MARINE & OFFSHORE TECHNOLOGY SCHOOL OF ENGINEERING

2014/2015 Final Year Project

Sterling

(Remotely Operated Vehicle Design)

Supervisor: Edwin Cheng Chua Wee

TSO: Richard Ho Yau Seng

Team member: Rahul Singh

Contents

Acknowledgments 4

Summary 5

Introduction 6

1.1 Project Introduction 6

1.2 Project Objectives 7

2 Designing Sterling 8

2.1 Software (Blender) 8

2.2 Ideas 9

2.3 Decision Matrix 12

2.3 Designing Sterling 13

2.4 Timeline planned 14

3 Literature Review 15

3.1 Arduino 15

3.2 Coding 17

Void setup(_) 18

Void loop () 18

3.3 Serial port 21

3.4 Processing 22

Void setup () 22

Void Draw () 23

Camera inside Processing 24

3.5 Add-ons 25

Accelerometer 25
Temperature and light dependent resistors 27
Controllers 28
MOSFETs 28
3.6 Remotely Operated Vehicles 30
3.7 Buoyancy of Sterling 32
4 Fabrication 33
4.1 Motors 34
4.2 LEDs and Camera 36
4.3 Chipsets and wiring 37
5 Learning Outcomes/ Future work 38
5.1 Battery 38
5.2 Arduino 38
5.3 PVC Pipe 38
5.4 Camera 38
6 Challenges 39
7 Conclusion 40
References 41
Appendix 42

Acknowledgments

Fabrication of the Remotely Operated Vehicle would not have been possible without the assistance of the unconditional support of Mr. Edwin and all the TSOs inside the workshop

Mr. Edwin, Thank you for your constant support on believing in me at doing such a huge project all by myself. Thanks to your constant reminders on the deadlines and frequent messages on my progress has allowed this project to reach where it is now. I would also like to thank your positivity throughout the 2 semesters of the project even though you understood that ROV would have not been easy for a 1 man job but you stayed positive and continued to boost my morale up.

TSO, deeply appreciate all the support given by the TSOs to assist in fabricating the Remotely Operated Vehicle. They have constantly given me an alternate way to make the design easier or simply help with the difficult to use machines in the workshop. I would also like to thank them for staying longer than they should to allow me to complete my project before the deadline.

Summary

Sterling (remotely operated vehicle) is designed to assist the dry docking process. When a ship is sailed into a dry dock, blocks are placed at the bottom of the ship. Before the water is drained, divers are sent below to inspect these blocks and Sterling has been designed to seek assistance in this process by replacing the need of divers going into the water to inspect.

By the use of Arduino microprocessors, Sterling has been programmed to run on the fingertips of the computer controls while providing a live feed of what is happening underwater. This report will aim to help in understanding how Sterling was designed and programmed to meet the needs of the objectives.

Introduction

1.1 Project Introduction

As technology continues to advance, the Singapore shipyards have still been using the old method of doing surveillance underwater through the aid of divers. Sterling has been programmed to reach the objective goal of minimizing the use of divers during the dry docking process. Through its simple design, it will reach the depths of dry docks to give a safe confirmation that the blocks have been placed correctly below the ship before the water is drained. Shipyards use divers to inspect these blocks which can often be a hassle. Often as the divers reach the bottom of the dry dock, there is limited vision of what is going on and there have been incidents where this has led to major accidents in which the divers lose their way up back to the top of the water. The safety measured used here is that divers are supposed to use a string of rope when they go down but sometimes these safety measures can be ignored causing them their lives.

During this process, the sea water in the dry dock often contains sea creatures that are not usually dangerous but there are always time when the more dangerous ones slide along and this can be a hazard for the divers.

This report will guide the reader through the journey on how Sterling came about from the designing process to the fabrication process in details to ensure full understanding on every feature of the Remotely Operated Vehicle and how the every feature aims on achieving a different set of goals.

1.2 Project Objectives

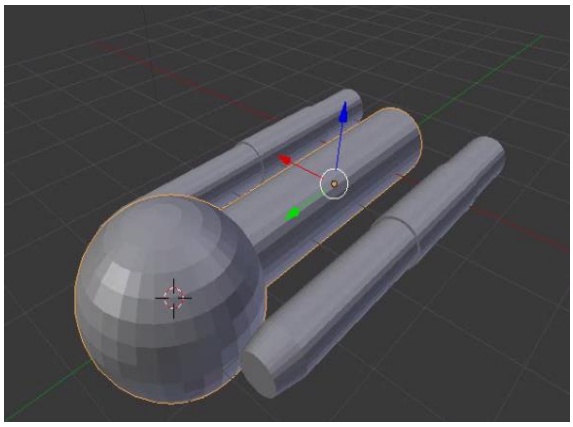
The main objectives will be described below as followed

Objective	Description
Minimize the use of divers	Divers are required for the current inspection of blocks during dry docks and sterling will aim to minimize them but not completely remove them as if there is a need to shift the blocks, a diver will be needed to go down and shift it. By doing this, they will have a clear objective on which block is off the grid and it will be not consume as much time as sending a team of divers to inspect and shift it after planning.
Reduce cost	Divers can often be expensive to hire due to the high risks in the job therefore Sterling will allow the shipyard to reduce costs in that department.
Be Portable	By using simple materials such as PVC pipes, Sterling is light weight and can be moved around easily without much hassle.
Easy to maintain	Being a ROV under water, troubles can occur during the time of functionality. As Sterling is controlled by a computer, the computer will have full readings on the status underwater and if anything were to go wrong, it can be easily be detected by the computer.

2 Designing Sterling

2.1 Software (Blender)

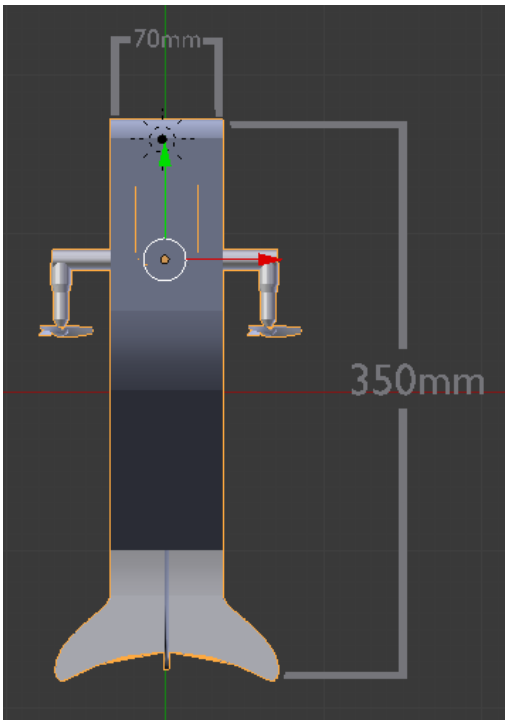
Throughout the ideation process, the main software that was used to design the core elements of the product was Blender. Blender in an open source 3D designing software that can be used to create animations, modeling, visual effects, rendering images and even creating video games. The software was released in 1995 and was written in C, C++ and python language of coding. Being an open source tool, it gives us the opportunity to import 3rd party plugins that help in making the journey of designing even more enjoyable and user friendly. The software uses simple shapes such as polygon, cylinder, cubes and by using the extort tool to form it into a live 3D image.



The software takes time getting used to but after getting a hang of it, designs can often be done with very minimal time required. Explaining ideas can be vital to the designing stage and using AutoCAD might have been challenging to explain out during presentation thus blender had helped in easing this process. Although blender does not provide the exact dimensions to the design but it provides the correct ratio to real life dimensions. The dimensions can be projected by using the textbox tool and simple rectangles placed at the correct 2D views of the design.

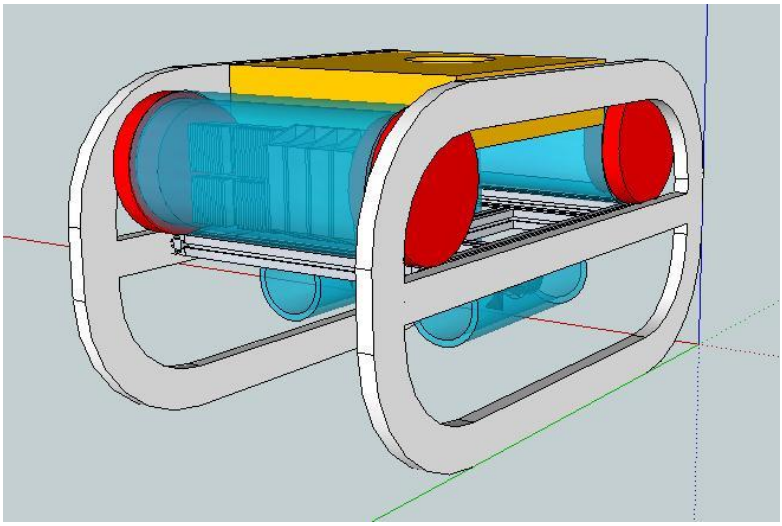
2.2 Ideas

Idea 1



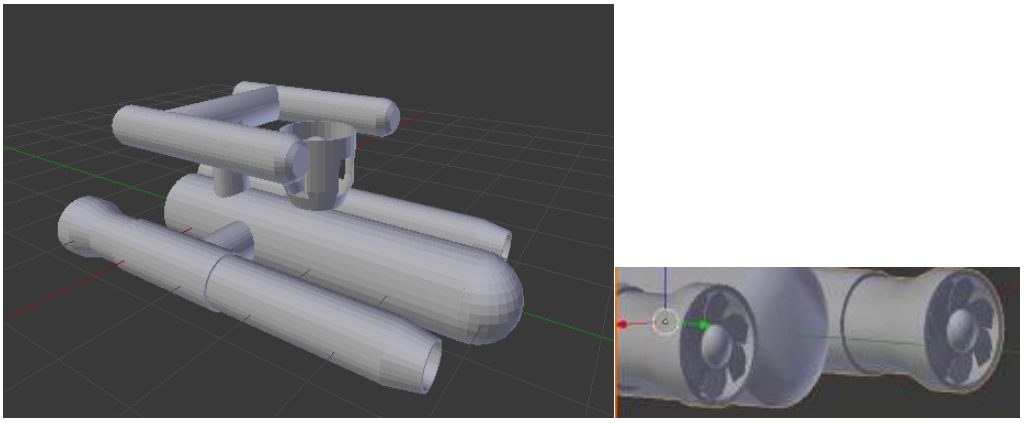
This design was the very first design that was created using blender. The mechanism for this idea was simple whereas it uses two motors and two servos to control the entire motion of the ROV. Although this design would work underwater, it was not practical to use it in a shipyard due to its small size. The thrusters provided speed but it would have been rather difficult to add cameras into it and getting it to neutral buoyancy. The ROV would have been able to provide enough speed to get to one place to another but due to the lack of weight; it is predicted to not be able to take the load of sea water especially deep under water where it might just crush due to the pressure. The design did follow the structure of a fish which gave it a very streamlined body which would completely reduce drag.

Idea 2



This idea was developed using google sketch up. This idea would have been designed from acrylic and aluminum mainly and would consist of 3 thrusters. The main components would be placed inside the 2 air chambers and the motors will have to be waterproof. Having the frame made of aluminum would boost up the cost of the product and at the same time increasing the weight of the ROV. The aluminum frame will also require special order and will require welding to form this shape. Due to the restrictions of the workshop and minimizing the time spent on ordering parts, it would not have been practical to build the machine. In addition, having an acrylic cylinder would restrict the ROV from going very deep in water as it might crack due to the water pressure and damage the electronic parts inside it.

Idea 3



The final idea was developed using blender and this was the idea that looked most practical to be developed further. Having a structure made of simple PVC pipes will enable it to be built at ease using PVC joints and caps. PVC pipes are sold in most hardware shops therefore it would not require much time to order these parts. The figure above shows how the back thrusters will be placed to power the ROV to move forward and backward. There is a thruster added to the top of the ROV to enable it to move in an upward and downward motion. The camera is placed in the middle hull at the front of it in a cylindrical shape. The design seemed rather easier and able to achieve all the objectives with the design itself therefore I decided to develop this idea further.

2.3 Decision Matrix

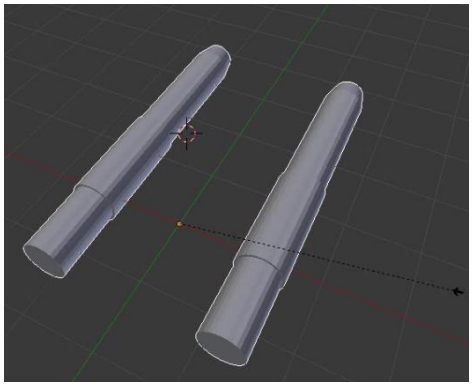
	1st idea	2nd idea	3rd idea
Construction	3	3	4
Creativity	5	4	4
Practical	2	4	4
Complication	2	3	3
Total	12	14	15

The decision matrix helps in assisting and portraying why idea 3 was chosen as compared to the rest of the ideas. Idea 3 has successfully managed to score the highest as compared to the rest and therefore from this matrix, it was decided that Idea 3 would be used for the Remotely Operated Vehicle Design.

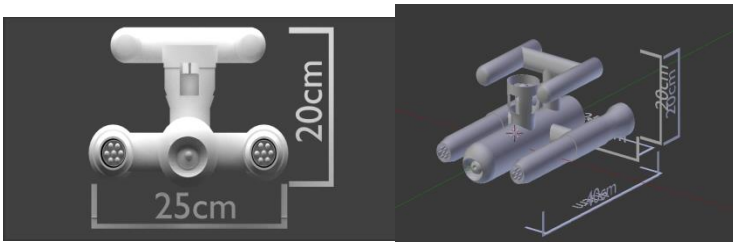
The main scales used for this matrix consisted of the construction, creativity, practical and complication of the design on a scale from 1-5 with 5 being the best and 1 being the least favorite.

2.3 Designing Sterling

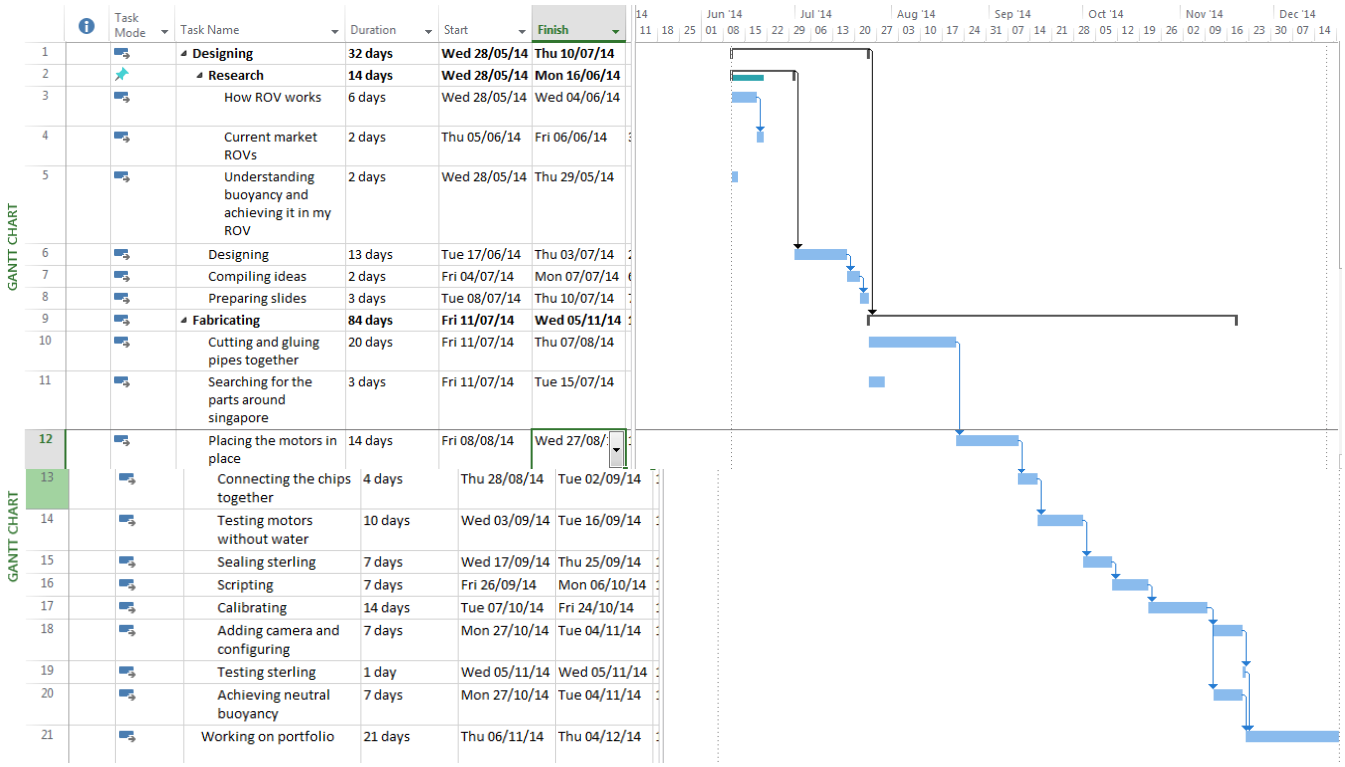
Sterling was designed using basic geometrical shapes which mostly consisted of cubes and cylinder. Tools such as Extrusion, Bevel and vertex edit tools were used with transformation tool being used the most.



Upon the completion of the design, dimensions were added from an external plugin tool which helped in illustrating the major dimensions in the ROV and upon completion of that, it had to be rendered. In order to get to the rendering process, lightings must be setup inside blender and the values of lighting had to be set which gave it the perfectly colored ROV. The render tool creates an artificial lighting inside the computer which usually takes up to 10 minutes to render a design like sterling. During this process, the graphic cards of the computer will run at maximum settings and upon completion will provide a beautiful image of the design in an illustration of lightings. The figure below shows the difference of a rendered and a non-rendered image.



2.4 Timeline planned



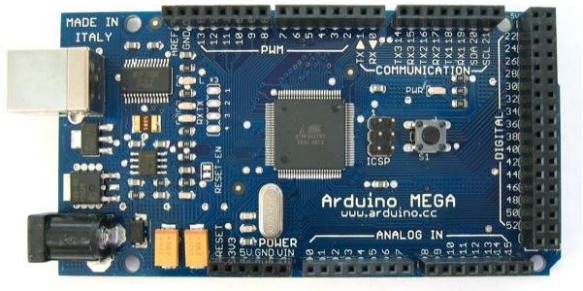
The plan at the point of time was to complete designing and research 10th July 2014 and finish fabricating by 4th November 2014.

3 Literature Review

3.1 Arduino

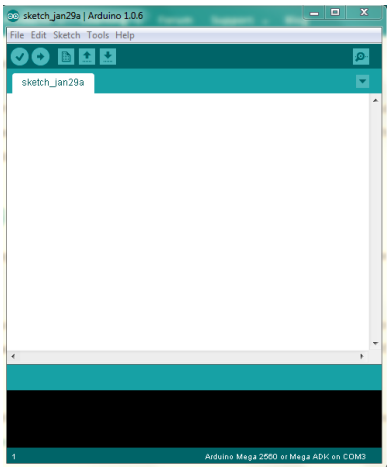
The motherboard of the ROV is known as the Arduino Mega 2560 where it is one of the most powerful chipsets with an open source market. The brilliance of the chip is that it can be programmed to do anything that you decide it to do. For example, before buying the motors or LED, Arduino used to be programmed to reply back to the questions asked inside the serial board. From everything learnt experience wise with Arduino, it works as a switch. It is the core chip to send out signals to the rest of the gadgets attached to it which is inclusive to the motor shields or MOSFETs. It works like a switch but in a much more complicated way where instead of pressing buttons on it, the user programs it on how he/she would like it to run.

The Arduino Mega 2560 consisted of 54 digital I/O pins and 16 analog input pins. The reason why Arduino is unable to control high voltage equipment directly is due to the lack of amperes in every pin which is used to send signals instead. Every pin on the board can supply up to 50mA but does not restrict it from controlling smaller gadgets such as light dependent resistors or accelerometers, where they can be connected to the 3.3V pins or the 5V pins.



Arduino Mega 2560

In order to upload any codes on the Arduino board, the user is required to install Arduino's official integrated development program which is written in java and can be used for writing the code and uploading it into the board. The software comes with over 40 examples for the user to learn from and every code is written in C/C++ language.



Arduino IDE

The chipset comes with a 128 kb flash memory for storing the codes inside it and an overcurrent feature in which the fuse will break the connection when more than 500mA is given to the USB, this is a crucial feature when it comes to learning Arduino from scratch whereas, if the polarities are switched or there is a short circuit, it will save the laptop from any damages. The chipset will only be able to start once the short circuit or any other issue has been resolved on the board. During the programming stage, this was often assumed that the board had been spoilt but after plugging out the wires, the Arduino had continued work like usual which helped in understanding the importance of the feature.

3.2 Coding

The basics of coding Arduino starts with defining the value which is also another form of adding a name to the value. This feature helps the compiler in replacing all the defined words into the value the user sets it to. An example of the feature in the consisting code will be defining the different motors.

```
#define MOTOR_A 0
#define MOTOR_B 1
#define MOTOR_C 2
```

After defining the values, Arduino is required to know which pins are being used by the motors. The line used to tell IDE is “const byte” which can be shown in the example below:-

```
const byte PWMA = 3;
const byte PWMB = 11;
const byte DIRA = 12;
const byte DIRB = 13;
const byte DIRC = 8;
const byte PWMC = 6;
```

As shown above, the shield requires each motor to have two pins where one of the pin is for the PWM control which is also known as speed while the other pin is used to control the direction. There are 13 PWM (speed) pins on the Arduino Mega. The PWM and DIR pins have been defined according to their respective motors, for example, PWMA is for motor A while PWMB is for motor B.

Void setup( )

The next feature would be setting up the pins to run as planned where this will require setting up values to every pin allocated while defining the motors.

The PWM and the DIR pins will have to be set as outputs in order for Arduino to send the signals and after the completion of that, their digital pins will require a value to be set. After all the definitions, the "Serial.begin(9600);" is added to let Arduino send data at 9600 baud rate

```
pinMode(PWMA, OUTPUT);
pinMode(PWMB, OUTPUT);
pinMode(DIRA, OUTPUT);
pinMode(DIRB, OUTPUT);
pinMode(DIRC, OUTPUT);
pinMode(PWMC, OUTPUT);
```

Setting the pins as output

```
digitalWrite(PWMA, LOW);
digitalWrite(PWMB, LOW);
digitalWrite(DIRA, LOW);
digitalWrite(DIRB, LOW);
digitalWrite(PWMC, LOW);
digitalWrite(DIRC, LOW);
```

Setting the pin values as low

```
Serial.begin(9600);
```

Void loop ()

"Void loop()" is a function used to declare to Arduino that anything inside the function will work in repetition. The repetition can be controlled by adding functions such as "delay(500);" below the line of code where it will require run the function for 500 milliseconds before restarting again. For the ROV code, the loop function is used to control the motors. Before adding any features to this function, Arduino will be required to read the incoming byte before performing any actions therefore Arduino will be told to read the seal port which is defined as stated below:-

```
if (Serial.available() > 0) {

   incomingByte = Serial.read();
```

Every byte sent in the Arduino comes in an ASCII (American Standard Code for Information Interchange) value which is also the values read by a computer. Every alphabet written on the computer is a numerical value in the code as the computer does not understand what alphabets are. The way these values are processed in the chipset is by changing it to an ACSII value.

DEC	OCT	HEX	BIN	Symbol	HTML Number	HTML Name	Description
65	101	41	01000001	A	&#65;		Uppercase A
66	102	42	01000010	B	&#66;		Uppercase B
67	103	43	01000011	C	&#67;		Uppercase C
68	104	44	01000100	D	&#68;		Uppercase D
69	105	45	01000101	E	&#69;		Uppercase E
70	106	46	01000110	F	&#70;		Uppercase F
71	107	47	01000111	G	&#71;		Uppercase G
72	110	48	01001000	H	&#72;		Uppercase H
73	111	49	01001001	I	&#73;		Uppercase I
74	112	4A	01001010	J	&#74;		Uppercase J
75	113	4B	01001011	K	&#75;		Uppercase K
76	114	4C	01001100	L	&#76;		Uppercase L
77	115	4D	01001101	M	&#77;		Uppercase M
78	116	4E	01001110	N	&#78;		Uppercase N
79	117	4F	01001111	O	&#79;		Uppercase O
80	120	50	01010000	P	&#80;		Uppercase P
81	121	51	01010001	Q	&#81;		Uppercase Q
82	122	52	01010010	R	&#82;		Uppercase R
83	123	53	01010011	S	&#83;		Uppercase S
84	124	54	01010100	T	&#84;		Uppercase T
85	125	55	01010101	U	&#85;		Uppercase U
86	126	56	01010110	V	&#86;		Uppercase V
87	127	57	01010111	W	&#87;		Uppercase W

ASCII values of alphabets

After Arduino processes the data into a byte, the user can then add functions such as “if (incomingbyte == ‘X’)”. By adding this function, it tells Arduino to hold any commands below it until the byte ‘X’ is executed. The term used to control the ROV is shown in the figure below:-

```
if (incomingByte == 'Q') {
driveArdumoto(MOTOR_C, CCW, 110);
}
if (incomingByte == 'W') {
driveArdumoto(MOTOR_A, CCW, 150);
driveArdumoto(MOTOR_B, CCW, 110);
}
if (incomingByte == 'S') {
driveArdumoto(MOTOR_A, CW, 150);
driveArdumoto(MOTOR_B, CW, 110);
}
if (incomingByte == 'A') {
driveArdumoto(MOTOR_C, CW, 110);
}
```

Every function is started by the use of ‘{‘ symbol and ended by ‘}’ symbol in the Arduino IDE. With the commands above, Arduino is told not move the motors until the incoming bytes are received. If the bytes are received, it will run the motor according to its direction and speed. The maximum speed of the motor known in the code is 256 and will run the motor at maximum throttle. Before the script is compiled, Arduino IDE does not know the meaning of “driveArdumoto” command used in the “void loop()” therefore it is necessary to define what the word means inside the code.

```
void driveArdumoto(byte motor, byte dir, byte spd)
{
  if (motor == MOTOR_A)
  {
    digitalWrite(DIRA, dir);
    analogWrite(PWMA, spd);
  }
  else if (motor == MOTOR_B)
  {
    digitalWrite(DIRB, dir);
    analogWrite(PWMB, spd);
  }
    else if (motor == MOTOR_C)
  {
    digitalWrite(DIRC, dir);
    analogWrite(PWMC, spd);
  }
}
```

In this function, “driveArdumoto” is being defined according to the type of motor, the direction and speed respectively.

3.3 Serial port

Arduino IDE serial port is the part of the software where commands are sent and received to or from Arduino. As stated in the program above, Arduino requires incoming bytes before letting the motor run and the serial port is the place to communicate with it. However, these bytes have to be sent with typing it out and then following up with the enter button pressed. This would not be practical to be used in an ROV where bytes have to be instantly sent without the usage of enter. Therefore another IDE would have to be used to send data over to Arduino.

3.4 Processing

Processing is an open source software which can be an alternative to Arduino's IDE serial port. The programming of Processing is also based on C/C++ but it is slightly more advanced than Arduino's IDE and will require a slightly deeper understanding into the language. Before starting the code on processing, the user is required to add the Arduino library into the compilation. This feature comes integrated with the software itself and can be simply be attached to the code by adding "import processing.serial.*;" at the top of the code.

```
import processing.serial.*;
```

Void setup ()

Just like the Arduino IDE, processing will require being setup and the user simply has to add a command to let processing know which port is being used by Arduino. A port on the computer is assigned to Arduino when it is installed successfully to the computer.

```
println(Serial.list());
port = new Serial(this, Serial.list()[0], 9600);
```

The port is being defined as 0 in the current computer therefore it is added and the baud rate has to match with the Arduino's one therefore it is set to 9600. In the case the user does not know his/her port that is allocated to Arduino, processing will type the serial lists available which will display the port being used by Arduino. "PrintIn" function is known as the Print Integer where the serial port is printed in the value of an integer.

Void Draw ()

Instead of using “void loop()”, processing will require the usage of “void draw()” where the user will be required to write the bytes into the port of Arduino.

```
void draw() {

    if(keyPressed) {
    if (key == 'q' || key == 'Q') {
      port.write('Q');
    }
    if (key == 'w' || key == 'W') {
      port.write('W');
    }
    if (key == 'e' || key == 'E') {
      port.write('E');
    }
    if (key == 'r' || key == 'R') {
      port.write('R');
    }
    if (key == 'o' || key == 'O') {
      port.write('O');
    }
```

In this function, Processing is being told that if the alphabets are pushed when the applications are running, it will be asked to write it on the Arduino’s serial port. This process eliminates the use of enter being pressed after every alphabet.

Camera inside Processing

For the benefit of the ROV, the camera can be launched with the processing application itself. In order to establish this feature, the user will be required to add the video library into processing which can be done by inserting a line on the top of Processing.

```
import processing.video.*;
import processing.serial.*;
```

The library is inclusive of the Arduino's "port writing" code. After adding the line, the camera will have to be setup under "void setup ();" where it can be displayed in the image below:-

```
void setup() {
  size(640, 480);

  String[] cameras = Capture.list();

  if (cameras.length == 0) {
    println("There are no cameras available for capture.
    exit();
  } else {
    println("Available cameras:");
    for (int i = 0; i < cameras.length; i++) {
      println(cameras[i]);
    }

    cam = new Capture(this, cameras[0]);
    cam.start();
  }
```

The code indicates the program to print the names of the cameras available in the computer. In the case that there are no cameras attached to the computer, the application will prompt and write "There are no cameras available for capture" and will exit the application. In the case that there are cameras available, the application will print the frames per second of the cameras and start it up in the application giving the user live feed.

The last step required will be to add camera's feed into the application box that is launched after running the application therefore will be written as shown:-

```
void draw() {
  if (cam.available() == true) {
    cam.read();
  }
  image(cam, 0, 0);
```

3.5 Add-ons

Accelerometer

Accelerometer can be added to check the accelerations at the different axis of X, Y, Z. The ADXL335 is an example of an accelerometer than can be used with Arduino. The code will simply require the user to define the pins, set the pins to output and give the grounpin a "low" value while giving the powerpin a "high" value.

```
const int groundpin = 18;
const int powerpin = 19;
const int xpin = A0;
const int ypin = A1;
const int zpin = A2;

void setup()
{
  Serial.begin(9600);

  pinMode(groundpin, OUTPUT);
  pinMode(powerpin, OUTPUT);
  digitalWrite(groundpin, LOW);
  digitalWrite(powerpin, HIGH);
}
```

After defining the pins and setting it up, the user is required to set the values to read in a loop with a delay of 15 milliseconds.

```
void loop()
{
  // print the sensor values:
  Serial.print(analogRead(xpin)+168);
  // print a tab between values:
  Serial.print(" ");
  Serial.print(analogRead(ypin));
  // print a tab between values:
  Serial.print(" ");
  Serial.print(analogRead(zpin));
  Serial.println();
  // delay before next reading:
  delay(15);
}
```

Upon opening the serial port, the values will be displaced in X Y Z format with spacing in between every value.

Angle	-90	-80	-70	-60	-50	-40	-30	-20	-10	0	10	20	30	40	50	60	70	80	90
Acceleration	662	660	654	642	628	610	589	563	537	510	485	455	433	408	390	374	363	357	355

The values are printed in accelerations which can be converted into degrees by a few calculations or by referring to the table above.

Temperature and light dependent resistors

```
float temp;
int tempPin = A15;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  temp = analogRead(tempPin);
  temp = temp * 0.48828125;
  Serial.print("TEMPRATURE = ");
  Serial.print(temp);
  Serial.print("*C");
  Serial.println();
  delay(1000);
}
```

Temperature sensor

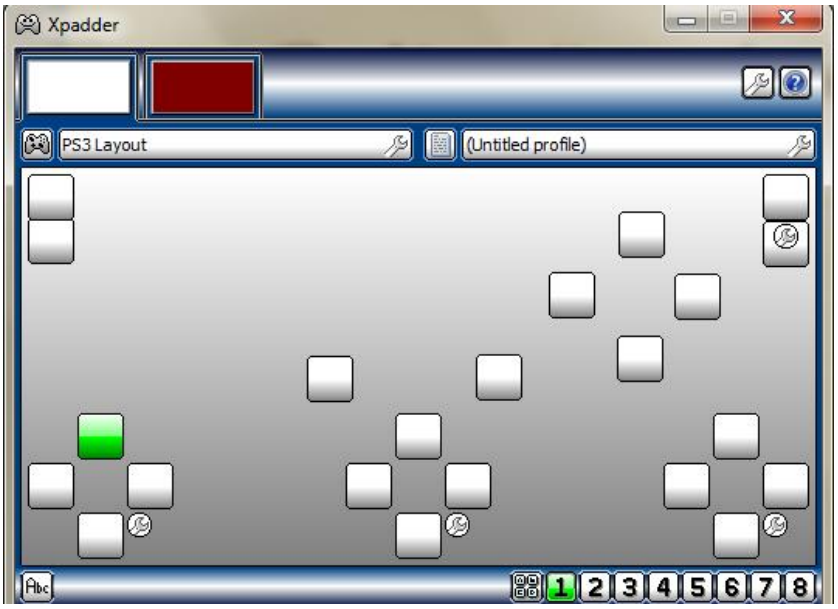
```
const int analogInPin = A8;
const int analogOutPin = 9;

int sensorValue = 0;
int outputValue = 0;

void setup() {

  Serial.begin(9600);
}

void loop() {
  // read the analog in value:
  sensorValue = analogRead(analogInPin);
  // map it to the range of the analog out:
  outputValue = map(sensorValue, 0, 1023, 0, 255);
  // change the analog out value:
  analogWrite(analogOutPin, outputValue);

  // print the results to the serial monitor:
  Serial.print("sensor = " );
  Serial.print(sensorValue);
  Serial.print("\t output = ");
  Serial.println(outputValue);
  delay(2);
}
```

Light dependent resistor

These sensors require the same processes where the pins have to be defined correctly and printed on the serial port. The light dependent resistor follows a slightly different format where the “sensorvalue” and “outputvalue” is defined before the function therefore making the line of coding look much neater and understandable.

Controllers

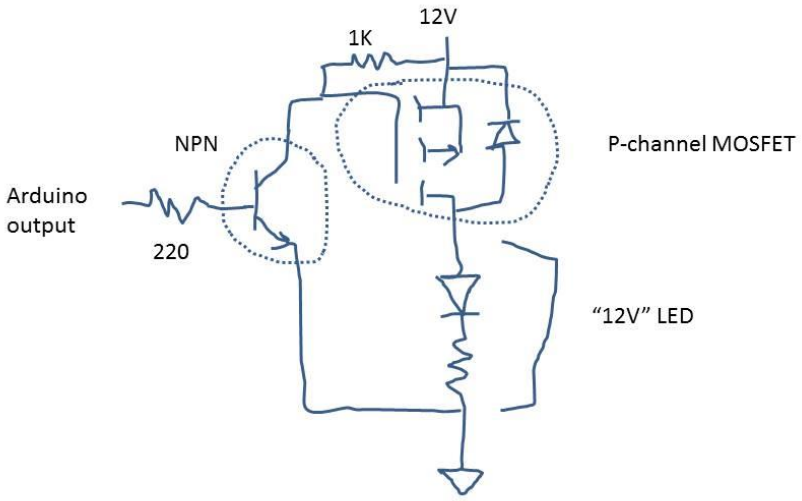
In order to send the signals using a controller into the port, a software called “Xpadder” can be used where it translates the button pressed on the controller into an ASCII format or an alphabet. However before using Xpadder, the keys have to be mapped according to the way you like them and the alphabets have to be linked to the press of the controller. The controller’s driver also has to be installed into the computer and detected in order for the process to complete.



MOSFETs

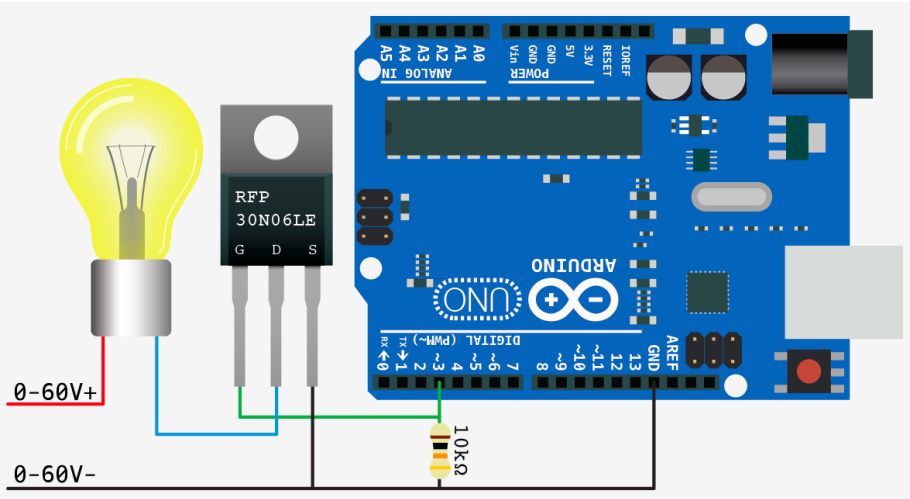
In order to control the LED, Arduino requires a MOSFET (Metal Oxide Semiconductor field effect transistor). These help in keeping up a 3 way switch known as the source, gate and body terminals. These transistors can control high current LEDs and comes in two different kinds known as a P channel MOSFET and N channel MOSFET. P-channel MOSFET, gate needs to swing between 12V (off) and 2V (Standard FET) or 7V (Logic level FET) to be on with the FET between 12V and the load. With a Logic Level N-channel FET, the gate needs to swing between 5V

(on) and 0V (off) with the FET between the load and Gnd therefore an NPN will be needed to pull the gate Low to turn it on, 1K resistor to 12V pulse gate to turn it off.



P channel MOSFET diagram

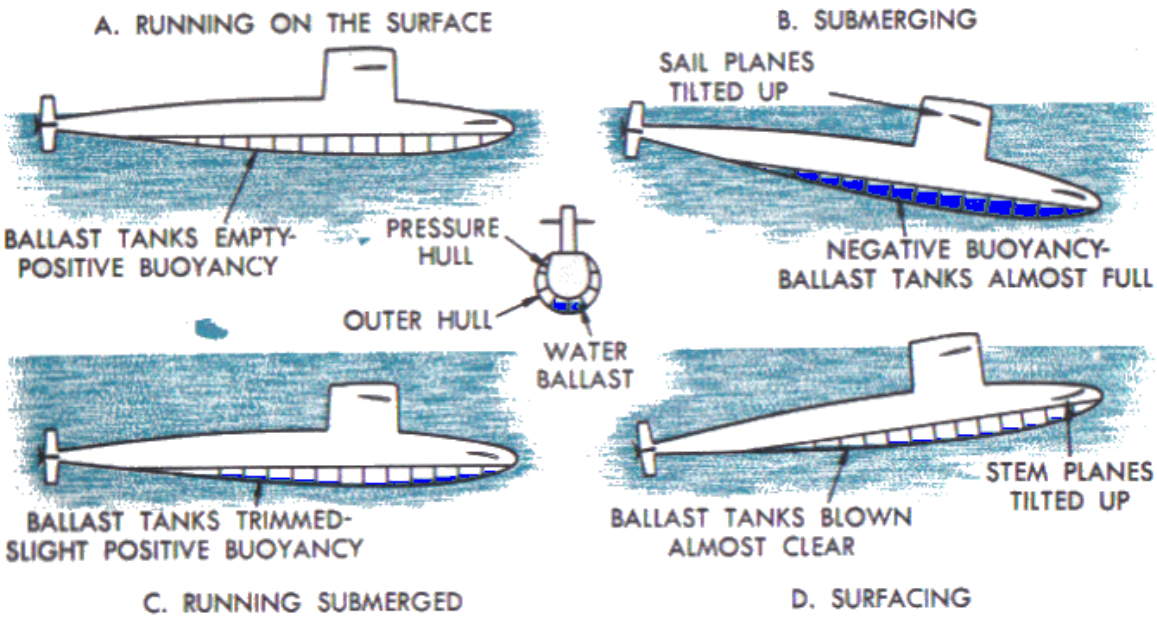
The N channel was therefore decided to be a better choice as it can directly be connected without the need of an NPN transistor.



N Channel connection to Arduino

3.6 Remotely Operated Vehicles

Every Remotely Operated Vehicle around the world serves a different purpose as designed by the user. They are known as the “underwater drones” where they can help inspect objects deep down in the water under the control of a human being above water. Every ROV has a different ballasting system according to the user designing it. The two type of ballasting is called dynamic/soft ballasting and static ballasting. Dynamic/soft ballasting is where the amount of water being displaced must be changed in the ballasting chamber in order for the ROV to move up and down.

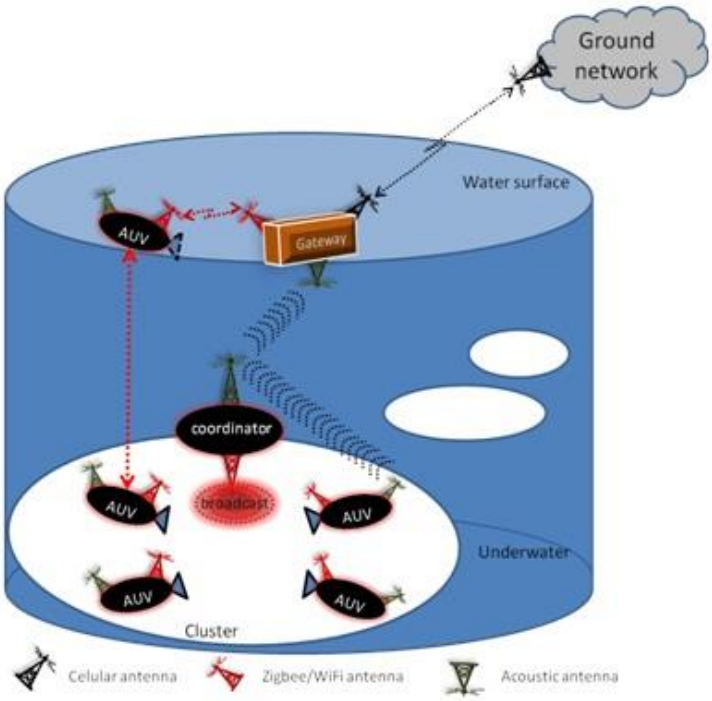


The image displays an example of how dynamic ballasting works inside submarines but are slightly modified but follows the same concept when it comes to ROVs.

Static ballasting is the ballasting system that is adapted by Sterling where the ROV is at neutral buoyancy and with the use of motors, it can be moved up and down.

The main forces consisted in any remotely operated vehicle is buoyancy, gravity and pressure. In a dynamic ballasting system, if the user requires the ROV to float, he/she will have to ensure that the buoyancy is greater than the gravity which will also mean that if pressure force is added, the ROV will sink.

During underwater surveillance, wireless communicators such as wifi and Bluetooth do not work too deep underwater therefore wires are the solutions used in those cases where the water is not very deep. If the ROVs are deeper than that, lasers sound and really low frequency radio signals are used to communicate with the ROV. Special cases such as Autonomous Underwater Vehicles (AUV) use a special technology to communicate underwater.



AUV are the next generations of ROVs which can go deep underwater but due to the power limitations so deep in water, the shapes are very

efficient to go underwater and can move around without using up much energy.

3.7 Buoyancy of Sterling

Middle hull radius = 3.5/2 = 1.75 "

Weight per foot = 1.41 lbs/ft

Weight per foot (lbs/inches) = 1.41/12= 0.1175 lbs/inch

Displacement: pi* 1.75*1.75*11.811 = 113.64 cubic inch

Upward acting buoyancy= 113.64*0.037059 lbs/cu = 4.21 lbs

Downward acting weight = 0.1175* 11.811 = 1.39 lbs

Net buoyancy = 4.21 – 1.39 = 2.82 lbs

Side hull radius = 2.375/2 = 1.1875"

Weight per foot= 0.68lbs/ft

Weight per foot (lbs/inch) = 0.68/12 = 0.05667

Displacement = pi*1.1875*1.1875*11.811 = 52.32 cubic inch

Upward acting buoyancy = 52.32*0.037059 = 1.94 lbs

Downwards acting weight = 0.05667*11.811 =0.669 lbs

Net buoyancy = 1.94 – 0.669= 1.271 lbs

Net buoyancy for both hulls = 1.271* 2 = 2.542 lbs

Total buoyancy = 2.542+2.82= 5.362 lbs

The total buoyancy of Sterling exclusive of the any equipment was 5.3 pounds which is close to 2.4 kilograms positive buoyancy.

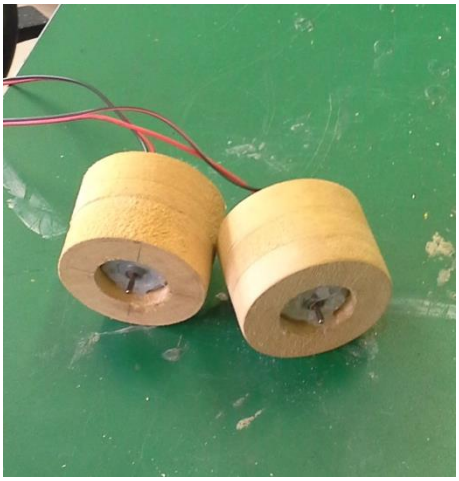
4 Fabrication

The first step to the fabrication process was to make sure the code was working before installing it on board. After purchasing the Arduino Mega and the Shield, two small 12V DC motors were purchased for the use of it. The initial plan was to use 4" PVC pipes as the main hull and 3" PVP pipes as the pontoons. The decision was adjusted to 3" PVC pipe as the main hull and 2" PVP pipe as the pontoon. This was mainly due to the ROV having too much air inside it which will cause the ROV to have a high rise in buoyancy.

SN	Parts	Quantity
1	Motors	3
2	Arduino Mega	1
3	Arduino driver shield	1
4	Arduino 2nd Driver	1
5	Power supply	1
6	PVC pipe 3" (400mm)	1
7	PVC Pipe 2" (400mm)	2
8	LED	1
9	MOSFET	1
10	Controller	1
11	Cement glue and Epoxy	1
12	Propeller	3
13	Shaft	1
14	Camera	1
15	Acrylic glass	1
16	Coupler	3

4.1 Motors

Due to the motor not being waterproof, it had to be seated inside the side hulls perfectly in in the center therefore they were cut exactly the size of the PVC pipe and the motors were forced and glued into the holes.

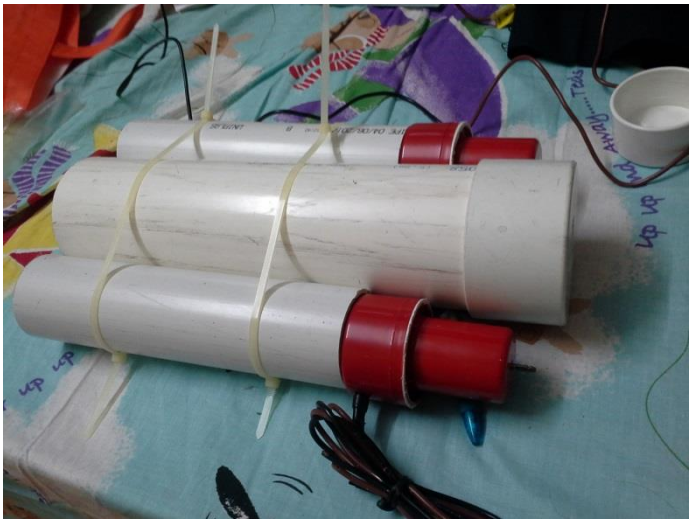


After the process, the PVC caps were drilled in the center so as for the shaft to enter. After testing the motors, it was found that one of the motor was having more friction that the other and this was due to the fact that the shaft was not properly aligned to the motor. For the ROV, if one of the motor rotated slower than the other, it would cause the ROV to turn around its own axis and which could lead to the failure in achieving the objective therefore these motors were replaced with bilge pumps. The bilge pumps had to be modified due to its primary function of being a pump. The outer shell was removed which just left the pump with the motor left which is fully waterproof.

Bilge pump before removing outer shell

Bilge pump after removing outer shell

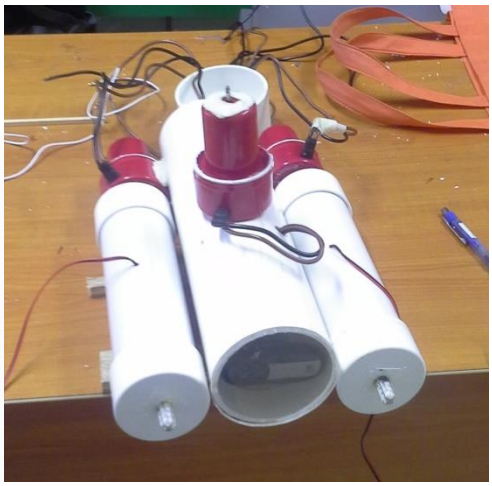
The bilge pumps were temporarily secured with cable ties before they were cemented onto the 3” PVC pipes. The motor to adjust the upward and downward moment was cemented onto the 3” PVC pipe.



4.2 LEDs and Camera

A Logitech C170 was purchased to provide live feed under water and the LEDs were 12V that require about 1.2A on full load. The webcam had to be secured onto something therefore a wooden panel that was half the size of the PVC pipe was made so it could easily be glued onto the sides of the PVC. The LED required holes onto the side pontoon caps where it was forced it and sealed tight to avoid any water leaks. A transparent acrylic was added to the front of the camera which would fully cover the PVC pipe but still allow vision for the camera. The sides of the PVC acrylic had to be sealed with cement as there were still tiny holes that would allow water to slip into the ROV.

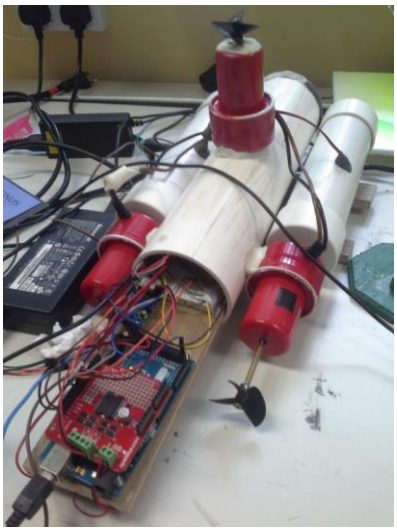
C170



After attaching the LED and camera

4.3 Chipsets and wiring

The Arduino board was then attached to a wooden plank that would allow ease of accessibility along with all the wires into the main hull.



After sliding the board in, the LED wires came off as they were only being held together with duct tape, therefore all the wires had to be soldered before putting it in to avoid any issues after it is sealed water tight. The propellers had also been glued onto the shafts and the motor wires were cut due to the shortage of space. Upon inserting everything, the ROV was then sealed and fully painted black. The ROV was then added into the water to test for any leaks.

5 Learning Outcomes/ Future work

5.1 Battery

Although the ROV is wired, using a poly lit ion battery would be recommended instead of a power supply from the plug

5.2 Arduino

Instead of using PS3 controller, it can be improved by using radio controller but the camera feed can't be done wirelessly underwater so an alternate solution would greatly improve the model.

5.3 PVC Pipe

Although there was enough space, 4" PVP pipe is greatly recommended for the main hull as the breadboard had to be cut in half in order to slot it in and the wires are in a mess.

5.4 Camera

Camera can be able to rotate with the simple installation of servos and an addition to the driver in Arduino with would help in better surveillance under water.

6 Challenges

Working alone	Although the ROV was possible to be done with just a single person, it could have reached greater heights by the ideation of more people and faster progress
Finding hardware	Finding hardware was considered one of the challenges in the journey of the project as most hardware shops lack of any website and have to be found by walking around or recommendations by friends or TSOs
3D modelling	Learning to use Blender was a new ground in which I completely had no clue about therefore it took some time understanding the software and modeling sterling
Coding	Coming from a course that does not specify in programming, it was challenging to understand the different functions on a self-taught basis.
Wiring the components	There were times where the Arduino had short circuited due to reversing the polarity or not attaching the wire properly.

7 Conclusion

The journey to reach to the end of the final year project has been challenging and along the way has continuously added skillsets to the students in various different ways. Everything learnt for the 3 years of a student's life is put to use during the project. Even after several failures, it is part of a learning journey to stay positive and reach towards the goal. With all the support provided by supervisors and TSOs, the ROV has become a successful one that has managed to meet the requirements from the objectives set on the first day of the project.

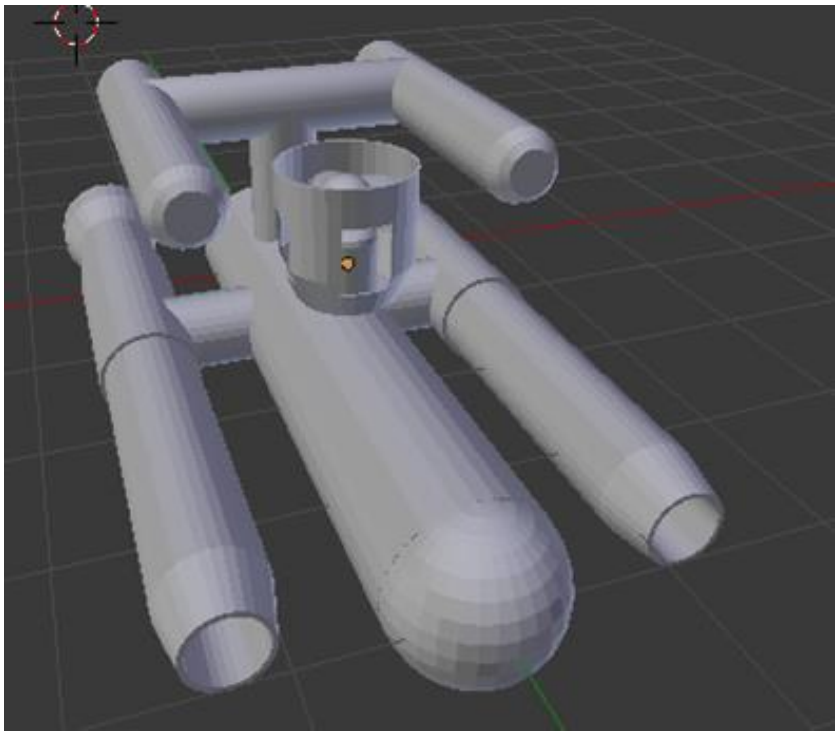
The ROV that has been created out of simple PVC pipes and Arduino chipset only, proves that brilliance lies in the simplicity of the design. The design will aim to serve divers for underwater inspections during the dry docking process which will eventually lead to making the shipyard a much better place to work in with much lower accidents.

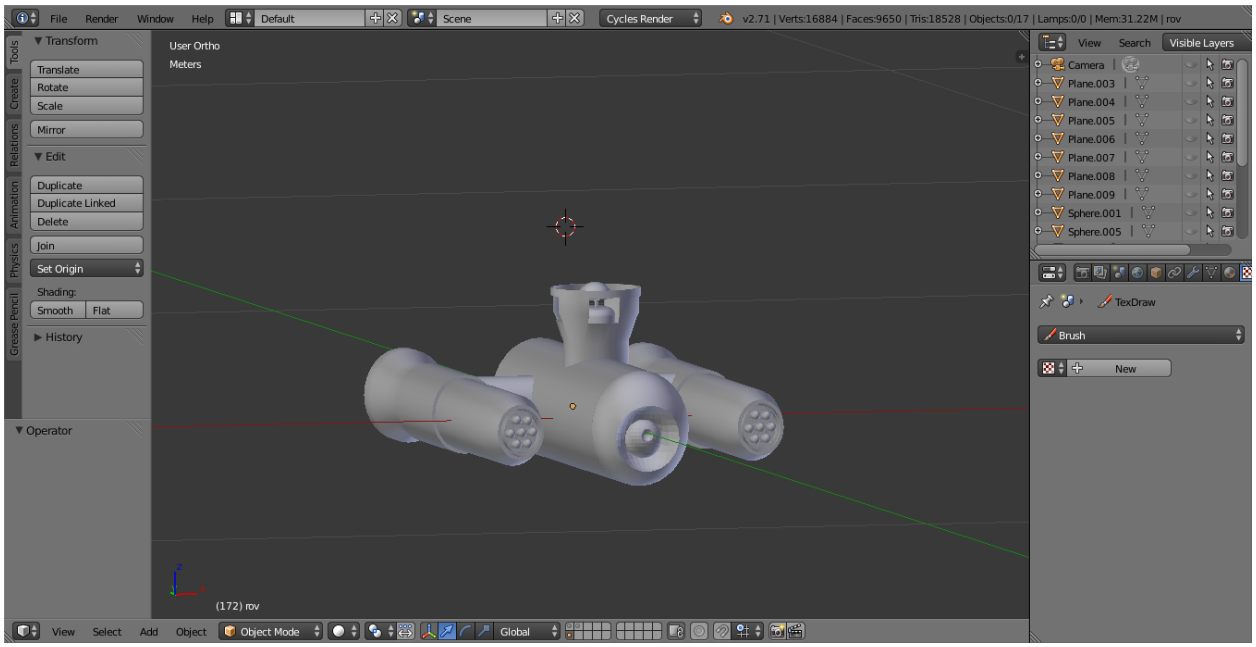
References

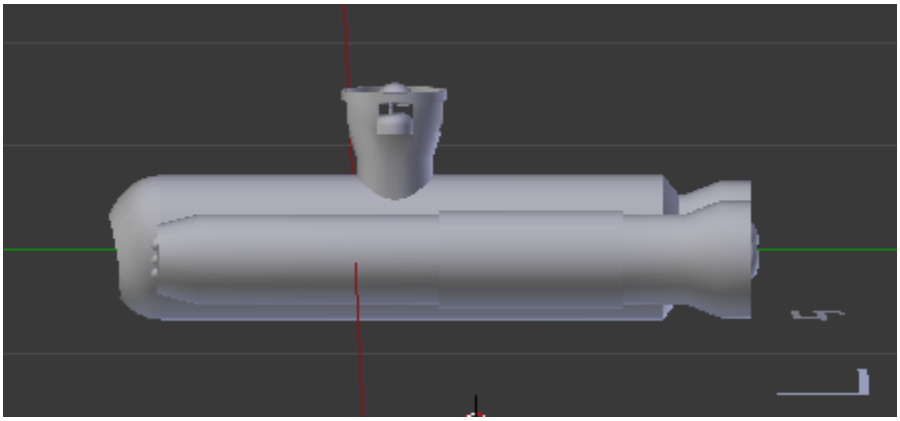
- Miami Science museum (2012). Introduction to Remotely Operated Vehicles. Retrieved from http://www.marinetech.org/files/marine/files/Curriculum/Other%20Curriculum%20Resources/RiseNet%20Intro%20to%20ROVs%20Learning%20Card_Final.pdf
- Rigzone 2010. Retrieved from https://www.rigzone.com/training/insight.asp?insight_id=343&c_id=17
- International Association for Great Lakes Research. Retrieved from http://www.iaglr.org/scipolicy/report/conclusion.php
- Tampan Mojidra (2013). Arduino temperature sensor LM35. Retrieved from http://www.instructables.com/id/ARDUINO-TEMPERATURE-SENSOR-LM35/
- Bildr 2004. High-Power Control: Arduino+ N channel MOSFET. Retrieved from http://bildr.org/2012/03/rfp30n06le-arduino/
- Kgdesign (2009). Need Help w/Gradual Motor Speed Increase. Retrieved from http://forum.arduino.cc/index.php?topic=44834.0
- JPWA (2014). JP ROV Update. Retrieved from http://www.homebuiltrovs.com/rovforum/viewtopic.php?f=18&t=1335
- Wracul (2011). Arduino underwater ROV. Retrieved from http://forum.arduino.cc/index.php/topic,22323.0.html

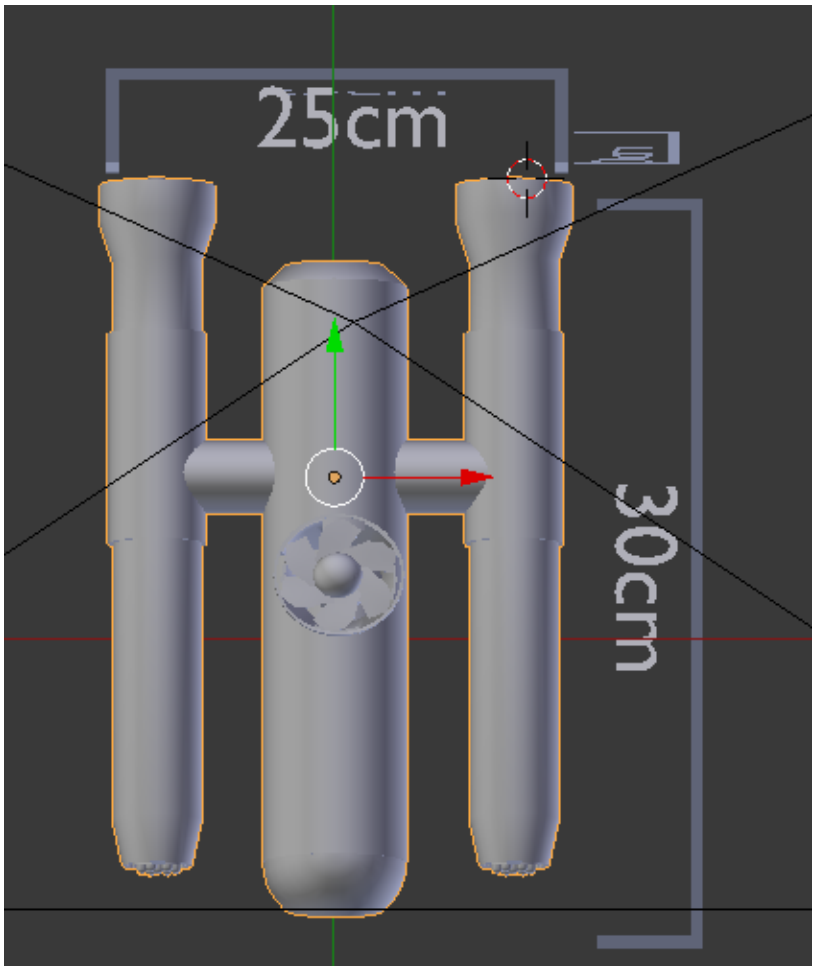
Appendix

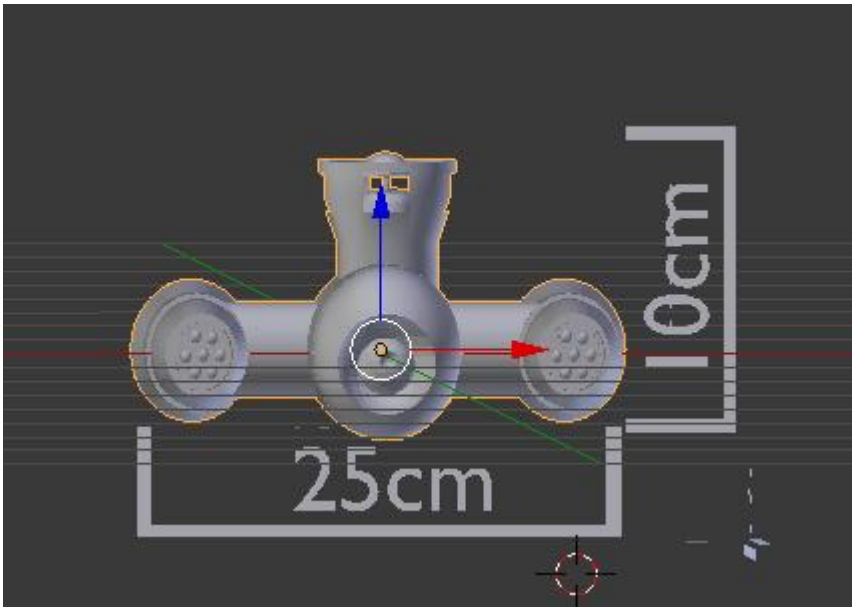
Initial designs



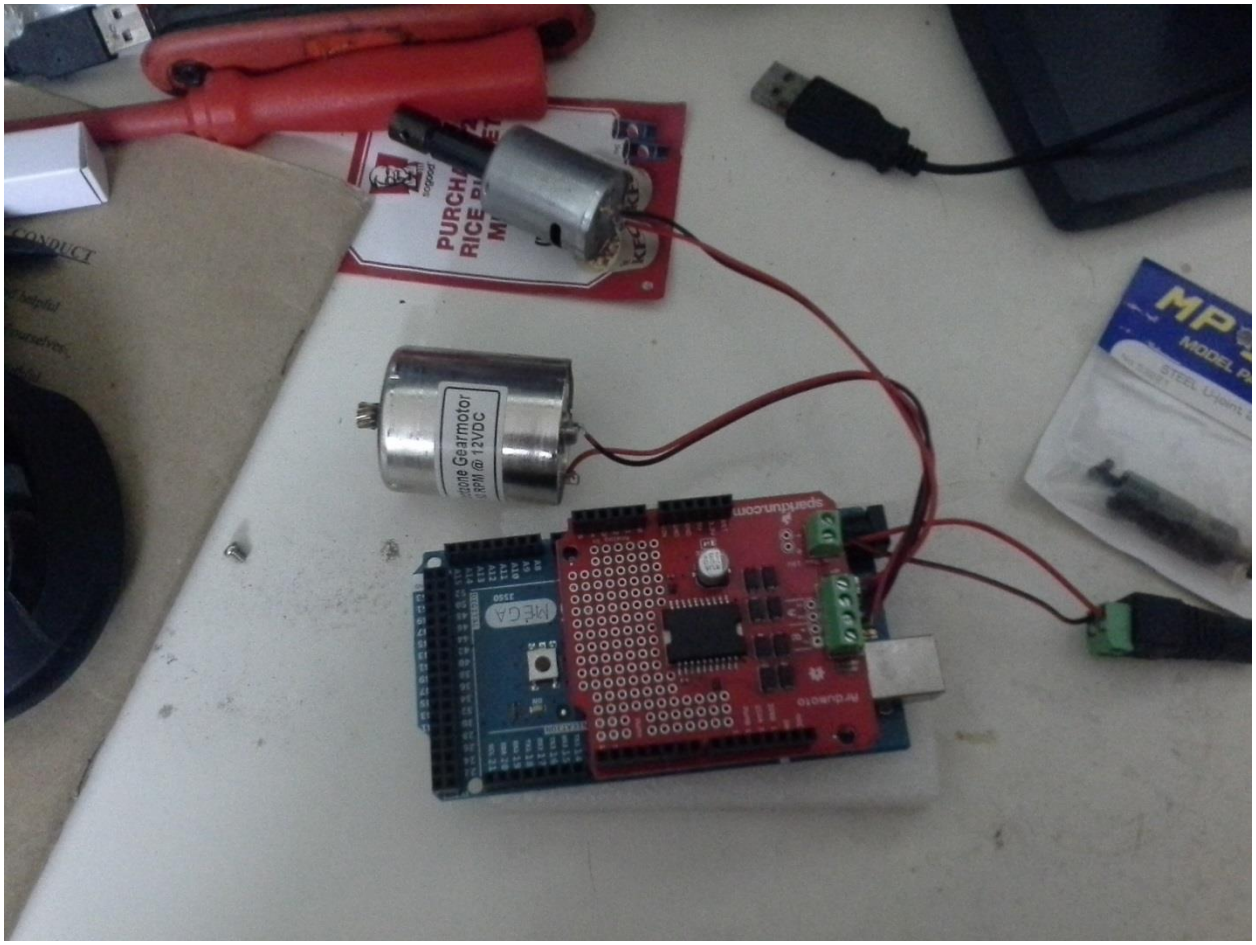




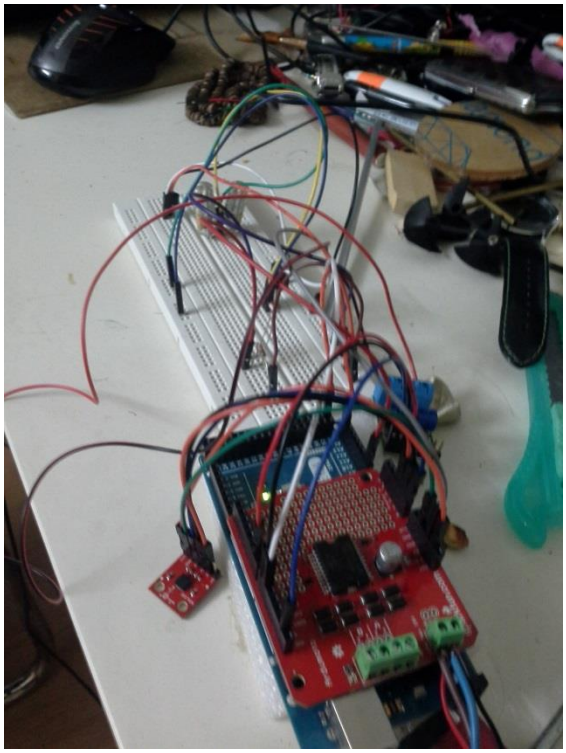

25cm
30cm



Arduino before being attached



Arduino’s coding stage



P channel MOSFET wiring