Development of a PIC16F877A-Based Robotic Fish for Oil Spill Detection and Location Communication

Michael Judrick Aringo, Anton Luis Boquer, John Matthew Enciso, and Robert Jaime Salvador

Department of Electronics and Computer Engineering
De La Salle University
2401 Taft Avenue, Manila, Metro Manila, Philippines
{michael_aringo, anton_boquer, john_enciso, & roberto_salvador}@dlsu.edu.ph

***Abstract* - Oil spills negatively affect aquatic ecosystems and environments, damaging marine wildlife in various ways, such as skin irritation, diseases, cancer, and even death. Additionally, aquatic habitats can get destroyed and ruined, leading to the relocation of other species and disrupting their natural behavior. To tackle this, this paper aims to develop a robotic fish to detect oil spills using PIC16F877A microcontrollers. It utilizes a light intensity/fluorescent sensor for oil spill detection and motors to control the fish's movement. Furthermore, it includes a GPS module for tracking location and an ESP8266 microcontroller for communication. This approach could provide a cost-effective solution for monitoring and responding to oil spills for the protection of marine environments. This successful design and implementation of the proposed robotic fish in simulation show its potential as a tool for detecting oil spills and minimizing the impact of oil spills on marine ecosystems.**

Index Terms* - *oil spill detection, fluorometer, PIC microcontroller, robotic fish;

I. Introduction

Oil is one of the fossil fuels that humans utilize to generate electricity, heat their houses, and provide power to several sectors of society, aiding in its economic growth. A specific type of fossil fuel is crude oil, which is obtained from ancient natural sources like plants and animals that are already in liquid form. This oil is found in reservoirs underneath the ocean floor, residing in the holes of ocean rocks. Oil companies transport this crude oil through trucks, pipelines, trains, or ships. In this operation, a leak may happen due to unforeseen events such as drilling malfunctions, pipeline breakage, or the most common one, the sinkage of a large oil ship or tanker. This disastrous event is largely known as an oil spill [1].

The occurrence of oil spills negatively affects the coastal and marine environment and other ecosystems. A small amount of oil can cause skin irritation, reproductive or developmental damage, liver disease, and alteration of the immune system in marine animals. Similarly, a large oil spill causes marine wildlife to acquire cancer or, worse, death. Indirect effects of oil spills include the relocation of species searching for food, increased hunt time, and life cycle disruptions [2].

In a recent event in the Philippines, a ship named MT Terranova sank on July 25, 2024, near Limay, Bataan. It was reported that the tanker was carrying industrial fule oil of 1.4 million liters. Because of this disaster, the government of Cavite declared a state of calamity as the oil spill reached their province. The governor estimated that 25,000 fisherfolk are negatively affected by this oil spill, which costs around one billion pesos in damage [3]. This event was preceded by a more catastrophic oil spill that happened in 2023 in Oriental Mindoro due to the sinking of an oil tanker named MT Princess Empress that was carrying industrial fuel oil weighing 800 thousand liters. This oil spill incident resulted in at least 41.2 billion pesos worth of damage to both coastal communities and the environment itself [4].

Because of this, several oil spill monitoring techniques and detection systems have been developed in the present time. Most of them are implemented as remote sensing algorithms on a satellite or other aerial vehicles or ships [5]. However, one emerging measuring instrument for oil dispersion or spill detection is the flourometer. A fluorometer is a commercially available, small-sized sensor that utilizes the fluorescence of the sample being tested. In simple terms, fluorescence measures the capacity of a sample to absorb electromagnetic radiation at one wavelength, often visible light, and reemit it at another wavelength that has a lower energy but a higher wavelength compared to the one that was absorbed by the sample. This property is heavily observed in crude oil, as almost all kinds of oil that were studied by scientists exhibited a strong fluorescence band when exposed to ultraviolet light. Thus, fluorometers are said to be one of the best choices for detecting oil spills or dispersion in bodies of water [6].

II. Design and Application

A. Objectives

The project primarily aims to develop a robotic fish that uses advanced control systems to detect and respond to environmental factors, specifically aimed at identifying oil spills. The secondary objectives are the following:

- To utilize a PIC16F877A microcontroller and an ESP8266 ESP-12E for managing the input from environmental sensors and output from navigation controls.
- To design and implement a communication system using UART protocol and GPS module that allows for real-time location tracking and data transmission.
- To integrate servo motors, a stepper motor, and a DC motor to simulate the natural movement of fish fins

and tails for effective propulsion and maneuverability.

- To develop a specialized sensor imitating a fluorometer that can detect oil spills based on their visual characteristics and initiate a response protocol.
- To create a user interface that allows remote control of the robotic fish via website, including manual override and automation features.

B. *3D Model*

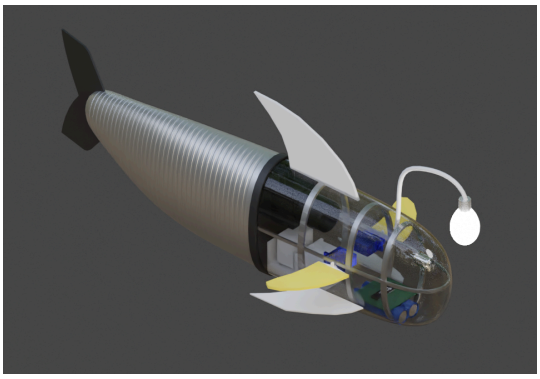


Fig. 1 The 3D model of the proposed robotic fish

The created model of the oil spill-detecting robotic fish is designed with a streamlined, aquatic form, closely mimicking the natural shape of a fish to ensure efficient hydrodynamics and smooth movement through water. The outer shell is constructed from glass, and the tail is made of silicone, providing protection to the internal components and ensuring the device can withstand harsh aquatic environments. The model focuses on illustrating the placement of components without including the wiring details.

C. *Block Diagram*

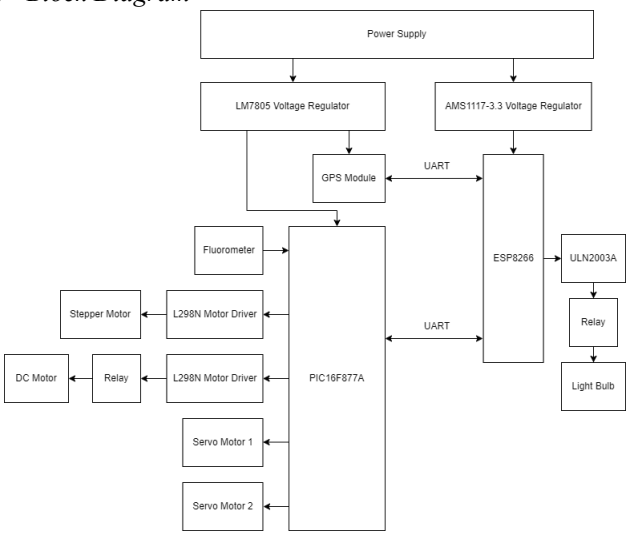


Fig. 2 The block diagram of the hardware setup

The figure above indicates the hardware connections of the proposed robotic fish and is patterned to [7]. The PIC16F877A microcontroller and GPS module are powered by the LM7805 voltage regulator, which supplies a fixed DC voltage of five volts. Meanwhile, the ESP8266 is connected to the AMS1117-3.3 voltage regulator, which provides a fixed DC voltage of 3.3 volts. Both of these regulators are connected to the main power supply, a 12-volt battery. In the PIC16F877A, the fluorometer is used as an input that will provide readings on the fluorescence of the water sample and, if present, the oil spill. The different motors are also interfaced in this microcontroller, including the two servo motors, the stepper motor, powered and controlled by a L298N motor driver, and the DC motor, which has a similar connection with the stepper motor with an additional relay. This microcontroller communicates with the ESP8266 via the UART protocol, which receives the fluorometer reading. A light bulb is interfaced to this module via the interconnection with ULN2003A and a relay. This module also produces the control signals from the buttons found in its webserver. The GPS module is also interfaced into this module to provide the real-time location of the robotic fish.

D. *Power Supply*

The main power supply of the proposed robot is a battery with a DC voltage of 12 volts. This will power the relays, motor drivers, motors, and other actuators. Connected to this voltage source is a LM7805 voltage regulator, which supplies five volts DC, powering the PIC microcontroller, servo motors, an operational amplifier, a voltage reference, and an analog switch. An AMS1117-3.3 voltage regulator is connected to this to obtain a DC voltage of 3.3 volts, which powers the ESP8266 module.

E. *Microcontroller*

The main microcontroller utilized in this project is the PIC16F877A microcontroller. This can be either a 40-pin or 44-pin package with five I/O ports, 15 interrupts, eight A/D input channels, and a parallel slave port. It operates with a DC voltage with an operating frequency of 20 MHz. It allows serial communication with other microcontrollers using the MSSP or USART protocol [8]. This device is used to obtain the fluorometer reading and control the motors, which mimic the movement of the fish.

Communicating with this microcontroller using the UART protocol is the NodeMCU ESP8266 module, which is a development board heavily utilized in applications of IoT technology. It is powered by a regulated DC voltage of 3.3 volts using a USB port. This is the reason why the FXMA108 voltage level shifter is used in the middle of their UART communication line. A shift between the voltages of 5 V and 3.3 V is required for them to have accurate communication. It has 16 GPIO (general purpose input-output) pins, four SPI communication pins, and two UART protocol interfaces, UART0 and UART2, having four pins. UART1 is utilized by the board to upload the program or firmware [9]. This board is used in the project to provide fish control and real time location sharing via its provided web server.

F. *Fluorometer*

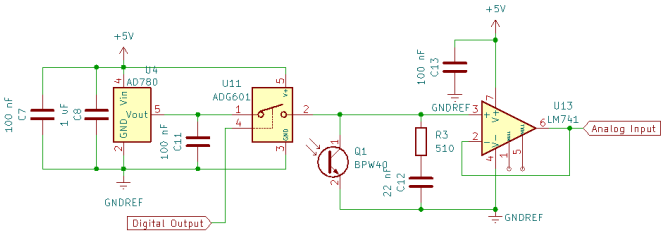


Fig. 3 The schematic diagram of the specialized fluorometer

The circuit for the measurement of the fluorescence of the liquid sample is patterned in [10]. It consists of a voltage reference IC, an analog switch, a phototransistor, and an operational amplifier. The voltage reference IC is powered by a regulated five-volt DC. The analog switch accepts the output voltage of this IC and is controlled by one of the digital output pins of the microcontroller, which turns it on and off. Connected to the other end of the switch is a phototransistor, which acts as a light sensor. A capacitor is connected to it in parallel, and the fluorescence can be measured by the discharge of this capacitor. An impedance amplifier is connected to this node in order for the analog input of the microcontroller to read this applied voltage properly without a high current flowing through it.

G. *DC Motor*

A DC motor is utilized in this robot to mimic the tail movement of the fish. The alternation of the voltage terminal corresponds to the flipping of the fish tail. Thus, in the project, the DC motor is interfaced in a relay that is further connected to a L298N motor driver, whose three pins are interfaced in the PIC microcontroller as output pins. The pins include the IN1, IN2, and EnA pins. These two external devices control and supply the necessary power to activate the DC motor properly.

H. *Stepper Motor*

A stepper motor is used in this project to provide balance and tilt control for the robotic fish. The stepper motor, similar to the DC motor, is interfaced with a L298N motor driver to control and provide the necessary power for the stepper motor. Four pins of the microcontroller are used as an output to control the stepper motor. With this, all of the input pins of the motor driver are utilized, and the enable pins for each pair are activated.

I. *Servo Motor*

Two servo motors are used in this robotic fish to provide the direction change for westward and eastward. Both of them utilized the PWM technique for precise positioning of both the left and right fins of the fish. With this, two pins of the PIC microcontroller are utilized as output pins, specifically CCP1 and CCP2.

In the program, the following PWM signals and values are used to control the two servo motors:

TABLE I

PWM CONTROL

Register	Bits	Description	Value	Explanation
PR2	All	PWM period register	0xFA	Sets the PWM period
CCP1C0N	5-4	Least significant 2 bits of 10 bit duty cycle	0x0C	Sets the PWM mode and the least significant bits of the duty cycle for CCP1
CCP2C0N	5-4	Least significant 2 bits of 10 bit duty cycle	0x0C	Sets the PWM mode and the least significant bits of the duty cycle for CCP2
T2C0N	2-0	Timer 2 ON/OFF, Prescaler	0x07	Sets the prescaler and the timer 2

J. *GPS Module*

A GPS module is utilized in this robotic fish to provide its real-time location. The project specifically made use of the NEO-6M GPS module. It is considered to be one of the most powerful GPS modules, as it can track around 22 satellites over 50 channels and has a -161 dB tracking sensitivity. Although it has these efficient features, it only requires a current of 45 mA. Also, it offers PSM, or power save mode, which alternately switches a specific part of the receiver, which reduces power consumption [11]. This is interfaced into the robotic system by using the UART protocol to communicate with the ESP8266 module. Since the two modules operate at different voltage levels, the FXMA108 voltage level shifter is used to shift between 5 V and 3.3 V.

K. *Light Bulb*

A 12-volt light bulb is used in the robotic fish, which activates whenever there is a detected oil spill. It is powered and controlled by ULN2003A and is interfaced in the ESP8266 module.

L. *ESP8266 Webserver*

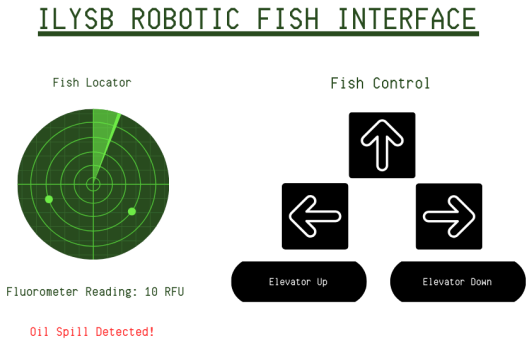


Fig. 4 A simple UI of the ESP8266 webserver

The webserver of the ESP8266 module hosts the location detector of the fish programmed as a radar system, as well as the buttons that will control the movement of the robotic fish. The fluorometer reading is also displayed underneath the radar system to view the current fluorescence of the liquid. When the threshold of this reading is reached, the phrase, “Oil Spill Detected!” is printed on the website. On the other hand, the buttons allow forward movement (up button), westward movement (left button), eastward (right button), and the elevator up and down of the robotic fish.

M. Motion of the Robotic Fish Oil Spill Detector

The design and functionality of the robotic fish oil spill detector involve a coordinated interplay between various actuators, including servo motors, a DC motor, and a stepper motor, each serving distinct roles in enabling the fish's movement and control. This section details the mechanisms by which these components work together to achieve effective motion in an aquatic environment.

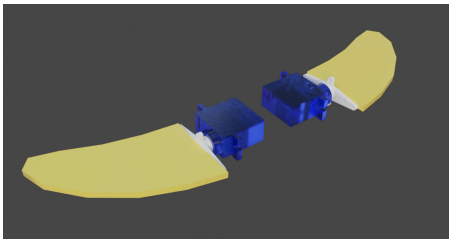


Fig. 5 The side fins with the servo motors

The robotic fish is equipped with servo motors that are strategically placed to control the side fins. These servo motors provide precise adjustments to the fins' angles, which are crucial for managing the fish's direction. By altering the orientation of the side fins, the robotic fish can execute smooth turns, dives, and ascents. The fins' movements are responsive to the inputs from the servo motors, allowing the robotic fish to maintain stability while swimming or to change its trajectory with minimal delay.

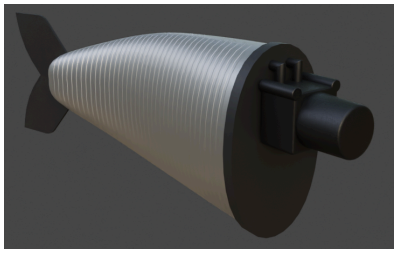


Fig. 6 The tail with the water pump

The primary propulsion of the robotic fish is driven by a DC motor that actuates the tail fin. The DC motor is responsible for oscillating the tail fin in a lateral motion, which generates the thrust required for forward movement. The amplitude and frequency of the tail fin's oscillations are controlled by the DC motor, allowing the fish to modulate its speed and direction of travel. By adjusting the tail fin's motion, the robotic fish can achieve both straight-line swimming and turning. When a turn is necessary, the tail fin's oscillations are altered in coordination with the side fins, enabling the fish to change direction efficiently.

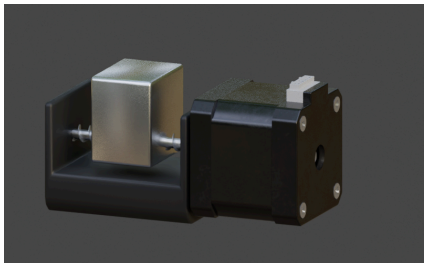


Fig. 7 The balance control unit

The stepper motor within the robotic fish controls a balance control unit, which plays a critical role in managing the fish's buoyancy and pitch. The balance control unit adjusts the internal center of gravity by moving a weight or altering an internal mechanism, effectively controlling the fish's pitch angle. This mechanism allows the robotic fish to dive or ascend by tilting its body upwards or downwards. The stepper motor's precise control enables fine adjustments, ensuring that the fish can maintain a desired depth or quickly adapt to changes in the underwater terrain. Additionally, the side fins assist in these movements by adjusting their angles in response to the fish's pitch, further stabilizing the motion.

N. Main Function Flowchart

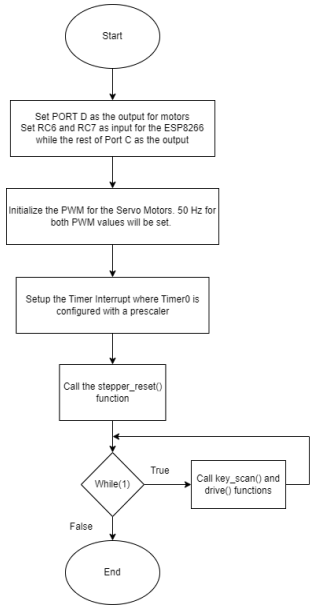


Fig. 8 Flowchart for the main function module

The figure above displays the flow of the main function module of the system itself. It is responsible for initializing the main components of the system, and it continuously controls the movement of the fish based on the inputs from the user. This module includes an initialization setup where, after its

completion, the module would enter an infinite loop where it would scan for user inputs via the ESP8266 module.

O. Oil Detection Flowchart

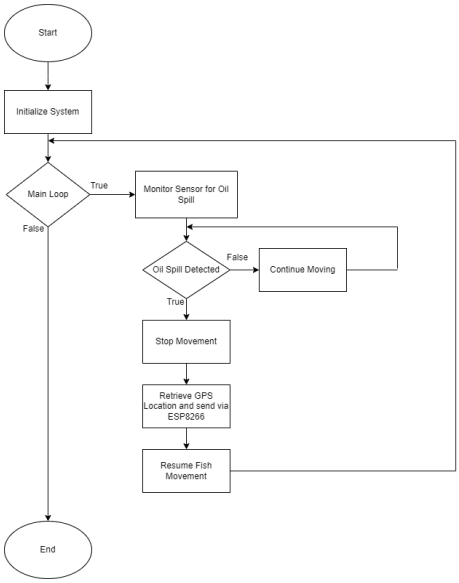


Fig. 9 Flowchart for the oil detection module

The figure above displays the general oil detection flowchart. Essentially, it is an infinite loop for the system to follow and search for oil spills within a given area. If the sensor of the system detects oil spills, it will stop its current movement and transmit the data from the GPS via the ESP8266. Otherwise, if it does not sense any oil spills, it will continue moving until the next discovery.

P. Drive Flowchart

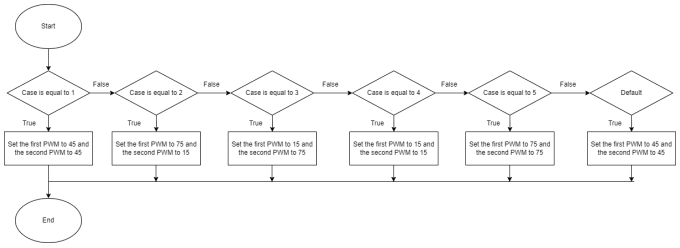


Fig. 10 Flowchart for the drive function module

The figure above displays the key_scan() function model of the system. This function is responsible for reading the current state of the keys or the inputs of the user. This would determine if the machine would go forward, up, down, left, or right.

III. Review of Related Literatures

The integration of biologically inspired control systems in robotic fish, such as the Central Pattern Generator (CPG)-based model developed by Korkmaz et al., offers valuable insights for enhancing the functionality of an oil spill-detecting robotic fish. In their study, the CPG model, inspired by Lamprey's spinal cord, enabled the robotic fish to execute realistic and rhythmic swimming patterns while adapting to environmental stimuli through a closed-loop fuzzy logic controller. This hierarchical control structure mimics the Central Nervous System (CNS), allowing the fish to respond, dynamically to various aquatic conditions. For an oil spill-detecting robotic fish, integrating a similar sensory feedback mechanism could optimize the detection and tracking of oil spills by ensuring smooth and efficient navigation through affected areas. The use of adaptive control systems based on biological principles can enhance the robotic fish's ability to operate autonomously in complex underwater environments, making it a valuable tool for environmental monitoring and disaster response [12].

In the development of a low-cost, portable fluorometer for environmental monitoring, Di Nonno and Ulber designed the "Portuino," an Arduino-based device capable of performing both photometric and fluorometric measurements. The device uses LEDs as a light source and a phototransistor as a light sensor, with a circuit that discharges a capacitor in response to photocurrent, allowing it to measure light intensity from fluorescent samples. Validation of the Portuino against laboratory equipment demonstrated its effectiveness in measuring various analytes, such as phosphate ions and fluorescent dyes, with similar correlation coefficients and limits of detection. The simplicity and affordability of the Portuino make it an excellent candidate for field applications, including oil spill detection, where real-time analysis and portability are crucial. This technology's adaptability, as evidenced by its use in various light ranges and with interchangeable LEDs, aligns well with the requirements for detecting and monitoring oil spills in marine environments [10].

A study conducted by Devalla et al. also developed a system that has the capability to detect oil traces within an aquatic environment. This system takes advantage of the fluorometric index in order to improve the general detection of oil substances. This system aims to indicate potential oil discharges such as fuel, lubricating oil, and crude oil at a distance from the sensor. With this, the study also explored the possibility of measuring oils within seawaters, which was accomplished by defining oil-to-water ratios based on the fluorescence indicator-fluorometric index (Flo/w). Flo/w was designed to achieve emission wavelengths from an excitation-emission spectrum (EEM) of oil-free seawater and water that is polluted with oil. The measurements ranges from a value of 50×10^{-9} to 200×10^{-9}. By using these values, a study was conducted over a five month period within a coastal water environment, demonstrating the dependence of the Flo/w and its ability to detect the presence of oil within seawater [13].

Another study that solidifies the effectiveness of fluoro-sensors is the one conducted by Mazumder et al. Since fluoro-sensors have the ability to effectively identify oil spills within different environments, such as water, ice, and snow, they play a crucial role in detecting oil seepages within oceans. This is possible since these sensors function by emitting ultraviolet lighting, which is absorbed by the oil,

emit a visible light. With this method, fluoru-sensors gain the ability to not only detect oil spills within an aquatic environment but also differentiate between light crude and heavy crude based on their fluorescent responses. This component is essential in detecting oil seepages as it uses a laser operating in the range of 0.3 to 0.355 μm, and if it discovers oil, the value would change within the range of 0.4 to 0.65 μm with a sharp peak at 0.48 μm [14].

An additional study was conducted to evaluate the feasibility of flourometric sensors for real-time oil spill detection in systems that are considered to be flow-through. The sensors were able to detect diesel oil for twenty days straight in conditions suitable for scientific conditions. However, the presence of algae-derived substances, CDOM, and turbidity significantly affected the capability of the sensors to detect the oil spill. Among these factors, substances that contain algae extracts have a substantial effect on oil spill detection as the fluorescence of the sample is enhanced, as seen by the sensors and other solutions. To further experiment with the sensors, the researchers integrated them into two different systems: a moored SmartBuoy and a FerryBox system. Both systems confirmed that the factors mentioned above interfere with the accurate reading of the fluometric sensors. Nevertheless, both of them operated properly for two months, providing real-time fluorescence data [5].

A study was administered concerning the development of a robotic fish and its algorithms when it comes to motion control. The robot is designed using the biomimetic principle, having four links, a propeller made using oscillating foil, a flexible posterior body, and being controlled via radio. By modulating the oscillation frequency of the joint of the fish, its propellation speed can be adjusted. Meanwhile, the orientation of the fish can be tuned by altering the different deflections of the said joint. Two control systems have been utilized to provide strong motion control of the fish amidst hydrodynamics and robotic operations. The PID control algorithm and a hybrid control technique were utilized for the implementation of the online speed control. On the other hand, a fuzzy logic controller was used by the robotic fish for its orientation control system. A point-to-point control algorithm integrated with an overhead vision system is utilized for real-time visual feedback [15].

Sodisetty and Reddy [16] focused on the design, propulsion mechanisms, and control dynamics of a robotic fish. Accordingly, robotic fish are envisioned for several areas of applications, such as those in ocean exploration, military exercises, and aquatic preservation; In all these cases, high-performance AUVs offering efficient propulsion and maneuverability are required. The robotic fish is powered by pectoral fins and a flexible tail connected serially by rotational springs to ensure efficient propulsion. The proposed control methodology combines propulsive waveforms with an inverse kinematics-based approach to generate movement of the fish. All the elements of the fish, including the head, body, and tail, are made from ABS material and linked by waterproof servo motors, which comprise the fish-like system. An Arduino Uno board controls the motion and makes the movements coordinated and smooth flowing. Results indicate that it provides high torque and balance in buoyancy and hydrodynamic forces.

A study by Ejofodomi and Ofuakagba [17] proposes a system named Underwater Robotic Oil Spill Surveillance, which is designed to provide constant monitoring of underwater pipelines for spills. The UROSS system utilizes an RFID tag and reader to identify pipeline sections for surveillance. Navigation is done with a gyrometer and an ultrasound sensor, while spill detection is done with a methane sensor. In the case of a spill, it captures images of the spill site by using a color camera and detects the position with the help of GPS. These data are sent via an Xbee Pro 900HP wireless link to the remotely located PC connected to the closest offshore platform. A RescueME Beacon Locator provides a second transmission of a distress signal for emergency services. The paper focused on the challenges of detecting subsea crude oil spills due to inaccessible locations and existing methods that had limitations, such as pipeline pressure monitoring and human surveillance. From there, the UROSS system patrols beside subsea pipelines and detects spills within minutes after onset while providing precise location data to authorities.

IV. Significance of the Project

The development of this study is aimed at providing relevant research with regards to spill pollution, more specifically with oil spills, by creating a system that has the main priority of detecting and locating oil spills within a given area. With this, the result and development of the study will be beneficial to, firstly, environmentalists**.** As people within the field aim to protect and improve the conditions of the environment, having a system that focuses on detecting oil spills can serve as another tool for the betterment of the country's aquatic life. Next are the fishermen. As these types of people's finances revolve around the sea, it is important to maintain the welfare of the environment. Hence the importance of this project to protect aquatic life within the environment in order to sustain the livelihoods of these people. Lastly, computer engineers and future researchers since this project was created by computer engineering students, this paper can serve as a basis for future research papers and projects with regards to the use of the respective microcontrollers and their application within an aquatic environment.

V. Recommendation

In the development of a robotic fish utilizing the PIC16F877A microcontroller for oil spill detection and location communication, several enhancements are recommended to elevate the device's functionality and sustainability. Integrating LoRaWAN, or cellular communication modules, would significantly enhance the range and reliability of data transmission, making real-time monitoring more effective across vast aquatic environments. These technologies, known for their low-power and long-range capabilities, are ideal for remote monitoring,

ensuring that data on oil spill locations can be efficiently communicated even in challenging conditions. Additionally, replacing the existing motor driver with DC motors and relays for tail motion control is proposed. This modification would allow for more precise and energy-efficient propulsion, improving the fish's ability to navigate effectively through water while reducing overall power consumption, which is crucial for extended field operations.

Furthermore, incorporating solar panels on the robotic fish's surface is recommended to provide a sustainable power source, enabling the fish to recharge its batteries while on the surface and thereby extending its operational life. This feature is particularly advantageous for long-term monitoring missions in remote or difficult-to-access locations, where access to power is limited. Coupling solar recharging capabilities with the use of energy-efficient sensors and components would optimize the overall power consumption, making the robotic fish more autonomous and capable of continuous environmental monitoring. These enhancements not only improve the operational efficiency of the robotic fish but also align with sustainability goals, making it a more reliable and eco-friendly tool for environmental monitoring and oil spill detection.

VI. Conclusion

In conclusion, the group was able to develop a system that detects oil spillages within a given marine environment by utilizing the PIC16F877A microcontroller as its main driver. Because of this microcontroller, it is able to control the movement of the fish with the use of two servo motors, a stepper motor, and a DC motor. The servo motors are responsible for helping the machine create either a left or a right motion, the stepper motor is responsible for either adjusting the fish in an upward or downward orientation, and the DC motor is responsible for moving the machine forward. Moreover, this system delivers and accepts data with the use of the ESP8266 ESP-12E module. This machine would remotely accept input from the five buttons of the user (i.e., Key 1, Key 2, Key 3, Key 4, and Key 5). This system also has the capability of detecting oil spills with the use of a GPS module and a specialized sensor imitating a fluorometer. With these components, the system is able to detect oil spillages and transmit the location to the remote location of the user.

Acknowledgment

The researchers would like to express their utmost gratitude to Dr. Jay Robert Del Rosario for his insightful comments and suggestions on implementing this study. His support is greatly appreciated by the researchers, as it helped them develop this study properly.

References

[1] "Oil spills," National Oceanic and Atmospheric Administration. https://www.noaa.gov/education/resource-collections/ocean-coasts/oil-spills

[2] "WEC285/UW330: Effects of oil spills on marine and coastal wildlife," Ask IFAS - Powered by EDIS. https://edis.ifas.ufl.edu/publication/UW330

[3] I. Gozum, "Was MT Terranova, other troubled ships in Bataan involved in oil smuggling?," RAPPLER, Aug. 05, 2024. [Online]. Available: https://www.rappler.com/philippines/was-mt-terranova-other-troubled-ships-bataan-involved-oil-smuggling/

[4] G. K. Cabico, "Mindoro oil spill damage valued at P41.2B — report," Philstar.com, Feb. 27, 2024. [Online]. Available: https://www.philstar.com/headlines/climate-and-environment/2024/02/26/2336272/mindoro-oil-spill-damage-valued-p412b-report

[5] S. Pärt, H. Kankaanpää, J.-V. Björkqvist, and R. Uiboupin, "Oil spill detection using fluorometric sensors: laboratory validation and implementation to a FerryBox and a moored SmartBuoy," Frontiers in Marine Science, vol. 8, Nov. 2021, doi: 10.3389/fmars.2021.778136.

[6] C. Abou-Khalil et al., "Field fluorometers for assessing oil dispersion at sea," Marine Pollution Bulletin, vol. 192, p. 115143, Jul. 2023, doi: 10.1016/j.marpolbul.2023.115143.

[7] N. Hou, "The Design and Simulation of Biomimetic Fish Robot for Aquatic Creature Study," arXiv preprint, 2021, arXiv:2110.07019.

[8] "PIC16F87XA Data Sheet," Microchip Technology Inc., https://ww1.microchip.com/downloads/en/devicedoc/39582b.pdf

[9] "NodeMCU ESP8266," Components101. https://components101.com/development-boards/nodemcu-esp8266-pinout-features-and-datasheet

[10] S. Di Nonno and R. Ulber, "Portuino—A novel portable Low-Cost Arduino-Based photo- and fluorimeter," Sensors, vol. 22, no. 20, p. 7916, Oct. 2022, doi: 10.3390/s22207916.

[11] LME Editorial Staff, "Interface ublox NEO-6M GPS Module with Arduino," Last Minute Engineers, Jun. 26, 2022. https://lastminuteengineers.com/neo6m-gps-arduino-tutorial/#google_vignette

[12] D. Korkmaz, G. O. Koca, G. Li, C. Bal, M. Ay, and Z. H. Akpolat, "Locomotion control of a biomimetic robotic fish based on closed loop sensory feedback CPG model," Journal of Marine Engineering & Technology, vol. 20, no. 2, pp. 125–137, Jul. 2019, doi: 10.1080/20464177.2019.1638703.

[13] E. Baszanowska and Z. Otremba, "Fluorometric Detection of Oil Traces in a Sea Water Column," Sensors, vol. 22, no. 5, pp. 2039–2039, Mar. 2022, doi: https://doi.org/10.3390/s22052039.

[14] S. Mazumder and K. K. Saha, "Detection of Oil Seepages in Oceans by Remote Sensing", Proceedings of the 6th International Conference & Exposition on Petroleum Geophysics.

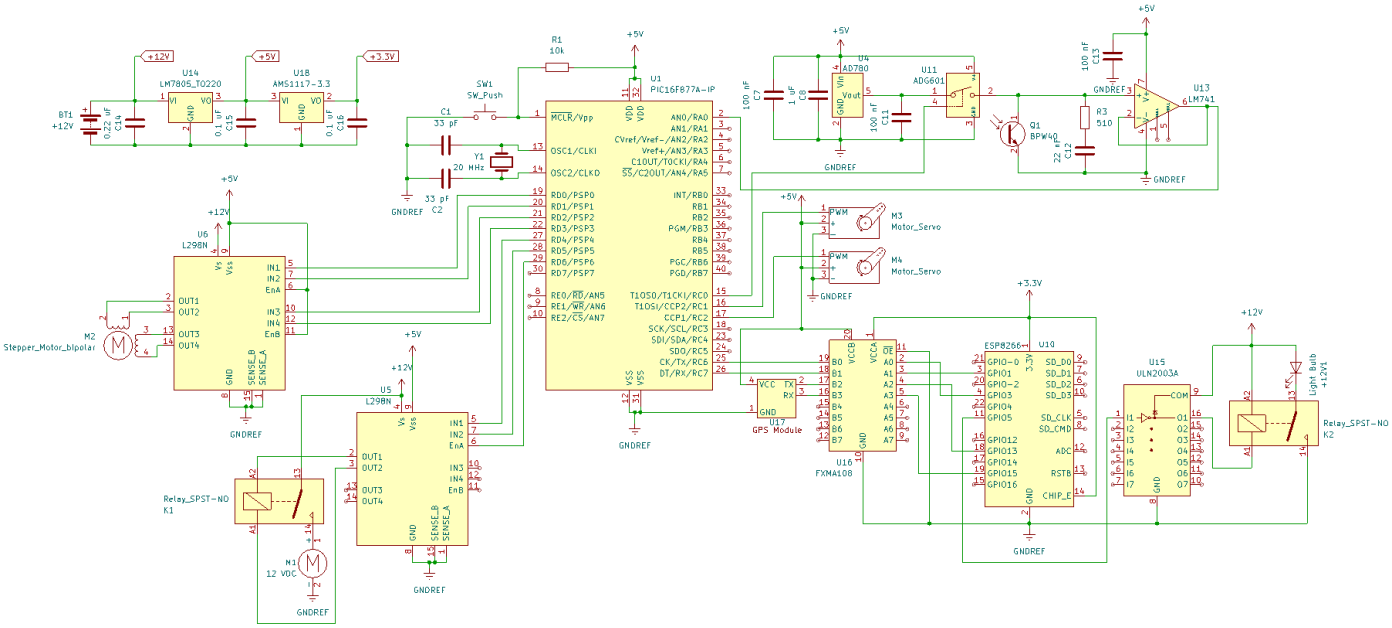
[15] J. Yu, M. Tan, S. Wang, and E. Chen, "Development of a biomimetic robotic fish and its control algorithm," IEEE Transactions on Systems Man and Cybernetics Part B (Cybernetics), vol. 34, no. 4, pp. 1798–1810, Aug. 2004, doi: 10.1109/tsmcb.2004.831151.

[16] V. N. B. P. Sodisetty and A. S. Reddy, "Design and analysis of controlling the robotic fish," IOP Conference Series Materials Science and Engineering, vol. 402, p. 012023, Sep. 2018, doi: 10.1088/1757-899x/402/1/012023.

[17] O. Ejofodomi and G. Ofualagba, "Subsea crude oil spill detection using robotic systems," European Journal of Engineering and Technology Research, vol. 4, no. 12, pp. 112–116, Dec. 2019, doi: 10.24018/ejeng.2019.4.12.1684.

APPENDIX

KICAD CIRCUIT



MIKROC CODE

```
#include <built_in.h>

// Define the input pins
#define KEY1 PORTB.F0
#define KEY2 PORTB.F1
#define KEY3 PORTB.F2
#define KEY4 PORTB.F3
#define KEY5 PORTB.F4

// Define ESP8266 pins
#define ESP_TX PORTC.F6
#define ESP_RX PORTC.F7

void sendCommand(char* command) {
    // Send command to ESP8266
    while (*command) {
        UART1_Write(*command++);
    }
}

void sendHttpRequest(char* command) {
    // Add HTTP request headers
    char buffer[100];
    sprintf(buffer, "AT+CIPSEND=%d\r\n", strlen(command));
    sendCommand(buffer);
    Delay_ms(100);
    sendCommand(command);
    Delay_ms(100);
}

void initESP8266() {
    UART1_Init(9600); // Initialize UART for ESP8266
    Delay_ms(100);    // Wait for ESP8266 to initialize
```

```
    // Configure ESP8266 for your Wi-Fi network
    sendCommand("AT+CWMODE=1\r\n");   // Set mode to Station
    Delay_ms(100);
    sendCommand("AT+CWJAP=\"Your_SSID\",\"Your_PASSWORD\"\r\n"); // Connect to Wi-Fi
    Delay_ms(2000); // Wait for connection

    // Set up the ESP8266 to use HTTP
    sendCommand("AT+CIPSTART=\"TCP\",\"your.server.com\",80\r\n"); // Replace with your server's
address and port
    Delay_ms(2000);
}

void main() {
    // Initialize input ports
    TRISB = 0xFF; // Set PORTB as input
    TRISC = 0x80; // Set RC6 as output (TX) and RC7 as input (RX)

    initESP8266(); // Initialize the ESP8266

    while (1) {
        // Check the input pins and send corresponding commands
        if (KEY1 == 0) {
            Delay_ms(2); // Debounce delay
            if (KEY1 == 0) {
                sendHttpRequest("GET /command?key=1 HTTP/1.1\r\nHost: your.server.com\r\n\r\n");
// Send command '1' for KEY1
                while (KEY1 == 0); // Wait until the key is released
            }
        }

        if (KEY2 == 0) {
            Delay_ms(2); // Debounce delay
            if (KEY2 == 0) {
                sendHttpRequest("GET /command?key=2 HTTP/1.1\r\nHost: your.server.com\r\n\r\n");
// Send command '2' for KEY2
                while (KEY2 == 0); // Wait until the key is released
            }
        }

        if (KEY3 == 0) {
            Delay_ms(2); // Debounce delay
            if (KEY3 == 0) {
                sendHttpRequest("GET /command?key=3 HTTP/1.1\r\nHost: your.server.com\r\n\r\n");
// Send command '3' for KEY3
                while (KEY3 == 0); // Wait until the key is released
            }
        }

        if (KEY4 == 0) {
            Delay_ms(2); // Debounce delay
            if (KEY4 == 0) {
                sendHttpRequest("GET /command?key=4 HTTP/1.1\r\nHost: your.server.com\r\n\r\n");
// Send command '4' for KEY4
                while (KEY4 == 0); // Wait until the key is released
            }
        }

        if (KEY5 == 0) {
            Delay_ms(2); // Debounce delay
            if (KEY5 == 0) {
                sendHttpRequest("GET /command?key=5 HTTP/1.1\r\nHost: your.server.com\r\n\r\n");
// Send command '5' for KEY5
                while (KEY5 == 0); // Wait until the key is released
            }
```

```
        }
    }
}

#include <built_in.h>

// Define constants and pin connections
#define step1 PORTD.F0
#define step2 PORTD.F1
#define step3 PORTD.F2
#define step4 PORTD.F3
#define motor_in1 PORTD.F4
#define motor_in2 PORTD.F5
#define motor_en PORTD.F6

// Global variables
int t = 0;
int a = 0; // Define time variable
int x = 0;
int key_value = 2; // Mode switch, 1, 2, 3 means straight, left, right
int model = 1;
int stepcountor = 0; // Recording the steps of stepper motor
int val1, val2; // Define sin wave, val1 has amplitude of 1, val2 has amplitude of 0.5

// Function prototypes
void flash();
void stepper_reset();
void stepper_up();
void stepper_down();
void drive();
void initMicrocontroller();
void processCommand(char command);

void interrupt() {
    if (TMR0IF_bit) {
        TMR0IF_bit = 0;
        TMR0 = 256 - 250; // Timer0 to overflow every 1 ms

        flash(); // Call the flash function every 1 ms
    }
}

// Fixed-point approximation of sine (scaled by 1000)
int approximate_sin(int x) {
    long x_squared = (long)x * x;
    return (x * 1000 - (x_squared * x) / 6000);
}

// Function to initialize the microcontroller
void initMicrocontroller() {
    TRISC = 0xFF; // Set PORTC as input (for RX from ESP8266)
    TRISD = 0x00; // Set PORTD as output for DC motor and stepper motor
    TRISA = 0x00; // Set PORTA as output

    // Initialize PWM for servo motors
    PWM1_Init(5000); // Initialize PWM with 5kHz frequency
    PWM2_Init(5000); // Initialize PWM with 5kHz frequency
    PWM1_Start();    // Start PWM1
    PWM2_Start();    // Start PWM2

    // Initialize UART module with 9600 baud rate for ESP8266 communication
    UART1_Init(9600);
    Delay_ms(100); // Wait for UART module to stabilize

    // Configure Timer0 for 1 ms interrupts
```

```
    OPTION_REG = 0x84; // Timer0 with prescaler 1:32
    TMR0 = 256 - 250;
    INTCON = 0xA0; // Enable Timer0 and global interrupts
}

// Flash function for generating PWM
void flash() {
    int pi = 3142; // 3.14159265358979323846 * 1000
    t++;
    x++;
    if (x > 800)
        x = 0;
    if (t > 25) {
        t = 0;
        a++;
        if (a > 40)
            a = 0;
        switch (model) {
            case 1:
                if (a < 20) {
                    motor_in1 = 0;
                    motor_in2 = 1;
                    val1 = 25 * approximate_sin(pi * a / 20) / 1000;
                } else {
                    motor_in1 = 1;
                    motor_in2 = 0;
                    val1 = 25 * approximate_sin(pi * (a - 20) / 20) / 1000;
                }
                break;
            case 2:
                if (a < 20) {
                    motor_in1 = 0;
                    motor_in2 = 1;
                    val1 = 25 * approximate_sin(pi * a / 20) / 1000;
                } else {
                    motor_in1 = 1;
                    motor_in2 = 0;
                    val1 = 4 * approximate_sin(pi * (a - 20) / 20) / 1000;
                }
                break;
            case 3:
                if (a < 20) {
                    motor_in1 = 0;
                    motor_in2 = 1;
                    val1 = 4 * approximate_sin(pi * a / 20) / 1000;
                } else {
                    motor_in1 = 1;
                    motor_in2 = 0;
                    val1 = 25 * approximate_sin(pi * (a - 20) / 20) / 1000;
                }
                break;
            default:
                val1 = 25 * approximate_sin(pi * a / 40) / 1000;
        }
    }
    if (t < val1)
        motor_en = 1;
    else
        motor_en = 0;
}

// Functions to control the stepper motor
void stepper_reset() {
    int i;
    for (i = 0; i < 100; i++) {
```

```
        step1 = 1; step2 = 0; step3 = 0; step4 = 1;
        Delay_ms(10);
        step1 = 0; step2 = 1; step3 = 1; step4 = 0;
        Delay_ms(10);
        step1 = 0; step2 = 0; step3 = 1; step4 = 1;
        Delay_ms(10);
        step1 = 1; step2 = 1; step3 = 0; step4 = 0;
        Delay_ms(10);
    }
}

void stepper_up() {
    int i;
    for (i = 0; i < 600; i++) { // 3 revs
        step1 = 1; step2 = 0; step3 = 0; step4 = 1;
        Delay_ms(10);
        step1 = 0; step2 = 1; step3 = 1; step4 = 0;
        Delay_ms(10);
        step1 = 0; step2 = 0; step3 = 1; step4 = 1;
        Delay_ms(10);
        step1 = 1; step2 = 1; step3 = 0; step4 = 0;
        Delay_ms(10);
    }
}

void stepper_down() {
    int i;
    for (i = 0; i < 600; i++) { // 3 revs
        step1 = 1; step2 = 1; step3 = 0; step4 = 0;
        Delay_ms(10);
        step1 = 0; step2 = 0; step3 = 1; step4 = 1;
        Delay_ms(10);
        step1 = 0; step2 = 1; step3 = 1; step4 = 0;
        Delay_ms(10);
        step1 = 1; step2 = 0; step3 = 0; step4 = 1;
        Delay_ms(10);
    }
}

// Drive function to control servo motors based on key_value
void drive() {
    int i;
    switch (key_value) {
        case 1: // Straight forward
            model = 1;
            for(i=0; i<50; i++) {
                PORTA.F1 = 1;
                PORTA.F2 = 1;
                Delay_us(1250); // pulse of 1250us
                PORTA.F2 = 0;
                PORTA.F1 = 0;
                Delay_us(18750);
            }
            break;
        case 2: // Left turn
            model = 2;
            for(i=0; i<50; i++) {
                PORTA.F1 = 1;
                PORTA.F2 = 1;
                Delay_us(1000); // pulse of 1000us
                PORTA.F2 = 0;
                Delay_us(400); // pulse of 1400us
                PORTA.F1 = 0;
                Delay_us(18600);
            }
```

```
            break;
        case 3: // Right turn
            model = 3;
            for(i=0; i<50; i++) {
                PORTA.F2 = 1;
                PORTA.F1 = 1;
                Delay_us(1000); // pulse of 1000us
                PORTA.F2 = 0;
                Delay_us(400); // pulse of 1400us
                PORTA.F1 = 0;
                Delay_us(19000);
            }
            break;
        case 4: // Elevator up
            model = 1;
            for(i=0; i<50; i++) {
                PORTA.F1 = 1;
                PORTA.F2 = 1;
                Delay_us(1000); // pulse of 1000us
                PORTA.F1 = 0;
                PORTA.F2 = 0;
                Delay_us(19000);
            }
            stepper_up();
            break;
        case 5: // Elevator down
            model = 1;
            for(i=0; i<50; i++) {
                PORTA.F1 = 1;
                PORTA.F2 = 1;
                Delay_us(1000); // pulse of 1000us
                PORTA.F1 = 0;
                PORTA.F2 = 0;
                Delay_us(19000);
            }
            stepper_down();
            break;
        default:
            stepper_reset();
    }
}

// Main function
void main() {
    initMicrocontroller(); // Initialize microcontroller and peripherals

    while (1) {
        if (UART1_Data_Ready()) {
            char command = UART1_Read(); // Read command from ESP8266
            processCommand(command); // Process the command
        }
        drive(); // Call the drive function continuously
    }
}

// Function to process commands received from the ESP8266
void processCommand(char command) {
    switch (command) {
        case '1':
            key_value = 1;
            break;
        case '2':
            key_value = 2;
            break;
        case '3':
```

```
            key_value = 3;
            break;
        case '4':
            key_value = 4;
            break;
        case '5':
            key_value = 5;
            break;
        default:
            key_value = 0; // Reset key_value to default if command is not recognized
            break;
    }
}
```