Actuator Control using Potentiometer based Joystick

Report on Summer Internship

Under the guidance of

Shri. N. Vedachalam

Scientist - F
Deep Sea Technologies Group
MOES- National Institute of Ocean Technology
Chennai

By

Mirza Ali Beg

Roll Number: 141104029
6th Semester, Bachelor of Engineering
Department of Electronics and Telecommunication
Goa College of Engineering, Goa

JULY 2017

"This Report is submitted as a part of curriculum for Bachelor of Engineering Programme at Goa College of Engineering, Goa"

CERTIFICATE

This is to certify that Mr. Mirza Ali Beg, 3rd year B.E. student (Electronics & Telecommunication Engineering) of Goa College of Engineering has completed two weeks of summer internship project work on the topic **"Actuator Control using Potentiometer based Joystick"** during the period 3rd July 2017 to 16th July 2017 under my supervision and guidance at the Deep Sea Technologies Group of National Institute of Ocean Technology, Chennai.

Date:-

Mr. N. Vedachalam

Scientist - F

Deep Sea Technologies group

National Institute of Ocean Technology

Ministry of Earth Sciences

Govt. of India

1. Acknowledgment

I would like to express the deepest gratitude and highest regard for my project guide Shri. N. Vedachalam for being the driving force behind this project. It was through his motivation and guidance that I was able to complete my project.

I would like to thank my mentors Mr. R. Ramesh and Mrs. V. B. N. Jyothi for helping me conceptualize the project and bestowing me with the requisite knowledge.

I am grateful to Dr. S. Ramesh (Scientist F) and Dr. G. A. Ramadass (Head of Deep Sea Technologies, NIOT) for giving me the opportunity to do this project as my summer internship in this reputed institute. This was a great exposure for me to understand how actual research and development is done in such prestigious institutes.

Heartfelt gratitude to my Head of Department Dr. H. G. Virani and Principal Dr. V. N. Shet of Goa Collage of Engineering for their constant support and encouragement.

2. Abstract

Remotely Operated Underwater Vehicle (ROV) constitutes half of the Unmanned Underwater Vehicles (UUV), the other half being Autonomous Underwater Vehicle (AUV).

As the name suggests ROVs are vehicles capable of operating underwater with the help of human input. The utility of ROVs have grown over the years ranging from applications in science to that in military.

A need has risen to establish an easier and reliable method of developing ROVs with the help of devices such as microcontrollers. In this project, we make use of a microcontroller and various other tools in an effort to develop an aspect of the propulsion systems used by ROVs.

3. Introduction

Unmanned underwater vehicles also called UUVs, are any vehicles that are able to operate underwater without a human occupant. These vehicles may be divided into two categories:

- Remotely Operated Underwater Vehicles (ROVs), which are controlled by a remote human operator.
- Autonomous Underwater Vehicles (AUVs), which operate independently of direct human input.

A **remotely operated underwater vehicle** (**ROV**) is a tethered underwater mobile device. This meaning is different from remote control vehicles operating on land or in the air. ROVs are unoccupied, highly maneuverable, and operated by a crew aboard a vessel. They are common in deep water industries such as offshore hydrocarbon extraction. They are linked to a host ship by a neutrally buoyant tether or, often when working in rough conditions or in deeper water, a load-carrying umbilical cable is used along with a tether management system (TMS). The TMS is either a garage-like device which contains the ROV during lowering through the splash zone or, on larger work-class ROVs, a separate assembly which sits on top of the ROV. The purpose of the TMS is to lengthen and shorten the tether so the effect of cable drag where there are underwater currents is minimized. The umbilical cable is an armored cable that contains a group of electrical conductors and fiber optics that carry electric power, video, and data signals between the operator and the TMS. Where used, the TMS then relays the signals and power for the ROV down the tether cable. Once at the ROV, the electric power is distributed between the components of the ROV. Most ROVs are equipped with at least a video camera and lights. Additional equipment is commonly added to expand the vehicle's capabilities. These may include sonars, magnetometers, a still camera, a manipulator or cutting arm, water samplers, and instruments that measure water clarity, water temperature, water density, sound velocity, light penetration, and temperature

i. ROV at work in an underwater oil and gas field.

3.1 ROV -Construction

Work-class ROVs are built with a large flotation pack on top of an aluminium chassis to provide the necessary buoyancy to perform a variety of tasks. The sophistication of construction of the aluminum frame varies depending on the manufacturer's design. Syntactic foam is often used for the flotation material. A tooling skid may be fitted at the bottom of the system to accommodate a variety of sensors or tooling packages. By placing the light components on the top and the heavy components on the bottom, the overall system has a large separation between the center of buoyancy and the center of gravity: this provides stability and the stiffness to do work underwater. Thrusters are placed between center of buoyancy and center of gravity to maintain the attitude stability of the robot in maneuvers. Various thruster configurations and control algorithms can be used to give appropriate positional and attitude control during the operations, particularly in high current waters. Thrusters are usually in a balanced vector configuration to provide the most precise control possible.

Electrical components can be in oil-filled water tight compartments or one-atmosphere compartments to protect them from corrosion in seawater and being crushed by the extreme pressure exerted on the ROV while working deep. The ROV will be fitted with cameras, lights and manipulators to perform basic work. Additional sensors and tools can be fitted as needed for specific tasks. It is common to find ROVs with two robotic arms; each manipulator may have a different gripping jaw. The cameras may also be guarded for protection against collisions. An ROV may be equipped with Sonar and LiDAR equipment.

3.2 ROV -Classification

Submersible ROVs are normally classified into categories based on their size, weight, ability or power. Some common ratings are:

- Micro - typically Micro-class ROVs are very small in size and weight. Today's Micro-Class ROVs can weigh less than 3 kg. These ROVs are used as an alternative to a diver, specifically in places where a diver might not be able to physically enter such as a sewer, pipeline or small cavity.
- Mini - typically Mini-Class ROVs weigh in around 15 kg. Mini-Class ROVs are also used as a diver alternative. One person may be able to transport the complete ROV system out with them on a small boat, deploy it and complete the job without outside help.
- General - typically less than 5 HP (propulsion); occasionally small three finger manipulators grippers have been installed, such as on the very early RCV 225. These ROVs may be able to carry a sonar unit and are usually used on light survey applications. Typically, the maximum working depth is less than 1,000 meters though one has been developed to go as deep as 7,000 m.
- Light Workclass - typically less than 50 hp (propulsion). These ROVs may be able to carry some manipulators. Their chassis may be made from polymers such as polyethylene rather than the conventional stainless steel or aluminium alloys. They typically have a maximum working depth less than 2000 m.
- Heavy Workclass - typically less than 220 hp (propulsion) with an ability to carry at least two manipulators. They have a working depth up to 3500 m.
- Trenching & Burial - typically more than 200 hp (propulsion) and not usually greater than 500 hp (while some do exceed that) with an ability to carry a cable laying sled and work at depths up to 6000 m in some cases.

3.3 The Control Station for an ROV

Control stations vary from large containers, with their spacious enclosed working area for work class systems, to simple PC gaming joysticks with PHDs (personal head-mounted displays) for some micro-ROV systems. All have in common a video display and some form of controlling mechanism (normally a joystick). On older analog systems, a simple rheostat controls the variable power to the electric motors, while newer digital controls are necessary for more advanced vehicle movements.

ii. Inside of a Control Station

With the rise of robotics as a sub-discipline within electronics, further focus highlighted the need to control robotic systems based upon intuitive interaction through emulation of human sensory inputs. Under older analog systems, a command of 'look left/go left' was a complex control command requiring the operation of several rheostats to gain vector thrusting to achieve the desired motion. As digital control systems arose, more complex control matrices could be implemented much more easily through allowing the circuit to proportionally control a thruster based upon the simple position of a joystick control.

The advent of the modern industrial joystick coupled with programmable logic circuits (or a microcontroller) has allowed easier control of the vehicle while operating through a much simpler and more intuitive interface. The more sensors available to the 'human' that allow intuitive interaction with the 'robot', the easier it is for the operator to figuratively operate the vehicle from the vehicle's point of view.

3.4 ROV -Applications

3.4.1 Science:

ROVs are also used extensively by the scientific community to study the ocean. A number of deep sea animals and plants have been discovered or studied in their natural environment through the use of ROVs. Science ROVs take many shapes and sizes. Since good video footage is a core component of most deep-sea scientific research, research ROVs tend to be outfitted with high-output lighting systems and broadcast quality cameras. Depending on the research being conducted, a science ROV will be equipped with various sampling devices and sensors. Many of these devices are one-of-a-kind, state-of-the-art experimental components that have been configured to work in the extreme environment of the deep ocean.

3.4.2 Military:

ROVs have been used by several navies for decades, primarily for minehunting and minebreaking. In October 2008, the U.S. Navy began to replace its manned rescue systems, based on the Mystic DSRV and support craft, with a modular system, the SRDRS based on a tethered, unmanned ROV called a pressurized rescue module (PRM). This followed years of tests and exercises with submarines from the fleets of several nations.

As their abilities grow, smaller ROVs are also increasingly being adopted by navies, coast guards, and port authorities around the globe, including the U.S. Coast Guard and U.S. Navy, Royal Netherlands Navy, the Norwegian Navy, the Royal Navy and the Saudi Border Guard. They have also been widely adopted by police departments and search and recovery teams. Useful for a variety of underwater inspection tasks such as explosive ordnance disposal (EOD), meteorology, port security, mine countermeasures (MCM), and maritime intelligence, surveillance, reconnaissance (ISR).

3.4.3 Survey:

Survey or Inspection ROVs are generally smaller than work class ROVs and are often sub-classified as either Class I: Observation Only or Class II Observation with payload. They are used to assist with hydrographic survey, i.e. the location and positioning of subsea structures, and also for inspection work for example pipeline surveys, jacket inspections and marine hull inspection of vessels.

3.5 Components of a Typical ROV

An ROV System consists of three primary components – underwater vehicle (the ROV), tether and control console. The tether is required to provide the real-time feedback to the operator including video and sensor data as well as control data for the ROV. Some (most) ROVs also are supplied with power through their tether, while some have onboard batteries. The actual underwater vehicle typically consists of the following components:

3.5.1 Sensors

Most industrial ROV systems provide the capability to transmit data from the submersible to the surface. This allows the ROV system to deliver a suite of instruments to the work site, powered by the vehicle, with data transmitted through the tether to the surface. Any combination of sensor and instrument (heading/gyro/depth etc.) is available as payload to the modern ROV system, assuming proper data protocol transmissions and power delivery are available. Common sensor packages placed aboard ROV systems are:

iii. Various Sensors for mounting to ROV

- Radiation Sensors
- CTD (conductivity/temperature/depth) sensors
- Pressure-sensitive depth transducer
- Magnetic flux gate compass module
- Slaved or rate gyro for heading stabilization
- Ultrasonic thickness gauges for measuring metal thickness and quality

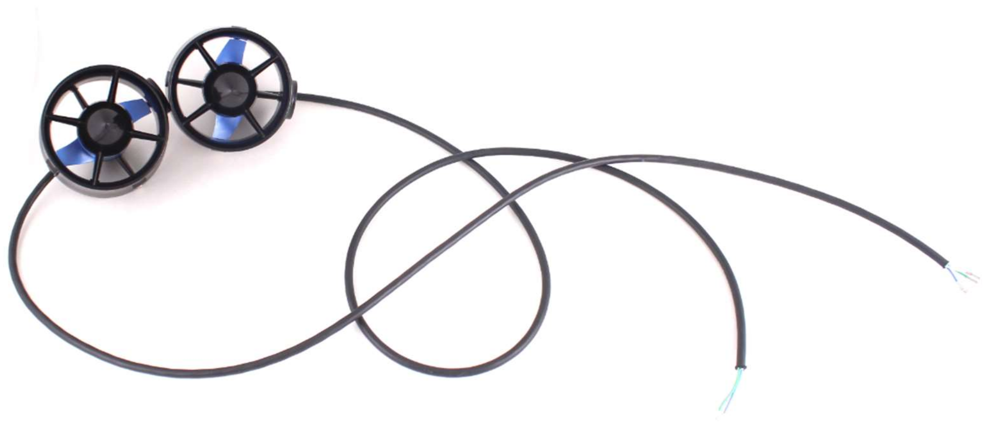
- Imaging sonar
- Acoustic positioning

3.5.2 Navigation

Radio waves cannot penetrate water very far, so as soon as an ROV dives it loses its GPS signal. Navigation can however be improved by using an underwater acoustic positioning system. When operating within a net of sea floor deployed baseline transponders, this is known as LBL navigation. With the help of a surface reference such as a support ship, ultrashort baseline (USBL) or short-baseline (SBL) positioning is used to calculate where the subsea vehicle is, relative to the known (GPS) position of the surface craft by means of acoustic range and bearing measurements. Estimates can be made using data from an Inertial Measurement Unit, and can be improved by adding a Doppler Velocity Log (DVL), which measures the rate of travel over the sea/lake floor.

3.5.3 Propulsion

Propulsion systems are classified as either Electric, Electro-hydraulic or ducted jet. Generally, the weight and relatively lower efficiency of an electro-hydraulic system effectively eliminates this system from consideration in vehicles weighing much less than 227 kg. In larger vehicles, however, it has the advantage of simplicity, ease of packaging, versatility, reliability, and low electrical noise. Although not a practical limitation in commercial operations, the higher acoustic noise inherent in the electro-hydraulic system may be important when considering the mission of the ROV, especially in the military mission of mine countermeasures.


iv. T100 Thrusters, a low-cost high performance thruster for marine robotics

Typical direct drive electric propulsion systems use a separate electric motor for each propeller, although a multiple output gearbox can be driven by a single motor. Electrical propulsion has weight advantages in small ROVs. Propulsion unit styles include:

- Continuous pitch propellers with constant speed motors (50/60 Hz)
- Variable frequency AC driven
- Universal motors with double reduction gear
- Brushless DC motors
- Permanent magnet brush type motors

3.5.4 Power

The type of power delivered to the submersible is a trade-off of cost, safety, and needed performance. Direct Current(DC) allows for lower cost and weight of tether components; Since inductance noise is minimal, it allows for less shielding of conductors in close proximity to the power line. Alternating Current(AC) allows longer transmission distances than that available to DC while using smaller conductors. Vehicles can be powered in any of the following three categories:

- *Surface-powered* vehicles must, by practicality, be tethered, since the power source is from the surface to the vehicle.
- *Vehicle-powered* vehicles store all of their power-producing capacity on the vehicle in the form of a battery, fuel cell, or some other means of power storage needed for vehicle propulsion and operation.
- *Hybrid system* involves a mixture of surface and submersible supplied power. Example of a hybrid system include the battery-powered submersible with a surface-supplied charger (through a tether) for recharging during times of less-than-maximum power draw.

3.5.5 Control System

The control system controls the different functions of the ROV, from controlling the propulsion system to switching of the light(s) and video camera(s). From simple relay control systems in the past to today's digital fiber optics, these systems are equipped with a microcontroller and subsystem control interface. The control system has to manage the input from the operator at the surface and convert it into actions subsea. The data required by the operator on the surface to accurately determine the position in the water is collected by sensors (sonar and acoustic positioning) and transmitted to the operator.

v. A mini ROV with its control console

4. ROSUB 6000

National Institute of Ocean Technology(NIOT) in collaboration with the Experimental Design Bureau of Oceanological Engineering(EDBOE) developed the Remotely Operable Submersible with 6000m depth capacity vehicle or the ROSUB 6000. It is an electric work class ROV equipped with under water luminaries and two manipulators with pay load capabilities of 150kg for mounting scientific and mission oriented tasks. The ROV has seven electric thrusters for maneuvering in manual or automatic mode. It has a Photonic Inertial navigation system (PHINS) supported by Doppler Velocity Log (DVL) and acoustic transponder/responder for position and navigation of ROV. Multibeam SONAR and depth sensor is also available for assisting navigation along with several video cameras and lights for monitoring. The ROSUB 6000 consists of the following subsystems.

vi. ROSUB, Deep sea water trial

- The Vehicle: Work Class Remotely Operated Vehicle (ROSUB 6000)
- Tether Management System (TMS)
- Handling Systems consist of Launch and Recovery System Crane (LARS), Umbilical Storage Winch, Traction Winch and Hydraulic Power Unit (HPU).
- Deck System consist of Containerized Control Console and Power House
- Electrical Power System
- Instrumentation and Data Telemetry System
- Navigation Control and Operational System (soft)

4.1 Specifications of the ROV -ROSUB 6000

Diving Depth	: 6000m
Size	: 2.53 x 1.8 x 1.7m
Weight	: 3700kg in air; -20kg in sea water
Mechanical Structure	: Al alloy (AA5056)
Floatation	: Syntactic foam
Payload	: up to 150kg
Propulsion	: Electrical (7 thrusters – 10kW each)
Forward Velocity	: 2 knots
Power	: Connected load 80kW: Nominal load 37kW)

- Pressure compensated step down transformer with rectifier unit
- National Instrument Compact Field Point Real Time controller with I/O modules
- Fiber optic media converter – 2 nos

- Camera : 3 Colour CCD, 2 B&W low light
- Manipulator : 1 no 5 function and 1 no 7 function
- Lights : 7 (LED, HID and Halogen)
- Subsea battery : 38 Ah
- Pressure compensator assembly with reservoir
- Dry pressure cases for electronics housing (AA6061 T6; Titanium alloy)
- Bulk heads, collectors and connector
- Capacitor bank in dry pressure case
- Navigation Equipment:
    - Doppler Velocity Log
    - Underwater-Photonic Inertial Navigation System
    - Sound Velocity Profiler
    - Depth Sensor
    - Acoustic Transponder
    - Radio beacon and flasher
- Scientific Payload:
    - RESON Multibeam Sonar 7125/7128
    - Oxygen Optode 3180
    - Conductivity Sensor 3919
    - Kinematic Methane Sensor from KMETS
    - Niskin Water Sampler
    - Short corer of 40cm length

5. Objective

To interface an analog joystick with a microcontroller in order to control the direction and speed of propulsion of an underwater vehicle with the help of servomotors and thrusters. This would require the use of a basic microcontroller such as the Arduino Uno, with the corresponding IDE installed in a computer to configure the microcontroller.

Hence, broadly we need to accomplish the following-

5.1 Interface the Joystick and Servomotor with the microcontroller
5.2 Establish control for the speed of a propeller/thruster
5.3 Assemble the vehicle
5.4 Test underwater

Here we shall primarily concentrate on the first two parts of this project i.e. Interfacing the joystick and servomotor with the microcontroller and establishing control for the speed of a propeller/thruster.

6. Microcontroller- The Main Processing Unit

A microcontroller (or **MCU** for microcontroller unit) is a small computer on a single integrated circuit. In modern terminology, it is a system on a chip or SoC. A microcontroller contains one or more CPUs (processor cores) along with memory and programmable input/output peripherals. Program memory in the form of Ferroelectric RAM, NOR flash or OTP ROM is also often included on chip, as well as a small amount of RAM. Microcontrollers are designed for embedded applications, in contrast to the microprocessor*s* used in personal computers or other general-purpose applications consisting of various discrete chips.

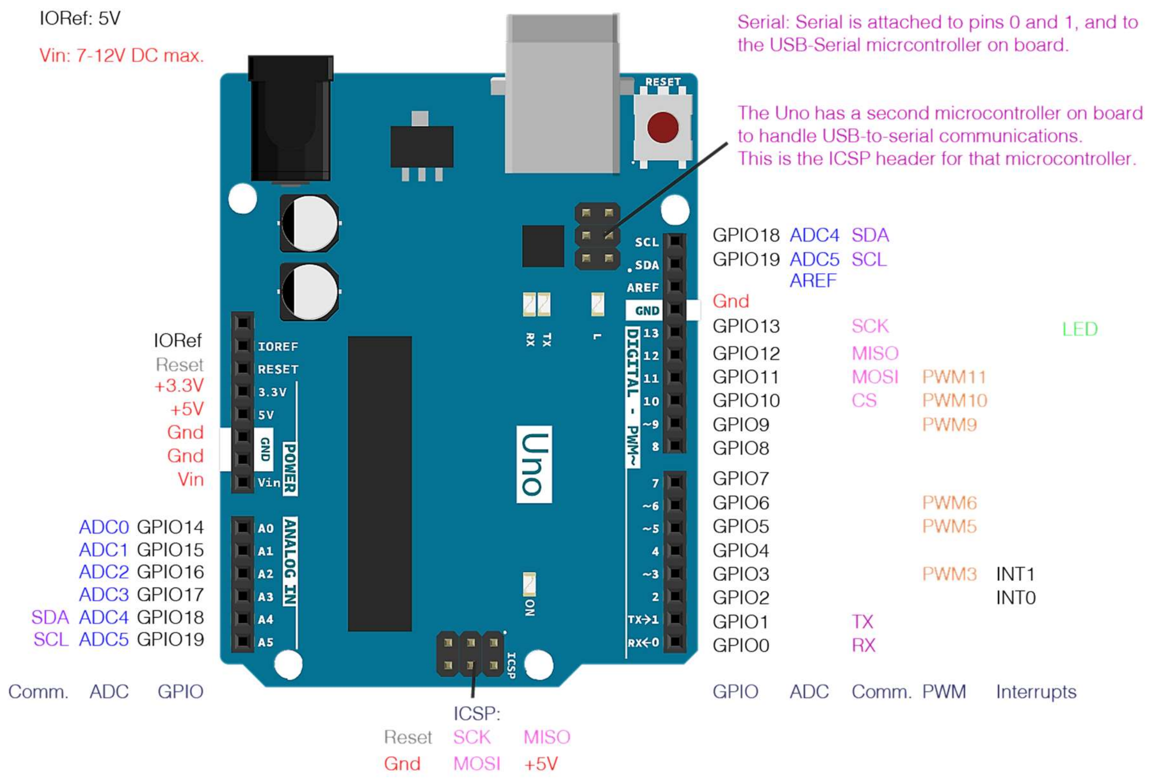
Microcontrollers are used in automatically controlled products and devices, such as automobile engine control systems, implantable medical devices, remote controls, office machines, appliances, power tools, toys and other embedded systems. By reducing the size and cost compared to a design that uses a separate microprocessor, memory, and input/output devices, microcontrollers make it economical to digitally control even more devices and processes. Mixed signal microcontrollers are common, integrating analog components needed to control non-digital electronic systems.

Some microcontrollers may use four-bit words and operate at frequencies as low as 4 kHz, for low power consumption (single-digit milliwatts or microwatts). They will generally have the ability to retain functionality while waiting for an event such as a button press or other interrupt; power consumption while sleeping (CPU clock and most peripherals off) may be just nanowatts, making many of them well suited for long lasting battery applications. Other microcontrollers may serve performance-critical roles, where they may need to act more like a digital signal processor (DSP), with higher clock speeds and power consumption.

A microcontroller is the main processing unit of an ROV as it is responsible for reading the input signals sent from the control station and then appropriately producing the desired action at the ROV as output. Microcontrollers are further used in other places within the ROV system, like in the control station itself to convert the human input into signals readable by the ROV. For example: the movement of the vehicle and its camera is controlled by a pair of joysticks which are interfaced with a microcontroller. Microcontrollers used in the field are heavy-duty and robust, but we can use a smaller microcontroller like the Arduino UNO for certain simulations.

6.1 Arduino Uno

Arduino Uno is a microcontroller board based on the ATmega328P (a single chip microcontroller). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller. It can be turned on by connecting it to a computer with a USB cable or by powering it with an AC-to-DC adapter or a battery.



vii. Arduino UNO

"Uno" means one in Italian and marks the release of Arduino Software (IDE) 1.0. The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino, now evolved to newer releases. The Uno board is the first in a series of USB Arduino boards, and the reference model for the Arduino platform.

Specifications:

- Microcontroller: ATmega328
- Operating Voltage: 5V
- Input Voltage (recommended): 7-12V
- Input Voltage (limits): 6-20V
- Digital I/O Pins: 14 (of which 6 provide PWM output)
- Analog Input Pins: 6
- DC Current per I/O Pin: 40 mA
- DC Current for 3.3V Pin: 50 mA
- Flash Memory: 32 KB (ATmega328) of which 0.5 KB used by bootloader

- SRAM: 2 KB (ATmega328)
- EEPROM: 1 KB (ATmega328)
- Clock Speed: 16 MHz

7. Procedure

For the entire process, we require the following main components: -

- A Computer/ Laptop with intel core i3 with minimum four gigabytes of RAM capable of running Arduino Software (IDE) 1.8.3
- An Arduino Uno board
- An analog joystick
- A high torque servo motor
- A high-performance thruster for marine robotics
- A power supply for the components
- A chassis and water tight enclosures for the components

Due to limitations of the resources available we will be simulating the process in basic conditions and hence cut down on a few requirements. Finally, we will be using the following components.

- A Laptop with intel core i3 with minimum four gigabytes of RAM capable of running Arduino Software (IDE) 1.8.3
- Arduino Uno board
- A 10k ohm potentiometer (in place of the joystick)
- Ultra Torque HS-645MG Servo from HI-tec
- The T100 Thruster and a Basic ESC-R1 speed controller from BlueRobotics
- A 12V adapter to power the thruster

The control for the movement of the servomotor and the speed of the thruster will require the same equipment therefore instead of using them simultaneously we shall consider them as separate cases.

Using the Arduino Software (IDE) 1.8.3 we will develop a system where the movement of the servomotor and thruster will be such that:

- Increase value of potentiometer(2.5V-5V)- The servomotor/thruster rotates Clockwise
- Decrease value of potentiometer(2.5V-0V)- The servomotor/thruster rotates Anti-Clockwise

Along with these actions the digital values corresponding to the varying resistance of the potentiometer (0-1023), the voltage supplied to the servomotor/thruster(0-5V), the angle of displacement of the servomotor (0-180 degrees) and a timestamp will be displayed.

We shall be using the following methods to achieve our end goal.

7.1. Using the ADC

The Arduino boards have a circuit inside called an *analog-to-digital converter or ADC* that reads this changing voltage and converts it to a number between 0 and 1023. If we consider a 10k ohm potentiometer connected to an analog pin with 5V as supply, when the shaft is turned all the way in one direction, there are 0 volts going to the pin, and the input value is

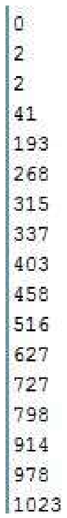
0. When the shaft is turned all the way in the opposite direction, there are 5 volts going to the pin and the input value is 1023. In between, analogRead() returns a number between 0 and 1023 that is proportional to the amount of voltage being applied to the pin.

This example shows how to read analog input from the physical world using a potentiometer. A potentiometer is a simple mechanical device that provides a varying amount of resistance when its shaft is turned. By passing voltage through a potentiometer and into an analog input on the board, it is possible to measure the amount of resistance produced by a potentiometer (or *pot* for short) as an analog value. In this example, the state of a potentiometer will be monitored after establishing serial communication between the Arduino UNO and a computer running the Arduino Software (IDE).

Connect the three wires from the potentiometer to the board. The first goes from one of the outer pins of the potentiometer to ground. The second goes from the other outer pin of the potentiometer to 5 volts. The third goes from the middle pin of the potentiometer to the analog pin A0. Plug the Arduino UNO board into the computer through USB and start the Arduino Software (IDE) and enter the code below.

```
// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.println(sensorValue);
  delay(1);        // delay in between reads for stability
}
```

Output:



```
0
2
2
41
193
268
315
337
403
458
516
627
727
798
914
978
1023
```

7.2. Using PWM

Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means. Digital control is used to create a square wave, a signal switched between on and off. This on-off pattern can simulate voltages in between full on (5 Volts) and off (0 Volts) by changing the portion of the time the signal spends on versus the time that the signal spends off. The duration of "on time" is called the pulse width. To get varying analog values, we change, or modulate, that pulse width. If this on-off pattern is repeated fast enough with an LED for example, the result is as if the signal is a steady voltage between 0 and 5v controlling the brightness of the LED.

This example shows how to read an analog input pin, map the result to a range from 0 to 255, use that result to set the pulse width modulation (PWM) of an output pin to dim or brighten an LED and print the values on the serial monitor of the Arduino Software (IDE).

Connect one pin from the potentiometer to 5V, the center pin to analog pin 0 and the remaining pin to ground. Next, connect a 220 ohm current limiting resistor to digital pin 9, with an LED in series. The long, positive leg (the anode) of the LED should be connected to the output from the resistor, with the shorter, negative leg (the cathode) connected to ground. Plug the Arduino UNO board into the computer through USB and start the Arduino Software (IDE) and enter the code below.

```
const int analogInPin = A0;  // Analog input pin that the pot is attached
const int analogOutPin = 9; //Analog output pin that the LED is attached
int sensorValue = 0;        // value read from the pot
int outputValue = 0;        // value output to the PWM (analog out)

void setup() {
  // initialize serial communications at 9600 bps:
  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);

  // wait 2 milliseconds before the next loop
  // for the analog-to-digital converter to settle
  // after the last reading:
  delay(2);
}
```

Output:

The Brightness of the LED increases when the voltage goes from 0 to 5V and it decreases from 5V to 0.

7.3. Controlling the Servomotor

Here we use a PWM output as discussed in the previous section

In this example, the position of a RC (hobby) servo motor is varied with an Arduino and a potentiometer.

Servo motors have three wires: power, ground, and signal. The power wire is typically red, and should be connected to the 5V pin on the Arduino board. The ground wire is typically black or brown and should be connected to a ground pin on the board. The signal pin is typically yellow or orange and should be connected to pin 9 on the board.

The potentiometer should be wired so that its two outer pins are connected to power (+5V) and ground, and its middle pin is connected to analog input 0 on the board. Plug the Arduino UNO board into the computer through USB and start the Arduino Software (IDE) and enter the code below.

```
#include <Servo.h>

Servo myservo;  // create servo object to control a servo

void setup() {
  Serial.begin(9600);
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}
void loop() {
  float sensorValue0 = analogRead(A0);
  float ang = 180 * sensorValue0 / 1023;
  myservo.write(ang);  //sets the servo position according to the scaled
value
  delay(1000);  // waits for the servo to get there
}
```

Output:

Here the Servo motor moves clockwise when voltage is increased from 2.5V to 5V and it moves anti-clockwise when decreased from 2.5V to 0.

7.4. Controlling Speed of Thruster

Here again we use a PWM output.

In this example, the speed of the thruster is varied with an Arduino and a potentiometer.

First the three wires of the T100 thruster is connected to Phase A(Red), Phase B(Black), Phase C(Yellow) wires of the basic ESC speed controller, then we connect the red wire on the other side with a 12V power supply and the black wire to the ground. Finally we connect the small signal (Yellow) wire to the PWM output pin of the arduino board. Plug the Arduino UNO board into the computer through USB and start the Arduino Software (IDE) and enter the code below.

```
#include <Servo.h>

byte servoPin = 9;
Servo servo;

void setup() {
  servo.attach(servoPin);
  Serial.begin(9600);
  servo.writeMicroseconds(1500); // send "stop" signal to ESC.
  delay(1000); // delay to allow the ESC to recognize the stopped signal
}

void loop() {
  float signal;
  float sensorValue = analogRead(A0);
  signal = map(sensorValue, 0, 1023, 1410, 1590);
  servo.writeMicroseconds(signal); // Send signal to ESC.
  delay(2000);

}
```

Output:

Here the speed of the thruster increases in clockwise direction when voltage is increased from 2.5V to 5V and it increases in anti-clockwise direction when decreased from 2.5V to 0.

7.5. Displaying Time Stamp

For this we require the PC/laptop to constantly send the date and time information to the Arduino. This is done by running a separate program on the PC/Laptop which will be run using the software “Processing 3.3.5”. Insert the following code in the software and run it after the Arduino has been configured.

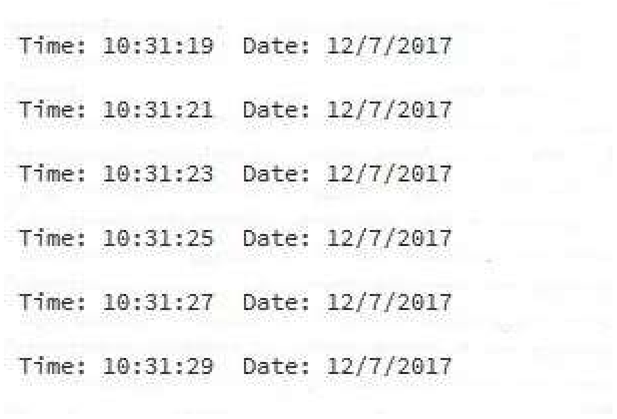
```
import processing.serial.*;
import java.util.Date;
import java.util.Calendar;
import java.util.GregorianCalendar;

public static final short portIndex = 0;  // select the com port
public static final String TIME_HEADER = "T";
public static final char TIME_REQUEST = 7;  // ASCII bell character
public static final char LF = 10;     // ASCII linefeed
public static final char CR = 13;     // ASCII linefeed
Serial myPort;     // Create object from Serial class

void setup() {
  size(200, 200);
  println(Serial.list());
  println(" Connecting to -> " + Serial.list()[portIndex]);
  myPort = new Serial(this,Serial.list()[portIndex], 9600);
  println(getTimeNow());
}

void draw()
{
  textSize(20);
  textAlign(CENTER);
  fill(0);
  text("Click to send\nTime Sync", 0, 75, 200, 175);
  if ( myPort.available() > 0) {  // If data is available,
    char val = char(myPort.read());      // read it and store it in val
    if(val == TIME_REQUEST){
       long t = getTimeNow();
       sendTimeMessage(TIME_HEADER, t);
    }
    else
```

```
      {
         if(val == LF)
             ; //igonore
         else if(val == CR)
           println();
         else
           print(val); // echo everying but time request
      }
    }
  }

  void mousePressed() {
    sendTimeMessage( TIME_HEADER, getTimeNow());
  }


  void sendTimeMessage(String header, long time) {
    String timeStr = String.valueOf(time);
    myPort.write(header);  // send header and time to arduino
    myPort.write(timeStr);
    myPort.write('\n');
  }

  long getTimeNow(){
    // java time is in ms, we want secs
    Date d = new Date();
    Calendar cal = new GregorianCalendar();
    long current = d.getTime()/1000;
    long timezone = cal.get(cal.ZONE_OFFSET)/1000;
    long daylight = cal.get(cal.DST_OFFSET)/1000;
    return current + timezone + daylight;
  }
```

Plug the Arduino UNO board into the computer through USB and start the Arduino Software (IDE) and enter the code below.

```
#include <TimeLib.h>
#define TIME_HEADER  "T"   // Header tag for serial time sync message
#define TIME_REQUEST  7    // ASCII character requests a time sync message

void setup()  {
  Serial.begin(9600);
  while (!Serial) ; // Needed for Leonardo only
  pinMode(13, OUTPUT);
  setSyncProvider( requestSync);  //set function to request sync
  Serial.println("Waiting for sync message");
}
void loop(){
  if (Serial.available()) {
    processSyncMessage();
  }
  if (timeStatus()!= timeNotSet) {
    digitalClockDisplay();
  }
  if (timeStatus() == timeSet) {
    digitalWrite(13, HIGH); // LED on if synced
  } else {
    digitalWrite(13, LOW);  // LED off if needs refresh
  }
  delay(1000);
}
void digitalClockDisplay(){
  // digital clock display of the time
  Serial.print(hour());
  printDigits(minute());
  printDigits(second());
  Serial.print(" ");
  Serial.print(day());
  Serial.print(" ");
  Serial.print(month());
  Serial.print(" ");
  Serial.print(year());
  Serial.println();
}
void printDigits(int digits){
  // utility function for digital clock display
  Serial.print(":");
  if(digits < 10)
    Serial.print('0');
  Serial.print(digits);
}
void processSyncMessage() {
  unsigned long pctime;
  const unsigned long DEFAULT_TIME = 1357041600; // Jan 1 2013

  if(Serial.find(TIME_HEADER)) {
     pctime = Serial.parseInt();
     if( pctime >= DEFAULT_TIME) { // check the integer is a valid time
       setTime(pctime); // Sync Arduino clock to the time received
     }
  }
}

time_t requestSync()
{
  Serial.write(TIME_REQUEST);
  return 0; // the time will be sent later in response to serial msg
}
```

Output:



```
Time: 10:31:19  Date: 12/7/2017

Time: 10:31:21  Date: 12/7/2017

Time: 10:31:23  Date: 12/7/2017

Time: 10:31:25  Date: 12/7/2017

Time: 10:31:27  Date: 12/7/2017

Time: 10:31:29  Date: 12/7/2017
```

Combining all the above methods we are able to complete the objective i.e. to control the servomotor/thruster with a potentiometer while displaying information like, the digital value corresponding to the variation in potentiometer value, input voltage, displacement angle/speed and the time stamp.

8. Final Code

8.1. For Servo Motor

```
#include <Servo.h>
#include <Time.h>
#include <TimeLib.h>

#define TIME_MSG_LEN 11 // time sync to PC is HEADER followed by Unix
time_t as ten ASCII digits
#define TIME_HEADER 'T' // Header tag for serial time sync message
#define TIME_REQUEST 7 // ASCII bell character requests a time sync
message
Servo myservo;  // create servo object to control a servo
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}
// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  float sensorValue0 = analogRead(A0);
  float sensorValue = 5 * sensorValue0 / 1023;
  float ang = 180 * sensorValue0 / 1023;
  myservo.write(ang);    // sets the servo position according to the
scaled value
  Serial.print("At Value= ");
  Serial.print(sensorValue0);
  Serial.print(",    The Voltage is= ");
  Serial.print(sensorValue);
  Serial.print("V");
  Serial.print("  and Angle is= ");
  Serial.print(ang);
  Serial.println(" Degrees");
  if (Serial.available()) {
    processSyncMessage();
  }
  if (timeStatus() == timeNotSet)
    Serial.println("waiting for time sync");
  else
    digitalClockDisplay();
  delay(2000);
  // delay in between reads for stability
}
void digitalClockDisplay() {
  // digital clock display of the time
  Serial.print("Time: ");
  Serial.print(hour());
  printDigits(minute());
  printDigits(second());
  Serial.print("  Date: ");
  Serial.print(day());
  Serial.print("/");
  Serial.print(month());
  Serial.print("/");
  Serial.print(year());
  Serial.println();
  Serial.println();
}
void printDigits(int digits) {
  // utility function for digital clock display: prints preceding colon
and leading 0
  Serial.print(":");
  if (digits < 10)
    Serial.print('0');
  Serial.print(digits);
}
void processSyncMessage() {
  // if time sync available from serial port, update time and return true
  while (Serial.available() >= TIME_MSG_LEN ) { // time message consists
of header & 10 ASCII digits
    char c = Serial.read() ;
    Serial.print(c);
    if ( c == TIME_HEADER ) {
      time_t pctime = 0;
      for (int i = 0; i < TIME_MSG_LEN - 1; i++) {
        c = Serial.read();
        if ( c >= '0' && c <= '9') {
          pctime = (10 * pctime) + (c - '0') ; // convert digits to a
number
        }
      }
      setTime(pctime); // Sync Arduino clock to the time received on the
serial port
    }
  }
}
```

8.2. For Thruster

```
#include <Servo.h>
#include <Time.h>
#include <TimeLib.h>

#define TIME_MSG_LEN 11 // time sync to PC is HEADER followed by Unix
time_t as ten ASCII digits
#define TIME_HEADER 'T' // Header tag for serial time sync message
#define TIME_REQUEST 7 // ASCII bell character requests a time sync
message
Servo myservo;  // create servo object to control a servo
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
  myservo.writeMicroseconds(1500); // send "stop" signal to ESC.
  delay(1000); // delay to allow the ESC to recognize the stopped signal
}
// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  float sensorValue0 = analogRead(A0);
  float sensorValue = 5 * sensorValue0 / 1023;
  float signal = map(sensorValue0, 0, 1023, 1410, 1590);
  myservo.writeMicroseconds(signal); // Send signal to ESC.
  Serial.print("At Value= ");
  Serial.print(sensorValue0);
  Serial.print(",    The Voltage is= ");
  Serial.print(sensorValue);
  Serial.print("V");
  Serial.print("  and thruster at= ");
  Serial.print(signal);
  Serial.println(" MicroSeconds");
  if (Serial.available()) {
    processSyncMessage();
  }
  if (timeStatus() == timeNotSet)
    Serial.println("waiting for time sync");
  else
    digitalClockDisplay();
  delay(2000);
  // delay in between reads for stability
}
void digitalClockDisplay() {
  // digital clock display of the time
  Serial.print("Time: ");
  Serial.print(hour());
  printDigits(minute());
  printDigits(second());
  Serial.print("  Date: ");
  Serial.print(day());
  Serial.print("/");
  Serial.print(month());
  Serial.print("/");
  Serial.print(year());
  Serial.println();
  Serial.println();
}
void printDigits(int digits) {
  // utility function for digital clock display: prints preceding colon
and leading 0
  Serial.print(":");
  if (digits < 10)
    Serial.print('0');
  Serial.print(digits);
}
void processSyncMessage() {
  // if time sync available from serial port, update time and return true
  while (Serial.available() >= TIME_MSG_LEN ) { // time message consists
of header & 10 ASCII digits
    char c = Serial.read() ;
    Serial.print(c);
    if ( c == TIME_HEADER ) {
      time_t pctime = 0;
      for (int i = 0; i < TIME_MSG_LEN - 1; i++) {
        c = Serial.read();
        if ( c >= '0' && c <= '9') {
          pctime = (10 * pctime) + (c - '0') ; // convert digits to a
number
        }
      }
      setTime(pctime); // Sync Arduino clock to the time received on the
serial port
    }
  }
}
```

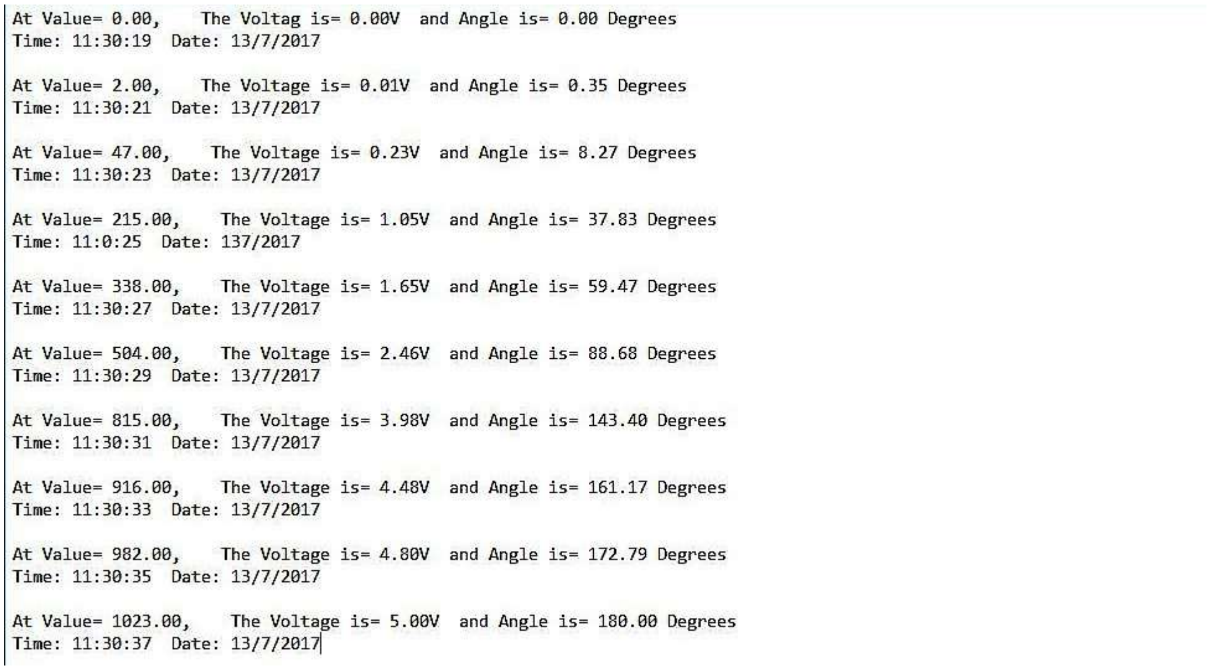
9. Output

The output comprises of:

9.1. For Servo Motor

- The physical motion of the servo motor according to the movement of the potentiometer as required. That is, clockwise movement when potentiometer value is increased and anti-clockwise movement when decreased.

- The display showing the digital value corresponding to the variation in the potentiometer, the input voltage, the displacement angle and the time stamp.

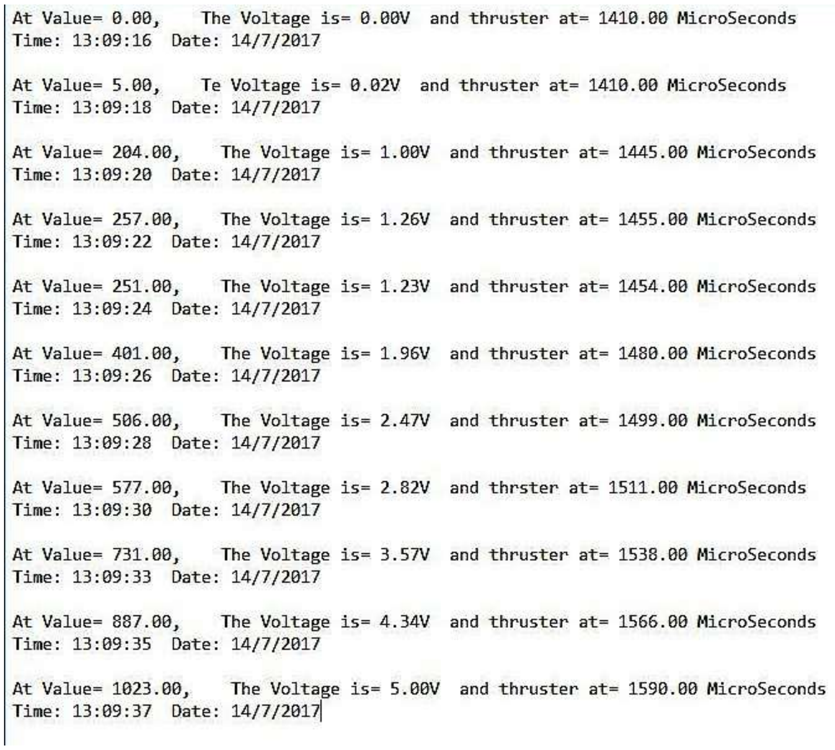


```
At Value= 0.00,    The Voltag is= 0.00V  and Angle is= 0.00 Degrees
Time: 11:30:19  Date: 13/7/2017

At Value= 2.00,    The Voltage is= 0.01V  and Angle is= 0.35 Degrees
Time: 11:30:21  Date: 13/7/2017

At Value= 47.00,    The Voltage is= 0.23V  and Angle is= 8.27 Degrees
Time: 11:30:23  Date: 13/7/2017

At Value= 215.00,    The Voltage is= 1.05V  and Angle is= 37.83 Degrees
Time: 11:0:25  Date: 137/2017

At Value= 338.00,    The Voltage is= 1.65V  and Angle is= 59.47 Degrees
Time: 11:30:27  Date: 13/7/2017

At Value= 504.00,    The Voltage is= 2.46V  and Angle is= 88.68 Degrees
Time: 11:30:29  Date: 13/7/2017

At Value= 815.00,    The Voltage is= 3.98V  and Angle is= 143.40 Degrees
Time: 11:30:31  Date: 13/7/2017

At Value= 916.00,    The Voltage is= 4.48V  and Angle is= 161.17 Degrees
Time: 11:30:33  Date: 13/7/2017

At Value= 982.00,    The Voltage is= 4.80V  and Angle is= 172.79 Degrees
Time: 11:30:35  Date: 13/7/2017

At Value= 1023.00,    The Voltage is= 5.00V  and Angle is= 180.00 Degrees
Time: 11:30:37  Date: 13/7/2017
```

9.2. For Thruster

- The physical motion of the thruster according to the movement of the potentiometer as required. That is, increased speed in clockwise direction when potentiometer value is increased and increased speed in anti-clockwise direction when decreased.

- The display showing the digital value corresponding to the variation in the potentiometer, the input voltage, the thruster speed.



```
At Value= 0.00,    The Voltage is= 0.00V  and thruster at= 1410.00 MicroSeconds
Time: 13:09:16  Date: 14/7/2017

At Value= 5.00,    Te Voltage is= 0.02V  and thruster at= 1410.00 MicroSeconds
Time: 13:09:18  Date: 14/7/2017

At Value= 204.00,    The Voltage is= 1.00V  and thruster at= 1445.00 MicroSeconds
Time: 13:09:20  Date: 14/7/2017

At Value= 257.00,    The Voltage is= 1.26V  and thruster at= 1455.00 MicroSeconds
Time: 13:09:22  Date: 14/7/2017

At Value= 251.00,    The Voltage is= 1.23V  and thruster at= 1454.00 MicroSeconds
Time: 13:09:24  Date: 14/7/2017

At Value= 401.00,    The Voltage is= 1.96V  and thruster at= 1480.00 MicroSeconds
Time: 13:09:26  Date: 14/7/2017

At Value= 506.00,    The Voltage is= 2.47V  and thruster at= 1499.00 MicroSeconds
Time: 13:09:28  Date: 14/7/2017

At Value= 577.00,    The Voltage is= 2.82V  and thrster at= 1511.00 MicroSeconds
Time: 13:09:30  Date: 14/7/2017

At Value= 731.00,    The Voltage is= 3.57V  and thruster at= 1538.00 MicroSeconds
Time: 13:09:33  Date: 14/7/2017

At Value= 887.00,    The Voltage is= 4.34V  and thruster at= 1566.00 MicroSeconds
Time: 13:09:35  Date: 14/7/2017

At Value= 1023.00,    The Voltage is= 5.00V  and thruster at= 1590.00 MicroSeconds
Time: 13:09:37  Date: 14/7/2017
```

10. Conclusion

ROVs have proved their usefulness time and again in various fields. It is of great importance to further the development of more and more efficient models. In this project, an effort was made to implement an easy and general control system for an ROV. With the help of hardware such as the Arduino Uno, HS-645MG Servo motor and a 10k ohm potentiometer, we were able to develop a simple and functional model of the remotely operated mechanism for control of the direction of propulsion of a small underwater vehicle.

11. References

Submersibles & Gas Hydrate Group, National Institute of Ocean Technology, Chennai – 600 100. Project Closure Report for “Development of Joint Technology for Unmanned Submersible Capable of Working up to 6000 Meters Water Depth (ROSUB 6000)” Volume-1, March 2011.

Submersibles & Gas Hydrate Group, National Institute of Ocean Technology, Chennai – 600 100. Project Closure Report for “Development of Joint Technology for Unmanned Submersible Capable of Working up to 6000 Meters Water Depth (ROSUB 6000)” Volume-2, March 2011.

Carafano, J., & Gudgel, A. (2007). The Pentagon’s robots: Arming the future [Electronic version]. Backgrounder 2093, 1-66. (https://en.wikipedia.org/wiki/Unmanned_underwater_vehicle)

Lin, P., Bekey, G., & Abney, K. (2008). Autonomous Military Robotics: Risk, Ethics, and Design[Electronic version]. (https://en.wikipedia.org/wiki/Unmanned_underwater_vehicle)

Robert D. Christ, Robert L. Wernli Sr. “The ROV Manual: A User Guide for Observation Class Remotely Operated Vehicles”.

Warner, Cody. "Improve Search and Recovery with ROVs". Deep Trekker. (https://en.wikipedia.org/wiki/Remotely_operated_underwater_vehicle).

"The Basic components of an ROV" (PDF). National Sun Yat-sen University. (https://en.wikipedia.org/wiki/Remotely_operated_underwater_vehicle).

Brian W. Evans. “Arduino Programming Notebook”. (playground.arduino.cc/uploads/Main/arduino_notebook_v1-1.pdf)

12. Figures

13. Glossary

1.	UUV	Unmanned Underwater Vehicle
2.	AUV	Autonomous Underwater Vehicle
3.	ROV	Remotely Operated Underwater Vehicle
4.	TMS	Tether Management System
5.	ROSUB 6000	Remotely Operable Submersible with 6000m depth capacity vehicle
6.	MCU	Micro-Controller Unit
7.	IDE	Integrated Development Environment
8.	PWM	Pulse Width Modulation
9.	ADC	Analog to Digital Converter