisor, Otar Akanyetu, for his expertise in this sector, and I am thankful that this project was both enjoyable and educational.

Abstract

This paper details the work that was done to establish a platform for testing 3D printed fishtails in order to further develop fish robots or AUVs. It also discusses the hardware that was required for this application as well as the limits of some of the technology. Specifically, a stepper motor controller was created to mimic the mobility of a fish tail in order to create a realistic simulator for testing and further developing AUV fish robots. When the platform is controlled by an IoT cloud-based user interface, it allows for greater adaptability and complete control over the motion and sensor data collected by the test platform.

Contents

Table of Figures

Table of tables

1. Introduction

1.1. Background

Simply described, biomimetics is the design of something to look or function in a way that is similar to that of a living entity. When scientists discovered that creatures in nature have changed over thousands of years to adapt to their environments, they coined the term "biomimetics," which has become increasingly popular in robotics research. In addition, as we see when developing robotics for specific scenarios, the applications for various types of robots are expanding. Legs, fins, or wings would be employed to propel a biomimetic robot, whereas many robots rely on a wheel system for movement.

In Boston Dynamics' recently released Spot agile mobile robot [1], which was built as a four-legged robot that can navigate tough terrain, this may be seen in action. Any situation in which it is possible to automate regular inspection and data collecting while maintaining safety. Then, when applying this approach to this endeavour and further researching the fish, it becomes clear that the principle is correct. Movement of the fish is characterised by its speed, agility, and silence.

Using an autonomous underwater vehicle (AUV), researchers want to collect data for a number of purposes while submerged in the ocean. As a result, rather of being directed by a person, an AUV is equipped with a system that is either pre-programmed or that acts on its own initiative. Not to be confused with a ROV (remotely operated vehicle), which is attached to a ship or vessel via a network of cables that serve as the primary means of communication and control for the vehicle [2]. It is controlled by a human and typically consists of a variety of sensors and an articulating arm that can be used to remove objects from the seabed or wreckages, whereas the AUV performs the work it was designed to do and then returns to the location where it began with the information it collected.

When it came to building an AUV fish robot, Marcus Tjomsaas collaborated with Dr. Otar Akanyeti, who served as his supervisor on the endeavour. Creating a fully functional 3D printed modular fish robot[3] was the ultimate goal of this project. It was invented in 2021 with the purpose of enhancing the robot's joint and mimicking the motion of a fish, and it is still in development today. Despite the fact that this project is complete and the robot has been completed, we want to develop it and build a platform where different parts of the fish robot may be tested and evaluated. Rather than focusing on optimising each component of the fish robot individually, Otar, the project supervisor, suggested that we concentrate on developing a platform where we can test and generate different tails, as this is the robot's principal mode of operation. A platform that allows users to test various 3D printed tails that are rigged and flexible, as well as the utilisation of various 3D printable materials, is now available to users.

After following the growth of last year's fish robot, I became intrigued by the prospect of developing another fish robot. Interested parties expressed a desire to cooperate with hardware and robotics development, in which case I would be able to combine my software and hardware development experience. The construction of a fish robot with oceanic applications, as well as how this may benefit future studies in the subject, are things that I'm interested in contributing to.

1.2. Requirements

When the project's requirements were first considered, it was determined that a platform for testing 3D printed or moulded fish fins was required early in the project's development cycle. This question was posed to me by my project supervisor, despite the fact that there had been a previous project in which a fish robot had been built, as indicated in the 1.1 background.

The following are the prerequisites for the project:

- Hardware that is appropriate for platform building
- Platform design that includes housing for the hardware and sensors that will monitor the fishtail
- A modular platform that can accommodate different fishtails
- Construct the motor controller.

1.3. Analysis

Determine which piece of the fish we want to utilise as a platform for research and enhancement, and then determine which type of fish tail we want to mimic and use on this system are the two most difficult decisions to make. The type of fish tail used in the previous project, which was revealed in background work 1.1, was a forked tail, which was also the type of tail used in the project that was based on a rainbow trout, which was revealed in background work 1.2. As a result, if this project were to be expanded further, the same tail design would be a solid choice for the tail design. Given that the forked tail possesses characteristics that allow for rapid swimming and sustained speed[4], we wish to test and investigate this further with the assistance of this platform, including testing various materials while 3D printing the fish tail using flexible 3D printing, rigid 3D printing, and resin prints.

Following a thorough understanding of why we were conducting this project and where we were heading, the next stage was to examine the hardware side of the endeavour. What kind of hardware would be most appropriate and effective for this application, as well as what kinds of sensors would be necessary, are all important questions. Given that we're talking about a fishtail that's going to be submerged in water, the first notion that came to me was that there has to be some kind of waterproofing for the electronic components. This was resolved by building the platform to house the electronics above water, after which it would be necessary to construct a drive train that would connect the motor above water to the location where the fish tail would be mounted.

In order to see the fish tails, the platform will require sensors and cameras to monitor the situation. It is hoped that data from the fishtail will be collected and compared to data from other types of fishtails that will be tested on the platform in the future. It serves as a platform for the development of additional 3D printed tails for the fish robot because it is the primary source of locomotion for the fish robot. As a result, a force sensor is desired after some preliminary research into it in order to quantitatively measure the mobility of the fishtail[5].

Furthermore, the selection of an appropriate motor to act as the drive train for the fish tail would be required for any further development of hardware that would be operated on this platform in the future. Other components would be required to make everything work together, and a controller would be required to control everything. It is likely that the Arduino platform, which runs the C++ programming language and is a good programming language on which to create[6], would be the most suited controllers for this application. When it comes to low-cost controllers, the Arduino is an excellent choice because its hardware is open-source and it supports a large range of components and sensors that may be used in conjunction with the controller. Although the Arduino does not have the computing capacity that other controllers may have in plenty, this does not appear to be a significant constraint at this time. Based on the early requirements of this platform, this should not be an issue. This will be powered by a motor controller, which will be in charge of supplying the heavy lifting of power.

In order to determine which motor would be the most appropriate for this platform, a servo motor was initially considered the best option[7]. In order to simulate the movement of a fish tail, the motor will be turned on and will pulsate rather than complete a full circle of motion. The motor will then travel back and forth. However, because stepper motors were the only hardware available for use on the platform, the motor was forced to settle for a compromise and use a stepper motor. A stepper motor is used to move an object step by step on a rail system; however, this motor can be programmed to match the characteristics of the platform being utilised.

If you're looking for a platform on which to place gear, you should read up on 3D modelling before you start. It was determined that this platform would be above water to allow you to test the system and fish tail. Product design software Fusion360 will be used to create this. Fusion360 is a CAD programme for creating product designs[8]. When used in this project, it was for the creation of a 3D model of a platform that would keep the motor, Arduino, and other electronic components above water and safe.
Using Fusin360, the platform and the initial prototype of the fish tail were created. The fish tail was intended to be modelled after a forked style tail.

1.4. Prototyping Model

Because not all of the project's requirements were apparent at the outset, the prototype model was chosen for its long life cycle. Therefore, the prototype model is appropriate, in which a prototype can be built, the user may provide feedback, and the prototype can then be adjusted with a new iteration, implementing a trial-and-error process between the developer and the user. During which I serve as developer and the user serves as project supervisor. Users provided feedback during our weekly meetings, which allowed them to view the progress and then provide feedback on changes or needs, which allowed us to develop new inputs for the following iteration.

1.5. Tools

This project was completed with the help of the following resources:

Arduino IDE (Integrated Development Environment): It is common to programme the Arduino microcontroller, which was used in this project, using the open-source Arduino software (IDE), which is available for free download on the internet. Because it is open-source software, the Arduino IDE is not only confined to the Arduino microcontroller, but it may also be used with a range of other microcontrollers.

Fuison 360: 3D modelling, CAD, CAM, CAE, and PCB layout are all included in Fusion 360's product design software suite. This application was used to model the hardware platform as well as the fish tail in this particular project.

3D Printer: The Sigma BCN3D and the Renkforce RF1000 3D printers were utilised for this project, and the required 3D-modeled parts were forwarded to a contact at Aberystwyth University for printing.

Fritzing: Fritzing is an open-source project to provide amateur or hobby CAD software for the design of electrical hardware, in order to assist designers and artists who are prepared to transition from prototyping to constructing a more permanent circuit. It was created at the Potsdam University of Applied Sciences.

GitHub: GitHub was utilized for this project as a source of version control and a means to share the code, with the option to also use it as cloud storage.

2. Hardware

2.1. Hardware selection

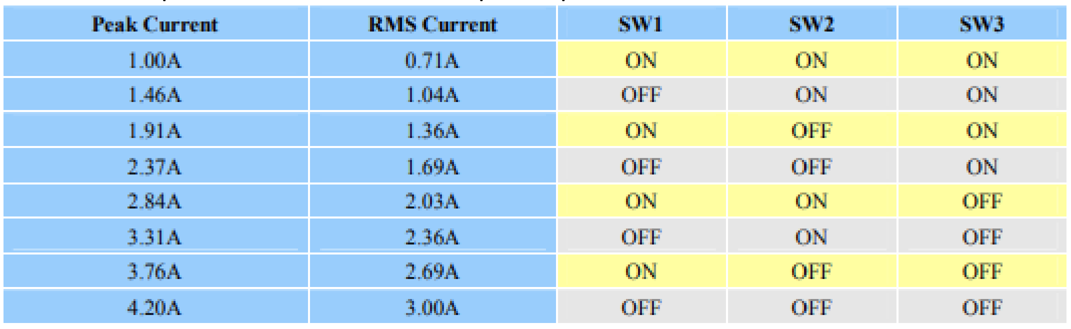
Because the purpose of this project was to build a testing platform for the fish tails, selecting the appropriate hardware was critical. Despite the fact that the first hardware options appeared to be slightly different from final hardware, the differences were in fact rather minimal. The conversion from a DC Planetary motor to a stepper motor was the most significant difference between the research hardware and the final hardware. The DC motor would have been ideal for the use for which the motor is intended, but we were forced to make a compromise due to the type of motor we had on hand.

2.1.1. Arduino Uno R3

The controller for this project is an Elegoo Arduino Uno R3 that has been coded in C++. Specifically, the ATMega16U2 microcontroller chip, which has a clock speed of 16Mhz, 32k flash memory, and 14 digital I/O pins, is utilised in the Arduino (6 PWM outputs)

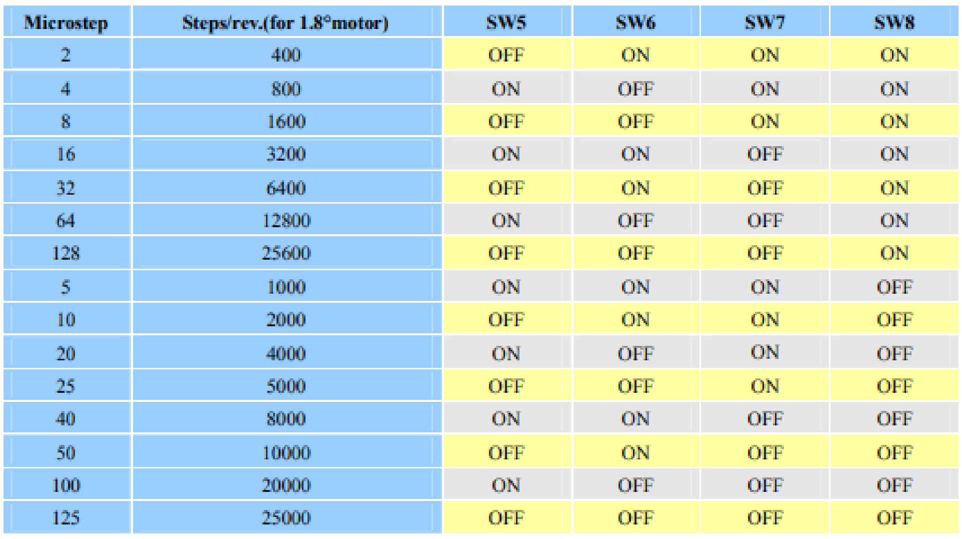
2.1.2. Microstep Driver DM542

For this project, the DM542 is a two-phase 20-50v controller with a maximum output current of 4.2A. It is the driver for the stepper motors that will be used. There are two of these running for this project, one for each of my stepper motors. The following setup is used by both drivers: peak current of 1.46A and pulses per second of 400 each.



Peak Current	RMS Current	SW1	SW2	SW3
1.00A	0.71A	ON	ON	ON
1.46A	1.04A	OFF	ON	ON
1.91A	1.36A	ON	OFF	ON
2.37A	1.69A	OFF	OFF	ON
2.84A	2.03A	ON	ON	OFF
3.31A	2.36A	OFF	ON	OFF
3.76A	2.69A	ON	OFF	OFF
4.20A	3.00A	OFF	OFF	OFF

Figure 1 Peak current setting for the DM542, where we can se the relation between Peak Current and RMS current



Microstep	Steps/rev.(for 1.8°motor)	SW5	SW6	SW7	SW8
2	400	OFF	ON	ON	ON
4	800	ON	OFF	ON	ON
8	1600	OFF	OFF	ON	ON
16	3200	ON	ON	OFF	ON
32	6400	OFF	ON	OFF	ON
64	12800	ON	OFF	OFF	ON
128	25600	OFF	OFF	OFF	ON
5	1000	ON	ON	ON	OFF
10	2000	OFF	ON	ON	OFF
20	4000	ON	OFF	ON	OFF
25	5000	OFF	OFF	ON	OFF
40	8000	ON	ON	OFF	OFF
50	10000	OFF	ON	OFF	OFF
100	20000	ON	OFF	OFF	OFF
125	25000	OFF	OFF	OFF	OFF

Figure 2 Microstep and Steps/rev setting for the DM542, where we can se the relation between Microstep and Steps/Rev

2.1.3. LIN Engineering 2A Stepper

An LIN Engineering 4118 Series WO-4119C-01 2A Hybrid Stepper motor was utilised in this application. In terms of performance, each motor has a current of 2 AMP and holding torque of 0.88Nm, as well as a Step angle of 1.8° and an axial load of 2.72kg.

2.1.4. ESP8266-01

The ESP01 is a Wi-Fi module that I'm including into this project in order to connect the Arduino to the internet. A total of nine GPIOs (General Purpose Input Outputs) are available on the ESP8266, which can be programmed to operate a variety of different hardware. However, the ESP01 that I'm using for this project only has two general purpose input/output (GPIO) pins. An integrated TCP/IP protocol stack and a 2.4 GHz Wi-Fi network with WPA/WPA2 compatibility are used by the ESP.

2.1.5. Power:

For this application, a 24V, 3A/4A power supply with a 24V to 5V stepdown is used to supply power to a variety of components on the platform.

2.1.6. Alternatives

However, despite the fact that there were alternative options, this was the hardware that was chosen for this project in the end. The alternate option, which was to use a DC motor instead of the stepper, was chosen because it was what was on hand and DC motors were difficult to come by at the time of the project. Using a stepper motor to operate with this system, on the other hand, was a compromise.

The ESP-R32 controller, which has many of the same functionality as the Arduino Uno and has a built-in WiFi module, was another choice for the controller's hardware configuration. Because the ESP8266-01 would be integrated into the controller, the ESP8266-01 would be no longer be required. With 4 MB of flash memory compared to 32kB for the Arduino Uno, and 520kB of SRAM compared to 2kB for the Arduino Uno, the R32 outperforms the Arduino Uno in terms of computational capacity as well as other factors. The R32 is programmed in the same coding language and with the same integrated development environment (IDE). However, because the requirement to swap the main controller was discovered late in the project's development, the adjustment was not put into effect.

2.2. Schematics

When the final hardware list had been completed, it was time to design the hardware layout, which included how all of the components would be connected as well as how the hardware would be connected. A programme known as Fritzing was used to develop all of the designs for the hardware layout and connections, which was detailed in greater depth in 1.5 Tools.

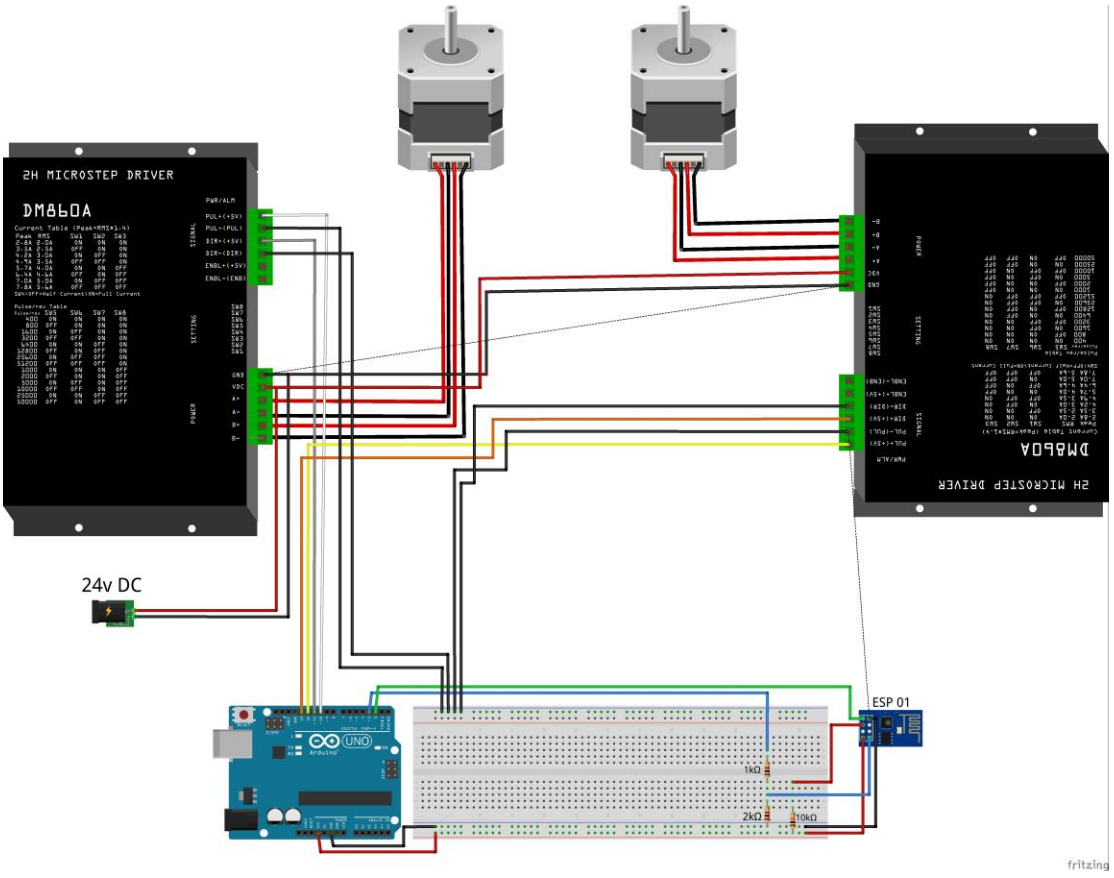


Figure 3 Breadboard view of the hardware

The platform's breadboard layout is depicted above, with two stepper motors and a DM542 being used to drive the platform. The DM860A model was utilised in the layout for demonstration purposes, however the connections are the same as on the real device. When the stepper motors are powered by a 24V power system, the connection between the stepper motors and the Arduino is made using digital pins 10 to 13. When connected to PUL+ on the driver1, digital pin 10 is connected to DIR+ on the driver1, digital pin 11 is connected to DIR+ on the driver1, digital pin 12 is connected to PUL+ on the driver2, and the last digital pin 13 is connected to DIR+ on the driver2.

Table 1 Connection table for the DM542 nr 1

DM542 Nr 1	
PUL+	Pin 10 Arduino Uno
PUL-	Ground (0V)
DIR+	Pin 11 Arduino Uno
DIR-	Ground (0V)
ENA+	Not Connected
ENA-	Not Connected

Table 2 Connection table for the DM542 nr 2

DM542 Nr 2	
PUL+	Pin 12 Arduino Uno
PUL-	Ground (0V)
DIR+	Pin 13 Arduino Uno
DIR-	Ground (0V)
ENA+	Not Connected
ENA-	Not Connected

The ESP-01 is then connected to the Arduino Uno by way of a separate power source, with the drivers and motors operating on 24V DC power supply. In order for the ESP-01 to operate properly at 3.3V, it must be powered separately; the ESP 01 is then powered by the Arduino Uno's 3.3V supply.

The ESP 01 was flashed with factory configuration AT software in order for it to be able to run AT commands in order to connect to a wireless network. This was accomplished by connecting the ESP 01 to a USB to pinout module and flashing it with an ESP8266 flasher programme to guarantee that it was running the most up-to-date software available on the chip at the time of development.

Table 3 Connection table for the ESP-01

ESP-01	
VCC	+3.3V
GND	Ground (0V)
GPIO0	Not connected
GPIO2	Not connected

CH_PD	Connect to +3.3V with 10kΩ Resistor
RST	Not Connected
RX	Pin 3 Arduino Uno
TX	Pin 2 Arduino Uno

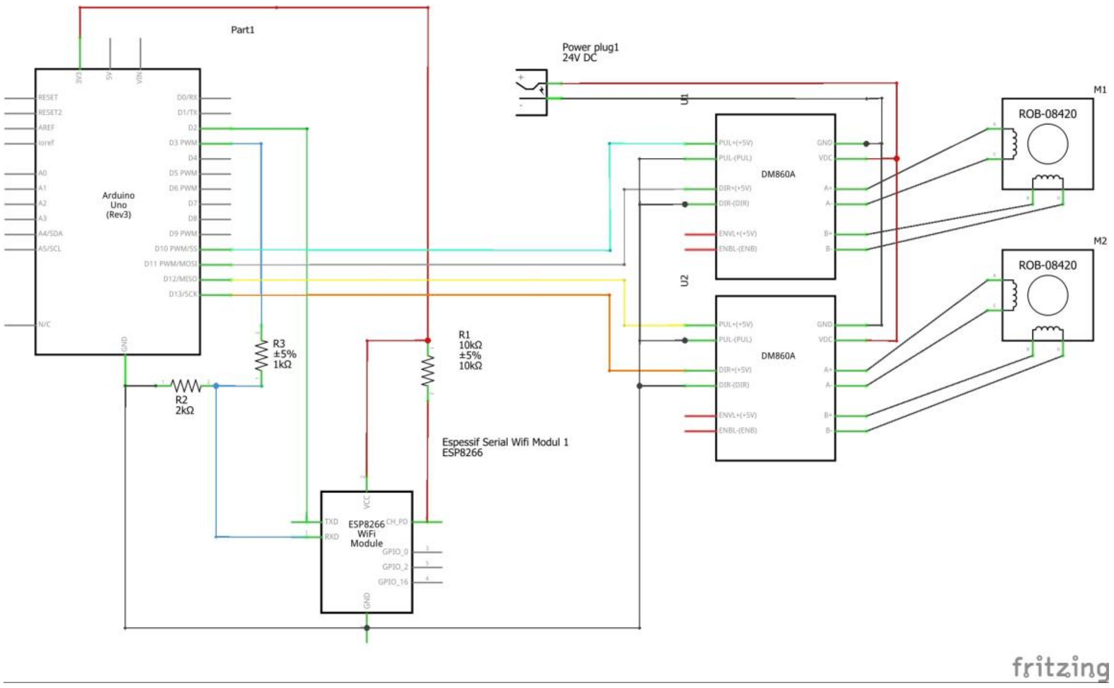


Figure 4 Full schematics for the hardware

Take a closer look at the component diagram for a more in-depth and detailed look at the resistors that connect the ESP module and the Arduino Uno. Furthermore, the RX and TX pins of the ESP module may be examined in greater detail, with the TX pin of the ESP module being directly linked to digital pin 2 on the Arduino Uno. Once the RX pin has been connected to the Arduino Uno's digital pin 3 through a 1k resistor and a 2k resistor to ground, the Arduino Uno's 5V output current is equalised with the current coming from the ESP module, which only handles 3.3V while the Arduino Uno's 5V output current is equalised with the current coming from the Arduino Uno's 3.3V output current. Therefore, these resistors are crucial in preventing damage to the ESP module from occurring. In this case, the TX and RX pins on the ESP module are utilised to read the serial information that is sent from the module.

A DC-DC voltage converter from 6.5V-40V to 5V 2A step down with usb output is one of the components that is not included in the schematics, and it is the component that is missing. The purpose of this component is to provide power to the Arduino Uno so that the platform can operate independently of a separate power source, and it is connected to the main power supply of 24V.

2.3. 3D Printing

2.3.1. Fish tail

The fish tail was created in Fusion 360, where it was modelled after a forket tail discovered in part 1.3 of the project's background research. This fishtail was printed mostly for demonstration purposes, as later in the project a drive train was created that was too large to accommodate this particular fishtail, and the fishtail was a lower priority for the project.

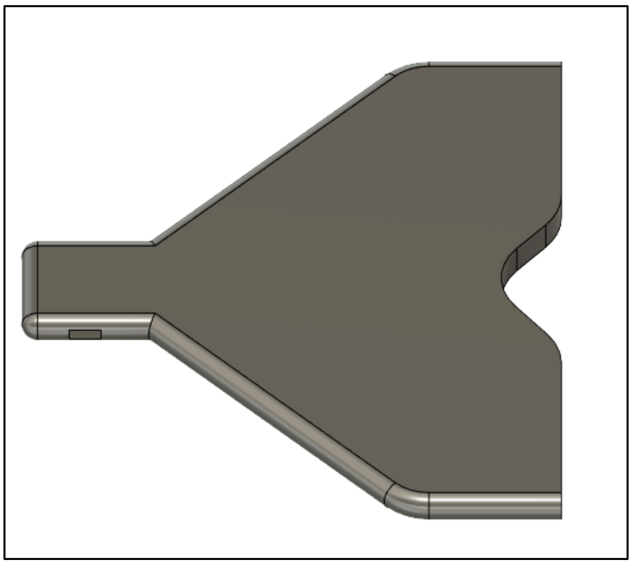


Figure 6 CAD model of the fishtail. Forked tail

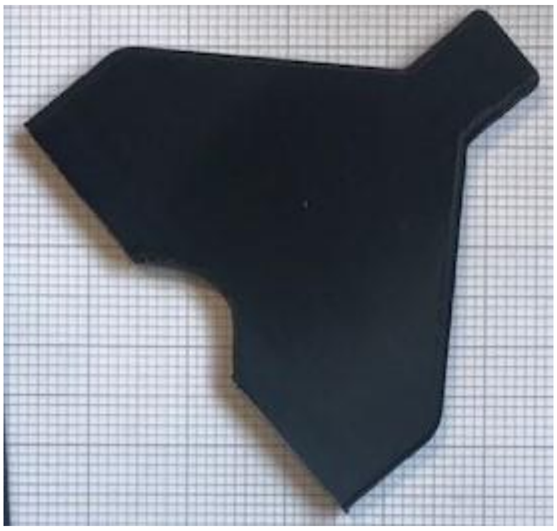


Figure 5 3D printed fishtail. Forked tail

2.3.2. Platform

Where the platform was developed in the same software as the fishtail, but the platform played a more significant role in the project because here is where all the hardware was to be installed, as it was of more significance to the project. This platform was subsequently designed with a bay for the Arduino Uno, a mount for the primary motor where it was intended to be positioned securely in the centre, and a cut-out for mounting a camera to capture footage of the fishtail to compare different tails. Then, the design was forwarded to a contact at Aberystwyth University for production, the platform was laser cut, and a separate Arduino Uno holder was printed to be attached to the laser cut platform. Below are photographs of the platform designed in Fusion 360 and the final product.

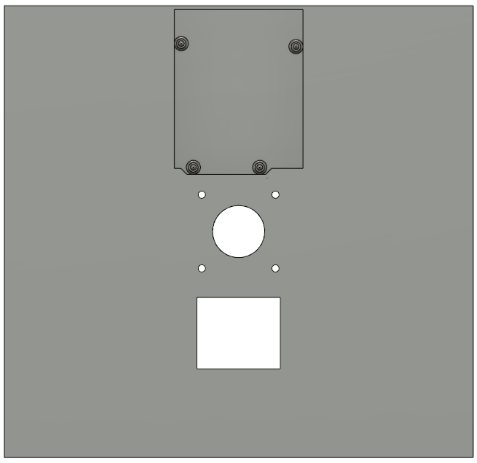


Figure 7 CAD Model of platform top view.

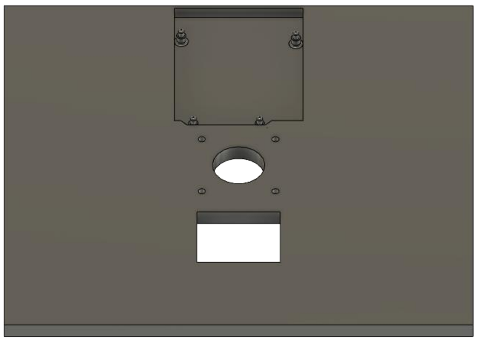


Figure 9 CAD model of the platform

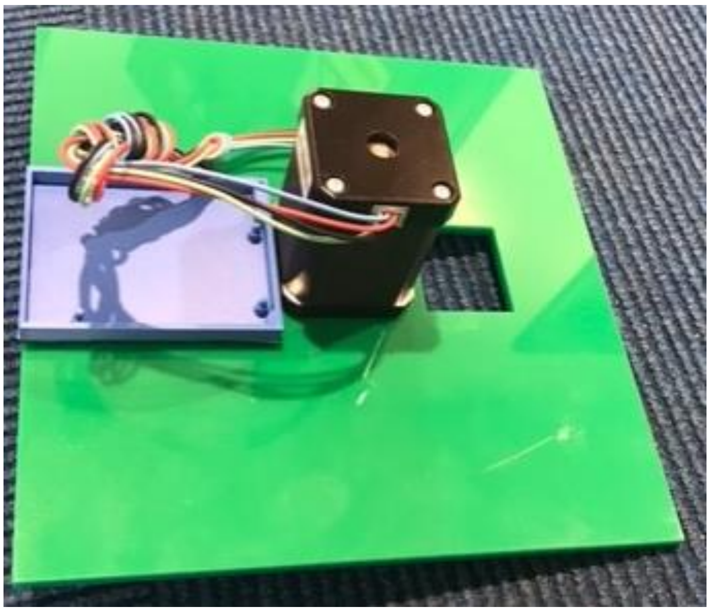


Figure 8 Printed platform

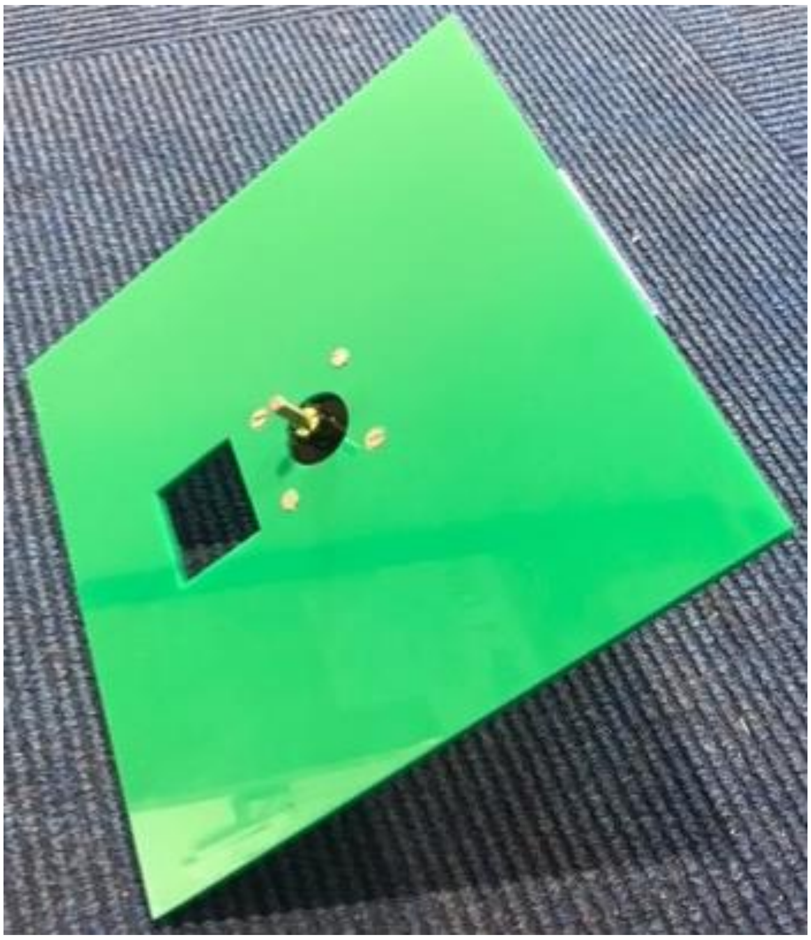


Figure 10 Printed platform, bottom view

3. Software

3.1. Languages

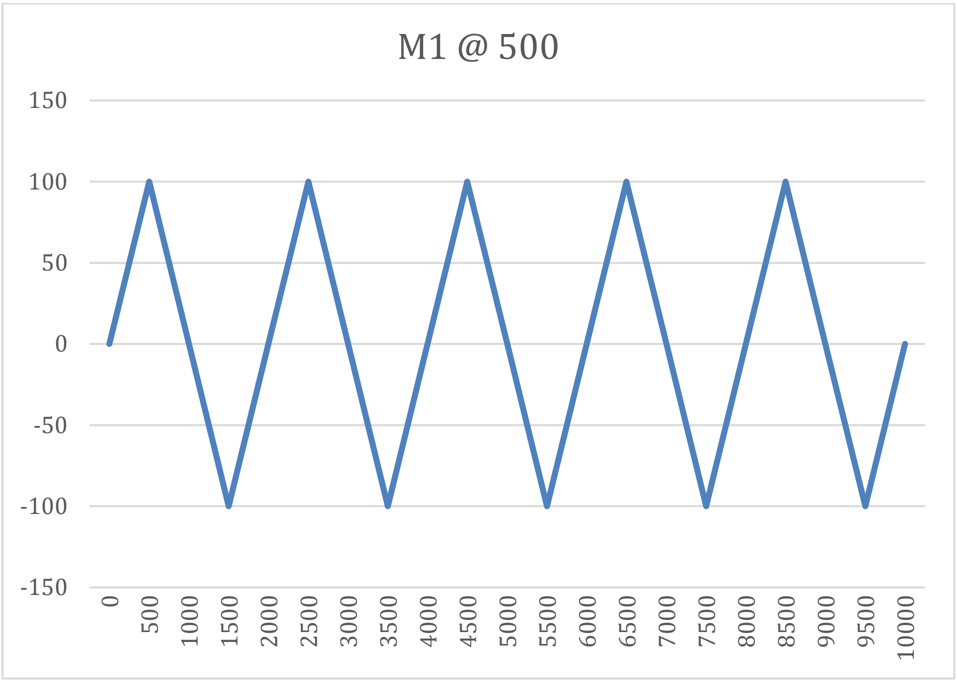
The C++ programming language is a general-purpose programming language that is an extension of the C programming language. For this application, the C++ programming language was selected as the foundation upon which to develop. It was decided on the programming language to use early on in the project, when the decision on the controller was being made. Whereas the Arduino Uno makes use of the C++ programming language, the C++ coding language is typically employed for the majority of same applications as well as for motor control.

3.2. Environment

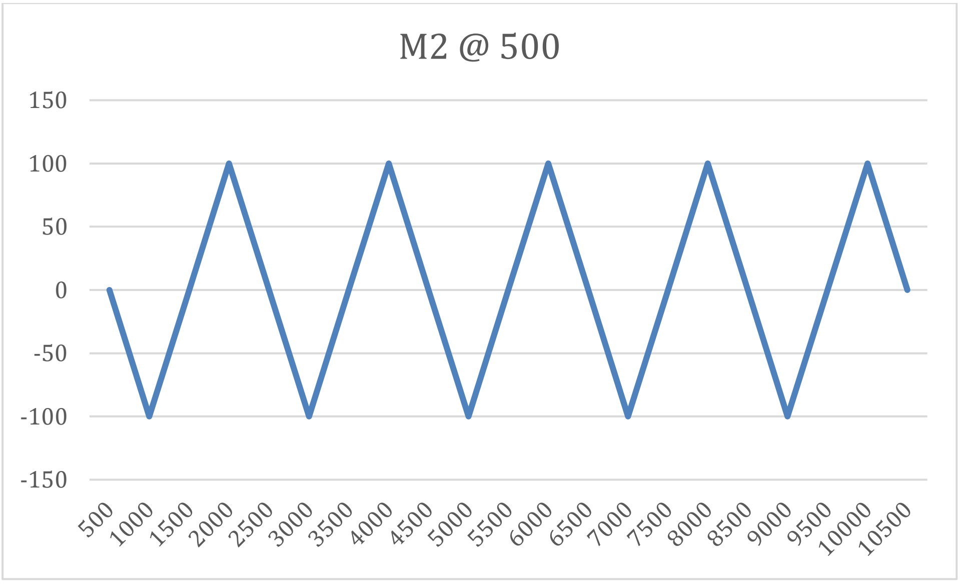
The Arduino IDE served as the development environment for this project, as it is compatible with a broad variety of devices and is not limited to the Arduino controller. Whereas the Arduino IDE contains a text editor for developing code as well as functions for connecting to authentic Arduino controllers and a wide variety of other controllers, it was previously proposed to replace the Arduino Uno controller with an ESP-R32, which uses the same Arduino IDE as the Uno. In addition, the IDE features a Serial Monitor that enables you to inspect the serial output from the controller while programming.

3.3. Motor Control

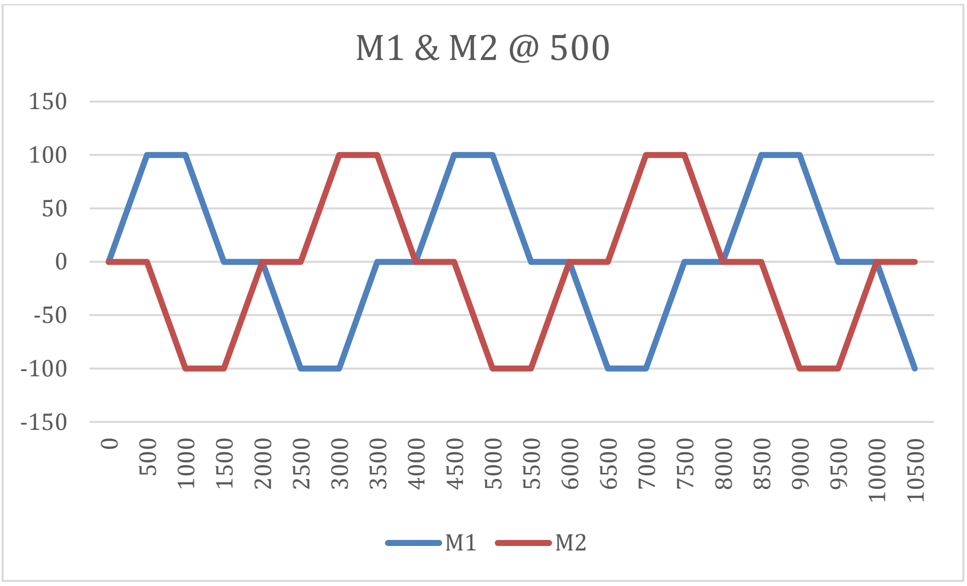
The motor control was created so that the user could have complete control over the stepper motor; however, the stepper motor does not contain internal position sensors. And in a motor action that is hard-coded, the stepper moves in a half-circle to imitate the fish tail motion. When developing the motor controller, the stepper motor's lack of internal position sensors was taken into consideration. The controller and position data is then taken to the 0 point where the stepper have been manually moved by the user, where the stepper motors have been set to 400 steps per revolution as described in 2.1.2, and the motor was programmed to move in a half circle 100 steps in each direction to simulate the fish tail movement. The motor controller was designed such that additional motors could be added to the hardware without altering the main motor controller; the only modification required was to call the motor controller's main function with different parameters. The controller took motor number, number of steps, direction, and speed as input parameters. The speed parameter then determines the delay between each step in microseconds, with the minimum delay being 350 microseconds; if the delay is less than 350, the motor stops since the stepper does not have sufficient time to step.



Here is a chart showing the motor's movement, where the 0 position corresponds to where the user sets the motor's starting point. Where the X axis represents time in microseconds and the Y axis represents the number of steps in each direction, + indicates clockwise rotation and – indicates counter clockwise rotation.



Here is a motion for the motor 2 set up in the same manner as the motor 1, so that it would operate independently and apart from the motor 1.

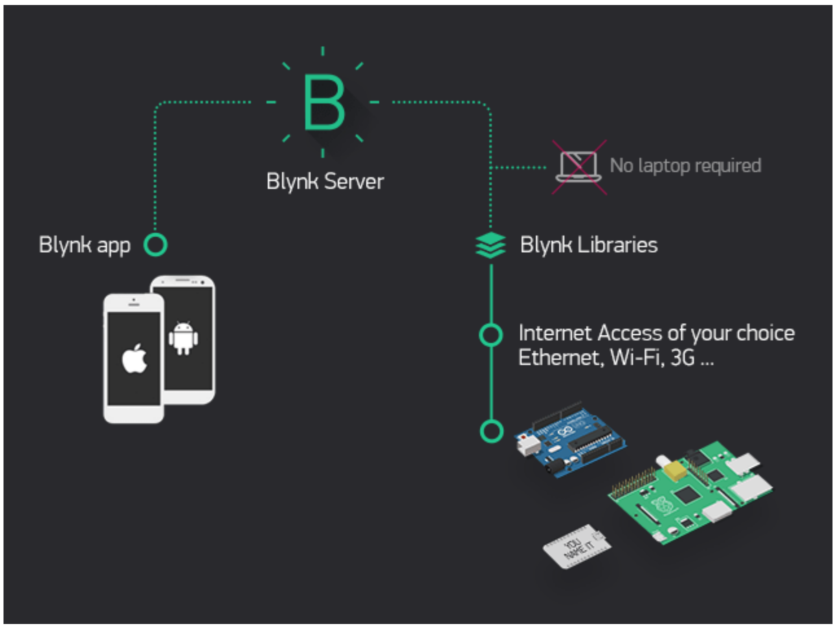


Here is a sequence in which motors 1 and 2 operate concurrently to replicate the movement of the fish tail, which is represented by M1, and the next joint of the fish, M2. Where time is specified in microseconds and oscillates between 100 and -100 steps. And where its movement is not continuous, it will periodically move in accordance with the controller function's call. The primary difference when running the motors in this mode is that they do not operate simultaneously; there are pauses between each motor movement. When motor 1 reaches its maximum distance of 100 steps, it pauses for 500 microseconds while motor 2 moves to -100 steps.

3.4. Blynk

Blynk is a fully integrated suite of Internet of Things (IoT) software that enables the management of hardware: device provisioning, sensor visualisation, remote control with mobile and web applications, Over-The-Air firmware updates, secure cloud, data analytics, user and access management, alerts, and automation.

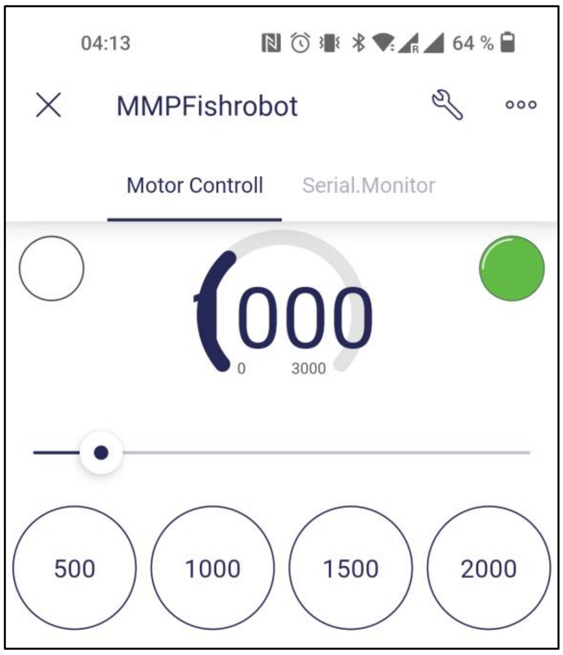
Where this involves the use of the Blynk servers for communication between the UI and the hardware, where the UI was generated using the Blynk application on a smartphone and the hardware, such as the Arduino Uno with stepper motors, is communicated using the Blynk servers. With this software, the Blynk Cloud was used in situations where Blynk offers the choice of using existing servers hosted by Blynk or setting up a private local server while still



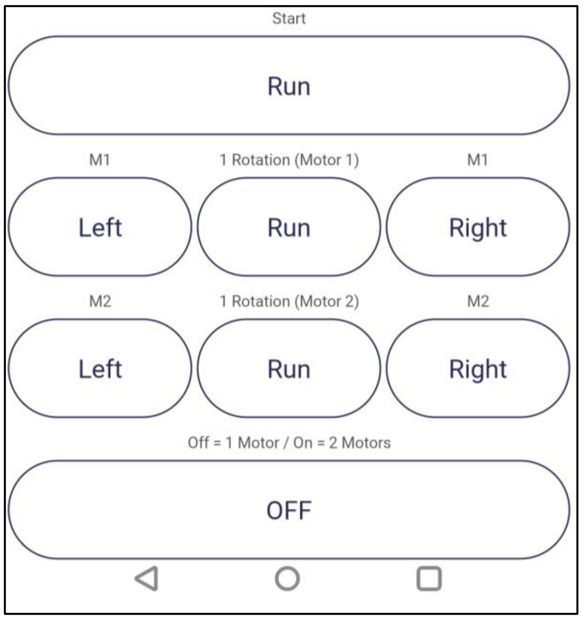
utilising the Blynk library. Whereas the Blynk library makes use of virtual pins and transfers serial data from the virtual pin to the Arduino Uno through the ESP-01 module, it does so by utilising virtual pins.

Here is a simplified depiction of how the Blynk library operates, illustrating how it is controlled via the Blynk app and online interface. Where the developer constructs the UI on the App or web interface or both, where the data passes through the Blynk servers regardless of whether it's the Blynk cloud or a private Blynk server, and then the data reaches the hardware. Using a cloud service can result in a delay between the user interface and the hardware, but in this case, where it was used to operate preprogramed sequences of motor programmes, the latency is justified by the features Blynk provides. Where it is feasible to have hardware control, sensor readings, and motor settings on a user interface that can be shared with other devices, without having to construct a website hosting for controls.

3.5. User Interface Design



The user interface was created using the Blynk App for Android, which was based on the Blynk platform. On the user interface, the user can configure the multiple speed settings for the motor by selecting preprogramed values of 500, 1000, 1500, and 2000 microseconds between each step, by entering more precise numbers for the speed, or by using the slider. As the variable is a global variable, the motor speed is then communicated to both motors and used to set the speed for both motors; the motor speed is also displayed in the UI as the dial at the top. This dial, which is not connected to the buttons or the slider, constantly checks the speed variable in the code for accurate motor speed variable. Also located at the top of the user interface are two virtual LEDs for displaying status; the left LED illuminates while the motors are running, and the green LED illuminates when the platform is linked to Wi-Fi and Blynk and is operationally ready.



On this portion of the user interface is the motor control, where the user may first pick the Run function, which starts the simulation that was coded into the software. Running this simulation then starts the motor in the motion specified in section 3.3. With the bottom switch button OFF, the simulation will run with only one motor, but with the switch button ON, the simulation will run with both motors doing the movement indicated in section 3.3. In the centre are the Left, Run, and Right buttons, which control the position of the separate motors and the manual control mentioned in section 3.3. This is true for both M1 and M2.

4. Implementation

When implementing the proposed design, there were a few issues with the 3D modelling of the fishtail and the platform, but these issues were primarily due to my lack of experience with 3D modelling, as this was an entirely new area I had never attempted before. Thus, this was more difficult than anticipated, and much of the software utilised for 3D modelling was difficult to use and intended for professionals only. This issue consumed a considerable amount of time at the beginning of the project, and as a result, the 3D modelling was scaled back, and as a result, only the platform that was 3D printed was really incorporated into the project, with the fishtail serving as a demonstration.

When the hardware was obtained, it was wired up in several phases, and the implementation of the hardware was relatively smooth. Whereas the initial stage involved an Arduino Uno and stepper motors with their drivers, the initial stage began with a single stepper motor. When designing the stepper motor controller for the simulation, the controller was intended to run the setup with only one stepper motor, and not with both stepper motors that were supplied. But as I was building it, I aimed to make it more flexible by not restricting the code to a single motor connection. And when utilising a stepper motor, there is an interesting library for stepper motors called <Stepper.h>, but this was not used due to its limited function set when the motor was intended to oscillate fast back and forth. When this was set up, the controller was then coded with this in mind. The first prototype of the UI was based on the Serial Monitor, where the platform had to be linked to a computer in order to be controlled, as it was a test platform that would not be autonomously driving about. At this time, Blynk IoT was discovered and implemented so that users could control the platform without requiring a direct connection between the platform and a PC. While implanting Blynk, the ESP-01 WiFi board had to be inserted into the platform. Although this appeared like an easy task at first, it proved to be more difficult than anticipated. The ESP module had to be flashed with factory AT firmware in order for it to connect and communicate with the Arduino Uno. Initially, I lacked the necessary equipment to flash the ESP module, as the serial to pinout module was required to connect the ESP module to. It was at this stage that the ESP-32 controller was discovered, which would solve the initial problems because WiFi was already integrated into the controller. However, the controller was not changed because it would have been a significant undertaking to do so at this point in the project's lifespan.

When implementing the UI into the code, it was quite simple due to the extensive documentation on the Blynk libraries and the controller implementation guide. The Blynk platform used the RX and TX pins on the ESP to pins 2 and 3 on the Arduino Uno for main communication, where the Blynk read the serial data from the ESP for the virtual pin values coming from the Blynk UI on the phone. It was implemented so that Blynk ran in the loop of the code and constantly checked the pins for serial data; if it read any information from the virtual pins, it would perform the action corresponding to the virtual pin; the set of instructions to perform the action corresponding to the virtual pin it read was then coded in. Problems arose, however, when it was desired to have a Terminal on the Blynk UI so that boot-up information could be displayed on the UI, allowing the device to be entirely independent of a computer and to verify that all connections were correct. When attempting to implement this feature, there was a problem with something flooding the Blynk library, which prevented the ESP module from booting and the software from running. This was eventually determined to be the consequence of the <Simpletimer.h> library. This was attempted late in the project's lifespan and abandoned since the controls were still under control.

The platform could have been tested with the fishtail attached if the fishtail had been linked to the motor's driving shaft. This was not implemented due to a lack of understanding of 3D modelling and 3D printing, and it was not prioritised in order for the project to proceed. Another implementation that was discussed in sections 1.2. and 1.3. was the use of different sensors to capture data as a force sensor to test the force output of different tails. However, this was not implemented due to the complexity of mounting this sensor to measure the force output and the fact that it was not feasible when the 3D-printed fish tail was given priority.

4.1. Final Hardware Iteration

As depicted in the image below, the final version of the hardware concludes with an Arduino coupled to two stepper motors and their drivers.

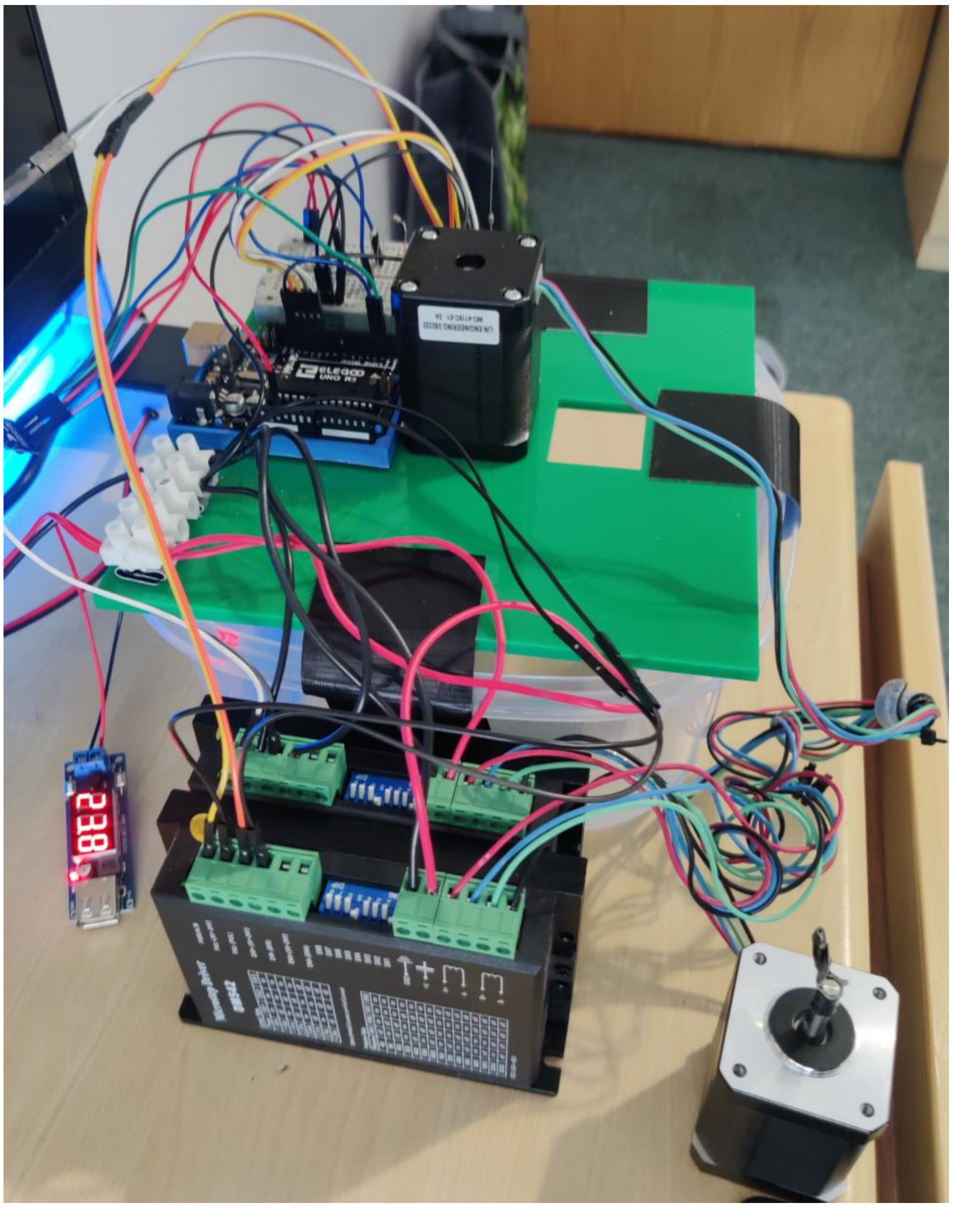


Figure 11 Picture of platform with hardware

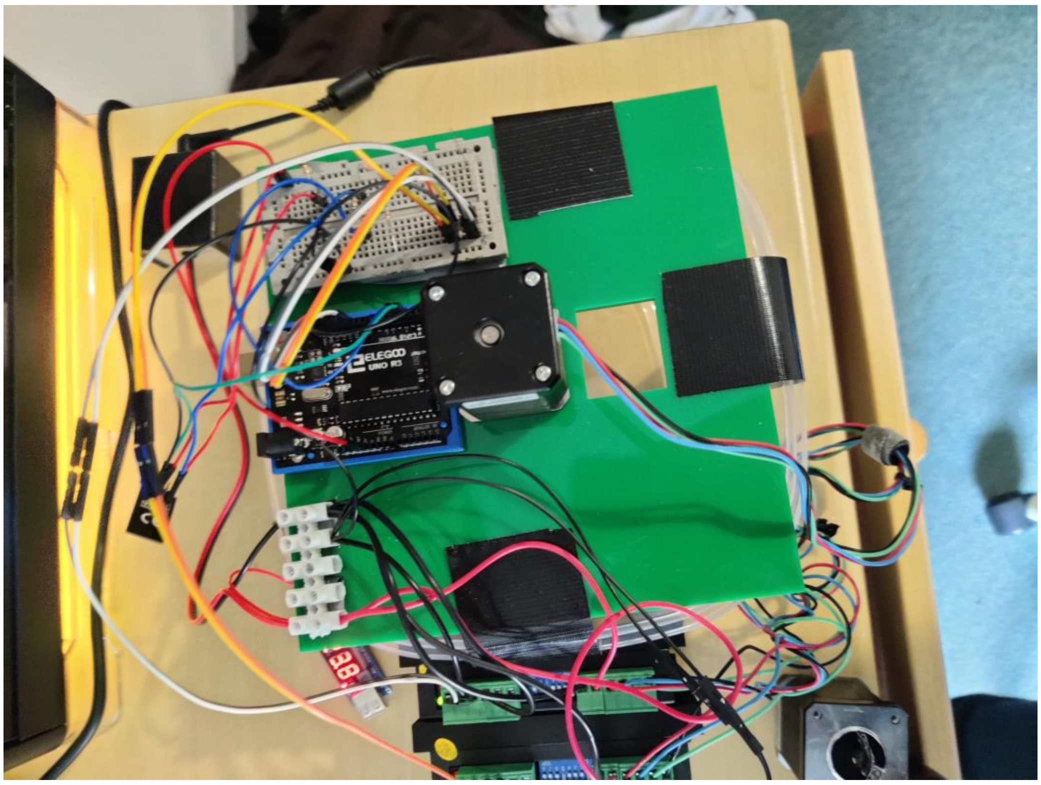


Figure 12 Picture of platform with hardware, top view

5. Testing

5.1. Overall Approach to Testing

Since this application consists of both hardware and software, act and reach testing was the most suitable method for evaluating it. Whereas, when the controller and the user interface were established, it was more of a test to check if the hardware was performing what it was supposed to do, and in certain cases to test the boundaries of the motor's speed. In one of the scenarios encountered during the development and testing of the controller, the minimum delay in microseconds was used to determine the speed. This resulted in the motor not moving at all, and steadily rising upwards in microseconds to find the absolute lowest delay achievable in microseconds. Multiple lines of code were output to the Arduino Serial Monitor port to confirm that the code was performing as intended. On the hardware level where the ESP-01 was tested, a few additional tests were conducted. The ESP-01 module was tested by using AT commands in a programme to read the chip without the

Arduino Uno, where it was possible to obtain data indicating that the module found WiFi connections and could connect, as well as confirmation in software that the module successfully connected to WiFi and that everything on that end was properly connected.

Where it would have been best to test the completed platform with all its intended components in an environment where the 3D-printed fish tail could be submerged in water, this did not occur throughout the project. Where we could test the fishtail in water with the drive train from the motor down to the water, where the drivetrain and the stepper motor mount were created, but where a 3D-printed fish tail that could be fitted to the drive train did not occur. Therefore, the testing conducted on the platform was merely a "dry" test, in which all components functioned as intended.

5.1.1. User Interface Testing

As with the rest of the software, the testing of the user interface consisted mostly of determining if the UI could control the hardware as planned. But there was still a bit more to test, as every function on the UI utilises virtual pins for each UI object. To test the set speed functions on the UI, it was sufficient to verify that it changed the global variable motor speed, and to continuously print the motor speed value to the Arduino serial monitor so that a visual confirmation could be obtained on the serial monitor when the value changed when using the UI. The same method was used to test the speed value on the user interface, by changing it manually from the serial monitor on the Arduino IDE and observing that the value on the user interface also changed to the right value.

6. Critical Evaluation

6.1. Requirements

Not all of the requirements outlined at the beginning of the project were met by the project's conclusion; the platform is still missing 3D-printed tails made of different materials, as well as sensors to collect sensor data from a running fish tail on the platform. However, the remaining conditions were met, since we have a platform to test 3D-printed fish tails that has mobility to move the tail in a fish-like manner. The intention was to place a camera on the platform for comparison purposes.

6.2. Future Work

This project has a number of areas that can be improved, and one of the modifications I would have made was to modify some of the hardware, as I described in the report. This was the initial sort of motors used in this project, and more work would be put into acquiring a DC motor or servo for this particular use. Where the DC motor would be more suitable for this application, as well as having position sensors and receiving data from the motors that can signal different things, such as when the device is tested in water. And simultaneously abandon the Arduino Uno controller in favour of the ESP-R32 controller, which is significantly more powerful and adaptable than the Arduino Uno.

I would also improve the 3D designs for the project, particularly the fish tail, where I would design a tail that would work with the drive shaft I have for the project so that testing in a water environment would be possible. In the same topic, I would work on incorporating more sensors into the platform; as stated earlier in the report, a force sensor would be a welcome addition. Therefore, additional time must be invested in determining how a force sensor would be installed and obtaining precise measurements of how alternative materials for a 3D-printed fish tail would perform.

6.3. Conclusion

In the end, a platform to test 3D-printed fish tails for AUVs was created, allowing for the development of a fish robot. The platform can test fishtails with the option of a camera equipped with a sensor to collect data on the fishtail and compare it to tails made from different materials. However, the hardware utilised for this project was not optimal; if I had the option, I would upgrade the hardware and either outsource 3D printing or educate myself further on 3D design and 3D printing.

7. References

[1] “Spot® - The Agile Mobile Robot,” *Boston Dynamics*. https://www.bostondynamics.com/products/spot (accessed May 02, 2022).

[2] N. O. and A. A. US Department of Commerce, “What is the difference between an AUV and a ROV?” https://oceanservice.noaa.gov/facts/auv-rov.html (accessed May 03, 2022).

[3] M. Tjomsaas, “The Development of a Modular 3D Printed Autonomous Robotic Fish.” May 07, 2021.

[4] “Structure and Function - Fish | manoa.hawaii.edu/ExploringOurFluidEarth.” https://manoa.hawaii.edu/exploringourfluidearth/biological/fish/structure-and-function-fish (accessed May 03, 2022).

[5] “Arduino - Force Sensor | Arduino Tutorial,” *Arduino Getting Started*. https://arduinogetstarted.com/tutorials/arduino-force-sensor (accessed Feb. 10, 2022).

[6] CppCon, *C++20: Reaching for the Aims of C++ - Bjarne Stroustrup - CppCon 2021*, (Dec. 16, 2021). Accessed: May 09, 2022. [Online Video]. Available: https://www.youtube.com/watch?v=15QF2q66NhU

[7] “DC Motor: What Is It? How Does It Work? Types, Uses.” https://www.iqsdirectory.com/articles/electric-motor/dc-motors.html (accessed May 09, 2022).

[8] “Fusion 360 | 3D CAD, CAM, CAE & PCB Cloud-Based Software | Autodesk.” https://www.autodesk.com/products/fusion-360/overview (accessed May 09, 2022).

8. Appendices

A. Third-Party Code and Libraries

The Arduino libraries follows:

- ESP8266_Lib.h
- BlynkSimpleShieldEsp8266.h
- SoftwareSerial.h

Where all of the libraries comes under the Blynk package and the selected ESP module for the project.

B. Readme file

INTRODUCTION

This project entails the expansion of a fish-shaped robot designed to test biological hypotheses. Although the objective of this project is not to design and build a complete fish robot, it is my obligation to create and test a variety of fish robot tails. This software, which is the primary programme for the platform, is built in C++ and mostly use the Blynk library.

REQUIREMENTS

This require some specific library, this is the Blynk library and here is a step-by-step guide on how this can be installed:

https://docs.blynk.io/en/legacy-platform/legacy-articles/what-do-i-need-to-blynk

or directly the library:

https://github.com/blynkkk/blynk-library/releases/tag/v1.0.1

CONFIGURATION

To make this work with your WiFi, it is necessary to adjust certain parameters; specifically, lines 27 and 28 must be modified to reflect the WiFi to which you wish to connect.

```
char ssid[] = "Your sssid here"; //You can replace the wifi name to your wifi
char pass[] = "Your password here"; //You can replace the wifi password to your wifi
```

CREATOR

Petter Wessel Kokkim
pwk1@aber.ac.uk