# ON DEMAND INFLATION SYSTEM FOR UNDERWATER SENSOR DEPLOYMENT

PRAVEEN ALOKA SENEVIRATNE

A dissertation submitted in partial fulfillment of the requirements for the Bachelor of Science Special Degree in Engineering Physics of the University of Colombo, Sri Lanka

MAY 2023

# DECLARATION

I declare that this dissertation does not incorporate without acknowledgement, any material previously submitted for a Degree or Diploma in any university and to the best of my knowledge and belief it does not contain any material previously published or written or orally communicated by another person except where due reference is made in the text.

P.A. Seneviratne

# ACKNOWLEDGEMENT

I would like to convey my special thanks and gratitude to my supervisor, Dr. Darshana Lakmal Weerawarne of the Department of Physics, University of Colombo who gave me this valuable opportunity to conduct this research project. I was able to deepen my knowledge in a broad spectrum of fields by carrying out this research under his supervision. Without his guidance and knowledge in every phase of the process, this project would never have been able to succeed. I would like to thank you very much for your assistance and support throughout this.

Also, I would like to appreciate my family, friends, and non-academic staff in the Department of Physics for their valuable time and support in helping me conduct this research project.

# ABSTRACT

Deep water bodies remain as one of the largest unexplored frontiers in the modern world. However, in recent times scientists and researchers have been making great strides in exploring and understanding deep water bodies and the effect they have on planet Earth as well as the living organisms on the planet. In order to research deep water bodies, various man-made devices are used to obtain data from regions which have previously been unreachable by humans alone. While these devices have helped propel the study of deep water bodies to new heights, they are not without their disadvantages. These disadvantages gave rise to the need for a new type of system which could operate in an isolated environment for a prolonged period of time and act autonomously in collecting data from deep sea bodies and then transmitting this information back to the device user once it reaches the surface.

To achieve the above-mentioned criteria, an on-demand inflation system for underwater sensor deployment was created whose movement depended on the variations in the volume of the system. This inflation system was created with compressed air as the fuel source and a nylon balloon as the volume-varying component. The inflation system was controlled using an Arduino Nano module which used readings to understand the depth of the system and control the device in a specified manner to take readings for a period of $(600.00 \pm 0.01)$ seconds before bringing the system back onto the surface. The ESP32 module on board the inflation system collects the data from the Nano module serially and processes it before saving it. This saved data is then transmitted via a WIFI network created from the ESP32 module. An external device could connect to this network and download this data automatically and signal the inflation to restart the data collection process once again. The movement speed of the device is affected primarily by the upthrust force acting on the system and the weight of the inflation system. The first prototype maintains an average velocity of $(0.1429 \pm 0.0005) ms^{-1}$ while the second prototype maintains an average velocity of $(0.1111 \pm 0.0005) ms^{-1}$. The depth of the second prototype was limited by the endurance of the waterproof electric box that was used. As it was IP66 rated, the maximum water pressure it could only endure pressure at a depth of 10.2 meters. The prototype successfully operates as an on-demand inflation system for underwater sensor deployment.

# Contents

# List of Figures

# List of Tables

# ABBREVIATIONS

AUV - Autonomous Underwater Vehicle

CSV - Comma-separated values

WiFi - Wireless Fidelity

IP - Ingress Protection

ROV - Remote Operated Vehicle

# 1 Introduction

Deep water research plays a critical role in expanding our understanding of the oceans and the complex ecosystems that exist within them. As most of the world's oceans are deep and largely unexplored, deep-water research is crucial for advancing our knowledge of the marine environment and the myriad of life forms that call it home. To conduct this research effectively, scientists have increasingly turned to the use of sensors to collect and analyze data. Sensors provide a wealth of information about the ocean, such as temperature, pressure, salinity, and oxygen levels, as well as information about the behaviour of marine animals and the physical and chemical characteristics of the water column. The use of sensors has revolutionized deep water research by allowing for more precise and accurate data collection over larger areas and depths. This information is vital for understanding the dynamics of the ocean and its ecosystems, including the effects of climate change, increasing acidic levels in the ocean, and pollution. Furthermore, sensors are often deployed in remote or inaccessible areas, where direct human observation is not possible, and can provide long-term monitoring of the ocean environment. This is particularly important for detecting and responding to natural disasters such as tsunamis, hurricanes, and human-caused incidents such as oil spills. However, conventional methods have a number of limitations, the first being the difficulty in accessing deep water regions and limited mobility in the currently used methods of obtaining data from deep sea bodies. In addition to this, the high expense and manpower that has to be incurred in travelling back and forth from the sensor to obtain data. This includes time taken to travel, expenses incurred for fuel and food, docking and un-docking expenses, etc. To address these challenges, an on-demand inflation system for underwater sensor deployment has been proposed which can be used to automate the process of data collection from deep water bodies. The proposed systems utilize a compact, inflatable structure to carry and deploy sensors in deep water environments and bring the collected data back to the surface and transmit it such that a device could record it at the surface of the water body. The system is designed to be deployed on demand, allowing for the deployment of sensors in remote and inaccessible regions. Additionally, the system offers a high level of mobility, allowing for relocating sensors as required and obtaining readings.

This research aims to explore the potential benefits and limitations of the on-demand inflation system for underwater sensor deployment and understand how the different techniques used here to collect data would affect the future recording of data in deep water bodies. Through thoroughly examining the system's design and capabilities,

this thesis will provide insight into the potential applications and benefits of the system in real-life situations, and identify areas for improvement if any. The significance of this research lies in the potential for the on-demand inflation system to revolutionize the deployment of underwater sensors. By addressing the limitations of conventional methods and providing a more efficient and effective solution, the system has the potential to greatly enhance our understanding of underwater environments and the processes that occur within them in a safer, more efficient and inexpensive manner. This treatise under the name Background is simply about the details of deep water bodies, currently used methods of data collection and their place in modern information collection from deep water bodies.

# 2 Background

## 2.1 Deep water bodies

Deep water bodies generally consist of lakes and seas spread across the ocean. The Earth's oceans are vast and diverse, covering more than 70 percent of the planet's surface. There are a variety of different sea bodies, ranging from shallow, tropical coral reefs to deep, dark abyssal zones. Each of these regions has its own unique characteristics, marine life, and ecological systems which require a multitude of sensors to properly analyze. Sea bodies are divided into two separate regions known as the pelagic region and the benthic region. In addition to the shallow coral reefs and intertidal zones in the Pelagic region, there are three main regions in the Benthic zone: the Bathyal zone, the Abyssal zone, and the Hadal zone. The Bathyal zone extends from the continental shelf to a depth of about 4,000 meters and is characterized by a steep decline in light penetration. The Abyssal zone begins at a depth of 4,000 meters and extends to the ocean floor, which can be several kilometres deep. This region of the ocean is characterized by complete darkness, near-freezing temperatures, and intense pressure. The Hadal zone is found in the trenches that occur where tectonic plates meet and are the deepest region of the ocean. The marine life in each of these regions is unique and adapted to the particular challenges of their environment. For example, coral reefs are home to a vast array of colourful fish and invertebrates, while the Intertidal zone is characterized by tide pools that support various small marine life. In the Bathyal zone, marine life activities include feeding on plankton and small fish, while in the Abyssal zone, scavenging is more common due to the lack of food sources. In the Hadal zone, organisms have to survive in extreme conditions, including high pressure, low temperatures, and limited food resources.

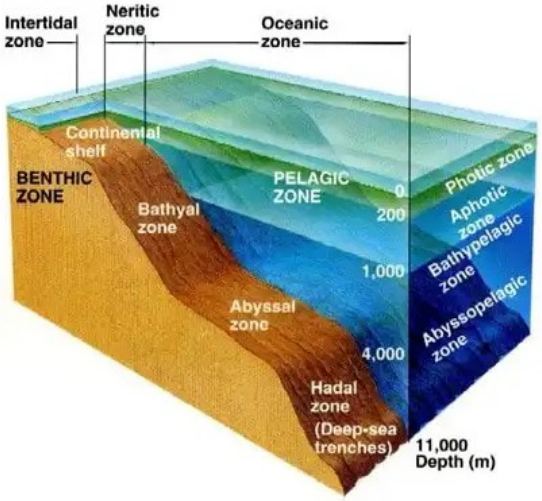


Figure 2.1: Zones the ocean is divided into. This image was taken from [4]

## 2.2 Study of Deep water bodies

The study of deep water bodies has allowed researchers to understand important factors which have a direct impact on the Earth as well as the living organisms on Earth. When considering the Earth's systems, Deep water bodies play a critical role in the Earth's systems, including the study and understanding of the water cycle, climate regulation, and oceanic and atmospheric currents. By studying deep water bodies, we can gain a better understanding of these systems and their interactions, which can help us make more informed decisions about managing our planet. In addition to this, the study of Deep water bodies helps us understand the ecology and biodiversity due to it being the home to a vast array of plant and animal species, many of which are unique to these environments. By studying these ecosystems, we can better understand the biodiversity and interdependence of species in the deep sea, which can inform conservation efforts. Another important outcome of studying Deep water bodies is the understanding of Resource management. Deep water bodies are a rich source of natural resources, including oil, gas, minerals, and fish. By studying these resources and their distribution, we can better manage them sustainably and minimize the impact of human activities on these ecosystems. Geological and geophysical researchers also pay attention to Deep water bodies as they can provide valuable information about the Earth's structure and history, including plate tectonics, seafloor spreading, and climate change. By studying the geology and geophysics of deep water bodies, we can gain insights into the Earth's past and present, and better predict future changes [11]

## 2.3 Methods used in collecting data from underwater sensors

There are numerous methods that have been implemented and used in collecting data from underwater sensors. Even though these methods have been used, they have their own advantages and disadvantages which make them inaccessible or unusable to certain research parties. Examples of such methods can be found below.

### 2.3.1 Autonomous Underwater Vehicles

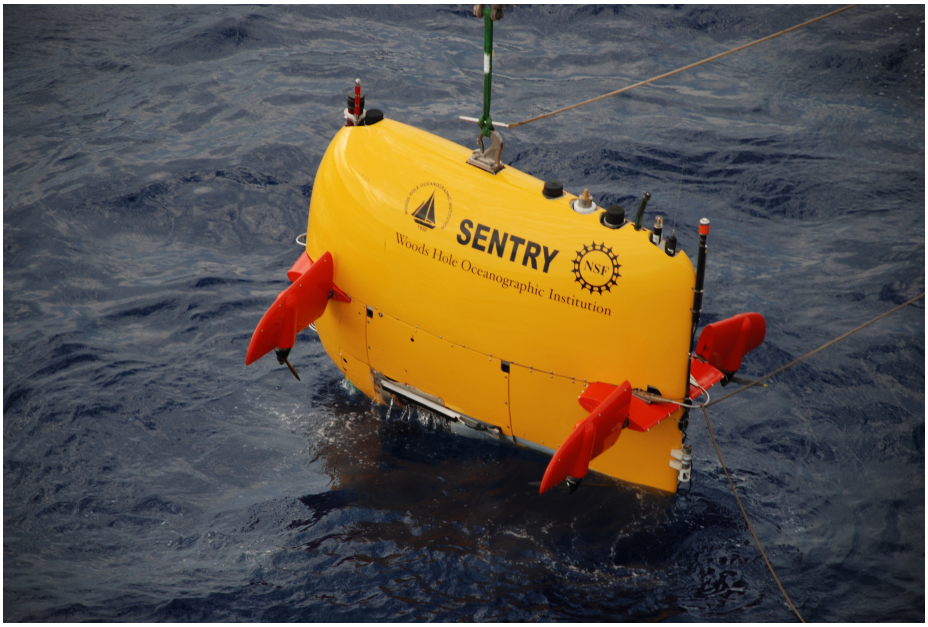


Figure 2.2: Autonomous underwater vehicles being lowered into the sea. This image was taken from [1]

AUVs are unmanned underwater vehicles that can be programmed to carry out specific tasks, such as collecting oceanographic data, mapping the seafloor, or monitoring marine life. AUVs are equipped with a range of sensors that can collect data on water temperature, salinity, dissolved oxygen, and other parameters. They can be used to survey large areas of the ocean and can operate at depths that are difficult or dangerous for human divers to reach. AUVs can be pre-programmed to follow a specific route or can be controlled remotely from a surface vessel. They can also be equipped with cameras, acoustic sensors, and other instruments to collect additional data on marine life or seafloor features. However, the AUVs that are currently in the market are very costly to produce as well as massive in their dimensions. This makes it hard to transport back and forth from the shore to its desired location and needs a large quantity of fuel to operate. [12]

UAV systems can be equipped with an underwater acoustic recorder to assist with monitoring harbour porpoises in Special Areas of Conservation around the globe. The UAV is capable of autonomous navigation, persistent landing, take-off and automatic data acquisition at specified waypoints. The system architecture that enables autonomous UAV flight including waypoint planning and control is described. A bespoke lightweight underwater acoustic recorder capable of transmitting the results of fast Fourier transforms (FFT) applied to incoming signals from a hydrophone was also designed. The system's

operation is successfully validated with outdoor experiments and indoor simulations demonstrating different UAVs capable of autonomously navigating and landing at specific waypoints while recording data in an indoor tank. Results from the recorder suggest that lightweight, relatively low-cost systems can be used instead of heavier more expensive alternatives.

### 2.3.2 Remotely Operated Vehicles

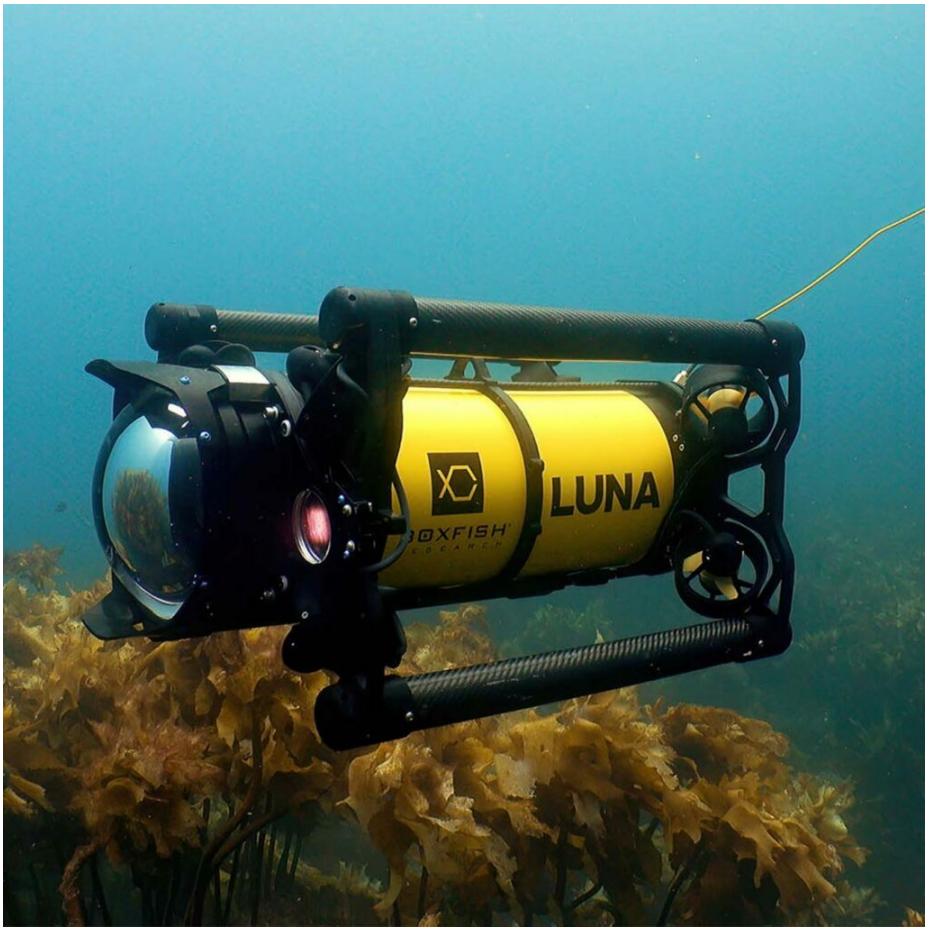


Figure 2.3: ROV collecting data in the sea. This image was taken from [5]

ROVs are unmanned underwater vehicles that are tethered to a surface vessel by a cable. They are controlled by human operators who can direct their movements and collect data in real-time. ROVs are equipped with high-resolution cameras, lights, and other sensors that can provide detailed images and data of underwater environments. They can be used for a variety of applications, including oceanographic research, exploration of deep-sea environments, and the maintenance and repair of underwater infrastructure. Some ROVs are also equipped with sampling tools that can collect water, sediment, or biological samples for further analysis. However, this is too costly to produce and massive in its dimensions. In addition to the transport and fuel costs

that accompany these types of vehicles, the cost of maintaining the vehicle operator needs to be factored in as well.

### 2.3.3 Use of Buoys


Figure 2.4: Buoy collecting data from the sea. This image was taken from [2]

Buoys are anchored or drifting devices that are used to collect data on a range of ocean parameters, including temperature, salinity, currents, and waves. Buoys can be equipped with a variety of sensors, including thermometers, conductivity sensors, current meters, and wave sensors. They can be powered by batteries, solar panels, or other renewable energy sources. Buoys can transmit data to shore-based stations via satellite, radio, or other wireless communication methods. They can be used to monitor conditions in specific locations over long periods of time and can provide valuable data for weather forecasting, oceanographic research, and other applications. Some buoys can also be equipped with sensors that can detect the presence of harmful algal blooms, oil spills, or other environmental hazards. While this method can be cheaper to produce than the other methods the readings being hyper-localized is of

its main disadvantages.

### 2.3.4 Use of fishing vessels

Fishing vessels are used to as vessels of opportunity where data acquisitions systems are attached to fishing vessels and they collect data whenever the fishing vessel is out at sea and brings back this data to be analyzed. While this is a low cost alternative to more expensive methods mentioned above, the collection of data is contingent on the fishing vessels and the places visited by it. [9]

# 3 Theory

## 3.1 Altitude measurement using Pressure Sensors

A pressure sensor is a device that measures the pressure of a fluid, indicating the force the fluid is exerting on surfaces in contact with it. Pressure transducers are used in many control and monitoring applications such as flow, airspeed, level, pump systems or altitude. To calculate pressure, the pressure transducer contains a force collector such as a flexible diaphragm which deforms when pressurized and a transduction element that transforms this deformation into an electrical signal. The shape and transduction methods are optimized to the requirements of the process being measured. These electrical signals are then relayed to controllers where they are processed and recorded. Pressure transducers use strain gauges to measure the force acting on them. The strain gauges undergo deformation, creating a change in voltage produced by it. The pressure measurement is based on the degree of change seen in the voltage. There are also advanced versions of pressure transducers that use capacitance or piezoelectric sensors instead of strain gauges. They are chosen based on the range, work environment and precision required from the pressure sensor.

Absolute pressure measures the pressure relative to a perfect vacuum, using absolute zero as a reference point. An example is a barometric pressure transducer. These also include a sealed gauge, where the signal has been offset to match the gauge pressure at the time of construction.

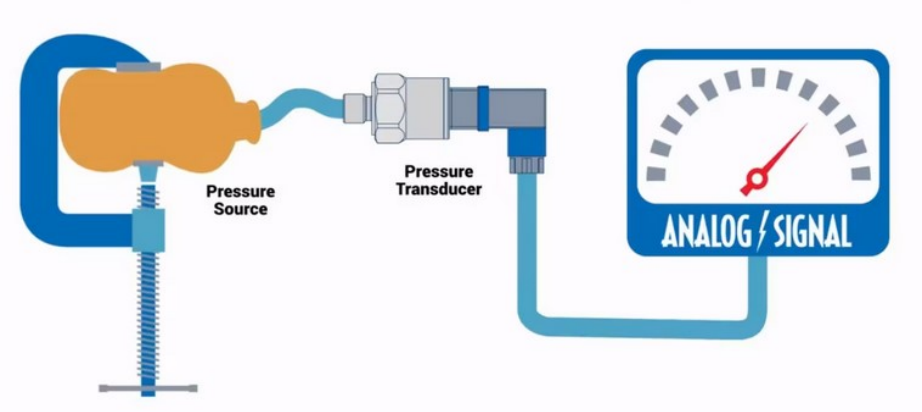


Figure 3.1: Operation of a Pressure sensor.

The pressure differences observed by the pressure sensor can be then converted into altitude readings. This conversion is done based on the pressure at the sea level in a specific area and follows the international barometric equation shown below.

$$A = 44330 \left(1 - \left(\frac{P}{P0}\right)^{\frac{1}{5.255}}\right) \tag{1}$$

Here,
P = Pressure at measuring altitude (hPa)
P0 = Pressure at sea level (hPa)

## 3.2 Solenoid valve

Solenoid valves are electro-mechanically operated valves that convert electric energy into mechanical energy. Their main purpose is to regulate the movement of gas or liquid. Solenoid valves consist of two basic parts: a solenoid (or electromagnet) and a valve. The valve body is made up of two or more openings. Whereas, the solenoid is home to several important parts, including a coil, sleeve assembly and plunger.

Solenoid valves work by employing the electromagnetic coil to either open or close the valve orifice. When the coil within the solenoid is energised, the plunger is lifted or lowered to open or close the orifice. This is what in turn controls flow, regulating the movement of gas or liquid. One of the main advantages of solenoid valves is their versatility. They can be used in an array of industries, for a wide variety of applications and are perfect for a broad range of liquids or gaseous media. They are also an extremely efficient way of controlling flow, as they require very little wiring, expense and effort compared to other valves. A solenoid valve's biggest flaw is its capability to handle dirty or contaminated fluids or gas. Foreign contaminants can collect in the solenoid valve's parts and impede operation. It is also very important that the correct voltage is applied to these valves.

## 3.3 Law of Archimedes and Upthrust

Archimedes' principle states that: When a body is immersed fully or partially in a fluid, it experiences an upthrust force in an upward direction which is equal to the weight of the fluid displaced by the body. Upthrust is defined as an upward force resulting from an object being wholly or partially immersed in a fluid. The intensity of this force is equal to the product of the density of the liquid, the gravitational acceleration and the volume of a part of the submerged body. This principle can be shown using the following equation.

$$U = \rho g V \tag{2}$$

$$U = Mg$$

In the above equation,
U = Upthrust acting on an object,
$\rho$ = Density of the fluid,
g = acceleration due to gravity,
M = Mass of the displaced fluid
V = Volume of an object on which the Upthrust is acting upon.

Therefore, a higher upthrust can be obtained by increasing the volume of the body and displacing more fluid.

## 3.4 Wi-Fi

Wi-Fi networks operate using radio waves with frequencies in the range of 2.4 – 5 GHz, landing somewhere between cell phone and microwave signals on the electromagnetic spectrum Wi-Fi's greatest benefit over more traditional forms of telecommunication connection is the fact that it doesn't require devices to be connected using wires - hence the name. The technology relies, as previously mentioned, on the transmission and receipt of radio waves, electromagnetic waves with frequencies in the Gigahertz range. Data is converted by a router or a wireless adaptor to either send or receive Wi-Fi radio waves. When this encoded radio signal is received by either the router or the adaptor, it is decoded back into the original data. This is a two-way process with the wireless adaptor and router working in tandem to encode and decode the radio Wi-Fi signals in the blink of an eye to transfer data between devices. [6]

## 3.5 Serial Communication

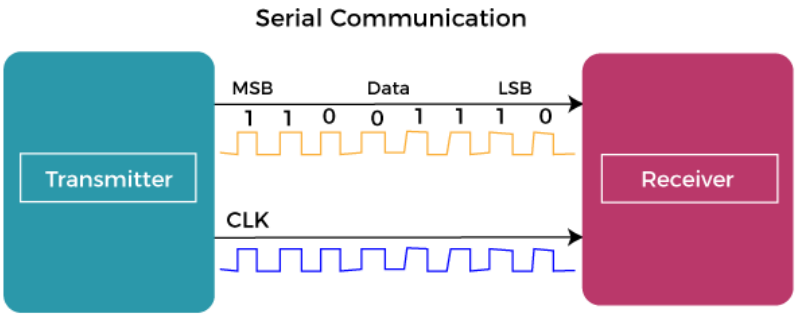


Figure 3.2: Serial Communication

Serial communication is a method of communication between devices that uses transmission lines to send and receive data between the two devices continuously one bit at a time. There are 2 main types of serial communication that are used in data communication. They are asynchronous serial communication and synchronous serial communication. Asynchronous serial communication occurs when data is transferred without support from an external clock signal. This transmission method is perfect for minimizing the required connections for communications, but more effort is necessary to reliably transfer and receive data. Synchronous serial communication occurs when the data is transferred in sync with an external clock signal. When the devices are using synchronous serial communication, all the communication that occurs happens with respect to the common external clock. [14]

## 3.6 Measuring of temperature using sensors

Temperature sensors measure temperature readings via electrical signals. They contain two metals that generate an electrical voltage or resistance when a temperature change occurs. These temperature sensors work by measuring the voltage across the diode terminals. When the voltage increases, the temperature also increases, which is then followed by a voltage drop between the transistor terminals and the emitter. There are 2 main types of temperature sensors which are namely contact temperature sensors and non-contact temperature sensors. Contact temperature sensors measure the degree of hotness or coldness of an object or substance via direct contact. They are generally used to detect a wide range of temperatures in different solids, liquids or gases. Non-contact thermometers are never in direct contact with an object or substance, therefore, they are widely used in hazardous environments such as power plant industries. They measure how hot or cold something is via radiation emitted by a heat source.

## 3.7 Velocity and Average velocity

Velocity is defined as the displacement of an object in a unit of time in a pre-specified direction with respect to a particular frame of reference. With respect to this frame of reference, the velocity throughout the journey may vary from one position to another. However, average velocity taken with respect to a particular frame of reference discards the variations of velocity that occur throughout the journey and look at the journey as a whole and calculate the velocity required for the object to complete the journey in the stipulated time. Velocity and average velocity can be calculated using the following equation.

$$s = ut + \frac{1}{2}at^2 \tag{3}$$

$$v = u + at \tag{4}$$

$$v^2 = u^2 + 2as \tag{5}$$

In the above equation,
v = Velocity of the object,
s = Distance travelled by the object,
t = Time taken by the object to travel this distance,
a = acceleration of the object,

## 3.8 Newton's second law of motion

Newton's second law of motion states that the time rate of change of the momentum of a body is equal in both magnitude and direction to the force imposed on it. The momentum of a body is equal to the product of its mass and its velocity. Momentum, like velocity, is a vector quantity, having both magnitude and direction. A force applied to a body can change the magnitude of the momentum or its direction or both. If a body has a net force acting on it, it is accelerated in accordance with the equation. Conversely, if a body is not accelerated, there is no net force acting on it. The second law of motion can be specified using the following equation.

$$F = ma \tag{6}$$

In the above equation,
F = Force acting on the object,
m = Mass on the object,
a = Acceleration of the object

# 4 Methodology

## 4.1 Introduction and general design of Inflation system design and development

With starting the development of the Underwater inflation system, the Design phase must be required for a better design which can fulfil all the research purposes. For that design, the phase was executed, and first, the design was created, and prototypes were made to check the suitability for the purpose and understand any changes to the design process that need to be made. The limitations and restrictions are described in section 4.2.3, and simply the essential limitations and restrictions can be listed as follows,

- Power efficiency
- Ability of equipment to withstand conditions underwater
- Cost

The resources used in the designing phase of the Inflation system are the simulating and designing tools used to obtain the design and results. From the output, it refers to the final design of the inflation system. This whole section is based on the methodology and this describes the processes which were carried out for the development of a better and more suitable inflation system design.

## 4.2 Design Process

The design process can be divided into 4 main areas. These areas encompass the purview of designing and developing the Inflation system

- Selecting suitable inflation system
- Selecting components that help the system's operation
- Comparing proposed prototypes
- Choosing the most viable prototype design

### 4.2.1 Selecting suitable inflation system

In order to build this inflation system, different methods were considered as possible solutions for the movement of the system in a body of water. This included methods such as the use of a tube which is connected to the surface of the water and the

system being at the lower end of it where the system travels up and down along the tube passing information, the use of a counterweight sitting at the top of a body of water which would move at a certain period of time whose momentum propels the system to the surface.

However, due to factors such as cost and portability of the device, the main method that was selected was the Creation of an inflation system which makes use of an expandable and contracting volume as it is the most cost-efficient, portable and practical way of achieving the required objectives.

### 4.2.2 Selecting of components that help the system's operations

When creating an inflation system which makes use of an expandable and contracting volume as its only source of movement, special attention had to be paid to components which facilitated the expanding and the contracting volume. As the change in volume had to be done on demand, a three-way solenoid valve was used as it could be controlled by giving an electrical signal to it via a transistor.

Figure 4.1: Solenoid switch used in experiment

The transistor used here was 2N2222 as it is capable of handling a current of up to 1A which is necessary for a proper current flow to the solenoid. A high-strength nylon balloon was used as the expanding and contracting volume in the device. To maintain the portability and cost-effectiveness of the system, a small yet affordable source of fluid was required to expand the balloon. A canister of Air Duster was used as a source as it fulfilled the aforementioned requirements.

An Arduino Nano was chosen to control the system using various peripherals due to its powerful core and compactness. An ESP-32 development board and a micro SD card were used to read, analyze and record data which could then be transmitted using WIFI. Since these electronic components need to be submerged in water as well they need to be guarded against water damage. This was accomplished by the use of 2 electric waterproof boxes. The barometric pressure and temperature sensor BMP180 was used as it is capable of withstanding high pressure and also provides accurate readings with an error of $\pm 0.25$ m. This pressure makes it possible for the pressure sensor to go down to a necessary depth and maintain itself at this depth while the sensor captures its readings. A micro SD card module was used to capture the readings obtained from the sensor.

### 4.2.3 Comparing prototypes designs for the proposed system

There are 2 main types of designs that were considered to be used in the on-demand inflation system. The 2 designs that were considered for this were the on-demand inflation system which operates by changing the volume of the system or creating an on-demand inflation system which operates by changing its density. The former method would work by inflating and deflating a balloon which would expand and contract on demand in order to move the system along a vertical axis while the latter would operate by taking in water into the cavity in the system to descend and pushing the water out of the cavity in order to ascend by changing the density of the system with respect to water, similar to a submarine.

### 4.2.4 Choosing the most viable prototype design

Out of the 2 proposed designs for the on-demand inflation system prototype mentioned above the system which operates by changing the volume of the system was chosen over the system which operates by changing its density. The reasons for choosing this are due to the ability of this design to overcome the limitations and restrictions that were listed above. The frequency with which the system is able to travel to record and transmit data is greater in the former design as it relies on an external fuel source to power the movement in contrast to the latter design which needs linear motors or actuators to change the density of the system. These motors require large amounts of power as well as time to complete one rotation of data collection and dissemination making it have a lower frequency. While considering the second limitation, it is more evenly matched. This is due to the former method being fully dependent on the inflating balloon for its movement. This singular point of failure brings into question the ability of this design to withstand conditions underwater while the latter

design also has a singular point of failure where its operation depends on the smooth operation of the linear actuators and the ability of the water to flow into the cavity without blockage. The final limitation that was considered is the cost of developing these 2 designs. As most of the components used in these designs were the same, only the different components used in the movement of these 2 systems along the vertical axis were considered. Since linear actuators are considerably more expensive than a solenoid valve with tubing and a balloon the former design was the clear choice. As the former design was better positioned to tackle the limitations that were identified earlier, it was chosen as the most viable prototype design to create an on-demand inflation system for underwater sensor deployment.

## 4.3 Building of the chosen prototype for the inflation system

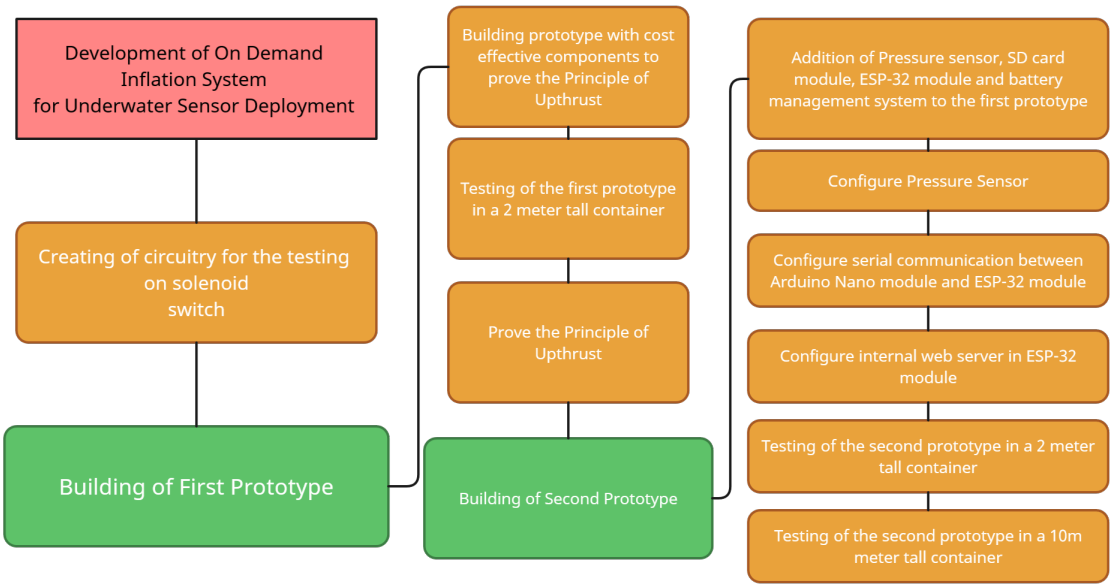


Figure 4.2: Flowchart of the build process of the inflation system

In the proposed method, the inflation system makes use of an expandable and contracting volume. There were 2 prototypes created for this method. The first prototype was created on a dot board. The dot board was made to ensure the compatibility of the components with each other and make the necessary changes to the components if required. It consisted of a waterproof electric box, solenoid valve, transistor, battery pack, Air Duster, Nylon Balloon and an Arduino Nano. Initially, the solenoid switch was individually tested by directly connecting the air duster and balloon to the solenoid switch using pneumatic tubes and powering the solenoid switch by connecting

it to an Arduino Uno module through a transistor, power diode and resistor. A code was then compiled to turn the solenoid switch on and off every 5 seconds and the air duster was turned on to check if the balloon would inflate and deflate as intended with the operation of the solenoid switch. [15]

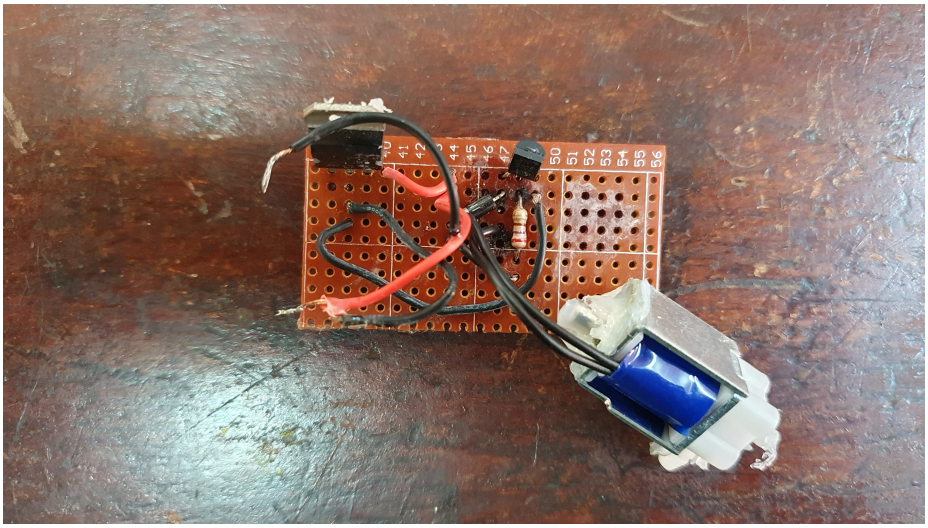


Figure 4.3: Solenoid switch circuit connected to Arduino UNO

Once the operation of the balloon was as expected, the circuit was transferred into a waterproof electric box and the holes were drilled on 3 faces of the box to allow the tubing necessary for the operation of the balloon to connect to the solenoid valve. The holes were then plastered with silicon gum to ensure that no leaks are left for the water to leak into the circuitry. The waterproof electric box used for this prototype was created using plastic due to its lightweight and low cost. To power this system 2 3.7V cells were used to power the prototype. A battery management system was not used as the primary goal of this prototype was to act as a proof of concept for the principle of upthrust and was created in the most cost-efficient manner. The device was turned on and off by placing and removing the batteries in the battery holder respectively. This prototype was created to understand the working principle for the locomotion of the inflation system and the constraints surrounding it. [8]

Figure 4.4: First prototype created as on-demand inflation system for underwater sensor deployment

Due to the buoyant nature of the components used in the creation of this prototype, it did not naturally sink when placed in a body of water. To overcome this issue, a nylon mesh was created which was then placed on around the prototype so that an external weight could be attached to the system so that the system would sink when placed in a body of water. This system of weights comprised of 5 equally weighted 100g weights which could be stacked individually until the weight which is just sufficient to submerge the system is reached. For this prototype, a weight of 300g was required which was accomplished by stacking all three 100g weights to reach the necessary weight.

Figure 4.5: Weight system used in the prototypes of the on-demand inflation system for underwater sensor deployment

This prototype was mainly created to prove that the law of buoyancy is held true and is capable of moving the system along a vertical axis on command. However, this system worked on a timer basis and was unable to suspend itself at a fixed depth and would rise back to the surface after a specified amount of time. This makes it unsuitable to be used in capturing data using a sensor. In addition to this, this prototype was controlled using only an Arduino Nano which only contains one core and is incapable of controlling the prototype, collecting data, processing the data and saving it on its onboard memory and transmitting the data to an external system simultaneously. This prototype lacked safeguards to protect the electronic components present in the system against potential leaks that may occur in high-pressure environments such as deep water bodies. Furthermore, due to the single battery pack that was used, the device was not able to operate independently for a prolonged period of time. Finally, due to the sensitive nature of the solenoid switches used in the creation of the system, they tend to malfunction quite often. This prototype was unable to provide feedback on the proper operation of the solenoid switch and made it difficult to troubleshoot the issue when the prototype doesn't function as expected.

To combat the shortcomings of the earlier prototype, another prototype was created in a printed circuit board (PCB). This prototype consisted of 2 waterproof electric boxes, a solenoid valve, a transistor, 2 battery packs connected in parallel, Air Duster, Nylon Balloon, an ESP-32 development board, a micro SD card module, an SD card,

BMP180 pressure and temperature sensor, 2 light emitting diodes and an Arduino Nano. The PCB design for this prototype was created using the EasyEDA website where all the components were loaded onto this website and then the connections were created between the components. [13]

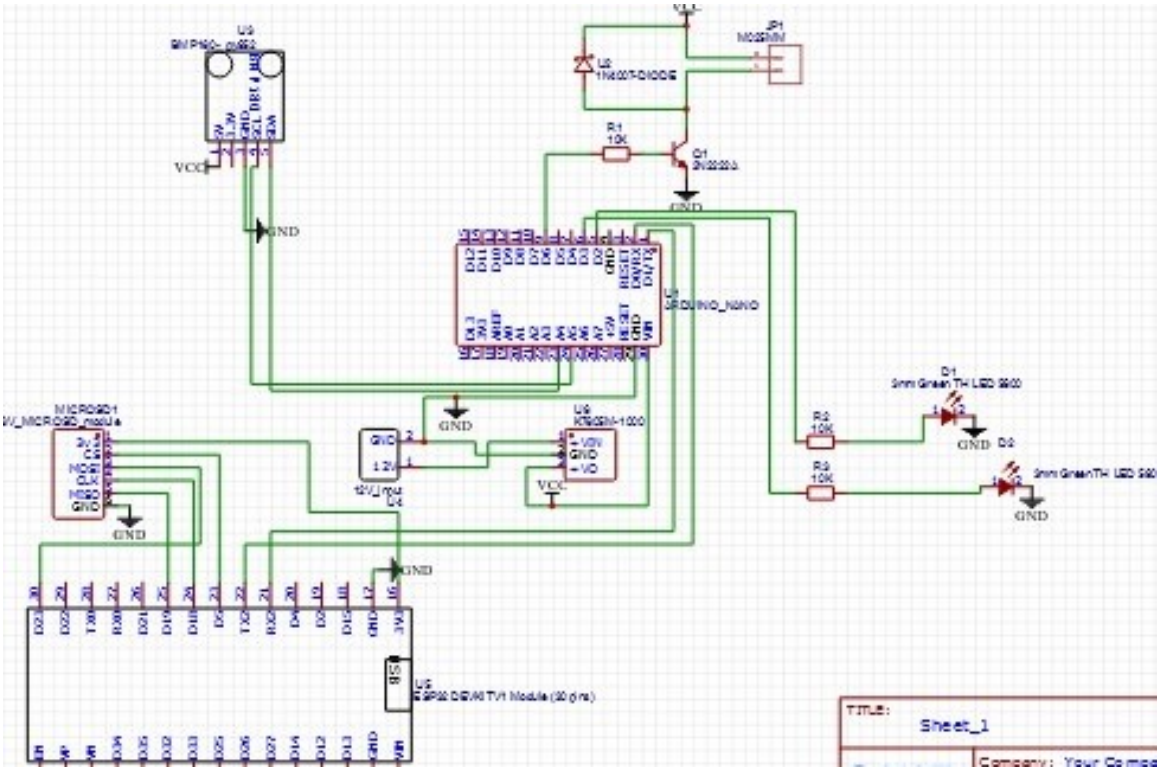


Figure 4.6: Connections between components in the second prototype

Once the components were correctly connected to each other the PCB design was loaded and the components were placed so that the routes could be properly etched. The initial design of this prototype consisted only of an Arduino Nano module. However, during the testing phase, it was required that the single core present in the Arduino Nano module was not powerful enough to control the system, save and process data and transmit it using an ESP01 module. To rectify this issue, another PCB design was created with the addition of an ESP-32 module which has a built-in WIFI transmitter and receiver and has 2 cores making it possible for it to save, process and transmit data with ease. Even though the website had a 100 percent success rate in etching the previous PCB module with only the Arduino module, the new design had only 96 percent success in etching the routes between the components after numerous attempts.

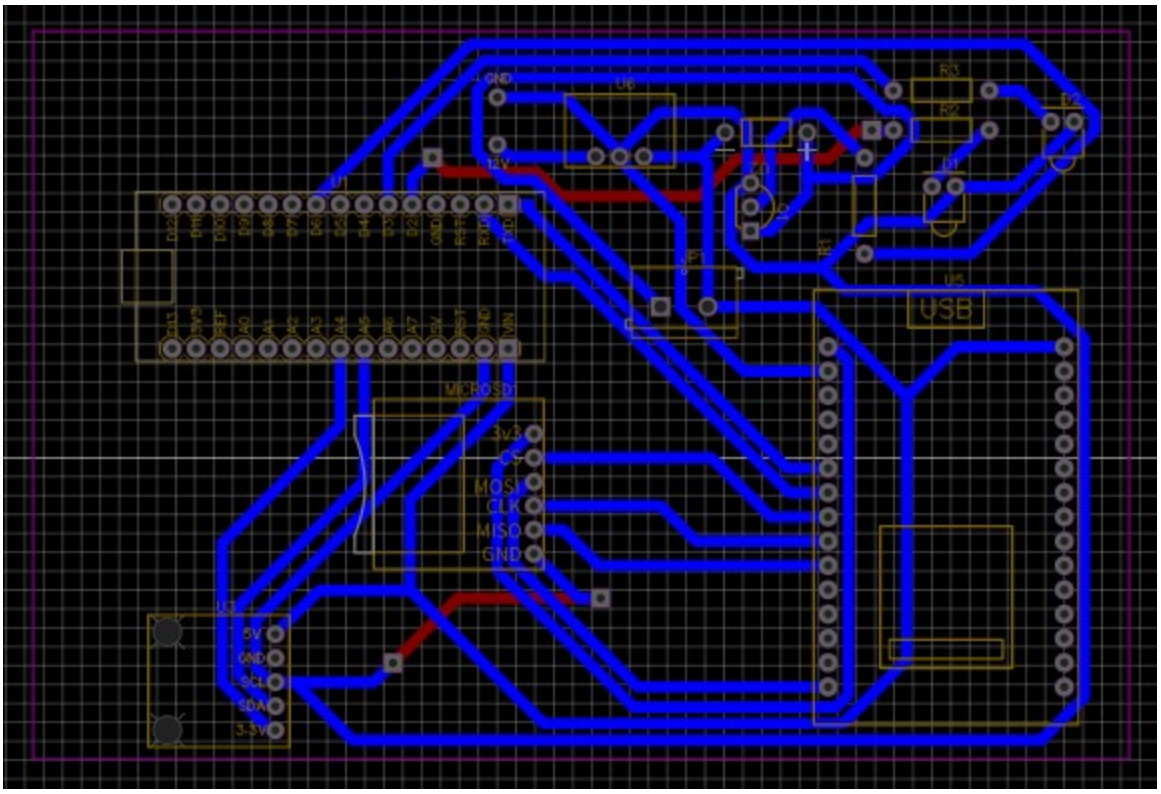


Figure 4.7: PCB design of the second prototype

The routes which could not be etched via the website were to be connected using two external wires. Once the design of the PCB was completed, the design was printed onto a thermal transfer paper. A copper-coated plate was then cut approximately to the size of the PCB and was cleaned using Brazzo to get rid of any impurities that lie on the copper plate. Once the plate was sufficiently cleaned, the thermal paper with the PCB design on it was kept on top of it and ironed until the design was transferred onto the copper plate. Once the drawing has completely transferred onto the copper plate, it was submerged in a solution of Ferric(III) chloride and stirred until the unwanted copper came off the plate. This process took 15 minutes. Once the unwanted copper had completely come off the plate, the plate was taken off the solution and washed and then cleaned using paint thinner to get rid of the print overlay that was covering the copper tracks. After this step, the broken copper tracks were reconnected by using solder and tested individually using the diode mode in a multi-meter to ensure that all the tracks are not broken. Once the tracks were seen to be properly connected, the copper board was drilled along the holes present in the design and the components were soldered onto the PCB. Once the components were soldered, the diode mode was used once again to ensure that there are no short connections between the components and the PCB. To power this prototype, four batteries were used with two sets of two batteries connected in series to create an

output voltage of 8.4V with a much longer operational time which is only possible with just two batteries connected in series. This is essential as this prototype would be isolated from the external environment and needs to operate without the need to be charged regularly. To ensure the protection of the system from an excess surcharge that may damage the system a 2-cell battery management system was connected between the battery pack and the system so that cells discharge as intended and can be charged in a safe manner.

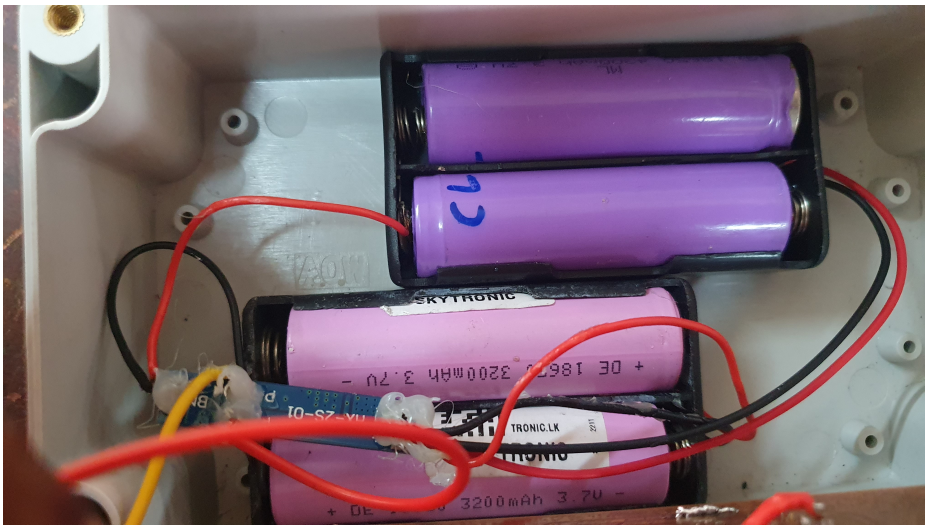


Figure 4.8: Battery pack and battery management system used in the second prototype for the on-demand inflation system for underwater sensor deployment

The output of the battery management system was connected to a switch which could be used to turn the system on and off when required to save battery power. This prototype makes use of both a timer and a pressure input in order to maintain the depth of the system. The pressure sensor sends its data to the Arduino Nano which then turns on or off the solenoid switch and controls the depth to which the system travels.

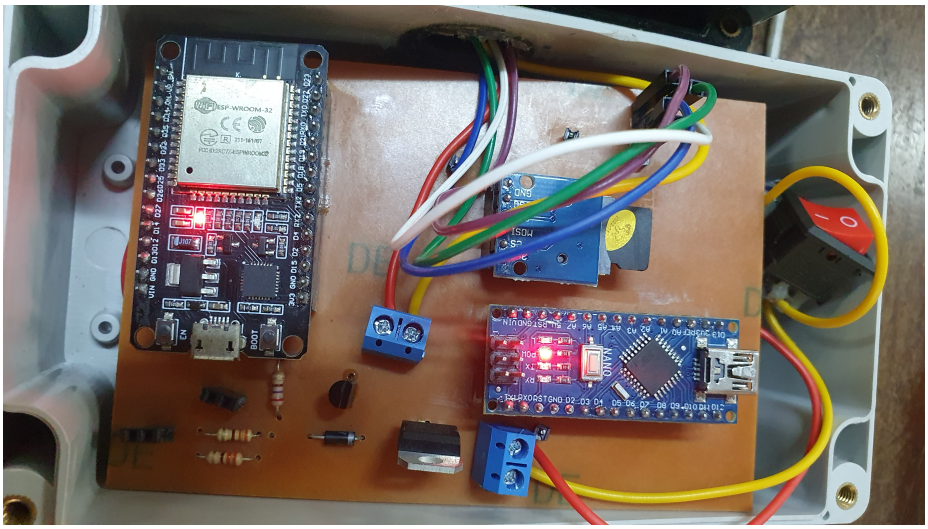

Figure 4.9: Circuitry of the second prototype created for the on-demand inflation system for underwater sensor deployment

Once it reaches this level the depth is maintained by continuously getting readings from the pressure sensor and changing the volume of the balloon accordingly. The internal timer of the Arduino Nano complements the pressure sensor readings allowing the Arduino Nano to decide on the movement of the system. The timer is used to understand when the system must submerge from the surface and how long the system must be kept underwater in order for the sensor to collect readings before coming back to the surface to transmit the collected data. The collected data is to be serially transmitted from the Arduino Nano to the ESP-32 module which in turn processes it and saves it as a CSV file on a micro SD card to be retrieved later.

In order for the system to operate as expected, the necessary Arduino code had to be developed. Using the Arduino code of the first prototype as the basis, the pressure sensor library, SD card library and ESP-32 development library were added to the Arduino IDE. Since the Arduino Nano module and the ESP-32 module were used to operate different sections of the system, two separate codes need to be built for the two modules. The Arduino code developed for the previous prototype was modified with the addition of the pressure sensor to it. This allowed this prototype to operate the solenoid switch making use of both a timer and a pressure input in order to maintain the depth of the system. The pressure sensor sends its data to the Arduino Nano which then turns on or off the solenoid switch and controls the depth to which the system travels. Once it reaches this level the depth is maintained by continuously getting readings from the pressure sensor and changing the volume of the balloon accordingly. The internal timer of the Arduino Nano complements the pressure sensor readings allowing the Arduino Nano to decide on the movement of

the system. The timer is used to understand when the system must submerge from the surface and how long the system must be kept underwater in order for the sensor to collect readings before coming back to the surface to transmit the collected data. The collected data is to be serially transmitted from the Arduino Nano to the ESP-32 module. The ESP-32 code was developed to receive the data from the Arduino Nano module serially and save it as a CSV file on a micro SD card to be retrieved later. To allow this the ESP-32 code operated with the SD card library. The ESP-32 code is made to separate the serial data that is coming from the Arduino Nano module and save it in the SD card module. Since the data coming serially into the module is separated using the comma, the data was separated and saved into an Excel sheet as a CSV file. In addition to this, an internal web server was created as a soft access point and a WIFI connection was created. Then, a web page was designed using HTML and CSS and was uploaded to the web server via the Arduino code. Now, when an external device connects to the internal ESP-32 server via the WIFI server they will be able to access the web page by accessing the IP address relevant to the web page. The created web page contains the option for the user to download the data from the sensor, upload new codes into the ESP-32 module via the web page and reset the entire device.

The process of collecting data in deep sea bodies exposes the device to uncontrollable environmental conditions such as pressure changes, contact with marine life and man-made structures or discards scattered across the ocean. In order to withstand these conditions the system needs to be made durable. Taking this factor into consideration, PVC waterproof electric boxes were used due to their light and durable nature and a mixture of silicon gum and hot glue was used to perfectly seal the boxes to ensure that no water leaks into the electronics present within the system. These safeguards ensure the safety of the system and the accuracy of the readings as there are not affected by external factors from the depths of the water bodies up until the surface of the deep water body.

Once the system is at the surface, the sensor data needs to be transmitted to a surface system. This is done by using the web server within the ESP-32 module. This web server in the soft access point mode (AP mode) allows up to 5 other devices and systems to connect to it while it is at the surface. The ESP-32 module creates a web page which can be accessed by anyone connected to its server and knows the IP address of the page to download the CSV file that has been updated up to the point of the download. The surface system that needs to connect to the ESP-32 web server via the created WIFI network needs to be programmed in such a manner that

it can automatically connect to the network once it is in range and then download the CSV file. Once the file download has been completed, a flag needs to be sent to the inflation system via the surface system which would then be used to repeat the process of submerging and collecting data. Once the process restarts, the previous data set will be removed in order to save memory for the new data that is to be recorded.

For the collection of this data and to transmit it onto a surface device the inflation system needs to move to the surface of the water body and then back to depths of the ocean repeatedly. In order to accomplish this task, compressed air is used as the fuel source obtained via air duster canisters. These air canisters were connected to the solenoid switch using transparent pneumatic tubing with silicon gum and super glue. However, in the first prototype, the pneumatic tube was connected to the air canister using a 3D-printed connector with a superglue coating. This method proved insufficient as water seeped into the tube through the connector and entered the inflating balloon and also the air from the canister was released manually. Therefore, in the second prototype that was created, this mechanism was changed into a mechanical pressing mechanism which releases air continuously once it is in the correct orientation and the pneumatic tube was connected to the air duster using an elbowed pneumatic connector. The large size of the air duster provided additional issues in the movement of the inflation system. This was due to the upthrust that was acting on the canister making it buoyant and in turn slowing the descent of the inflation system. As a workaround to this problem, the air canister was replaced by an inhaler filled with compressed air. Its small size provides less upthrust making it easier for the inflation system to operate as intended. Furthermore, 4 holes were drilled in one waterproof electric box and one hole in the other in order to connect the air duster, the inflating balloon and the air outlet with the solenoid valve and the final hole were to pass the wiring from one waterproof electric on to the other respectively. These holes were made using a drill press having drill bits of 5mm and 8mm. Once the pneumatic tubes passed through the holes, they were made waterproof by applying silicon gum on either side of each hole and then the two waterproof electric boxes were fastened together using an epoxy resin.

The pressure used to calibrate the movement of the inflation system is attached to the top of one waterproof electric box which already contains the pneumatic tubes. The internal calculations of the pressure sensor are done using the universal constant for the atmospheric pressure at sea level which is 1013.25 hPA. However, in Sri Lanka, the atmospheric pressure ranges between 1008 hPa to 1011.5 hPA. Therefore, whenever

the prototype is to be tested or deployed the most current atmospheric pressure at sea level needs to be obtained in order to calibrate the pressure sensor accurately. If the body of water that is used for testing is not the ocean, the height of the water body from the ocean needs to be taken into consideration as well when calibrating the pressure sensor so that system remains stagnant at the required depth and does not malfunction. This pressure sensor was attached to the electric box by drilling a hole in it for a male-to-female header and covering the hole using silicon gum and super glue. The pressure sensor was connected to the male headers which are facing towards the outside of the box and was covered with a coating of silicon gum to ensure that no water comes in contact with the sensor. The pressure sensor was placed outside the waterproof electric boxes as the isolated nature inside the box causes the pressure variations inside the box to be disproportional to pressure variations occurring outside of the box which leads to the pressure readings being disproportional with the depth of the system. The coating of super glue and silicon gum affects the accuracy of the readings taken by the pressure sensor. In order to rectify this shortcoming, the readings taken by the sensor were multiplied by a constant of value 1.3402 so that the output matches the original value. This constant was obtained by taking a set of 100 data readings before the application of glue on the pressure sensor and obtaining another set of 100 data readings after the application of glue which were then compared by getting the average of each data set and obtaining the multiplication constant from it.

The wiring of the sensor was connected to the female end of the header inside the box and taken through the connecting opening to the other electric box which is connected to the Arduino Nano. The opening between the electric boxes was covered with a coating of silicon gum once all the wiring was correctly done to ensure no water leaked into the electronics.


Figure 4.10: Completed second prototype of the on-demand inflation system for underwater sensor deployment

This prototype was also buoyant due to the components used in it. To overcome this limitation, a nylon mesh was once again used to cover the prototype so that a weight could be attached to it. The setup comprising equal weights was used for this prototype as well. In order to get the prototype to submerge a weight of 200g was required. Therefore, two 100g plates were stacked on top of each other to achieve the required weights.

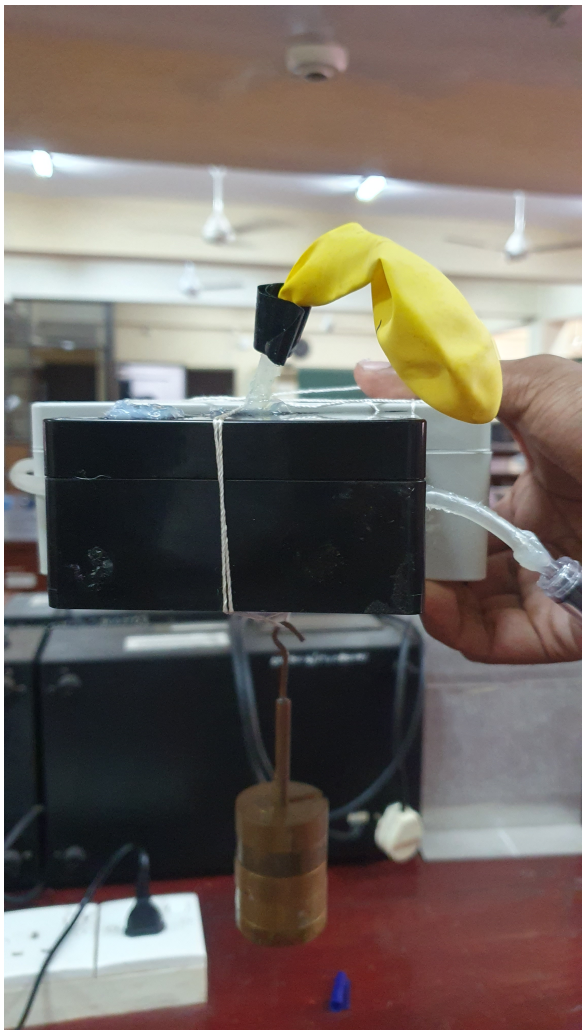

Figure 4.11: Attachment of the weights onto the second prototype of the on-demand inflation system for underwater sensor deployment

## 4.4 Operation of the created prototypes

The first prototype created was made in a simplistic manner as mentioned above. Since this prototype contains only one waterproof electric box which is loaded with operational electronics and has an air duster connected to it in order to power the movement of the system along the vertical axis. Once the first prototype was fully assembled, with air-tight connections among the tubing, a weight of 300g was added to the system to offset the buoyant nature of the system. This buoyant nature was mainly due to the lightweight and high volume of the air duster. The weight was attached to the system by a mesh of nylon strings covering the system. Once the weight was securely fastened, the entire system was taken initially to a body of fresh water with a depth of 2 m and the device was kept on the water body after turning it on. Due to the attached weight, the system started to sink as soon as it came

in contact with the body of water. The system lay stagnant at the bottom of the container for a time period of 10 minutes. Then, the solenoid valve turned on and started inflating the balloon for a period of 20 seconds. The change in the volume of the balloon created enough upthrust to raise the system from the bottom of the container and bring it close to the surface. The entire system doesn't reach the surface of the body of water entirely because when a part of the balloon moves into the atmosphere, the volume on which the upthrust acts on the system decreases until the system reaches an equilibrium. The system stays in this equilibrium for a period of 2 minutes. Then the solenoid switch turns off and the air present in the balloon starts to leave through the exit in the solenoid switch and out of the system. This causes the balloon to deflate and slowly lower the system back into the base of the container. The process then started back from the top and started repeating. Hence, this prototype proved that the principle of upthrust can be used to move the inflation system along a vertical axis.

Once the principle of inflation was proven, the second prototype was built as mentioned above. While building this prototype, the basic operations of the first prototype were integrated into the second prototype and additional features were added to it in order to closely resemble a fully operational on-demand inflation system for underwater sensor deployment and minimise the limitations of the first prototype. For the first experiment, the code was modified so that the system would sink to a depth of $(1.00 \pm 0.25)$ m and hold its depth for a period of 10 minutes. Due to the uncertainty present in the sensor, a range of $(0.75 \pm 0.25)$ m meters to $(1.25 \pm 0.25)$ m meters was given for the system to maintain its depth. Then a weight of 200g was attached to the system using the nylon mesh used in the first prototype. The system was then turned on and put into the 2-meter container. The system began to sink immediately once it lay on the body of water until it reached the specified range and the solenoid switch turned on until it started moving upwards due to the upthrust. At the point the system moved above $(0.75 \pm 0.25)$ m meters, the solenoid switch turned off and the air was let go from the system until the object came back into the range. This process repeated itself until the system stayed stagnant within this range. After the 10 minutes were over the solenoid switch turned on and the balloon began inflating until the pressure sensor reading became less than 0.25 meters. Then the solenoid switch turned off and the system stayed just below the surface. The internal server was then accessed via WIFI using a laptop and the data which was processed using the ESP-32 module was collected wirelessly and processed data was downloaded. Then the device began to sink back into the depths of the container and restart the process of collecting data at a depth of 1 m.


Figure 4.12: The webpage created in the internal web server that allows the data to be downloaded remotely

As the second prototype operated as expected in the 2 m container, the device was once again tested in a 10m deep water body. Before submerging the system, the code was once again modified so that it would remain stagnant at $(9.00 \pm 0.25)$m. Due to the uncertainty present in the pressure sensor, the range of $(8.75 \pm 0.25)$ m meters to $(9.25 \pm 0.25)$ m meters was given as the range at which the device would stay remain constant. The device was then turned on and submerged in the body of water and it immediately began to sink until it reached the specified depth range. Once it passed the depth of $(9.25 \pm 0.25)$ m meters, the solenoid switch turned on and started inflating the balloon so that the system started moving towards the surface until it passed the depth of $(8.75 \pm 0.25)$ m meters. At this point, the solenoid switch turned off and started letting air out of the balloon until it came within the specified range and the system held constant at this range. It stayed in this range for a period of 10 minutes before the solenoid switch turned on and started inflating the balloon once again until the system reached the surface once the pressure sensor started reading the depth to be less than 0.25m the solenoid switch turned off and kept the system at the surface at the body of water. Once again, a laptop was used to connect to the internal server of the ESP-32 module via WIFI and download the sensor data that was processed by the ESP-32 module and send the flag to the system in order to restart the process. Once the flag was received by the server, the balloon started deflating and submerging the system back into the deep water body until it reached the specified depth and the process was repeated.

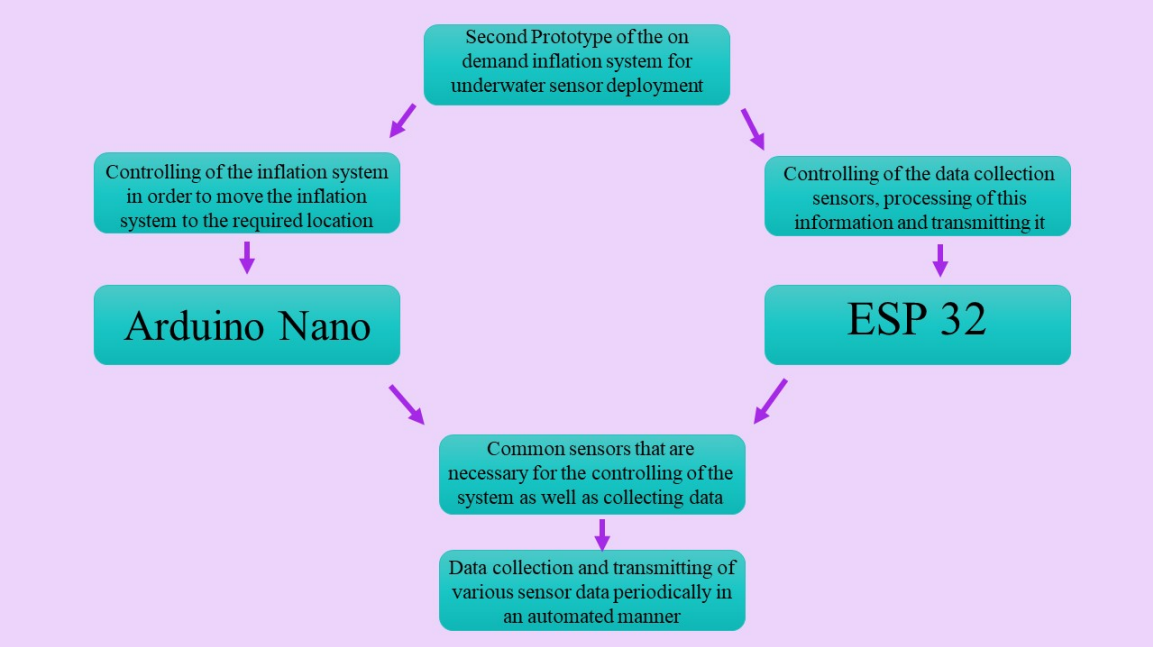


Figure 4.13: Design ideology of the second prototype of the on-demand inflation system for underwater sensor deployment

# 5 Result and Analysis

## 5.1 Results and the Analysis of the 2 prototypes

### 5.1.1 Results and Analysis of the first Prototype

This section provides the results of the first prototype created for the on-demand inflation system for underwater sensor deployment. This was made without any sensors to input data into the Arduino Nano. It operated using the internal timer in the Nano module. The operation of the device starts once the device is turned on by inserting the batteries into the battery holder, the internal clock would run for 10 minutes at which point it would turn on the solenoid switch which in turn inflates the balloon and brings the inflation system to the surface. The device rests at the surface for a period of $(30.00 \pm 0.01)$ seconds before the solenoid switch turns off again and the air starts leaving the balloon. When the air leaves the balloon the device starts slowly sinking back into the deep sea body. Due to the inflation system being only controlled using the internal timer in the Arduino Nano module, the depth to which the inflation system would sink could not be controlled. Only once the 10 minutes of inactivation was completed would the device begin moving back towards the surface.

When the first prototype was turned on by inserting the batteries and placed in water, due to the weights attached to the inflation system via the nylon mesh, the system started to sink. It sank to the bottom of the container and stayed at the bottom for a period of $(600.00 \pm 0.01)$ seconds and then the solenoid switch turned on and the balloon started inflating for a period of $(20.00 \pm 0.01)$ seconds. With the inflating balloon, the device raised off the ground and started moving towards the surface which took a time period of $(10.00 \pm 0.01)$ seconds. Once it reached the surface it stayed there for a further period of $(30.00 \pm 0.01)$ seconds. This timing was only held through for a container with a depth of 2 meters. Once the depth of the water container changed, the inflation device varied on how long it floated at the surface of the water. This is because even though the device was programmed to stay at the surface for $(30.00 \pm 0.01)$ seconds, this is contingent on the depth of the device. If the depth of the device is different than 2 meters, it would take more or less time than 20 seconds to reach the surface and since there is no external source for the system to gauge if the object has reached the surface of the water body, it would go through the entire specified time cycle which causes issues in the operation of this prototype. This prototype was analyzed on its movement along the vertical axis with the inflation and deflation of the balloon due to the switching of the solenoid switch according to the time periods specified in the code of the on-demand inflation system

which works as anticipated.

Table 1: Table of Distance (m) and Time taken (s) related to the first prototype

| Reading N.o | Distance travelled (±0.01)m | Time taken (±0.01)s |
|---|---|---|
| 1 | 2.00 | 14.08 |
| 2 | 1.99 | 13.97 |
| 3 | 2.00 | 14.00 |
| 4 | 2.00 | 13.92 |
| 5 | 2.01 | 14.03 |

The average velocity of the balloon in the upwards direction is $(0.1429 \pm 0.0005) ms^{-1}$.

Table 2: Table of Diameter of the balloon (m) and Volume of the balloon in the first prototype($m^3$)

| Reading N.o | Diameter of balloon (±0.01) m | Volume of the balloon (±0.02)$m^3$ |
|---|---|---|
| 1 | 0.31 | 0.02 |
| 2 | 0.26 | 0.01 |
| 3 | 0.30 | 0.01 |
| 4 | 0.34 | 0.02 |
| 5 | 0.29 | 0.01 |

Assuming the balloon to be spherical in shape, the upthrust acting on the system when the balloon is inflated to a diameter of $(0.30 \pm 0.01)$ m volume of the balloon was $(0.01 \pm 0.02) m^3$ is $(100.00 \pm 0.02)$ N.

The average acceleration of the system was calculated to be $(0.02 \pm 0.01) ms^{-2}$. The above results of the first prototype have been obtained by considering five random repetitions that occur in the cycles of gathering and transmitting sensor data from the inflation system to an external collection source.

### 5.1.2 Results and Analysis of the second Prototype

This section focuses on the second prototype built for the on-demand inflation system for underwater sensor deployment. Unlike the first prototype, the second prototype makes use of both a timer and a pressure sensor in order to understand when to operate the solenoid switch. In addition to this, the second prototype uses 2 modules of Arduino Nano and ESP32 in order to control the device and save and process the sensor data respectively so that system works as intended without any delay or information overloading. An SD card was used to store the sensor data and the

internal server created in the ESP32 module was used to transmit the processed data to the external device which in this case was a laptop and return a complete flag after the download is completed.

Initially, the second prototype was submerged in the $(1.00 \pm 0.25)$m container that was used in the first prototype.

Table 3: Table of Distance (m) and Time taken (s) related to the second prototype in 2m container.

| Reading N.o | Distance travelled ($\pm$0.01) m | Time taken ($\pm$0.01) s |
|---|---|---|
| 1 | 0.98 | 9.00 |
| 2 | 1.02 | 9.09 |
| 3 | 1.03 | 8.95 |
| 4 | 0.97 | 8.92 |
| 5 | 1.00 | 9.02 |

The average velocity of the balloon in the upwards direction is $(0.1111 \pm 0.0005) ms^{-1}$. Since the distance travelled is (1.00$\pm$0.01) m and the time taken to travel this distance is $(8.99 \pm 0.01)$ s.

The second prototype was then switched on and put into a deep body of water of a depth of $(10.00 \pm 0.01)$ meters. It was programmed to sink to a depth in the range of $(8.75 \pm 0.25)$ meters to $(9.25 \pm 0.25)$ meters.

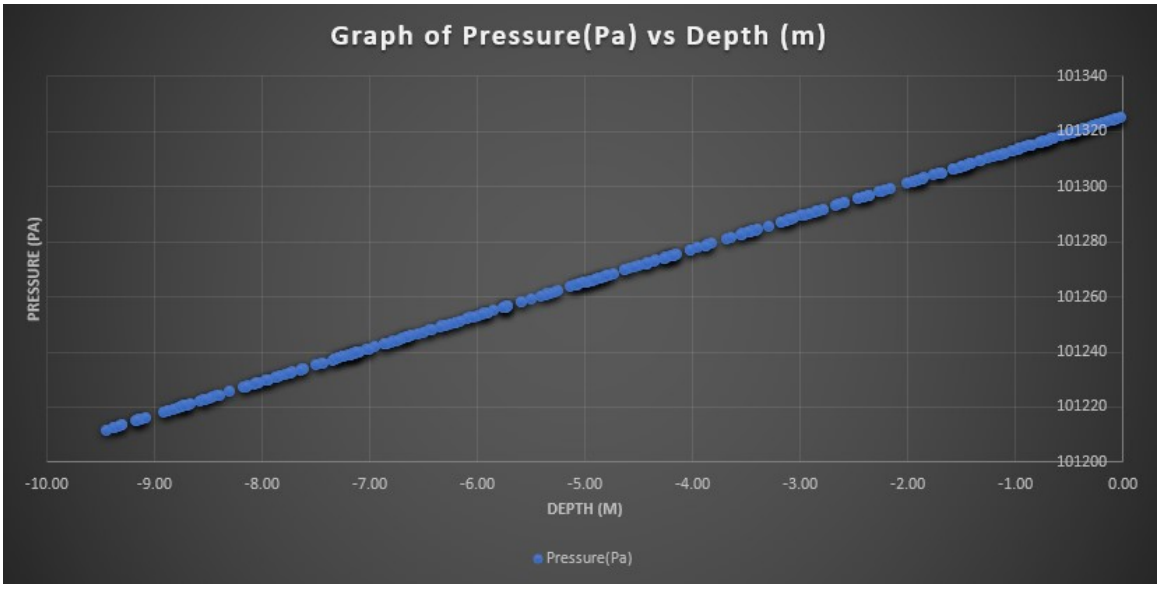


Figure 5.1: Graph of Pressure (Pa) vs Depth (m) in the second prototype of the on-demand inflation system for underwater sensor deployment

Once the pressure sensor read the depth was greater than $(9.25 \pm 0.01)$ meters it closed the solenoid switch and started inflating the balloon continuously until the depth was less than $(8.75 \pm 0.01)$ meters which in turn made the Arduino Nano open the solenoid switch until the system came back into this region.

Table 4: Table of Distance (m) and Time taken (s) related to the second prototype in 10m container.

| Reading N.o | Distance travelled ($\pm$0.01) m | Time taken ($\pm$0.01) s |
|---|---|---|
| 1 | 9.12 | 26.63 |
| 2 | 8.89 | 26.31 |
| 3 | 9.05 | 26.60 |
| 4 | 8.97 | 26.40 |
| 5 | 9.02 | 26.48 |

The average velocity of the balloon in the upwards direction is $(0.3399 \pm 0.0005) ms^{-1}$. Since the distance travelled is (9.00$\pm$0.25) m and the time taken to travel this distance is $(26.48 \pm 0.01)$ s.

This process was repeated until the system stabilized in this region. The device then stayed in this region for a period of $(600.00 \pm 0.01)$ seconds continuously gathering data, processing it and saving it on the micro SD card in real-time. Once the time period is over, the solenoid switch turns on regardless of the depth of the inflation system and brings it back to the surface. Once the device had reached the surface the internal server created on the ESP32 module starts looking for a connection via WIFI which gives the connected device to directly download the sensor data file once connected to the server. The laptop was able to successfully connect to the ESP32 server and download the data from the server. Once the data was downloaded it sent a flag which made the system restart the process again and sink back into the deep water body. The above results of the second prototype have been obtained by considering five random repetitions that occur in the cycles of gathering and transmitting sensor data from the inflation system to an external collection source.

The maximum depth to which the inflation system could submerge is limited by the lowest maximum pressure that the components can endure. In this prototype, maximum depth is limited by the waterproof electric box that was used which was rated as IP66. The IP66 rating for waterproof electric boxes is rated to a maximum water pressure of 100 kPa. A pressure increment of 100 kPa occurs every 10.2 meters. Therefore the theoretical max depth that could be reached is 10.2 meters. [3]

# 6 Discussion

The main objective of the research titled "On Demand Inflation System for Underwater Sensor Deployment" is to find a compact, cost-efficient and operational alternative to the systems and devices such as Autonomous Underwater Vehicles, Remotely Operated Vehicles, and the use of Buoys that are in operated across the globe to collect data from and deploy sensor into deep sea bodies such as oceans, rivers and lakes. To fulfil these objectives, an inflation system was created and the first step of creating this inflation system was to understand the types of systems that could be created in order to accomplish these objectives and the limitations of each type of system which could prevent it from achieving the aforementioned objectives. The two main types of systems that were considered in order to build the inflation system were an inflation system that operates by changing the volume of the system and an inflation system that operates by changing the density of the system. The common potential limitations that both of these systems faced were the cost of creating the system, the ability to withstand harsh underwater conditions, interactions with marine life and the efficiency of the system in collecting and transmitting data from beneath the deep water bodies to the surface of these water bodies. While considering how the two types of systems would overcome the limitations, it was clear that creating an inflation system that operates by varying its volume was more suitable to tackle these limitations as it is more cost-effective as well as power efficient due to varying mechanisms used in the proposed types. The former would use an external fuel source in the form of compressed air which is controlled by the system accordingly to make the system move as necessary along the vertical axis of the water body by inflating and deflating a balloon while the latter uses linear actuators inside the system to fill a cavity inside the system to vary its volume in order to move along the vertical axis of the water body. The use of linear actuators is more costly and well as less power efficient compared to the alternative which is solenoid switches. Due to this, the system which operates by varying its volume was chosen as the method on which the inflation system would be based upon.

Once the type of system was chosen, the next step in the research was to identify the mechanism that would be used to change the volume of the system. As a solution for this issue, a 2-position 3-way solenoid air valve was used. This air valve is capable of controlling the direction of an airflow across the valve depending on if the valve is switched on or off. This mechanism allows air to flow into the balloon when the switch is turned off and for the air to flow out of the balloon and out of the system when the switch is turned on. Before integrating this air valve into the system it needed to be

tasted to ensure that it could withstand the air pressure that would be flowing through it. Therefore, a separate circuitry was created which comprised of a power transistor, a diode and a resistor which was controlled using an Arduino Uno to turn the solenoid valve on and off every 10 seconds. With the use of pneumatic tubes, the air duster containing the compressed air and the inflating balloon was connected to the solenoid valve using superglue to ensure that there are no leaks in the created mechanism. With the use of a zip tie, the air duster was made to be in the continuously open position and the device was powered up. It could be observed that when the solenoid valve is in the on state the balloon started inflating and after a period of 10 seconds the solenoid valve would turn off and the balloon would start deflating for a period of 10 seconds before it started to inflate again. When the operation of this device was observed, it could be seen that there were minor air leaks inside the solenoid valve which indicates that the pathways aren't perfectly sealed even though it is not in use. In addition to this, these solenoid valves are very sensitive and need surge protection added for their safety and proper operation. However, it was deemed that these leaks are minor and the sensor was sturdy enough to operate in an isolated environment and would not affect the overall operation of the final design. Therefore, this model of the solenoid air valves was decided to be used in the creation of the inflation system.

Once the operation of the inflation mechanism was decided upon, the next step was to create a basic prototype for the inflation system. The primary objective of this prototype was to prove the principle of upthrust and its applicability to the inflation system. Therefore, the first prototype was built in a cost-efficient manner with only the essential components included in it. In an attempt to make this prototype compact as possible, the Arduino Uno module was switched to an Arduino Nano module. The small size of the Arduino Nano module compared to an Arduino Uno module while providing the same processing power and its ability to seamlessly interface with the Arduino internal development environment made it the logical choice as a substitution. Other components that were used in the first prototype were a waterproof electric box, a solenoid valve and its accompanying circuitry, a 500 g weight and a battery pack with 2 cells connected in series along with a 5 V regulator. The code was developed in the Arduino IDE such that the Nano module would send a signal to the solenoid valve to turn on after a time period of 10 minutes the balloon would start to inflate for a period of 20 seconds and the expansion of the volume in the balloon creates enough upthrust for the prototype to reach the surface of the water body. The system was then taken near a container of a depth of 2 meters and turned on and sealed before keeping it on the body of water.

The system started sinking immediately once kept on the water body and sank straight to the bottom of the container. Then once the 10 minutes had passed, the balloon began to inflate for a period of 20 seconds. After 9 seconds of inflating, the system began to rise off the base of the container and move towards the surface and remain at the surface for a period of 30 seconds. However, it was observed that once the system reached close to the surface and then a portion of the balloon moved above the body of water, the upthrust weight began to reduce and become stagnant once the upthrust became equal to the weight of the system. This shortcoming could be solved in future iterations of prototypes created using this type of inflation system would be to attach the inflation object to the base of the system such that the upthrust doesn't reduce when the system reaches the surface. Once the system stayed at the surface for a period of 30 seconds the balloon began to deflate and slowly move the device back into the bottom of the container. This basic prototype proved the applicability of the principle upthrust on the type of inflation system that we aim to create.

The primary issue of the first prototype is the lack of control that the Arduino Nano module has over the movement of the system. This is primarily due to the lack of peripherals in the system which would allow the Nano module to identify how the system is moving and react accordingly. Once this issue is rectified it would allow the system to sink to a specified depth and maintain itself at this depth while obtaining readings and measurements via sensors. The second issue that arose is the effect that power surges had on the solenoid valve. Due to the absence of a battery management system power surges that arose from the battery rendered the solenoid valve unusable and had to be replaced. The final issue that came up while operating this prototype was the insufficiency of the seal provided by the waterproof electric box. After 5 repetitions of the device moving to the bottom of the container and back to the surface, the device was taken out of the water and opened up. Once opened, water drops could be seen on the inside of the system container. To overcome this issue, during the second round of testing the system was once again sealed and a coating of silicon gum was applied along the base of the lid and allowed to dry before putting the device back into the water container. At the end of the second round of testing, the system was opened up and the interior of the system was completely dry. Since the primary objective of the first prototype was successfully achieved, it was decided to create another prototype which built upon the first prototype but also addressed the shortcomings that could be observed in the first prototype.

The second prototype was built upon the design of the first prototype with the shortcomings of the first prototype in mind. The first step in creating this prototype

was to design a PCB which comprised all the components necessary for the second prototype. The components used in the second prototype include the components of the first prototype and in addition to this, an ESP32 module, a micro SD card module and a BMP180 pressure sensor. Once the components were correctly connected, the corresponding PCB design was created. Here an additional ESP32 module was used to ease the load which would have fallen on a singular Arduino Nano module as its single processing core cannot handle the controlling of the system, collecting information and processing it simultaneously. The 2 cores allow the system to be future-proofed in the case of adding and processing data from more sensors without hindering the operation of the system. Once the PCB design was successfully created the design was then printed on a thermal transfer paper and etched onto a copper-plated board which had dimensions similar to the waterproof electric box. The board was then immersed in a solution of Ferric(III) chloride solution so that the additional copper is stripped from the board subsequently the board was drilled and components were attached to the board.

The next step was to configure the pressure sensor and the micro SD card module. The pressure sensor was connected to the Arduino Nano module and the SD card was connected to the ESP32 module. The pressure readings taken from the sensor were read by the Arduino Nano module and serially transferred to the ESP32 module which in turn processes this data and saves it in the SD card. In order to accomplish this separate codes needed to be developed to individually control the Arduino Nano module and the ESP32 module. The code for the Arduino Nano module was built upon the code that was created from the first prototype. The readings of the pressure sensor along with the parameters in which the system could move in a deep water body were added to this code. These parameters allowed the inflation system to submerge to a required depth and remain constant at this depth for a specified period of time. In the case of this prototype, the parameters were given such that it would submerge for a period of 10 minutes at a depth range of 8.75 meters to 9.25 meters. Once the 10 minutes had passed, the balloon would begin inflating until the pressure read a depth value of less than 0.25 meters. Since this pressure sensor has an accuracy of 0.25 meters and makes its calculation based on the value specified as the sea level pressure, the changes in the sea level from time to time led to the device not reaching the surface of the water body. Therefore, in further iterations, either a more accurate pressure sensor needs to be used or a different type of sensor needs to be used to minimize issues that may occur from changes to the surface air pressure at sea level. Once the sensor data is serially transmitted from the Arduino Nano module to the ESP32 module, the ESP32 module then processes this data and divides into individual

data and saves it as a CSV file in the SD card module. It continues this process until the device reaches the surface. Once the system reaches the surface that is when the internal web server of the ESP32 module becomes accessible for external devices to connect. Once the external device connects to it and downloads the data, a flag is automatically sent to the ESP32 to trigger the reset pin on the Arduino Nano module to restart the entire process which in turn starts deflating the balloon and thereby slowly sinking the device back onto the depths of the water body. The second issue that occurred in the testing phase of the second prototype was the failure of the SD card mount while the device was in the process of collecting data which had to be replaced before continuing the testing of the system. In order to overcome this issue, an additional SD card could be used so that a backup of the data is present in case of a failure in the primary data-saving mechanism. Thirdly, some external devices were unable to connect to the IP address of the web page containing the sensor information as they were unable to resolve the IP address. However, since the majority of the external devices were able to connect this seems to be an issue with those devices rather than that of the web server. Therefore, when selecting an automated device to collect data from the inflation system, the device chosen must be compatible with the web server created by the ESP32 server. Additionally, the ESP32 module only supports a WIFI network of 2.4GHz. However, the 2.4GHz bandwidth has an high absorption rate and reduces the ability of the network to penetrate water bodies. In future iterations, the WIFI network used must be of bandwidth 5GHz as it is not absorbed as much by water which helps it travels further throught water bodies. [7]

One of the main issues that need to be addressed in the second prototype is the inability of the entire inflation to come off the water body. Due to this inability, the ESP32 module would remain underwater thereby creating a weak WIFI network on the above surface of the water. To connect to this WIFI server, the external device had to be brought close to the surface of the water in order to avoid connection interruptions, This could be resolved by attaching the inflating mechanism to the bottom of the system in a manner which doesn't capsize the system and elevates the entire system above the surface of the water body. Also, to increase the range of the WIFI network an antenna could be added to the circuit so as to make the connection between the inflation system and the external device stronger the process of transferring data more efficient. Additionally, another issue with creating a compact device filled with many components is the tendency of these components to short circuit due to terminals of various components coming into contact with each other. This issue was resolved by applying a layer of glue at every terminal and on the bottom of the PCB to create an insulation layer among the components. A useful

addition to this prototype in future iterations would be the addition of controllable motors along a GPS module which could be used to help guide the inflation system to a specified location with the use of coordinates once it reaches the surface making it easier for the external device to locate the inflation system.

This system is also useful in confining the data collection to a specific region in the water body as readings from unspecified locations may vary from what actually needs to be researched. Finally, an internal troubleshooting system needs to be created for this system such that if an error is found during an internal diagnostic session, the inflation system would abandon the process of collecting data and return to the surface and remain there such that it could be retrieved and the error is dealt with. This diagnostics must include a feature to measure the available battery power as well. The details of the internal diagnostics could be transmitted onto the unmanned external device which arrives periodically to collect data to inform the researchers regarding the errors that are present in the inflation system. [10]

The fully operational second prototype manages to achieve the primary objective of the research to create an on-demand inflation system for underwater sensor deployment which is compact, cost-effective and power efficient. This prototype is capable of collecting data from various sensors added onto it simultaneously and saving it in a storage device after processing it. This data could then be transmitted to an external device which ideally would be an unmanned device which could travel into the ocean or another deep water body and automatically connect to the web server created by the ESP32 module and download the data automatically once connected and bring this data to researchers while the inflation system subtly lowers back into the ocean to repeat the process.

# 7 Conclusion

The main objective of the research titled "On Demand Inflation System for Underwater Sensor Deployment" is to find a compact, cost-efficient and operational alternative to the systems and devices such as Autonomous Underwater Vehicles, Remotely Operated Vehicles, and the use of Buoys that are in operated across the globe to collect data from and deploy sensor into deep sea bodies such as oceans, rivers and lakes. In order to accomplish this objective, 2 prototypes were created which relied on varying the volume of the inflation system to move the device along the vertical axis. Compressed air was used as the fuel to move the inflating mechanism and a balloon was used as the inflating object. The second prototype acts as an extension of the first prototype and has the capability of maintaining the depth of the inflation system at specified depths in water bodies in order to obtain data. This was done by obtaining regular readings using a pressure sensor to understand the depth to which the inflation system has sunk. Once the readings have been obtained for a specified period of time the inflation system would move back towards the surface and subsequently repeat this process.

The data collected from the Arduino Nano module is sent serially to the ESP32 module. The data received by the ESP32 module is then processed by the same module and saved in the micro SD card. Data was saved as a CSV file so that it could be more easily analyzed for information. The CSV file was made accessible to the internal web server that was created in the ESP32 module. The web server was created in such a way that the data file could be downloaded by an external device that connects to the web server via a WIFI network. Once the file is downloaded by the external device a flag is sent back to the ESP32 module via the web server to reset the Arduino Nano module. Once this flag is received the input to the reset pin of the Nano module is made low for a second before it is made high again. This cause the Arduino Nano model to run the code once again from the beginning and start sinking back into the depth and thereby repeating the process of information collection.

This prototype accomplishes the main objectives required of an on-demand inflation system for underwater sensor deployment. However, even though it is capable of submerging into a deep water body and maintaining itself at a specified depth while collecting readings and then transmitting them to an external device, this prototype still needs to be improved upon in order to act as a truly isolated system and act as a replacement to the methods of underwater data collection that are used

presently. These improvements include would allow the inflation system to remain operational in harsh conditions while collecting the necessary data from various sensors simultaneously and counter any unforeseen circumstances in real time which would lead to this device being used in various uncontrolled environments for reliable, compact and cost-effective data collection and transferring of data in a larger distance than what is currently possible in this prototype. In conclusion, the prototype created as an on demand inflation system for underwater sensor deployment operates successfully in collecting data from deep water bodies and transmitting it to the end user once it reaches the surface. However, additional components need to be added to the prototype in future iterations in order to use this in uncontrolled environments.

# References


[1] Autonomous Underwater Vehicles. oceanexplorer.noaa.gov/technology/subs/auvs/auvs.html.

[2] Buoy. www.usgs.gov/media/images/university-maine-ocean-observing-system-surface-buoy.

[3] IP Ratings Explained. www.clarionuk.com/resources/ip-ratings/.

[4] The Deep Sea.

[5] Underwater Robotic Vehicles (ROV  AUV) for Inspections, Observation and Videography.

[6] An Introduction to 5GHz Technology, 2021.

[7] Can Water Interfere With WiFi Signals? Exploring The Science Behind WiFi And Water Interactions, 2023.

[8] Share Alike. Arduino Nano V2.3 User Manual. *Arduino*, pages 1–5, 2008.

[9] S. Aronica, N. Gabriele Gallì, Bernardo Patti, I. Fontana, P. Calandrino, G. Giacalone, G. Basilone, S. Mazzola, and A. Bonanno. The autonomous underwater data acquisition system for physical and chemical parameters (AUDAS-PCP) onboard a fishing vessel. *Journal of Operational Oceanography*, 9:s58–s65, jun 2016.

[10] Daniel Babatunde, Simon Pomeroy, Paul Lepper, Ben Clark, and Rebecca Walker. Autonomous deployment of underwater acoustic monitoring devices using an unmanned aerial vehicle: The flying hydrophone. *Sensors (Switzerland)*, 20(21):1–21, nov 2020.

[11] G. R. Bigg, T. D. Jickells, P. S. Liss, and T. J. Osborn. The role of the oceans in climate. *International Journal of Climatology*, 23(10), 2003.

[12] Clemens Deutsch. *On the Performance of Long-Range Autonomous Underwater Vehicles : Enhancing the Endurance of AUVs*.

[13] Espressif. ESP32 Series Datasheet. *Espressif Systems*, pages 1–69, 2022.

[14] Li Li Li, Jing Yu He, Yong Peng Zhao, and Jian Hong Yang. Design of microcontroller standard SPI interface. In *Applied Mechanics and Materials*, volume 618, pages 563–568. Trans Tech Publications Ltd, 2014.

[15] Product Reference Manual. Arduino UNO R3 Features. pages 1–13, 2022.

# 8 Appendix

The Arduino code created to operate the Arduino Nano module and the pressure sensor.

```
#include <Wire.h>
#include <Adafruit_BMP085.h>
#define seaLevelPressure_hPa 1011.50

Adafruit_BMP085 bmp;

int t=0;

void setup() {
  Serial.begin(9600);
  pinMode(6, OUTPUT);
  if (!bmp.begin()) {
    Serial.println("Could not find a valid BMP085 sensor, check wiring!");
    while (1) {}
  }
  Serial.println("Booted");
  }

void loop() {

  int x = bmp.readAltitude(seaLevelPressure_hPa * 100);
  Serial.print(bmp.readTemperature());
  Serial.print(",");
  int32_t pressure = bmp.readPressure()*1.3402;
  Serial.print(pressure);
  Serial.print(",");
  Serial.print(bmp.readAltitude());
  Serial.print(",");
  Serial.print(bmp.readSealevelPressure()*1.3402);

  Serial.print(",");
  Serial.print(bmp.readAltitude(seaLevelPressure_hPa * 100));
  Serial.print(",");
  Serial.print(t);
```

```
  Serial.println("");

  if ((t<=60000) && (x<=-0.25)){
    if (-9.75>=x){
      digitalWrite(6, LOW);
    }
    else if(x<=-10.25){
      digitalWrite(6, HIGH);
    }
    else{
    digitalWrite(6, HIGH);
    }

  }
  else if((t<=60000) && (x>-0.25)){
    digitalWrite(6, LOW);
  }
  else{
    digitalWrite(6, HIGH);
  }
delay(1000);
t++;
}
```

The Arduino code created for the ESP32 module and the SD card module.

```
#include "FS.h"
#include "SD.h"
#include "SPI.h"

#include <WiFi.h>
#include <ESP32WebServer.h>
#include <ESPmDNS.h>

#include "CSS.h"

ESP32WebServer server(80);

#define RXp2 16
#define TXp2 17
const int ledPin = 21;

#define servername "research"
#define SD_pin 16

bool   SD_present = false;

String data = "";

void listDir(fs::FS &fs, const char * dirname, uint8_t levels){
    Serial.printf("Listing directory: %s\n", dirname);

    File root = fs.open(dirname);
    if(!root){
        Serial.println("Failed to open directory");
        return;
    }
    if(!root.isDirectory()){
        Serial.println("Not a directory");
        return;
    }

    File file = root.openNextFile();
```

```
    while(file){
        if(file.isDirectory()){
            Serial.print("  DIR : ");
            Serial.println(file.name());
            if(levels){
                listDir(fs, file.path(), levels -1);
            }
        } else {
            Serial.print("  FILE: ");
            Serial.print(file.name());
            Serial.print("  SIZE: ");
            Serial.println(file.size());
        }
        file = root.openNextFile();
    }
}

void createDir(fs::FS &fs, const char * path){
    Serial.printf("Creating Dir: %s\n", path);
    if(fs.mkdir(path)){
        Serial.println("Dir created");
    } else {
        Serial.println("mkdir failed");
    }
}

void removeDir(fs::FS &fs, const char * path){
    Serial.printf("Removing Dir: %s\n", path);
    if(fs.rmdir(path)){
        Serial.println("Dir removed");
    } else {
        Serial.println("rmdir failed");
    }
}

void readFile(fs::FS &fs, const char * path){
    Serial.printf("Reading file: %s\n", path);
```

```
    File file = fs.open(path);
    if(!file){
        Serial.println("Failed to open file for reading");
        return;
    }

    Serial.print("Read from file: ");
    while(file.available()){
        Serial.write(file.read());
    }
    file.close();
}

void writeFile(fs::FS &fs, const char * path, const char * message){
    Serial.printf("Writing file: %s\n", path);

    File file = fs.open(path, FILE_WRITE);
    if(!file){
        Serial.println("Failed to open file for writing");
        return;
    }
    if(file.print(message)){
        Serial.println("File written");
    } else {
        Serial.println("Write failed");
    }
    file.close();
}

void appendFile(fs::FS &fs, const char * path, const char * message){
    Serial.printf("Appending to file: %s\n", path);

    File file = fs.open(path, FILE_APPEND);
    if(!file){
        Serial.println("Failed to open file for appending");
        return;
    }
    if(file.print(message)){
```

```
        Serial.println("Message appended");
    } else {
        Serial.println("Append failed");
    }
    file.close();
}

void renameFile(fs::FS &fs, const char * path1, const char * path2){
    Serial.printf("Renaming file %s to %s\n", path1, path2);
    if (fs.rename(path1, path2)) {
        Serial.println("File renamed");
    } else {
        Serial.println("Rename failed");
    }
}

void deleteFile(fs::FS &fs, const char * path){
    Serial.printf("Deleting file: %s\n", path);
    if(fs.remove(path)){
        Serial.println("File deleted");
    } else {
        Serial.println("Delete failed");
    }
}

void testFileIO(fs::FS &fs, const char * path){
    File file = fs.open(path);
    static uint8_t buf[512];
    size_t len = 0;
    uint32_t start = millis();
    uint32_t end = start;
    if(file){
        len = file.size();
        size_t flen = len;
        start = millis();
        while(len){
            size_t toRead = len;
            if(toRead > 512){
```

```
                toRead = 512;
            }

            file.read(buf, toRead);
            len -= toRead;
        }
        end = millis() - start;
        Serial.printf("%u bytes read for %u ms\n", flen, end);
        file.close();
    } else {
        Serial.println("Failed to open file for reading");
    }


    file = fs.open(path, FILE_WRITE);
    if(!file){
        Serial.println("Failed to open file for writing");
        return;
    }

    size_t i;
    start = millis();
    for(i=0; i<2048; i++){
        file.write(buf, 512);
    }
    end = millis() - start;
    Serial.printf("%u bytes written for %u ms\n", 2048 * 512, end);
    file.close();
}

void setup(){
    Serial.begin(9600);
    Serial2.begin(9600, SERIAL_8N1, RXp2, TXp2);
    digitalWrite(ledPin, HIGH);
    if(!SD.begin()){
        Serial.println("Card Mount Failed");
        return;
    }
    uint8_t cardType = SD.cardType();
```

```
    WiFi.softAP("Research", "12345678");

  if (!MDNS.begin(servername))
  {
    Serial.println(F("Error setting up MDNS responder!"));
    ESP.restart();
  }

  Serial.print(F("Initializing SD card..."));

  if (!SD.begin(SD_pin))
  {
    Serial.println(F("Card failed or not present, no SD Card data logging possible..
    SD_present = false;
  }
  else
  {
    Serial.println(F("Card initialised... file access enabled..."));
    SD_present = true;
  }

  /*********  Server Commands  **********/
  server.on("/",         SD_dir);
  server.on("/upload",   File_Upload);
  server.on("/fupload",  HTTP_POST,[](){ server.send(200);},
  handleFileUpload);

  server.begin();

  Serial.println("HTTP server started");
  pinMode(ledPin, OUTPUT);
  deleteFile(SD, "/data.csv");
  appendFile(SD, "/data.csv", "Temperature,Pressure,Altitude,
  Pressure at sealevel (calculated),Real altitude, Seconds\n");
}

void loop(void)
```

```
{
  data = Serial2.readString();
  Serial.print(data);
  appendFile(SD, "/data.csv", data.c_str());
  server.handleClient(); //Listen for client connections
}

void SD_dir()
{
  if (SD_present)
  {
    if (server.args() > 0 )
    {
      Serial.println(server.arg(0));

      String Order = server.arg(0);
      Serial.println(Order);

      if (Order.indexOf("download_")>=0)
      {
        Order.remove(0,9);
        SD_file_download(Order);
        Serial.println(Order);
      }

      if ((server.arg(0)).indexOf("delete_")>=0)
      {
        Order.remove(0,7);
        SD_file_delete(Order);
        Serial.println(Order);
      }
    }

    File root = SD.open("/");
    if (root) {
      root.rewindDirectory();
      SendHTML_Header();
      webpage += F("<table align='center'>");
```

```
      webpage += F("<tr><th>Name/Type</th><th style='width:20%'>Type File/
      Dir</th><th>File Size</th></tr>");
      printDirectory("/",0);
      webpage += F("</table>");
      SendHTML_Content();
      root.close();
    }
    else
    {

      SendHTML_Header();
      webpage += F("<h3>No Files Found</h3>");
    }
    append_page_footer();
    SendHTML_Content();
    SendHTML_Stop();
  } else ReportSDNotPresent();
}

//Upload a file to the SD
void File_Upload()
{
  append_page_header();
  webpage += F("<h3>Select File to Upload</h3>");
  webpage += F("<FORM action='/fupload' method='post'
  enctype='multipart/form-data'>");
  webpage += F("<input class='buttons' style='width:25%' type='file'
  name='fupload' id = 'fupload' value=''>");
  webpage += F("<button class='buttons' style='width:10%'
  type='submit'>Upload File</button><br><br>");
  webpage += F("<a href='/'>[Back]</a><br><br>");
  append_page_footer();
  server.send(200, "text/html",webpage);
}

void printDirectory(const char * dirname, uint8_t levels)
{

  File root = SD.open(dirname);
```

```
if(!root){
  return;
}
if(!root.isDirectory()){
  return;
}
File file = root.openNextFile();

int i = 0;
while(file){
  if (webpage.length() > 1000) {
    SendHTML_Content();
  }
  if(file.isDirectory()){
    webpage += "<tr><td>"+String(file.isDirectory()?"Dir":"File")+"
    </td><td>"+String(file.name())+"</td><td></td></tr>";
    printDirectory(file.name(), levels-1);
  }
  else
  {
    webpage += "<tr><td>"+String(file.name())+"</td>";
    webpage += "<td>"+String(file.isDirectory()?"Dir":"File")+"</td>";
    int bytes = file.size();
    String fsize = "";
    if (bytes < 1024)                    fsize = String(bytes)+" B";
    else if(bytes < (1024 * 1024))   fsize = String(bytes/1024.0,3)+" KB";
    else if(bytes < (1024*1024*1024)) fsize = String(bytes/1024.0/1024.0,3)+"MB";
    else

    fsize = String(bytes/1024.0/1024.0/1024.0,3)+" GB";
    webpage += "<td>"+fsize+"</td>";
    webpage += "<td>";
    webpage += F("<FORM action='/' method='post'>");
    webpage += F("<button type='submit' name='download'");
    webpage += F("' value='");
    webpage +="download_"+String(file.name());
    webpage +=F("'>Download</button>");
```

```
      webpage += "</td>";
      webpage += "<td>";
      webpage += F("<FORM action='/' method='post'>");
      webpage += F("<button type='submit' name='delete'");
      webpage += F("' value='");
      webpage +="delete_"+String(file.name()); webpage +=F("'>Delete</button>");
      webpage += "</td>";
      webpage += "</tr>";

    }
    file = root.openNextFile();
    i++;
  }
  file.close();
}

void SD_file_download(String filename)
{
  if (SD_present)
  {
    File download = SD.open("/"+filename);
    if (download)
    {

      server.sendHeader("Content-Type", "text/text");
      server.sendHeader("Content-Disposition", "attachment; filename="+filename);
      server.sendHeader("Connection", "close");
      server.streamFile(download, "application/octet-stream");
      download.close();
      delay(1000);
      digitalWrite(ledPin, LOW);
      delay(1000);
      digitalWrite(ledPin, HIGH);
    } else ReportFileNotPresent("download");
  } else ReportSDNotPresent();
}
```

```
File UploadFile;

void handleFileUpload()
{
  HTTPUpload& uploadfile = server.upload();

  if(uploadfile.status == UPLOAD_FILE_START)
  {
    String filename = uploadfile.filename;
    if(!filename.startsWith("/")) filename = "/"+filename;
    Serial.print("Upload File Name: "); Serial.println(filename);
    SD.remove(filename);
    UploadFile = SD.open(filename, FILE_WRITE);
    filename = String();
  }
  else if (uploadfile.status == UPLOAD_FILE_WRITE)
  {

    if(UploadFile) UploadFile.write(uploadfile.buf, uploadfile.currentSize);
  }
  else if (uploadfile.status == UPLOAD_FILE_END)
  {
    if(UploadFile)
    {
      UploadFile.close();
      Serial.print("Upload Size: "); Serial.println(uploadfile.totalSize);
      webpage = "";
      append_page_header();
      webpage += F("<h3>File was successfully uploaded</h3>");
      webpage += F("<h2>Uploaded File Name:");
      webpage+= uploadfile.filename+"</h2>";
      webpage += F("<h2>File Size:");
      webpage += file_size(uploadfile.totalSize) + "</h2><br><br>";
      webpage += F("<a href='/'>[Back]</a><br><br>");
      append_page_footer();
      server.send(200,"text/html",webpage);
    }
    else
```

```
    {
      ReportCouldNotCreateFile("upload");
    }
  }
}

void SD_file_delete(String filename)
{
  if (SD_present) {
    SendHTML_Header();
    File dataFile = SD.open("/"+filename, FILE_READ);
    if (dataFile)
    {

      if (SD.remove("/"+filename)) {
        Serial.println(F("File deleted successfully"));
        webpage += "<h3>File '"+filename+"' has been erased</h3>";
        webpage += F("<a href='/'>[Back]</a><br><br>");
      }
      else
      {
        webpage += F("<h3>File was not deleted - error</h3>");
        webpage += F("<a href='/'>[Back]</a><br><br>");
      }
    } else ReportFileNotPresent("delete");
    append_page_footer();
    SendHTML_Content();
    SendHTML_Stop();
  } else ReportSDNotPresent();
}


void SendHTML_Header()
{
  server.sendHeader("Cache-Control", "no-cache, no-store, must-revalidate");
  server.sendHeader("Pragma", "no-cache");
  server.sendHeader("Expires", "-1");
  server.setContentLength(CONTENT_LENGTH_UNKNOWN);
  server.send(200, "text/html", "");
```

```
  append_page_header();
  server.sendContent(webpage);
  webpage = "";
}


void SendHTML_Content()
{
  server.sendContent(webpage);
  webpage = "";
}

void SendHTML_Stop()
{
  server.sendContent("");
  server.client().stop();
}

void ReportSDNotPresent()
{
  SendHTML_Header();
  webpage += F("<h3>No SD Card present</h3>");
  webpage += F("<a href='/'>[Back]</a><br><br>");
  append_page_footer();
  SendHTML_Content();
  SendHTML_Stop();
}

void ReportFileNotPresent(String target)
{
  SendHTML_Header();
  webpage += F("<h3>File does not exist</h3>");
  webpage += F("<a href='/"); webpage += target + "'>[Back]</a><br><br>";
  append_page_footer();
  SendHTML_Content();
  SendHTML_Stop();
}
```

```
//ReportCouldNotCreateFile
void ReportCouldNotCreateFile(String target)
{
  SendHTML_Header();
  webpage += F("<h3>Could Not Create Uploaded File (write-protected?)</h3>");
  webpage += F("<a href='/"); webpage += target + "'>[Back]</a><br><br>";
  append_page_footer();
  SendHTML_Content();
  SendHTML_Stop();
}

//File size conversion
String file_size(int bytes)
{
  String fsize = "";
  if (bytes < 1024)                 fsize = String(bytes)+" B";
  else if(bytes < (1024*1024))      fsize = String(bytes/1024.0,3)+" KB";
  else if(bytes < (1024*1024*1024)) fsize = String(bytes/1024.0/1024.0,3)+" MB";
  else       fsize = String(bytes/1024.0/1024.0/1024.0,3)+" GB";
  return fsize;
}
```

The CSS code which complements the ESP32 code to build the website in the internal webserver.

```
<html>
        <head>
                <title>Inflation System Server</title>
        <meta name='viewport' content='user-scalable=yes,initial-scale=1.0,
    width=device-width'>
        <style>//From here style:
        body{max-width:65%;margin:0 auto;font-family:arial;font-size:100%;}
        ul{list-style-type:none;padding:0;border-radius:0em;overflow:hidden;
    background-color:#d90707;font-size:1em;}
        li{float:left;border-radius:0em;border-right:0em solid #bbb;}
        li a{color:white; display: block;border-radius:0.375em;padding:0.44em 0.44em;
    text-decoration:none;font-size:100%}
        li a:hover{background-color:#e86b6b;border-radius:0em;font-size:100%}
        h1{color:white;border-radius:0em;font-size:1.5em;padding:0.2em 0.2em;background:
    #d90707;}
        h2{color:blue;font-size:0.8em;}
        h3{font-size:0.8em;}
        table{font-family:arial,sans-serif;font-size:0.9em;border-collapse:collapse;
    width:85%;}
        th,td {border:0.06em solid #dddddd;text-align:left;padding:0.3em;border-bottom:
    0.06em solid #dddddd;}
        tr:nth-child(odd) {background-color:#eeeeee;}
        .rcorners_n {border-radius:0.5em;background:#558ED5;padding:0.3em 0.3em;width:
    20%;color:white;font-size:75%;}
        .rcorners_m {border-radius:0.5em;background:#558ED5;padding:0.3em 0.3em;width:
    50%;color:white;font-size:75%;}
        .rcorners_w {border-radius:0.5em;background:#558ED5;padding:0.3em 0.3em;width:
    70%;color:white;font-size:75%;}
        .column{float:left;width:50%;height:45%;}
        .row:after{content:'';display:table;clear:both;}
        *{box-sizing:border-box;}
        a{font-size:75%;}
        p{font-size:75%;}

        </style>
        </head>
        <body>
                <h1>Inflation System Server</h1>
                <ul>
                <li><a href='/'>Files</a></li>
                <li><a href='/upload'>Configuration</a></li>
                </ul>
```

```
        </body>
</html>
```