



University of Southern Queensland  
Faculty of Health, Engineering and Sciences

# APPARATUS FOR MEASURING MICROPLASTIC AEROSOLS IN THE ATMOSPHERE

Dissertation submitted by

**Mr James Beecham**



In fulfilment of the requirements of

**Course ENG4111/ENG4112 – Research Project**

towards the degree of

**Bachelor of Engineering (Honours) (Mechatronic Engineering)**

**October 2023**

## ABSTRACT

Microplastics are a topic of increasing concern, with research finding these tiny plastic particles everywhere from Antarctic ice to human blood. Recently microplastics have been observed in the upper atmosphere, plausibly the transport mechanism which carries them to remote regions. Research in the area is new, so a standard apparatus has yet to be developed. The lack of standardisation makes it difficult to compare the results of various studies to model the transport mechanism (Beaurepaire et al. 2021).

This project aims to design, implement and test an apparatus which can be carried by a weather balloon to draw known quantities of air through a filter to collect samples of particulate matter.

As the field is new, it is not yet clear what additional data may be required. Therefore the control system's operation must be easily modified by researchers and extra sensors easily added. Of utmost importance is the ability to be operated by researchers with no presumed knowledge of electronics or computer coding, and the minimum possible computer skills requirement. For easy replication, the apparatus must use common and affordable components and avoid bespoke parts.

A binderless glass fibre filter is used, as it contains no plastic and can be heated to remove any existing microplastic contamination (Song et al. 2021). To draw air through it, a cooling fan for computer servers is used in conjunction with an automotive mass airflow (MAF) sensor for feedback. By combining the MAF sensor with pressure and temperature sensors, the volumetric flow rate of air through the filter is calculated. By controlling the fan accordingly, the control system can then draw a known quantity of air through the filter.

To make the apparatus adaptable and accessible, the control system is based around a Raspberry Pi Pico running the CircuitPython programming language. Python is already widely used for data processing in research environments. The control system is programmed by dragging the code file onto it like a USB flash drive, with no software required. Retrieving the comma separated variables (CSV) format data logs is done in the same way.

Mounting parts were 3D printed and laser cut, and a simple circuit board with basic components was made to facilitate hardware interfaces and power requirements. The apparatus was assembled and tested in free air and in a vacuum chamber to assess its operation in a partial vacuum which simulates some of the conditions of that atmosphere at high altitude.

The apparatus met its goals for automatic operation, weight, cost, ease of construction and use, and sample collection performance in both free air and a vacuum chamber, showing promise for deployment in further research.

**University of Southern Queensland  
Faculty of Health, Engineering and Sciences  
ENG4111/ENG4112 Research Project**

## **LIMITATIONS OF USE**

The Council of the University of Southern Queensland, its Faculty of Health, Engineering & Sciences, and the staff of the University of Southern Queensland, do not accept any responsibility for the truth, accuracy or completeness of material contained within or associated with this dissertation.

Persons using all or any part of this material do so at their own risk, and not at the risk of the Council of the University of Southern Queensland, its Faculty of Health, Engineering & Sciences or the staff of the University of Southern Queensland.

This dissertation reports an educational exercise and has no purpose or validity beyond this exercise. The sole purpose of the course pair entitled “Research Project” is to contribute to the overall education within the student’s chosen degree program. This document, the associated hardware, software, drawings, and other material set out in the associated appendices should not be used for any other purpose: if they are so used, it is entirely at the risk of the user.

**University of Southern Queensland  
Faculty of Health, Engineering and Sciences  
ENG4111/ENG4112 Research Project**

**CERTIFICATION OF DISSERTATION**

I certify that the ideas, designs and experimental work, results, analyses and conclusions set out in this dissertation are entirely my own effort, except where otherwise indicated and acknowledged.

I further certify that the work is original and has not been previously submitted for assessment in any other course or institution, except where specifically stated.

A solid black rectangular box used to redact the signature of Mr. James Beecham.

Mr James Beecham

## ACKNOWLEDGEMENTS

This research was conducted under the supervision of Dr Craig Lobsey & Dr Tobias Low. I'd like to thank Craig for the idea behind this project and both of them for providing guidance throughout.

## TABLE OF CONTENTS

Contents	Page
<b>ABSTRACT</b>	<b>2</b>
<b>LIMITATIONS OF USE</b>	<b>3</b>
<b>CERTIFICATION OF DISSERTATION</b>	<b>4</b>
<b>ACKNOWLEDGEMENTS</b>	<b>5</b>
<b>LIST OF FIGURES</b>	<b>8</b>
<b>NOMENCLATURE AND ABBREVIATIONS</b>	<b>9</b>
 <b>CHAPTER 1 INTRODUCTION</b>	 <b>10</b>
1.1 Outline	10
1.2 Introduction	10
1.3 The Problem	10
1.4 Aim	10
1.5 Conclusions	11
 <b>CHAPTER 2 PROJECT PLANNING</b>	 <b>12</b>
2.1 Aims	12
2.2 Objectives	12
2.3 Timeline	13
2.3.1 Risk Assessment	13
2.3.2 Environmental Sustainability	13
2.3.3 Project Disposal	14
 <b>CHAPTER 3 LITERATURE REVIEW</b>	 <b>15</b>
3.1 Introduction	15
3.2 Microplastic Aerosols	15
3.3 Apparatus Design and Testing Procedures	15
3.3.1 Particulate Sample Collection Configuration	15
3.3.2 Control System	16
3.3.3 Microplastic-Specific Procedures	16
3.4 Summary	16
 <b>CHAPTER 4 METHODOLOGY</b>	 <b>17</b>
4.1 Design requirements	17
4.1.1 General requirements	17
4.1.2 Sample collection filter	17
4.1.3 Physical design	17
4.1.4 Electronic hardware	18
Fan	18
Airflow sensor	18
Pressure sensor	18
Battery	18
Microcontroller	18
4.1.5 Software control system	18
4.2 Design Implementation	19

4.2.1 Sample collection filter	19
4.2.2 Electronic hardware	19
Fan	19
Airflow sensor	20
Pressure sensor	21
Batteries	22
4.2.3 Physical design	24
4.2.4 Control system	28
Microcontroller	28
Programming language	28
Software code structure	29
Hardware interfaces	29
Altitude calculation	30
Volumetric flow rate calculation	31
Data logging	32
Circuit board	33
4.3 Testing and Results	35
<b>CHAPTER 5 CONCLUSIONS</b>	<b>36</b>
5.1 Conclusions	36
5.2 Challenges	37
5.3 Further Research	37
<b>APPENDICES</b>	<b>39</b>
Appendix A – Risk assessment	39
Appendix B – Control system software code	40
Appendix C – Control system software boot code	51
Appendix D – Ground test data	52
Appendix E – Vacuum chamber test data	54
Appendix F – Calibration output	58
<b>REFERENCES</b>	<b>59</b>

## LIST OF FIGURES

Figure 1: A Gantt chart showing the project timeline.	13
Figure 2: The Silverstone FHS 80X fan.	19
Figure 3: The Siemens VDO 5WK9605 MAF sensor.	20
Figure 4: The circuit diagram for a simple MAF sensor.	21
Figure 5: A standard 18650 lithium battery and a simple single cell charger.	23
Figure 6: The fan to MAF sensor adaptor and mount (CAD model).	24
Figure 7: The filter grille component (CAD model).	25
Figure 8: The electronics enclosure component (CAD model).	25
Figure 9: The battery holder mounting plate (CAD model).	25
Figure 10: The FDM 3D printer used to produce physical parts of the apparatus.	26
Figure 11: Microplastic fibres attached to an FDM 3D printed part.	26
Figure 12: The assembled apparatus with all 3D printed parts.	27
Figure 13: A chart comparing the piecewise formula with the actual relationship of P and H.	31
Figure 14: The piecewise formula used in the control system software.	31
Figure 15: The formula relating density $\rho$ (), pressure P (Pa) and temperature T (K).	31
Figure 16: The derivation of the formula used to convert MAF, P and T to VAF.	32
Figure 17: The sampled volume is determined using rectangular integration.	32
Figure 18: The circuit diagram of the control system.	34
Figure 19: The completed circuit board of the control system.	35



## NOMENCLATURE AND ABBREVIATIONS

Abbreviation	Technical term	Description
3D printer	Three dimensional printer	A computer-controlled machine which uses one of several methods to fabricate a part by adding material.
-	Aerosol	A mixture of small particles in suspension with a gas.
-	Arduino	A microcontroller development platform. Term describes the hardware device, the programming environment and the associated programming language.
CNC	Computer numerical control	A computer-calculated cutting path is applied to a machine tool by the control of motors which actuate the axes of the machine.
CSV	Comma separated variables	A computer file format which allows the simple storage of data separated by commas. This can be input to a spreadsheet software which automatically interprets the commas and line endings to place the values into cells.
FDM	Fused deposition modelling	A type of 3D printing technology. A filament of material (usually plastic) is melted and extruded onto the part being fabricated, where it solidifies.
HAB	High altitude balloon	Sometimes called a “weather balloon”. A large balloon made of rubber designed to reach high altitude. Filled with helium or hydrogen. Usually unmanned.
IC	Integrated circuit	A miniaturised electronic circuit contained within a casing and designed to be included into other higher level circuits.
MAF sensor	Mass airflow sensor	A sensor which uses an electrically heated wire grid. The sensor is cooled relative to the mass of air flowing through it, resulting in a measurable change in current draw.
-	Microlitter	Small pieces of human-made litter, but not limited by material. Can include microplastics, natural fibres, sawdust, soot etc.
-	Microplastic	Small pieces of plastic ranging up to five millimetres in length. Created either intentionally, as a byproduct of manufacturing processes or through wear or degradation to plastic materials during their service life or after disposal.
PHT sensor	Pressure, humidity and temperature sensor	A combined sensor which measures pressure, humidity and temperature of the surrounding air.
PWM	Pulse width modulation	A method of controlling electrical devices where variable length pulses of electrical current are supplied. This allows the speed of motors to be controlled.
SLA	Stereolithography	A type of 3D printing technology. A liquid resin is exposed to ultraviolet light in a specific pattern created by an LCD screen for each layer. The resin hardens where exposed to the light, allowing complex 3D geometry to be fabricated.

# CHAPTER 1

## INTRODUCTION

### 1.1 Outline

This dissertation presents the design process for an apparatus which may be used in further research into the occurrence and distribution of airborne microplastics in the atmosphere, as well as the development of models to understand the atmospheric transport mechanism of microplastics. The design constraints involved are also discussed, and the apparatus developed for prior research efforts in this field are analysed.

### 1.2 Introduction

Microplastics are pieces of plastic less than 5mm in size. While sometimes manufactured at this size, they are also often generated from the breakdown of larger plastic objects in the environment, from wear of plastic products in use or as waste from manufacturing processes. Due to their small size, they are easily able to be carried through natural processes across vast distances. First identified in oceans in the 1970's, they have now been discovered in various locations which one would expect to be pristine or sterile, such as remote polar regions (Kelly et al. 2020) and even the human bloodstream (Leslie et al. 2022).

### 1.3 The Problem

In 2015 the transport of microplastics through the air was first identified (Dris et al. 2016). The literature in this space is quickly evolving, but a lack of standardisation in methodology and apparatus is proving to be a challenge in establishing a consensus on the extent of the problem (Rosso et al. 2023). Every study requires the researcher(s) to design their own apparatus and consequently the results are not easily comparable. For example one study may include microplastics down to 1 micron while another may find a lower rate of occurrence because it only considers microplastics larger than 10 microns.

### 1.4 Aim

This dissertation presents the design process for an apparatus which could be used in ongoing research of microplastics in the atmosphere. As well as the operation of the apparatus in collecting atmospheric samples, the design will consider the contamination-free requirements of the preparation of the apparatus and the requirements of the sample processing required to obtain useful results.

## 1.5 Conclusions

The apparatus was designed, constructed and tested both at ground level atmospheric pressure and in a vacuum chamber to provide the reduced pressure seen during a high altitude balloon flight. The apparatus hardware and software performed as expected, automatically triggering the sampling routine, collecting a sample and logging data accordingly. The design shows considerable promise for use in atmospheric microplastics research, although more testing is recommended.

The physical design of the apparatus allowed easy fabrication and assembly using only a basic 3D printer and screwdrivers. The circuit board required some specialised skills due to the prototype nature of the project, but could easily be simplified if more examples were required for atmospheric microplastics research. The electronic components were easy and quick to source online for a total cost of \$139.79 including shipping. The cost and sourcing of parts would not inhibit the deployment of this design. The completed apparatus weighed 628g, light enough to easily fly on a high altitude balloon and accommodate modifications or additional payloads.

## CHAPTER 2

### PROJECT PLANNING

#### 2.1 Aims

This project aims to design, implement and test an apparatus which can draw a measured volume of air through a filter to capture a sample of microplastic fibres and particles for research. The apparatus must be carried by an unmanned high altitude balloon to collect a sample at a specific altitude. To encourage repeated studies in the current replication crisis (Wilson 2022), the apparatus must be easy to implement by researchers from many backgrounds in any global region.

#### 2.2 Objectives

In order to be suitable for its intended use, the apparatus must have a filter capable of collecting a sample of microplastics without contaminating the sample with microplastics originating from the apparatus itself. To this end, no plastic parts should be placed upstream of the filter and the filter must be entirely free from plastic. It is also imperative that the apparatus be light enough to be carried by a high altitude balloon.

The apparatus must employ a suitably powerful fan to draw air through the filter, with a digital control system using feedback from an airflow sensor to control the total volume of air sampled. A

It must also be designed to be affordably constructed. Therefore it should use affordable materials and components and be produced and assembled using affordable tools. To make the apparatus easy to implement, parts should be easy to source and all steps of constructing and setting up the apparatus should be as easy as possible.

To test the operation of the apparatus, it will be run in free air to collect a sample. Another test will be run with the apparatus within a vacuum chamber to test the automatic triggering and sampling performance with reduced pressure as seen in the upper atmosphere.

## 2.3 Timeline

The project was planned to follow the timeline set out in the Gantt chart in Figure 1.

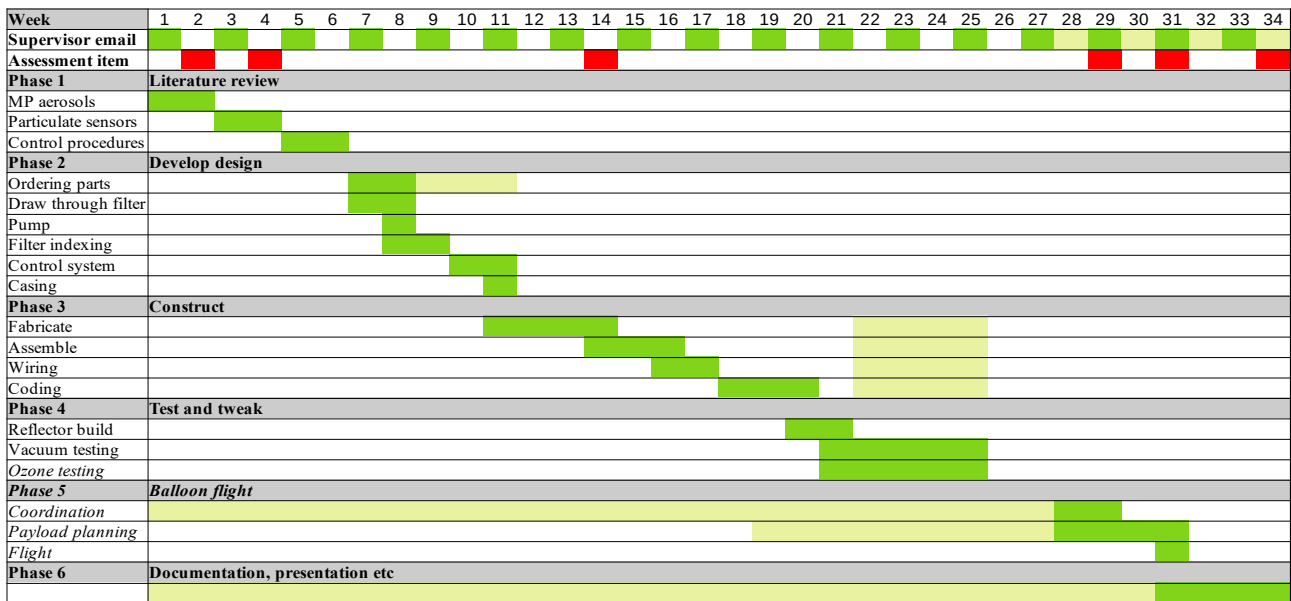


Figure 1: A Gantt chart showing the project timeline. Pale green shows allowable overruns and pre-planning tasks.

### 2.3.1 Risk Assessment

A risk assessment was conducted to determine the risks associated with this research work. This is attached as Appendix A

### 2.3.2 Environmental Sustainability

By the nature of this apparatus design, it is expected that the environmental impact of the device itself will be insignificant. Plastics must be avoided where possible in the design and the apparatus must be recovered to assess the results, so it is unlikely that any litter or microlitter will be generated.

For a high altitude balloon launch there is necessarily some environmental impact, as a substantial volume of lifting gas must be released and the large balloon made of latex rubber ruptures before the descent. There are two options for lifting gas. Helium is non-flammable and doesn't contribute to global warming, but is also a non-renewable fossil resource of which the earth has a very limited supply. Hydrogen is highly flammable and has a low global warming potential of 5.8 (MIT 2006), but is renewable when combusted. Being lighter than helium, hydrogen gas allows a smaller balloon to provide the same buoyancy. This in turn allows a launch to be conducted with a lower input of latex, a natural polymer produced from tree sap which is farmed in vulnerable rainforest environments. For these reasons, hydrogen would be the preferable lifting gas if safety requirements regarding its high flammability would allow. To further improve the sustainability of hydrogen as a lifting gas, an ignition source could be provided prior to the balloon rupture to combust the gas into water vapour, as unburnt hydrogen has a low global warming potential and can escape the atmosphere into space as helium does.

### 2.3.3 Project Disposal

After the research project was completed, the prototype was disassembled to avoid any consequential effects of its misuse. Notes have been embedded in all code and CAD models to ensure that the imperfect state of the project is clear to anyone who may attempt to use components of it.

## CHAPTER 3

### LITERATURE REVIEW

#### 3.1 Introduction

This literature review examines the literature on airborne microplastics, with a focus on apparatus design for the collection of atmospheric samples. Particular attention is paid to the sensors and control systems used in such apparatus. To understand the context of the apparatus design, the literature on processing of microplastic samples is also reviewed.

#### 3.2 Microplastic Aerosols

The morphology of microplastics varies widely, but they can be divided into two main categories, particles and fibres (Conkle, Baez Del Valle & Turner 2018). Particles can take on a myriad of shapes. While fibres can contain any number or type of bends, the cross section of synthetic fibres tends to be uniform along their length. Sizes range from 5mm down to undetectably small as further breakdown takes place in the environment. Nanoplastics are microplastics in the sub-micron range (Gigault et al. 2018). Research into nanoplastics is extremely limited due to the difficulty of processing these samples. This is discussed further in section 3.3.3. The concentration of airbourne microplastics in the atmosphere is roughly in the range of one to more than 1000 parts per cubic metre (O'Brien et al. 2023).

#### 3.3 Apparatus Design and Testing Procedures

##### 3.3.1 Particulate Sample Collection Configuration

Details of apparatus designs specifically suited to the collection of particulate samples for the study of airborne microplastics are lacking in the literature. Habeck, Flaten & Candler (2020) described the design of an apparatus for measuring the concentrations of particulate in flight, where no sample is collected, for their study relating to the aerodynamics of hypersonic aircraft. Lateran et al. (2016) used a small vacuum pump to draw air through an Andersen cascade sampler to collect a particulate sample on several adhesive surfaces. The Andersen cascade sampler separates particles by size during sample collection by varying the airflow velocity in each chamber. No feedback system was used to determine the airflow. Bryan et al. (2014) used an off the shelf pollen sampling system in their high altitude balloon microbial study. Fonseca et al. (2003) used an industrial MAF sensor for feedback in an analog control system circuit to provide a consistent mass flow rate.

### 3.3.2 Control System

Previous control systems for atmospheric particulate sampling apparatus designs by Lateran et al. (2016) and Bryan et al. (2014) have used the Arduino Mega 2560. This is a development platform based on an 8 bit Atmel ATmega microcontroller. This popular but dated platform was the norm at the time, but the recent literature indicates that better options are now available. Habeck, Flaten & Candler (2020) used a Raspberry Pi in their study of atmospheric aerosol droplets. Anderson et al. (2023) developed a balloon based system for monitoring wildfires using a Raspberry Pi Pico W for the control system, citing its “accessibility, inexpensive cost and adaptability”.

Single use lithium AA batteries (Bryan et al. 2014), lithium ion batteries (Habeck, Flaten & Candler 2020) and have been successfully used in similar high altitude apparatus designs in the past. Lateran et al. (2016) used lithium polymer batteries but housed them inside a foam descent glider which was sealed to provide insulation.

### 3.3.3 Microplastic-Specific Procedures

Binderless glass fibre filters are usually used. These filters are made from glass fibres which are joined together by melting so that no binding agent is required. This construction means they contain no plastic and allows them to be heated to just below their glass transition temperature to burn away any microplastics or residue which may be present before samples are collected (Song et al. 2021).

Samples are manually processed to determine the results. This process is very laborious, as samples must be examined under a microscope to find all particles and fibres (Beaurepaire et al. 2021). Each piece of microlitter must then be examined and tested in various ways to determine if it is synthetic or natural in origin. One such test involves holding a heated needle adjacent to a fibre, as synthetic fibres will curl toward the heat source and natural fibres will not (Beckingham et al. 2023).

## 3.4 Summary

A knowledge gap has been identified in the application of a control system for an atmospheric sampling apparatus with a low barrier to entry. Previous apparatus designs have used the Arduino platform, which requires code to be written in the C++ programming language and flashed to the controller using specialised software. Control systems based on a Raspberry Pi have been used, which is a complete computer requiring considerable setup and more susceptible to unpredictable performance than a microcontroller. Analog control systems have also been used, which require specific design for the exact use case and cannot be easily modified.

While MAF sensors have been used for atmospheric sampling apparatus, a knowledge gap has been identified in the use of a MAF sensor in a digital control system for metering of known air quantities. Another gap exists in the use of an automotive MAF sensor, which is considerably more affordable and accessible than industrial instruments.

The most notable knowledge gap is a detailed description of a particulate sampling apparatus suited for the study of atmospheric microplastics.



## CHAPTER 4 METHODOLOGY

### 4.1 Design requirements

#### 4.1.1 General requirements

Some design requirements were determined in order to ensure the apparatus would be capable of collecting useful samples, and be as accessible and replicable as possible for researchers from any region and background. These requirements are organised by the aspect of the design they relate to.

All components should be affordable and readily available, and wherever possible conform to a standard which is widely used by multiple manufacturers. This provides the ability to source equivalent components from many avenues and provides redundancy if one manufacturer were to stop producing a particular component.

To reduce the impact of limitations in available skilled trades, any custom designed components should be able to be produced with standard digital fabrication techniques such as 3D printing, CNC laser cutting and CNC machining.

The apparatus should be as light as possible to limit the required lifting gas in the balloon and maximise the altitude reached. Some jurisdictions also place weight limits on balloon launches, so adherence to international standards is important for the wide applicability of the apparatus. In the USA this is 6lbs (2.7kg) for a single payload (StratoStar 2022).

#### 4.1.2 Sample collection filter

The sample collection filter should be capable of capturing solid particles and fibres down to a reasonable size. As the processing of these samples involves manual sorting of microlitter piece-by-piece to differentiate microplastics from other microlitter, it is impractical to attempt to process any components of a sample below around 2 microns in size. Therefore a filter should be used which can capture particles down to below 2 microns. Attention should be paid to the rated flow rate and corresponding pressure drop of the filter to ensure that the air movement device is rated for both a flow rate and static pressure to suit.

#### 4.1.3 Physical design

To ensure that no microplastics generated by the apparatus itself find their way onto the sample collection filter, no plastic-containing materials should be used upstream of it. Any plastic parts produced for the apparatus should use a distinctive colour which allows them to be easily identified if they happen to contaminate the results.

#### 4.1.4 Electronic hardware

##### Fan

A fan or other air movement device must be specified to draw air through the sample collection filter. The selected air movement device must provide a sufficient flow rate to collect a statistically significant number of microplastic fibres/particles over a reasonably short range of altitude. As the average ascent rate of a high altitude balloon is often around 300m/min and the expected occurrence of microplastic fibres/particles is in the order of <1 to 1000 parts per cubic metre (O'Brien et al. 2023), a flow rate of 0.3 cubic metres per minute will allow a sample of roughly <1 to 1000 fibres/particles to be collected over an altitude range of 1,000m.

##### Airflow sensor

The volume of air which is drawn through the filter by the fan must be controlled, requiring an airflow sensor to provide feedback. This must work at a range of air pressures down to near-vacuum.

##### Pressure sensor

A pressure sensor will be required to act as an altimeter to track the altitude of the apparatus during its flight. This can also be used to keep track of the ascent rate of the balloon and return to a low power state after landing to keep the device transmitting its beacon as long as possible to assist in researchers recovering the apparatus.

##### Battery

The apparatus will need to be powered by a battery. This battery must be capable of withstanding the low temperatures and pressures at high altitude, and powering the apparatus for at least three hours with several minutes of continuous fan operation during sampling. A simple and affordable charging method is necessary as well.

##### Microcontroller

A microcontroller is required to read the sensors, record values and trigger the sample collection at the appropriate altitude. Keeping this as simple and accessible as possible is important, as researchers should be able to easily change the sampling parameters or reprogram the control system to add extra sensors with the lowest possible barrier to entry.

#### 4.1.5 Software control system

There are several requirements which must be met by the control software design. It must use the pressure sensor data to determine its altitude. It must be capable of using the MAF sensor signal to determine the volumetric flow rate of air through the filter and therefore determine the total volume collected during a sampling operation. It must also record data to complement the samples and to verify that the samples were collected correctly.

## 4.2 Design Implementation

### 4.2.1 Sample collection filter

Best practice in this field of research is to use a binderless glass filter which can be both washed and heated in a kiln to vaporise any present microplastics before a sample is collected (Song et al. 2021).

Unfortunately the rated flow rate and corresponding pressure drop of the filter could only be found for water (Finetech Research and Innovation 2023), so while the pressure drop observed will be lower with the lower viscosity of air, no exact figures are available. This makes closely matching an air movement device difficult, and instead some assumptions must be made and the fan is likely to be more powerful than required, resulting in a weight penalty.

### 4.2.2 Electronic hardware

#### Fan

It was determined that the most reliable way to source a low-cost device for drawing air through the filter would be to utilise a computer cooling fan. These components are available at a very low price point, are energy efficient and include advertised ratings for air flow rate and static pressure which will assist in specifying a fan to match the characteristics of the sample collection filter.

The selected fan is the Silverstone FHS 80X (Figure 2), a powerful fan designed for computer servers with a rated airflow of 83.66 cubic feet (2.369 cubic metres) per minute and rated static pressure of 50.77mmH<sub>2</sub>O (SilverStone 2023). While no empirical data is available on the relationship between flow rate and pressure, these figures suggest the fan's performance is approximately correct for this application.

The fan accepts a speed control signal which is a simple 5V pulse width modulation (PWM) control. As the fan is intended for continual cooling use in a computer server, the PWM control is ignored by the fan's internal control circuit when a PWM signal below 20% is received, resulting in a minimum speed still being provided as long as the fan is powered. As such, circuit elements to switch the 12V power to the fan was also included on the circuit board.



*Figure 2: The Silverstone FHS 80X fan.*

## Airflow sensor

The most basic type of airflow sensor is an anemometer, which uses a small turbine which turns due to the flow of air over its blades. These sensors are not designed to work at reduced air pressure and would likely be inaccurate if employed for this apparatus.

When selecting a solid state airflow sensor to provide feedback to the control system, cost is a major concern. Industrial-grade flow sensors are available, but at prices (RS Group 2023a) which could discourage the replication which is necessary for atmospheric microplastics research to yield definitive far reaching conclusions. The decision was made to use a cheap and widely available MAF sensor instead. While in the automotive industry these are commonly called mass airflow sensors, the same operating principle is used in industrial control components where they are usually referred to as thermal dispersion flow switches (RS Group 2023b). A Siemens VDO 5WK9605 MAF sensor (Figure 3) was chosen, as it is available as a replacement part for a wide range of BMW and Hyundai two to three litre engines which have been sold around the world. This makes it an easy part to source from countless suppliers in any region long term. While the sensor has a 3 pin plug connection, the cable connector for this could not be readily sourced on the required timeframe of this project, so the device was opened and a standard pin header connector was added. The cover for the electronics was fixed back into place using thermoplastic welding.



*Figure 3: The Siemens VDO 5WK9605 MAF sensor.*

This type of MAF sensor uses a resistive heating element and thermistor in a balanced circuit which inherently controls for air temperature. A circuit diagram of this layout is shown in Figure 4. Therefore the only factor which influences the output of the circuit is the mass airflow past the heating element. As the apparatus control system requires feedback for the volumetric airflow, this can be calculated using the mass airflow and the air pressure signal provided by the pressure, humidity and temperature (PHT) sensor.

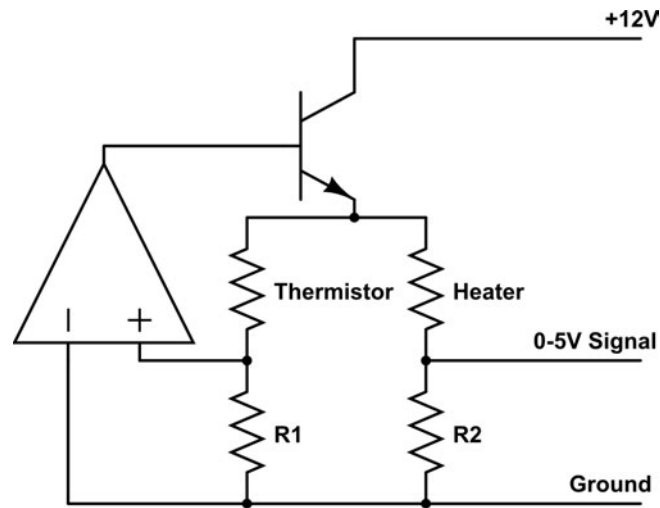


Figure 4: The circuit diagram for a simple MAF sensor.

The selected MAF sensor is powered by 12V and provides an internally-amplified 0-5V analog output. This can be read using the internal analog-to-digital converter (ADC) in the Raspberry Pi Pico W. The Raspberry Pi Pico W runs at a 3.3V logic level, meaning that it should not receive a signal of 5V. In testing it was found that the 0-5V analog signal actually never exceeds 3.3V even with directed streams of cold gas, so no level shifting circuit is required. Due to the heating element in this sensor, circuit elements were included on the circuit board to allow the sensor to be turned off when not in use.

## Pressure sensor

The apparatus control system needs to be able to measure its altitude as well as the surrounding air pressure. Altimeters generally function by measuring the air pressure, which decreases exponentially as the altitude increases. While small changes in air pressure can result from weather, these changes only translate to a minimal shift in the indicated altitude of a few hundred metres at most. As the various attributes of air are commonly used in conjunction with each other, digital sensors designed to collect this information are often offered as a combined pressure, humidity and temperature (PHT) sensor. The added data points of temperature and humidity may also be helpful to researchers and add very little overhead to collect. Individual pressure sensors and digital altimeters are also available.

While a very wide range of pressure sensors, PHT sensors and digital altimeters are on the market, only a select few are capable of being used for this apparatus due to the need for operation at very high altitude (and hence very low pressures). When calculating based on the minimum rated pressure, most available sensors were found to have a maximum usable altitude of 60,000ft (18.3km), but this design must measure to near the maximum altitude of a high altitude balloon, roughly 100,000ft or 30km. The sensor chosen was a TE Connectivity MS8607 PHT sensor, which provides pressure readings down to 10mbar, extending the usable altitude to just above 100,000ft (30.48km) (TE Connectivity 2023). The MS8607 communicates over an I2C bus, which is standardised, making replacement of this component with a different model due to sourcing difficulties relatively straightforward. While this is only an 8 pin integrated circuit (IC), it is a small surface mount package, so a commonly available MS8607 breakout board from Adafruit (Adafruit 2023a) was used to reduce assembly complexity by providing standard 0.1" spaced pins. This

sensor uses very low power so it can be powered from a digital output pin of the Raspberry Pi Pico W directly, allowing it to be turned on or off in software if required and removing the need for the battery system to consider this sensor's voltage requirements.

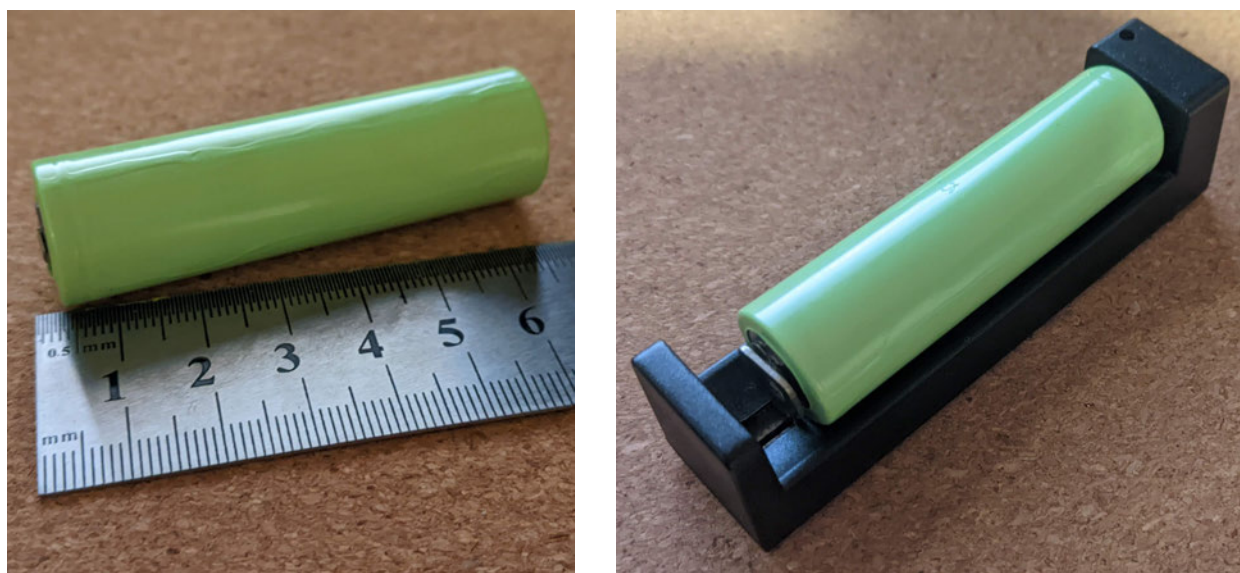
## Batteries

While batteries are available in various chemistry designs, lithium batteries are the highest performing in most categories. Lithium batteries offer the best energy density, meaning longer battery life for a given weight. They also offer the best power density, meaning the most powerful equipment can be powered for a given weight. Lithium batteries also work in the lowest temperatures, although still not as low as those observed at high altitude. While often grouped into a single category, there are various chemistry designs based upon lithium. Lithium ion is the most common, while lithium polymer offers higher energy density and power density at the expense of thermal stability and long term endurance. Lithium iron phosphate batteries are the most stable and have the longest life cycle, but with a weight penalty. Lithium titanate batteries are often overlooked and rarely used due to their reduced energy density, but they offer very high power density and extremely fast charge times. They also offer a feature for which there is limited demand, stability in very low operating temperatures (Gree Altairnano New Energy 2023). Lithium titanate batteries are very well suited for high altitude use, but their unique chemistry and lack of popularity introduce some issues. Unlike lithium ion and lithium polymer batteries which operate in the range of 3.0V-4.2V per cell, lithium titanate batteries operate at within a voltage range of 1.8V-2.85V (Cadex 2023). Many battery chargers, battery management systems and battery charging ICs are specifically designed for the voltage limits of lithium ion cells. This makes sourcing equipment compatible with lithium titanate cells difficult, even at the component level. The cells themselves could also prove difficult to source in remote locations or regions with difficult customs requirements. The decision was made to use lithium ion batteries for their ease of sourcing and wide compatibility. The substitution of lithium titanate batteries could be proven necessary by further low temperature testing, but lithium ion batteries have been used successfully (Habeck, Flaten & Candler 2020).

Lithium battery cells are available in two main packaging types. Pouch cells have the thin film-like material which makes up the cell folded into layers within a foil pouch. Cylindrical cells are rolled into a cylinder within a metal casing similar to a common AA cell. The foil pouch for which the former is named provides effectively zero structural support to the cell, making it prone to damage. An internal short circuit can result from crushing or puncturing of the cell often leading to thermal runaway, a condition in which the battery will create a toxic, self-sustaining fire which destroys the battery and can easily damage surrounding equipment. In the case of an unmanned balloon flight, this would also create a risk of wild fires. While these risks still exist with cylindrical cells, their metal casing provides considerable added ruggedness. Another consideration is that all lithium batteries produce some gas while charging or in use. Pouch cells accommodate this gas internally, eventually inflating toward the end of the cell's life, while cylindrical cells have a one-way vent to allow the gas to escape. It is anticipated that under the partial vacuum of high altitude flight, the batteries will exhibit some additional off-gassing. In the case of the pouch cell, this gas could inflate the pouch to the point that the cell within delaminates. The cylindrical cell's metal casing on the other hand contains the cell material and any gas generated is incapable of separating the layers. Cylindrical cells are also available in standardised sizes. The most common of these is the 18650 (Figure 5), a cell 18mm in diameter and 65mm in length. These cells were widely used within



laptop batteries of years past and are still the dominant form factor for cordless power tool and vacuum batteries, USB power banks and electric vehicle batteries. In recent years, the use of 18650 cells in their standalone form has been gaining traction, particularly for flashlights and electronic cigarettes. Chargers designed for one or more individual cells are now widely available. Multiple cell holders are available from most electronics suppliers, providing a convenient method for interfacing them with the apparatus. These cells and their associated accessories are also very competitively priced. It was decided to use cylindrical 18650 cells for the apparatus due to the various advantages discussed above.



*Figure 5: A standard 18650 lithium battery and a simple single cell charger.*

A single cell lithium battery provides between 1.8V and 4.35V, depending on the specific chemistry and the state of charge. The fan and MAF sensor both require 12V power while the microcontroller requires 1.8-5.5V. Any type of single cell lithium battery could directly power the microcontroller, with a boost converter used to produce the 12V power required by the fan and MAF sensor.

Alternatively, multiple cells could be connected in series to provide 12V to the fan and MAF sensor and a linear voltage regulator or buck converter could be used to drop this to 5V. Separate batteries for these different devices within the apparatus could be used, but it was determined that this should be avoided, as it may falsely give the appearance that the apparatus is functioning, causing it to be launched without having the capability to power the fan when it eventually reaches the sampling altitude. As the microcontroller draws little current while the fan draws considerably more, it was decided that it would be more efficient to convert from a 12V battery down to 5V for the microcontroller. The conversion from a higher voltage to a lower one can also be done with solid state components, while increasing the voltage commonly require electrolytic capacitors, which are unlikely to give reliable performance at very low temperatures and pressures.

Although lithium titanate cells get close, no commonly available battery is capable of operating reliably at such extreme low temperatures as observed at high altitude. Therefore it was determined that the best solution would be an insulated battery enclosure as used by Lateran et al. (2016). The most appropriate material to achieve this cheaply would be mineral insulation wool, as it offers good thermal insulation performance and is available around the world at a reasonable price. If further testing shows this to be inadequate, a suitable dust-free aerogel product is available (Aerogel Technologies 2023), providing aerospace-level performance although at a relatively high price.

### 4.2.3 Physical design

As 3D printers are widely available to researchers and have low barriers to entry in both cost and skill requirement, it was decided that 3D printing should be used where practicable. As 3D printing requires the use of plastic and can create microplastics, it was decided that laser cut wood should instead be used in any location upstream of the sample collection filter. The apparatus should also be oriented to draw air in the downward direction so that any microplastics shed by the downstream components of the apparatus are less likely to be swept toward the sample collection filter as the apparatus rises through the air.

The general layout of the fan and MAF sensor were noted and measurements of key features were taken using vernier calipers. Various parts were then designed in 3D CAD using Autodesk Fusion 360. This software was chosen as at the time of writing it is provided free to students and hobbyists, and has a fairly affordable institutional licensing structure. It is noted that designing the part in OpenSCAD would allow an entirely free and open source software to be used, and for key dimensions of parts to be simply modified by changing parametric variables. OpenSCAD does however have a steep learning curve which may in fact reduce accessibility of the design.

A part to mate the fan and MAF sensor and smoothly duct air between them was designed, shown below in Figure 6. Fillets were included where appropriate to maximise the strength of of this part.

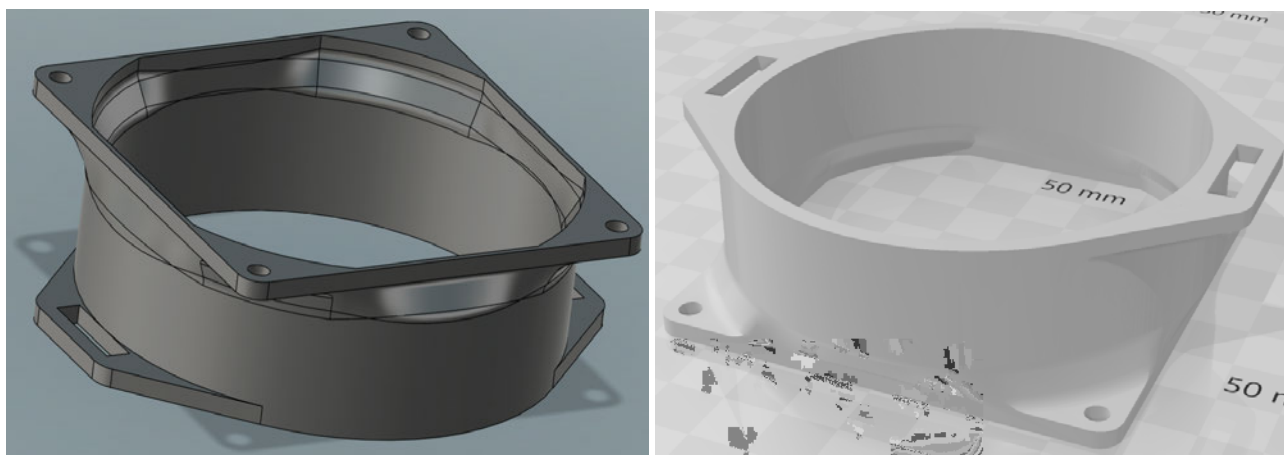
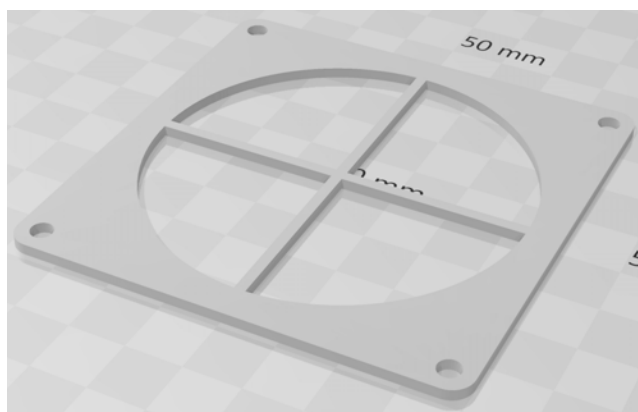


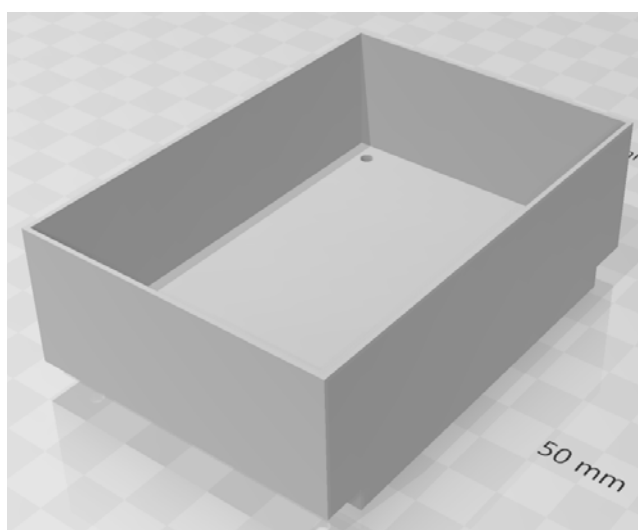
Figure 6: The fan to MAF sensor adaptor and mount (CAD model).

A grille to support the sample collection filter was designed for the other side of the fan (Figure 7). An enclosure for the electronic control system circuit board was also modelled (Figure 8). A mounting plate to fix the battery holder to the fan and MAF sensor assembly was also designed (Figure 9). A step in the base of the electronics enclosure and battery holder mount was included to allow it to mate with adhesive against the flat face of the fan and clear the wider round section of the fan to MAF sensor adaptor. While these parts could have been designed to mount with screws to the flanges of the fan, this would require threaded rods and nuts to be acquired in addition to the screws included with the fan. Retaining a large flat section at the lowest point of the electronics enclosure also allows it a large area to adhere to the 3D printer's bed. Challenges with print adhesion are common and considering this in 3D printed part design is considered best practice, especially when accessibility and replicability are priorities.

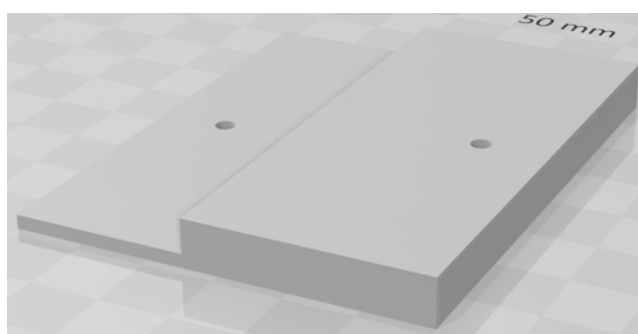




*Figure 7: The filter grille component (CAD model).*



*Figure 8: The electronics enclosure component (CAD model).*



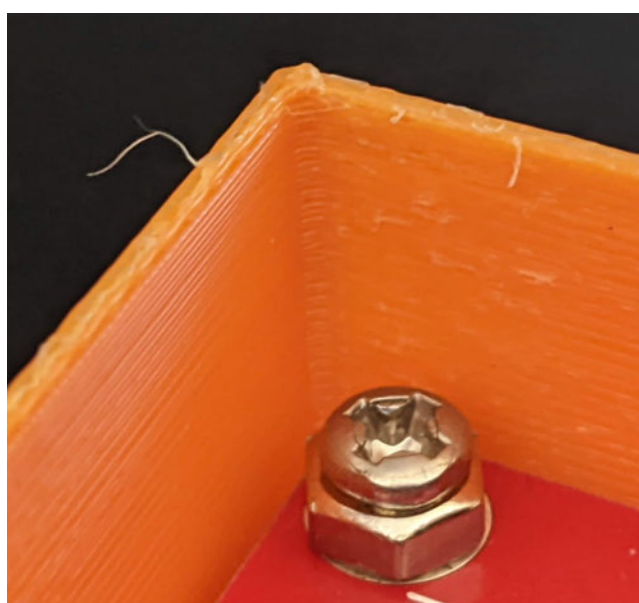
*Figure 9: The battery holder mounting plate (CAD model).*

These parts were then all produced using a simple fused deposition modelling (FDM) 3D printer, shown in Figure 10. This is a particularly low cost example which can be considered a worst case for research environments. This type of 3D printer is the most affordable and widespread, using cheap plastic filament which is extruded through a heated nozzle to build up parts layer-by-layer by tracing a path with a thin stream of melted plastic. Because the stream is thin and must start and stop, thin fibres of plastic hanging from the finished part are a common occurrence (Figure 11), and have the potential to contaminate samples. Orange filament was chosen for its high visibility when

searching for the landed apparatus in the natural environment after a flight. Orange is also helpful as its distinctive colour will allow researchers to infer the origin of any microplastic contamination generated by the apparatus itself. Phosphorescent filaments are also available and may provide even better distinction from atmospheric microplastics. If issues with contamination are detected, this part can also be produced using a 3D printer of the stereolithography (SLA) type. This technology instead forms parts by selectively solidifying a light-reactive resin into a single layer at a time. Because the part is formed as a monolithic piece of solidified resin, it is much less likely to become a source of microplastic contamination.



*Figure 10: The FDM 3D printer used to produce physical parts of the apparatus.*



*Figure 11: Microplastic fibres attached to an FDM 3D printed part. M3 screw for scale.*

The apparatus was then assembled by connecting the electronic components to the 3D printed parts. The fan included self-tapping screws which pass through the 3D printed parts and thread into its plastic mounting flanges. The MAF sensor slips inside the adaptor mount and is secured with cable ties through the two mounting flanges. The battery holder mounting plate and electronics enclosure are mounted with adhesive to the flat faces of the fan. A ruler is visible in the bottom right corner of the top image, showing measurements from 18 to 23 cm.



*Figure 12: The assembled apparatus with all 3D printed parts.*

## 4.2.4 Control system

### Microcontroller

The control system for the apparatus was designed around a Raspberry Pi Pico W microcontroller board. This device has also been used by Anderson et al. (2023) on a balloon sensor system. This device uses a modern 32 bit ARM CPU architecture with two cores running at 133MHz and two megabytes of onboard flash storage (Raspberry Pi Foundation 2023). Various data buses and an internal ADC are available for interfacing with sensors. The Raspberry Pi Pico is available in a regular or “W” variant, with the W including WiFi and Bluetooth Low Energy. While neither of these communications standards have a range useful for communicating with the apparatus in flight, a WiFi signal could potentially be used as a short range beacon to help locate the apparatus upon landing. The Raspberry Pi Pico microcontroller range have support for compiled C/C++ code or the interpreted MicroPython or CircuitPython languages.

The Raspberry Pi Foundation has committed to continue producing the Pico range until 2034 (Raspberry Pi Foundation 2023), which makes this microcontroller a sensible choice to give the apparatus long-term serviceability and replicability.

### Programming language

It was decided to program the control system using the CircuitPython interpreted programming language, which is provided as open source under the MIT License (CircuitPython 2023) by Adafruit Industries. This has several distinct advantages over the other offerings.

CircuitPython is based upon the Python programming language. Python is used widely in data science and research applications, and is quickly overtaking alternatives. Python has many advantages which make it easy to use, such as allowing the implicit declaration of variables. CircuitPython adds to this the ability to ignore data types in most circumstances. The result is a programming language which lends itself to use by those without a background in computer science or software engineering.

The CircuitPython interpreter program consists of compiled code which runs on the microcontroller and executes CircuitPython code by interpreting it in real time. The interpreter also provides USB communications to a file system on the microcontroller’s internal two megabyte flash storage. This allows the control system software to be accessed for editing in the same way as a text file on a regular USB flash drive. The control system software can then also write data log files to this file system to be accessed in the same manner.

Adafruit Industries, a manufacturer and distributor of hobby electronics, makes available a very large selection of open source hardware libraries under the MIT License. These libraries make interfacing with hardware devices, often sensors and actuators, much simpler and faster to implement. These libraries are available in C/C++ and their own CircuitPython, but not MicroPython. While a library can be translated to run in MicroPython, using CircuitPython for the control system opens the possibility of easily integrating a new library into the code to change the pressure sensor for a different model or to add additional sensors to suit researchers’ needs.

These features allow for extreme accessibility for the use and modification of the apparatus by researchers with all levels of computer literacy. The ability to edit the control system code without any specific computer software is very convenient in institutional environments where not all users have the ability to install software on the computer system, and not all computers on the system will have a specific software installed.

## Software code structure

The general layout of the control system software is that of a state machine. A state variable is used to store the current mode of operation, or state. One state is applied to each phase of the sample collection flight: pre-launch, ascent, sampling, descent and recovery. Data is logged during all states except for the recovery state, and energy efficiency is the main priority in all states except for the sampling state. The loop speed of the state machine is controlled independently by the current state. This allows fast looping for accurate measurements in the sampling state and increased efficiency with slower looping for the other states.

While delay statements in code are generally to be avoided, in the case of the CircuitPython language, they are a convenient method to put the microcontroller into a low power sleep state. This battery saving technique is preferable in most of the states of this apparatus' functionality, as it waits for the launch to take place, ascends, descends or attempts to stay powered while waiting for researchers to recover it from its landing site. Only when sampling does the loop speed need to be increased and the loop time measured accurately. To save further power, the MAF sensor is turned off when not in use, as it includes a heating element as part of its design. When a sample is to be collected, a short delay is provided to allow the sensor to heat up and stabilise before the fan is turned on.

Functions are used for obtaining sensor readings and to send control outputs. This allows the main program to be concise and easy to interpret by others. This modular approach also makes it easier to change hardware while retaining the layout of the software code.

## Hardware interfaces

The output of the MAF sensor is an analog signal in the range of 0-5V being linearly related to the mass airflow. This 0-5V analog signal is common for automotive sensors. This signal is read using the internal ADC in the Raspberry Pi Pico W. While the ADC is only rated for 0-3.3V, early testing of the MAF sensor showed that its actual output range ever exceeded 3.3V, so no level shifting circuit has been included on the circuit board.

The fan accepts a PWM speed control signal, but to completely stop the fan its power must be cut. These two actions are performed by different pins, so a function has been written to ensure that a zero speed command is always accompanied by cutting the power and a non-zero speed command is always accompanied by restoring power. All control of the fan is then done through this function.

The PHT sensor communicates with the Raspberry Pi Pico W through an I2C bus. The communication protocol is somewhat complex, but Adafruit provides a library (Adafruit 2023b) under the MIT License which can be used to simplify the communications. This also allows a replacement sensor to be used if the MS8607 is taken out of production in future, by switching out the library for one compatible with the alternate sensor.

## Altitude calculation

To use the pressure sensor to find altitude, a mathematical formula needed to be found which reflects the relationship between altitude and air pressure. This formula can be found quite easily from various sources including The National Weather Service (National Weather Service 2023), but this relationship begins to diverge from available data (Table 1) at high altitudes within the stratosphere. This is likely because the formula is only intended to be applied at regular manned flight altitudes within the troposphere. To accurately represent the relationship at high altitudes, a formula was derived from the data in Table 1 (The Engineering ToolBox 2003) by entering the values into a spreadsheet, applying a trend line and accessing the formula for the trend line.

Table 1: U.S. Standard Atmosphere vs. Altitude (The Engineering ToolBox 2003)

Altitude (m)	Temperature ( $^{\circ}\text{C}$ )	Absolute Pressure (mbar)	Density ( $\text{kg}/\text{m}^3$ )	Dynamic Viscosity ( $10^{-5} \text{ N s}/\text{m}^2$ )
-1000	21.50	1139	1.347	1.821
0	15.00	1013	1.225	1.789
1000	8.50	898.8	1.112	1.758
2000	2.00	795.0	1.007	1.726
3000	-4.49	701.2	0.9093	1.694
4000	-10.98	616.6	0.8194	1.661
5000	-17.47	540.5	0.7364	1.628
6000	-23.96	472.2	0.6601	1.595
7000	-30.45	411.1	0.5900	1.561
8000	-36.94	356.5	0.5258	1.527
9000	-43.42	308.0	0.4671	1.493
10000	-49.90	265.0	0.4135	1.458
15000	-56.50	121.1	0.1948	1.422
20000	-56.50	55.29	0.08891	1.422
25000	-51.60	25.49	0.04008	1.448
30000	-46.64	11.97	0.01841	1.475
40000	-22.80	2.87	0.003996	1.601
50000	-2.50	0.7978	0.001027	1.704
60000	-26.13	0.2196	0.0003097	1.584
70000	-53.57	0.052	0.00008283	1.438
80000	-74.51	0.011	0.00001846	1.321



It was observed that no standard trend line would fit the relationship closely enough, so a piecewise formula (Figure 14) was instead used, combining the low altitude formula from The National Weather Service and the formula derived from the table which is accurate for high altitudes. The outputs of these two sub-functions are plotted in Figure 13.

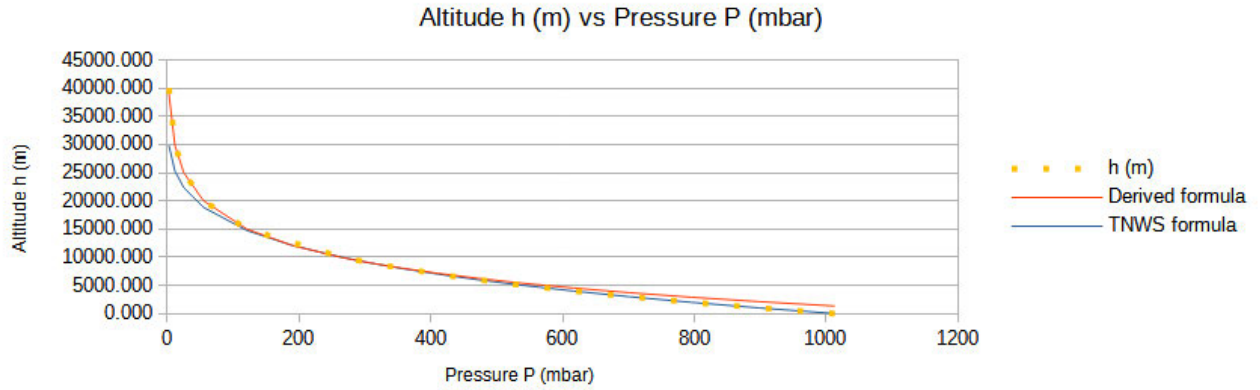


Figure 13: A chart of the actual relationship between pressure and altitude compared with the two sub-functions of the piecewise formula.

The piecewise formula should cross over at the point where the two formulas intersect, so as to not have a sudden change in calculated altitude. This point was found using the spreadsheet. The complete piecewise formula for calculating altitude from air pressure is shown in Figure 14.

$$h = \begin{cases} 44307.69 \left( 1 - \left( \frac{P}{1013.25} \right) \right)^{0.190284} & P > 189.94 \\ -6417.004605 * \ln(P) + 45755.99158 & P \leq 189.94 \end{cases}$$

Figure 14: The piecewise formula used in the control system software to determine altitude from air pressure.

## Volumetric flow rate calculation

The MAF sensor provides an output linearly corresponding to the mass flow rate of air through the sensor. As the apparatus must sample known volumes of air, the volumetric flow rate of air must be found using the air density. While the density of air at a given pressure and temperature can easily be calculated using the formula in Figure 15, this formula relies on a gas constant,  $R$ , relating to the mix of gases in air. While the mix of gases is different at high altitude, this has a negligible effect on the value of  $R$ . This was confirmed by calculating  $R$  from the density, pressure and temperature values in Table 1.

$$\rho = \frac{P}{(RT)} = \frac{m}{V}$$

Figure 15: The formula relating density  $\rho$  ( $\text{kg/m}^3$ ), pressure  $P$  (Pa) and temperature  $T$  (K) for a specific mix of gases with constant  $R$ .

As a mass flow rate is a ratio of mass to time, a volumetric flow rate is a ratio of volume to time and density is a ratio of mass to volume, a relationship exists between the three variables. By substituting the formula for density, a relationship between mass airflow ( $dm/dt$ ), volumetric airflow ( $dV/dt$ ), pressure and temperature is derived (Figure 16).

$$\frac{m}{t} * \frac{V}{m} = \frac{V}{t} \Rightarrow \frac{m}{t} * \rho^{-1} = \frac{V}{t} \Rightarrow \frac{dm}{dt} * \left( \frac{P}{RT} \right)^{-1} = \frac{dV}{dt} \Rightarrow \left( \frac{dm}{dt} \right) * \left( \frac{RT}{P} \right) = \frac{dV}{dt}$$

Figure 16: The derivation of the formula used to convert mass airflow, pressure and temperature to volumetric airflow.

When collecting a sample, the total collected volume must be determined using the flow rate. A numerical integration of the flow rate is performed using a rectangular approximation (Figure 17). An internal timer of the microcontroller is used to determine the loop time  $\Delta t$ . The MAF sensor is read and the volumetric airflow ( $dV/dt$ ) is calculated every loop. Storing a list of values to sum after the sampling would quickly exceed the Raspberry Pi Pico W's 256kB of RAM at high sampling rates. Instead a running sum is kept during the sampling period so that only a single value must be stored. When the desired sample volume has been collected, the fan is turned off.

$$\int \frac{dV}{dt} dt = V_{\text{sampled}} = \lim_{\Delta t \rightarrow 0} \left( \sum_{t_{\text{sample start}}}^{t_{\text{sample end}}} \left( \frac{dV}{dt} * \Delta t \right) \right)$$

Figure 17: The sampled volume is determined using rectangular integration.

To minimise  $\Delta t$ , the loop must be run as fast as possible. During development it was found that logging to the CSV file during the loop severely impacts loop speed, due to the speed of the microcontroller's internal flash memory. To improve loop speed, an option was included to disable data logging during sampling.

## Data logging

The two megabytes of internal flash storage holds the compiled CircuitPython interpreter program and also a file system where the control system software and experimental data logs are stored. This file system can be accessed by the interpreter and by the computer it is connected to. Catastrophic data corruption will occur if both the interpreter and computer attempt to write to the file system at the same time. Consequently, only one of these can have the file system mounted with write permissions while the other must be mounted as read-only. In order for the control system software to log data, the file system must be mounted with write permissions by the interpreter, making the file system read-only for the computer. This means however that the computer will not be able to edit the control system software. The solution to this was for a program to be written which runs as the microcontroller boots. This program checks the value of a pin on the microcontroller and either takes write permissions if the signal is low, or leaves them for the computer to take if the signal is high. A circuit was included on the circuit board to automatically disable data logging and give write permissions to the computer if a USB cable is connected. A jumper to override this behaviour is also included.



Data is logged in the CSV file format. This is the same as a text file, but with particular placement of commas and line ending characters to make the data easily import into a spreadsheet later.

CircuitPython allows a text file to be opened and written to in various modes. By opening in the append mode, data is written after existing data. Therefore previous experimental data will not be overwritten but instead added to, and the data is written as measurements are taken, so a power failure of the apparatus will not result in data loss.

As the airborne transport mechanism of microplastics is not yet understood, it is not clearly defined what additional data may be useful to collect. Therefore it was determined that it would be beneficial to collect all data which is readily available and may conceivably be used to continue research in this field. By observing correlation between variables, researchers can infer likely chains of causation for further study. For example, by recording the air temperature, humidity and balloon's ascent rate, researchers may discover a higher prevalence of microplastics in warm moist ocean thermals than the rest of the atmosphere.

## Circuit board

The circuit diagram of the control system is provided in Figure 18.

As mentioned in section 4.2.2, the output of the MAF sensor is a common 0-5V analog automotive sensor signal. No level shifting circuit has been included on the circuit board however as testing the output of this sensor showed that its actual output range is 0-3.3V. The inclusion of a level shifting circuit would only reduce the effective resolution of the sensor.

To switch the power to the MAF sensor and fan, a ULN2003 Darlington transistor array is used. The ULN2003 offers seven NPN Darlington transistor pairs in a single IC with a common ground and internal biasing. This provides a convenient way to switch higher currents from the microcontroller without greatly increasing the part count or number of solder joints to be made. Multiple channels of the transistor array can be connected in parallel to add together their current capacity for higher current devices like the fan.

While not a design requirement, a very bright red LED array designed to be powered directly by 5V was included. This was connected to two spare channels of the ULN2003 so that it can be switched by the control software. This LED array is used as a flashing aircraft clearance light during flight to provide an additional level of safety. During the recovery stage, the LED gives a short double flash every three seconds to aid researches in finding the apparatus while conserving battery power.

A 7805 linear voltage regulator IC is used to supply 5V power to the microcontroller. While not the most efficient way to generate 5V, the 7805 reduces the part count of the design and increases its reliability. While a buck converter circuit using a charge pump would be more efficient, these circuits often require the use of an electrolytic capacitor, a component which would not reliably operate at very low temperatures and partial vacuum. The current draw of the microcontroller is also not very high, so the 7805's efficiency is not a particularly great concern. The Raspberry Pi Pico W has a separate power input which is isolated from the USB power source by diodes. This allows the board to be powered by either source without one source feeding into the other. The 7805 regulator powers the board via this separated input, allowing the batteries to be installed while the USB cable is connected, which is useful when testing the MAF sensor and fan.

The software code run at boot time uses the Raspberry Pi Pico W's pin GP2 to determine whether the interpreter or a computer will have write permissions for the file system on the internal flash storage. A voltage divider takes the 5V supply from the USB port and outputs a 3.3V signal to the pin when the USB port is powered. A jumper is included to override this. While the voltage divider is not strictly necessary, as the pins of the Raspberry Pi Pico W are 5V tolerant, one resistor would be required anyway to prevent the jumper shorting the USB port to ground.

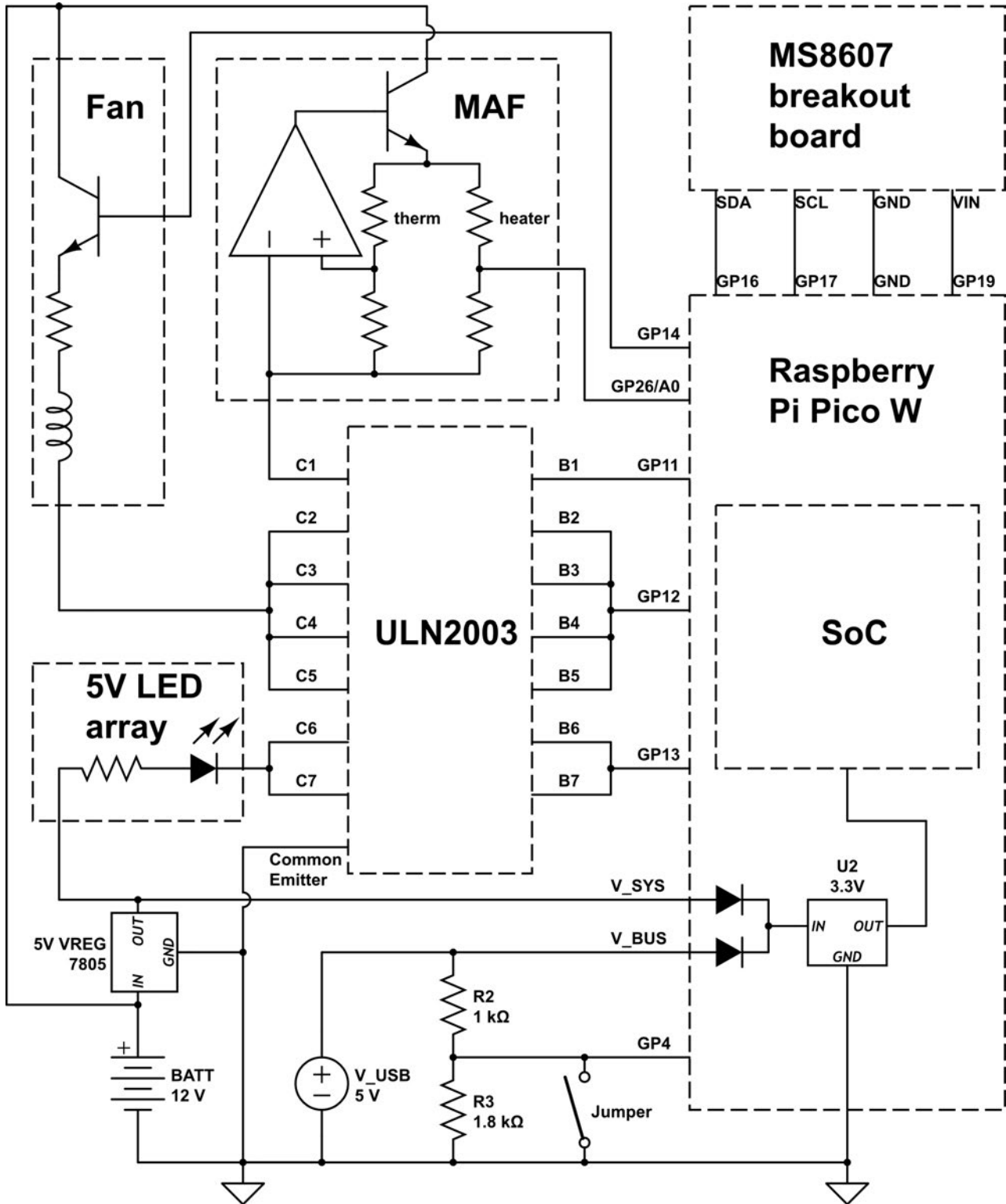


Figure 18: The circuit diagram of the control system.

The full circuit as shown in Figure 18 was constructed on a prototyping circuit board which mimics the layout of a standard breadboard used in electronics prototyping (Sparkfun 2023). The completed board is shown in Figure 19. This allows the circuit to be adaptable and easily understood by others.

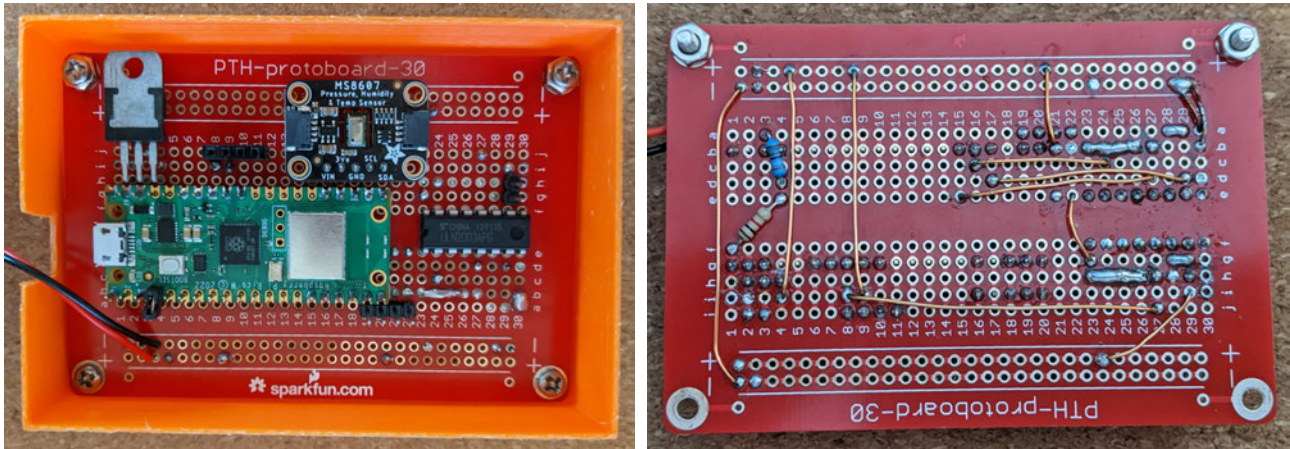


Figure 19: The completed circuit board of the control system.

### 4.3 Testing and Results

The apparatus was tested at ground level atmospheric pressure by inflating a bag to directly measure the air volume. This was quite close to the volume calculated by the digital control system, however a calibrated flow bench would be required to calibrate the apparatus more accurately. The automotive MAF sensor and the algorithm for conversion to volumetric flow provided reliable feedback to the control system. The fan was successfully turned off by the control system when the calculated volume reached the target volume as per the design, providing a metered sample of air. During this test, a “TestData.csv” file was created on the apparatus USB storage file system, which has been provided as Appendix D.

The apparatus was then tested in a vacuum chamber which had the air pressure reduced to 68.4hPa, equivalent to 18,654m in altitude. Higher equivalent altitude tests were planned but the vacuum pump available was unable to reduce the pressure further. During the test a sample of 1,000.3L was collected between 10,402m and 10,943m of equivalent altitude after being automatically triggered to begin at 10,000m of equivalent altitude. The 402m overshoot from the target altitude is due to the speed of the vacuum pump which far exceeds the flight speed of the balloon in its rate of pressure change. No faults with the apparatus were detected and it performed as expected in every regard. The automotive MAF sensor performed reliably in the low pressure environment and the fan pushed air through it as expected. During this test, a “TestData.csv” file was created on the apparatus USB storage file system, which has been provided as Appendix E.

## CHAPTER 5

### CONCLUSIONS

#### 5.1 Conclusions

The apparatus was tested both at ground level atmospheric pressure and in a vacuum chamber to provide the reduced pressure seen during a high altitude balloon flight, as detailed in section 4.3. During these tests, the apparatus hardware and software performed as expected, automatically triggering the sampling routine, collecting a sample and logging data accordingly. The design shows considerable promise for use in atmospheric microplastics research, although more thorough testing would be prudent prior to deployment in order to ensure the accuracy of results and reliability of the system.

The physical design of the apparatus performed as expected, with parts being easy to 3D print with a low-end consumer grade 3D printer and finally assemble with only a few screwdrivers. The circuit board required some specialised skills to produce such as soldering and electronic circuit design, but could be reduced to entry-level with the production of a kit including a custom made circuit board which can be easily and cheaply sourced from a plethora of fabrication services. A fully populated circuit board could also be ordered to completely remove the need to perform electronics assembly. This could potentially be achieved without increasing costs, as individual ICs such as the MS8607 pressure and temperature sensor and Raspberry Pi RP2040 microcontroller (the main component of the Pi Pico W) could be sourced at a lower price than the ones used which come on carrier breakout boards with marked up pricing. Installing the code and using the completed control system was as extremely simple and would not present difficulties to anyone with basic computer skills. Further development and testing of the circuit board design to make it more reliable and easier to construct would be advisable before proceeding with producing examples for atmospheric microplastics research.

All electronic components used were purchased from consumer websites with global shipping options for a total cost for all major components of \$139.79 inclusive of shipping. Minor components used to craft the prototype circuit board were not included in this tally as they were sourced from a spare parts bin and are generally very cheap and easily acquired. All parts arrived within two weeks of being ordered. As discussed in section 4.2.2, a female connector to suit the 3 pin male connector on the MAF sensor could not be sourced quickly enough for the project timeline and a workaround involving specialised skills was required. This should not be an issue in future as global supply chain issues are remedied following the COVID-19 pandemic. The achieved price point and ease of sourcing parts should not be an impediment for the vast majority of researchers around the world.

The completed apparatus weighed 628g with batteries. While no regulatory requirements for balloon launches in Australia are available and approval is instead given on a discretionary basis, compliance with international requirements is necessary for wide accessibility of the apparatus design. In the USA this limit is 6lbs (2.7kg) for a single payload, leaving more than 2kg of overhead for the addition of a camera to monitor the apparatus' operation, a GPS tracking system, parachute, and the implementation of extra sensors and features.

## 5.2 Challenges

It was originally planned to launch the apparatus as an additional payload on a proposed high altitude balloon launch. The launch has been delayed until after the end of this project, so the only testing possible was inside a vacuum chamber to simulate the low air pressure at high altitude. The main factor of the high altitude environment that cannot be readily recreated in vacuum chamber testing is the extremely low temperature.

The stated temperature range of the ICs used is  $-40^{\circ}\text{C}$  to  $+85^{\circ}\text{C}$ , which is not quite adequate to endure the extremely cold temperatures of altitudes above roughly 8km (26,000ft) (Table 1). Further testing could prove that some chips malfunction at these temperatures, however it is expected that some self-heating will occur due to power consumption, especially with the reduced cooling provided by very low density air. This temperature range could also prove to be inaccurate and the chips could function correctly over a far greater range than stated, as this is a standard industrial IC temperature range. This standard range would be applied to any chip other than those intended for military use and would also be applied to any chip incapable of operating above  $85^{\circ}\text{C}$  regardless of its low temperature endurance.

As explained in section 4.2.4, Python is an interpreted programming language. The speed of the CircuitPython interpreter on the Raspberry Pi Pico W was found to be insufficient to achieve a short enough loop time to apply an effective filtering algorithm to the MAF sensor signal. As a result this signal has some noise. This noise may be due to the buffeting effect of the fan blades being in close proximity to the sensor, or may be inherent to the sensor. This noise may decrease the accuracy of the sensor, although the averaging effect of the running sum created from its output may abate this to some degree. As this signal is analog, an analog filter could be applied by including a capacitor between the MAF signal pin and ground. The Raspberry Pi Pico W also includes programmable IO modules which act similarly to a basic co-processor, allowing timing-intensive tasks to be offloaded from the main CPU and run much faster in assembly language. It may be possible to implement sensor filtering on these modules while retaining the ease of use of the CircuitPython language for the main program.

## 5.3 Further Research

Collecting samples of microplastics is only one part of the problem. Processing these samples is laborious and therefore costly, as samples must be manually sorted through and each piece of microlitter examined and tested individually. Some automated method to handle and test the sample to differentiate plastics from other materials at this scale is a research topic ripe for expansion. Nanoplastics (plastic microlitter in the sub-micron range) are a neglected field of study at this time, as nanoplastics are too small to manually process samples whatsoever. This problem might be addressed with the development of a microelectromechanical device to move samples into a microscope's view and computer vision and artificial intelligence to determine the outcome of automated tests.

The cold endurance of electronic components in the design has not been able to be tested. If problems with low temperature performance arise, an insulated enclosure and switchable high wattage resistor could be included to provide a self-heating function to the control system components.

By adding a GPS unit, researchers could also collect data on the direction of wind in which the sample is collected, perhaps providing insight into the origin of microplastics and the wind currents that carry them. As mentioned in section 5.1, high altitude balloons in Australia are regulated on a discretionary basis, however a GPS tracking system is a common inclusion for safety and readily locating the landed payload. In the USA this is a legal requirement. If a GPS is already included in another payload, this data would simply need to be made accessible to the apparatus control system. This could be readily achieved using Bluetooth's serial port profile, which is a feature of the Raspberry Pi Pico W microcontroller board used in this project (BlueKitchen 2023).

To accurately test the fan's operation at high altitude in the laboratory environment, the vacuum chamber needs to be chilled as well as evacuated. One way to achieve this would be to bring the chamber as close to absolute vacuum as possible and then add a calculated mixture of liquid nitrogen and room temperature air to bring the pressure back up to the intended simulated altitude and the corresponding temperature. As vacuum chambers are pressure vessels which need considerable strength for safe operation, the chamber must be made from a material which can safely be operated at these low temperatures. The glass would also be exposed to temperatures around 25°C on the outside and as low as -56.5°C on the inside. This 81.5K temperature differential through the material would subject the vacuum chamber to considerable thermal loading. It is anticipated that the glass vacuum chamber used in this testing would be unsafe to operate with this temperature differential, and if cooling the chamber by adding liquid nitrogen, the susceptibility of glass to thermal shock may also be a cause for concern.


A printed circuit board design could be produced, allowing researchers to simply order the board from one of many online made-to-order fabrication stores. This would mean that no soldering or placement of components would be required and the assembly could be entirely "plug and play".

As computer fans are nearly exclusively dark in colour, and the apparatus must operate in high ultraviolet exposure of roughly 135% of that at sea level with limited conductive cooling provided by the low density atmosphere, it may be observed that shading the fan is insufficient to provide reasonably reliable operation. The fan should also operate for limited periods separated by periods of rest to allow for passive cooling.

As mentioned in section 5.2, flight testing of the apparatus has been delayed and is planned to continue as further research.

## APPENDICES

## Appendix A – Risk assessment

2473	RISK DESCRIPTION			TREND	CURRENT	RESIDUAL
	ENG4111 Research Project - James Beecham				Medium	Low
RISK OWNER		RISK IDENTIFIED ON	LAST REVIEWED ON		NEXT SCHEDULED REVIEW	
James Beecham		22/05/2023				
RISK FACTOR(S)	EXISTING CONTROL(S)	CURRENT	PROPOSED CONTROL(S)	TREATMENT OWNER	DUE DATE	RESIDUAL
A device of less than 3kg falling from a height of no more than 50km. Very unlikely (less than lightning) to hit a person.	Control: A parachute is attached to the device to slow its fall.	Low	No Control:			Low
A 40 watt class 4 CO2 laser as part of a CNC laser cutter machine.	Control: An interlock is installed on the cover of the machine so that the laser cannot fire while the cover is ajar. The cover contains a clear viewing window made of acrylic plastic, which infrared should not pass through.	Low	The viewing window has been covered with a material which is opaque to infrared and visible light. A camera system has been fitted for remote monitoring of fire risk.			Low
The CO2 laser has the potential to burn objects.	Control: A viewing window exists to allow monitoring of the cutting process so that intervention can take place while the fire is limited to candle-like proportions. No cutting is allowed to be left unattended.	Medium	An air assist has been fitted to the laser cutter to greatly reduce the risk of ignition. Cool compressed air constantly blows on the area exposed to the laser beam. Monitoring will still be required via a camera, with the student supervising within close proximity.			Low
The CO2 laser will vaporise small amounts of the objects it cuts. This can release various chemicals depending on what is cut. Most notably, materials containing the element chlorine can release chlorine gas, which is known to cause lung damage. Mainly a concern for occupational (ongoing) exposure.	Control: A fume extraction system is fitted to draw vapours, dust and gases out of the machine and vent them away from people.  Control: All materials are to be tested for chlorine before cutting. No material containing chlorine is allowed to be cut.	Low	No Control:			Low
Large (possibly as long as 200 pages) documents will be typed. Code will be typed. CAD modeling with a computer	Control: Nothing is usually done about this.	Medium	Frequent breaks will be taken from the tasks.			Low

powered by riskware.com.au

commercial in confidence

mouse.						
Soldering iron use for electronics assembly. Liquid solder can spring off of components towards the face.	Control: Protective goggles will be worn during all soldering works.	Low	No Control:			Low
Soldering fumes can contain lead.	Control: Mainly a concern for occupational (ongoing) exposure.	Low	A fume extractor will be used when soldering works are performed.			Low
There is a very small risk of the glass vacuum chamber imploding if it is damaged.	Control: The risk is very low and glass is unlikely to be ejected.	Low	The vacuum chamber will be inspected for damage prior to use.			Low
Small cuts and bruises related to hand tool use and the manual installation of small electronic components with sharp metal pins.	Control: Nothing is usually done about this.	Low	Protective leather gloves will be worn where practicable.			Very Low

powered by riskware.com.au

commercial in confidence

## Appendix B – Control system software code

The following is the contents of code.py, the CircuitPython software code which runs as on the Raspberry Pi Pico W as the main program of the control system.

```
''' Atmospheric Microplastics Measurement Apparatus Controller
By James Beecham for dissertation project (ENG4111 & ENG4112)
University of Southern Queensland, 2023
This code is produced as part of an educational exercise only and
should not be used for any other purpose.
No warranty or guarantees apply.
'''

#-----Test Settings-----
#What altitude (in metres) do you want to collect the sample at?
#(For ground testing at atmospheric pressure, enter negative alt.)
SampleALT = 10000
#What volume (in litres) of air should be sampled?
SampleVOL = 1000

#MAF sensor calibration. What reading does the sensor give at what
#flow rate, temperature and air pressure during calibration test?
CalibrationTest = False          #True: calibration testing mode
MAFcalVal = 36023                 #MAF sensor value
MAFcalos = 16471                 #MAF sensor offset value
MAFcalT = 28.04                  #calibration temp degrees celcius
MAFcaldvdt = 2369                #flow rate dv/dt in m^3 per min
MAFcalP = 1015.14                #calibration pressure in mbar/hpa

#Fast sampling increases sampling rate (and therefore accuracy) by
#only writing to the CSV file after sampling. Disabling this
#assists in debugging but slows the loop time due to the speed of
#the internal flash memory.
FastSampling = True

#-----Imports-----
#basic libraries
import time
import supervisor
import board
import digitalio
import pwmio
import busio
import math
from analogio import AnalogIn
if CalibrationTest == True:
    import gc

#PHT sensor library
from adafruit_ms8607 import MS8607
```



```

#-----Set pin assignments-----
PHTpwrPin = board.GP19      #turns on the PHT sensor board
I2Csc1Pin = board.GP17      #I2C bus SCL pin
I2CsdaPin = board.GP16      #I2C bus SDA pin

MAFpwrPin = board.GP11      #heats the MAF sensor when on
MAFanalogPin = board.A0     #analog reading from MAF sensor

FANpwrPin = board.GP12      #12V switched power for fan
FANpwmPin = board.GP14      #PWM speed control for fan
#FANSensePin = board.GP15    #connected but not used

BeaconLEDPin = board.GP13    #5v switched beacon LED

#-----Board LED-----
BoardLED = digitalio.DigitalInOut(board.LED)  #set pin number
BoardLED.direction = digitalio.Direction.OUTPUT #set pin to output
BoardLED.value = True                        #turn on board LED

#-----Beacon LED-----
BeaconLED = digitalio.DigitalInOut(BeaconLEDPin) #set pin number
BeaconLED.direction = digitalio.Direction.OUTPUT #set pin to output
BeaconLED.value = False                        #turn off beacon LED

#-----PHT Sensor Setup-----
PHTpwr = digitalio.DigitalInOut(PHTpwrPin)      #set pin number
PHTpwr.direction = digitalio.Direction.OUTPUT   #set pin to output
PHTpwr.value = True                             #turn on PHT sensor
time.sleep(1)                                   #wait for PHT boot
i2c = busio.I2C(I2Csc1Pin, I2CsdaPin)           #set up i2c bus
PHTsignal = MS8607(i2c)                         #setup MS8607 lib

def PRES():                                     #pressure function
    global PHTsignal                            #import global var
    return (PHTsignal.pressure)                 #return

def HUM():                                     #pressure function
    global PHTsignal                            #import global var
    return (PHTsignal.relative_humidity)        #return

def TEMP():                                   #temp. function
    global PHTsignal                            #import global var
    return (PHTsignal.temperature)              #return

def ALT():                                     #piecewise alt func
    if PRES() > 189.94: #TNWS formula for <40,000ft
        altitude = 44307.69*(1-(PRES()/1013.25)**0.190284)
    else:                                     #Derived trend form
        altitude = ((-6417.004605*math.log(PRES())) + 45755.99158)
    #debugging overrides. Un-comment below line for groundnd tests
    #altitude = int(input("\nenter altitude for debugging: "))
    return (altitude)                         #conv pres. to alt.

MaxALT = 0                                    #Init MaxALT
LaunchALT = 0                                #Init LaunchALT

```

```

#-----MAF Sensor Setup-----
MAFpwr = digitalio.DigitalInOut(MAFpwrPin)    #set pin number
MAFpwr.direction = digitalio.Direction.OUTPUT #set pin to output
MAFpwr.value = False                          #turn off MAF
MAFsignal = AnalogIn(MAFanalogPin)           #set analog pin

#Interpret calibration data. Real dmdt = dvdt * P/RT.
MAFcaldvdt = MAFcaldvdt/60/1000              #L/min to m^3/s
MAFcalP = MAFcalP*100                        #change to Pascals
MAFcalT += 273.15                           #change to Kelvin
MAFcaldmdt = MAFcaldvdt*MAFcalP/(287.058*MAFcalT)#find dmdt (kg/s)
#MAFcal is a ratio of dmdt in kg/s to MAF sensor reading
MAFcal = MAFcaldmdt / (MAFcalVal-MAFcalos)

def MAF():
    global MAFsignal
    global MAFpwr
    global MAFcal
    output = MAFsignal.value - MAFcalos        #remove offset
    output = abs(output)                       #ban negatives
    output = MAFcal * output                   #scale to kg/s
    if MAFpwr.value == False:                 #floating voltage
        output = 0
    return(output)                            #return result

def VAF():
    global MAFcal
    #dmdt*(RT/P)=dvdt, *1000: m^3->L, +273.15:C->K, *100:mbar->Pa
    output=1000*MAF()*287.058*(TEMP()+273.15)/(100*PRES())
    return(output)                            #return result

CurrentVOL = 0                               #init CurrentVOL

#-----Fan Setup-----
#set FANpwr pin assignment, off to start
FANpwr = digitalio.DigitalInOut(FANpwrPin)    #set pin number
FANpwr.direction = digitalio.Direction.OUTPUT #set pin to output
FANpwr.value = False                          #turn off fan

#set pin assignment, frequency (25kHz), minimum speed to start
FANSpeed = pwmio.PWMOut(FANpwmPin, frequency=25000, duty_cycle=0)

def FAN(state):
    global FANpwr
    if state == True:                          #to turn on fan
        FANpwr.value = True
        FANSpeed.duty_cycle = 2**16-1
    elif state == False:                      #to turn off fan
        FANSpeed.duty_cycle = 0
        FANpwr.value = False
    elif state < 1:                           #to set fan speed %
        FANpwr.value = True
        FANSpeed.duty_cycle = state * 65535 / 100

```

```

#-----Logger Setup-----
'''Checks whether writing to a data logging file will cause an
error. This WILL happen if you have USB connected and don't have a
bypass jumper from pin 3 to pin 4/GP2. See boot.py for details'''

RWswitch = digitalio.DigitalInOut(board.GP2)
RWswitch.direction = digitalio.Direction.INPUT

if RWswitch.value == True:                                #if code has no RW
    outputCSV = False                                     #disable CSV output
    print("\nDATA LOGGER DISABLED.")
    print("SEE BOOT.PY FOR DETAILS.\n")
    time.sleep(3)                                         #wait to read msg
else:
    outputCSV = True                                       #enable CSV output
    print("\nDATA LOGGER ENABLED.")
    print("SEE BOOT.PY FOR DETAILS.\n")
    time.sleep(1)                                         #wait to read msg

#-----Calibration Test-----
#Calibration mode. Reads the MAF sensor for 10 seconds in still
#air, then runs the fan and reads the MAF sensor with max airflow.
#Averages sensor values, saves to CSV and shows them in REPL.
if CalibrationTest == True:
    #Test with the fan off to get average MAF offset reading.
    #Wrapped in function so that variables are culled after each.
    def CalibNoFan():
        global MAFcalos                                     #import global var
        global MAFcalP                                     #import global var
        global MAFcalT                                     #import global var
        print("\n10 second still air test starts in 3 seconds.\n")
        MAFpwr.value = True                                #turn MAF on
        time.sleep(3)                                       #wait for heat up
        MAFvals = [MAFsignal.value]                        #array 1st val
        PRESvals = [PRES()]                                #array 1st val
        TEMPvals = [TEMP()]                                #array 1st val
        CalibStartTime = supervisor.ticks_ms()             #record start time
        while True:
            time.sleep(0.05)                                #0.05sec loop time
            MAFvals.append(MAFsignal.value)                 #append MAF value
            PRESvals.append(PRES())                         #append PRES value
            TEMPvals.append(TEMP())                         #append TEMP value
            if (supervisor.ticks_ms() - CalibStartTime) > 10000:
                break
        MAFcalos=sum(MAFvals)/len(MAFvals)                 #calc average MAF
        MAFcalP=sum(PRESvals)/len(PRESvals)                #calc average PRES
        MAFcalT=sum(TEMPvals)/len(TEMPvals)                #calc average TEMP

```

```

#Test with the fan on to get average MAF reading
def CalibFan():
    global MAFcalVal                                #import global var
    print("\n10 second FAN ON test starts in 3 seconds.\n")
    print("\n!!! FINGERS CLEAR !!!.\n")
    time.sleep(3)                                    #wait for safety
    FAN(True)                                         #turn FAN on full
    time.sleep(3)                                    #wait for spin up
    MAFvals = [MAFsignal.value]                      #array 1st val
    CalibStartTime = supervisor.ticks_ms()           #record start time
    while True:                                     #10 sec with fan
        time.sleep(0.05)                             #0.05sec loop time
        MAFvals.append(MAFsignal.value)              #append MAF value
        if(supervisor.ticks_ms() - CalibStartTime) > 10000:
            break
    FAN(False)                                       #turn FAN off
    MAFpwr.value = False                           #turn MAF off
    MAFcalVal=sum(MAFvals)/len(MAFvals)              #calc average MAF

CalibNoFan()                                       #calibrate offset
gc.collect()                                     #clean up RAM
CalibFan()                                       #calibrate MAF
gc.collect()                                     #clean up RAM

#Display data in REPL CLI for convenience if user has REPL on.
print("MAF calibration offset = MAFcalos = %f" % MAFcalos)
print("MAF value = MAFcalVal = %f" % MAFcalVal)
print("Pressure (hPa/mbar) = MAFcalP = %f" % MAFcalP)
print("Temperature (C) = MAFcalT = %f" % MAFcalT)
print("Vol flow rate (L/min) = MAFcaldvdt: From flow bench.")
print("\nCalibration complete.\n")

#Write data to CSV file for no REPL or no PC on flow bench).
if outputCSV == True:                             #write CSV headers
    with open("/Calibration.txt", "w") as TXT: #w=overwrite
        TXT.write('MAF calibration offset = MAFcalos = ')
        TXT.write('{0:f}', '.format(MAFcalos))
        TXT.write('\n')
        TXT.write('MAF value = MAFcalVal = ')
        TXT.write('{0:f}', '.format(MAFcalVal))
        TXT.write('\n')
        TXT.write('Pressure (hPa/mbar) = MAFcalP = ')
        TXT.write('{0:f}', '.format(MAFcalP))
        TXT.write('\n')
        TXT.write('Temperature (C) = MAFcalT = ')
        TXT.write('{0:f}', '.format(MAFcalT))
        TXT.write('\n')
        TXT.write('Vol flow rate (L/min) = MAFcaldvdt = ')
        TXT.write('Flow bench data')
        TXT.write('\n')
        TXT.write('No flow bench? Fan rated flow is kinda ok')
        TXT.flush

time.sleep(65535)                                #sleep "forever"

```

```

#-----Main Loop-----
#The main loop is a state machine. Loop speed is defined by state.
#Note that CircuitPython lacks switch/case, as based on Python 3.8
#CPU timer overflow is >6 days; no need to handle.
state = 1                                #init state var
stateswitch = True                        #init as true
# 1: Pre-Launch Routine
# 2: Ascent Routine
# 3: Sample Routine
# 4: Descent Routine
# 5: Recovery Routine
# state=0|state>5: allow state to be entered in REPL for testing

#Check if CSV output works & add separator from any old test data
if outputCSV == True:
    try:                                  #test for errors
        with open("/TestData.csv", "a") as CSV:
            CSV.write('\n\nNEW TEST DATA FOLLOWS\n\n\n')
            CSV.flush
        print("\nTest write successful.\n")
    except OSError as Error:              #if errors thrown
        outputCSV = False
        print("\nERROR WRITING TO CSV.")
        print("Errorr code: %i" % Error.args[0])#print error code
        print("DATA LOGGER DISABLED.\n")

while True:
    if state == 1:
        #=====Pre-Launch Routine=====
        #This is state for pre-launnnch checks
        #Don't flash LED (which would blind the launch crew).

        #First loop in this state:
        if stateswitch == True:            #first loop only
            stateswitch = False            #only run this once
            LaunchALT = ALT()              #record launch alt
            PLaunchsttime = supervisor.ticks_ms() #record start t
            print("Launching from %.2f M altitude" % LaunchALT)
            if outputCSV == True:          #log beginning of pre-launch
                with open("/TestData.csv", "a") as CSV:
                    CSV.write('\n\nPre-Launch Stage:\n')
                    CSV.write('Launching from altitude (m):,')
                    CSV.write('{0:f}'.format(LaunchALT))
                    CSV.write('@CPU time (ms):,')
                    CSV.write('{0:f}'.format(PLaunchsttime))
                    CSV.write('To collect a sample of (L):,')
                    CSV.write('{0:f}'.format(SampleVOL))
                    CSV.write('At altitude (m):,')
                    CSV.write('{0:f}\n'.format(SampleALT))
                    #Column headers for repeated data logging
                    CSV.write('CPU uptime (ms),')
                    CSV.write('Altitude (m),')
                    CSV.write('Pressure (hPa),')
                    CSV.write('Temperature (C),')
                    CSV.write('Humidity (%),')
                    CSV.write('MAF sensor (kg/s),')
                    CSV.write('VAF (L/s),')
                    CSV.write('Sampled volume (L)\n')
                    CSV.flush

```

```

#LED handling:
BeaconLED.value = False          #don't use LED

#Slow loop time to save power and avoid excessive logging
time.sleep(3)                    #wait 10sec

#Escape conditions:
if ALT() >= (LaunchALT + 10):      #if ascending
    state = 2                     #switch to ascent
    stateswitch = True            #for 1 time code
if SampleALT <= LaunchALT:        #allow gnd testing
    state = 2                     #switch to ascent
    stateswitch = True            #for 1 time code

elif state == 2:
    #=====Ascent Routine=====
    #This is state for ascent by balloon
    #Flash beacon LED at 0.5Hz for aircraft clearance

    #First loop in this state:
    if stateswitch == True:        #first loop only
        stateswitch = False       #only run this once
        print("\nAscending now!")
        Ascentstime = supervisor.ticks_ms() #record start t
        if outputCSV == True:     #log beginning of ascent
            with open("/TestData.csv", "a") as CSV:
                CSV.write('\n\nAscent Stage:\n')
                CSV.write('Ascending @CPU time (ms):,')
                CSV.write('{0:f}\n'.format(Ascentstime))
                #Column headers for repeated data logging
                CSV.write('CPU uptime (ms),')
                CSV.write('Altitude (m),')
                CSV.write('Pressure (hPa),')
                CSV.write('Temperature (C),')
                CSV.write('Humidity (%),')
                CSV.write('MAF sensor (kg/s),')
                CSV.write('VAF (L/s),')
                CSV.write('Sampled volume (L)\n')
                CSV.flush

    #LED handling:
    BeaconLED.value = not BeaconLED.value #toggle LED
    time.sleep(1)                        #1sec loop time

    #Escape conditions:
    if ALT() >= SampleALT:          #at sample altitude
        if CurrentVOL < SampleVOL: #if sample not done
            state = 3              #switch to sample
            stateswitch = True     #for 1 time code
    if ALT() < MaxALT - 1000:       #if fallen > 1km
        state = 4                 #switch to descent
        stateswitch = True        #for 1 time code
    if (ALT() < (LaunchALT + 120)) & (CurrentVOL > 0):
        state = 5                 #switch to recovery
        stateswitch = True        #for 1 time code
        Descentstime = supervisor.ticks_ms()

```

```

elif state == 3:
    #=====Sample Routine=====
    #This is state for during in flight sample collection
    #Use the FAN to draw a sample through the filter.
    #Use the MAF to keep track of the sample size.
    #Flash beacon LED at 0.5Hz for aircraft clearance
    #Maximum loop time for accuracy, accurately measured.

    #First loop in this state:
    if stateswitch == True:                #first loop only
        stateswitch = False                #only run this once
        looptime = 0.0                     #dt var
        toggletime = 0                    #LED loop timer var
        lasttime = supervisor.ticks_ms()   #last time var
        print("\nSampling now!")
        Samplestime = supervisor.ticks_ms() #rec start t
        SamplestALT = ALT()                #rec start altitude
        RestoreopCSV = outputCSV           #to restore if FS
        if FastSampling == True:            #for fast sampling
            if outputCSV == True: #log beginning of sampling
                with open("/TestData.csv", "a") as CSV:
                    CSV.write('\n\nSampling Stage:\n')
                    CSV.write('Start sampling @ time (ms):,')
                    CSV.write('{0:f}\n'.format(Samplestime))
                    CSV.write('Fast sampling on = no logs.\n')
            outputCSV = False               #disable logging
        if outputCSV == True:               #log beginning of sampling
            with open("/TestData.csv", "a") as CSV:
                CSV.write('\n\nSampling Stage:\n')
                CSV.write('Start sampling @ time (ms):,')
                CSV.write('{0:f}\n'.format(Samplestime))
                #Column headers for repeated data logging
                CSV.write('CPU uptime (ms),')
                CSV.write('Altitude (m),')
                CSV.write('Pressure (hPa),')
                CSV.write('Temperature (C),')
                CSV.write('Humidity (%),')
                CSV.write('MAF sensor (kg/s),')
                CSV.write('VAF (L/s),')
                CSV.write('Sampled volume (L)\n')
                CSV.flush
            MAFpwr.value = True              #turn MAF on
            time.sleep(3)                   #wait for heat up
            FAN(True)                       #turn FAN on full

    #LED handling:
    #If 1s since LED toggle, toggle and start timing again
    if (supervisor.ticks_ms() - toggletime) >= 1000:
        BeaconLED.value = not BeaconLED.value #toggle LED
        toggletime = supervisor.ticks_ms() #update toggletime
        #also write to the data log every 1s:

    looptime = supervisor.ticks_ms() - lasttime #find looptime
    lasttime = supervisor.ticks_ms()           #store last time

    #Running sum to integrate flow rate WRT loop time
    #This gives the total volume which has flowed
    CurrentVOL = CurrentVOL + (VAF() * (looptime / 1000))

```

```

#Escape conditions:
if CurrentVOL >= SampleVOL:          #after sampling
    Sampletime = (supervisor.ticks_ms()-Samplestime)/1000
    FAN(False)                        #turn FAN off
    MAFpwr.value = False              #turn MAF off
    state = 2                         #switch to ascent
    stateswitch = True                #for 1 time code
    SamplefnALT = ALT()               #rec finish alt
    DescentDuringSampling = False     #no error logged
    outputCSV = RestoreopCSV          #restore logging
if ALT() < MaxALT - 1000:              #if fallen > 1km
    Sampletime = (supervisor.ticks_ms()-Samplestime)/1000
    state = 4                         #switch to descent
    stateswitch = True                #for 1 time code
    FAN(False)                        #turn FAN off
    MAFpwr.value = False              #turn MAF off
    SamplefnALT = ALT()               #rec finish alt
    outputCSV = RestoreopCSV          #restore logging
    #Throw an error in the data log as the balloon has
    #popped during sampling and the test is spoiled
    DescentDuringSampling = True      #error logged
    if outputCSV == True:
        with open("/TestData.csv", "a") as CSV:
            CSV.write('\n\nERROR!\n')
            CSV.write('Descended during sampling.\n')
            CSV.write('Results likely unreliable.\n')

elif state == 4:
    #=====Descent Routine=====
    #This is state for descent by parachute
    #Log altitude and descent rate for parachute testing
    #Flash beacon LED at 0.5Hz for aircraft clearance

    #First loop in this state:
    if stateswitch == True:            #first loop only
        stateswitch = False           #only run this once
        print("\nDescending now!")
        Descentstime = supervisor.ticks_ms() #record start t
        if outputCSV == True:          #log beginning of descent
            with open("/TestData.csv", "a") as CSV:
                CSV.write('\n\nSampling Stage:\n')
                CSV.write('Started descent @CPU time (ms):,')
                CSV.write('{0:f}\n'.format(Descentstime))
                #Column headers for repeated data logging
                CSV.write('CPU uptime (ms),')
                CSV.write('Altitude (m),')
                CSV.write('Pressure (hPa),')
                CSV.write('Temperature (C),')
                CSV.write('Humidity (%),')
                CSV.write('MAF sensor (kg/s),')
                CSV.write('VAF (L/s),')
                CSV.write('Sampled volume (L)\n')
                CSV.flush

    #LED handling:
    BeaconLED.value = not BeaconLED.value #toggle LED
    time.sleep(1)                          #1sec loop time

    #Escape conditions:
    if ALT() < (LaunchALT + 120):        #under 120m/400ft
        state = 5                       #switch to recovery
        stateswitch = True               #for 1 time code

```



```

elif state == 5:
    #=====Recovery Routine=====
    #This is the final state to help researchers finding it
    #Short infrequent LED beacon flashes to save power
    #This may look messy, but the sleep function saves battery

    #First loop in this state:
    if stateswitch == True:                #first loop only
        stateswitch = False                #only run this once
        print("\nRecovery time!")
        #log beginning of recovery stage with total time
        #log stage times and summary of flight

        #calculate stage times
        PLaunchtime = (Ascentsttime-PLaunchsttime)/1000
        Ascenttime = (Descentsttime-Ascentsttime)/1000
        Descenttime=(supervisor.ticks_ms()-Descentsttime)/1000
        Recoverysttime = supervisor.ticks_ms()
        Totaltime = (supervisor.ticks_ms()-Ascentsttime)/1000
        if outputCSV == True:              #log beginning of descent
            with open("/TestData.csv", "a") as CSV:
                CSV.write('\n\nRecovery Stage:\n')
                CSV.write('Started recovery @CPU time (ms):,')
                CSV.write('{0:f}\n'.format(Recoverysttime))
                #log stage times
                CSV.write('Pre-Launch took (s):,')
                CSV.write('{0:f}\n'.format(PLaunchtime))
                CSV.write('Ascent took (s):,')
                CSV.write('{0:f}\n'.format(Ascenttime))
                CSV.write('Descent took (s):,')
                CSV.write('{0:f}\n'.format(Descenttime))
                CSV.write('Stayed airbourne for (s):,')
                CSV.write('{0:f}\n'.format(Totaltime))
                #log peak altitude
                CSV.write('Peaked at altitude(m):,')
                CSV.write('{0:f}\n'.format(MaxALT))
                #Log sampling results
                CSV.write('Collected sample of (L):,')
                CSV.write('{0:f}\n'.format(CurrentVOL))
                CSV.write('Altitude window (m):,')
                CSV.write('{0:f},'.format(SamplestALT))
                CSV.write('to (m):,')
                CSV.write('{0:f}\n'.format(SamplefnALT))
                CSV.write('Sampling took (s):,')
                CSV.write('{0:f}\n'.format(Sampletime))
                if DescentDuringSampling == True:
                    CSV.write('\n\nERROR!\n')
                    CSV.write('Descended during sampling.\n')
                    CSV.write('Results likely unreliable.\n')
            CSV.flush
            outputCSV = False                #stop data logging
        #LED handling:
        BeaconLED.value = not BeaconLED.value #toggle LED value
        if BeaconLED.value == False:          #wait 3s while off
            time.sleep(3)
        else:                                #double flash
            time.sleep(0.05)                  #50ms on
            BeaconLED.value = False
            time.sleep(0.05)                  #50ms off
            BeaconLED.value = True
        #<WiFi SSID broadcast to help finding not yet implemented>

```

```

else:
    #=====Invalid State=====
    #For state machine debugging.
    #Pause the program and wait for input of a state variable.
    state = int(input("\nEnter a valid state variable. "))

#=====Runs in all states=====
if ALT() > MaxALT:
    #update max alt
    MaxALT = ALT()

#Data logging:
if outputCSV == True:
    with open("/TestData.csv", "a") as CSV:
        CSV.write('{0:f}'.format(supervisor.ticks_ms()))
        CSV.write('{0:f}'.format(ALT()))
        CSV.write('{0:f}'.format(PRES()))
        CSV.write('{0:f}'.format(TEMP()))
        CSV.write('{0:f}'.format(HUM()))
        CSV.write('{0:f}'.format(MAF()))
        CSV.write('{0:f}'.format(VAF()))
        CSV.write('{0:f}'.format(CurrentVOL))
        CSV.write('\n')
        CSV.flush
#END OF PROGRAM

```

## Appendix C – Control system software boot code

The following is the contents of `boot.py`, the CircuitPython software code which runs as the Raspberry Pi Pico W is booting up. The purpose of this code is to enable the hardware pin GP2 to select whether the file system on the internal flash storage will be mounted as read-only or with write permissions for either the connected computer or the CircuitPython interpreter itself.

```
...
```

This code runs at boot to choose how the storage is mounted.

Either the `code.py` file OR the computer can have read-write permissions, but not both. The other will be read-only. Data logging into CSV files requires read-write access for `code.py`. Updating code by dragging and dropping `code.py` requires read-write access for the computer.

A jumper from pin 3 (GND) allows `code.py` read-write access always, to allow REPL to be used without blocking data logging.

With <USB> AND <no jumper> (pin 4 is pulled high):

- A computer can edit/update the program through file dropping.
- `code.py` CANNOT write test data CSV files to the storage drive.

With <no USB> OR <USB + jumper> (pin 4 is pulled low):

- A computer CANNOT edit/update the program through file dropping.
- `code.py` can write test data CSV files to the storage drive.

```
...
```

```
import board
import digitalio
import storage
```

```
RWswitch = digitalio.DigitalInOut(board.GP2)
RWswitch.direction = digitalio.Direction.INPUT
```

```
# Mount root as RW if GP2 is pulled low or as read only if high.
# In this case, we mean mounting for code.py not for USB.
storage.remount("/", RWswitch.value)
```

## Appendix D – Ground test data

The following is the contents of TestData.csv, which was produced by the control system software when the sampling altitude was set to -100m and the apparatus was on the test bench at 100m elevation. On the next page, the same data is shown after importing the CSV file into a spreadsheet.

### NEW TEST DATA FOLLOWS

#### Pre-Launch Stage:

Launching from altitude (m):,-16.289337,@CPU time (ms):,2962755.000000,To collect a sample of (L):,1000.000000,At altitude (m):,-100.000000  
CPU uptime (ms),Altitude (m),Pressure (hPa),Temperature (C),Humidity (%),MAF sensor (kg/s),VAF (L/s),Sampled volume (L)  
2966143.000000,-17.852776,1015.409912,26.389999,36.331696,0.000000,0.000000,0.000000,

#### Ascent Stage:

Ascending @CPU time (ms):,2966521.000000  
CPU uptime (ms),Altitude (m),Pressure (hPa),Temperature (C),Humidity (%),MAF sensor (kg/s),VAF (L/s),Sampled volume (L)  
2967958.000000,-18.782394,1015.459961,26.540001,36.400360,0.000000,0.000000,0.000000,

#### Sampling Stage:

Start sampling @ time (ms):,2968339.000000  
Fast sampling on = no logs.  
2998785.000000,-  
19.859894,1015.520020,26.949997,35.782379,0.000000,0.000000,1005.723633,

#### Ascent Stage:

Ascending @CPU time (ms):,2999165.000000  
CPU uptime (ms),Altitude (m),Pressure (hPa),Temperature (C),Humidity (%),MAF sensor (kg/s),VAF (L/s),Sampled volume (L)  
3000603.000000,-  
19.268326,1015.510010,26.709999,35.751862,0.000000,0.000000,1005.723633,

#### Recovery Stage:

Started recovery @CPU time (ms):,3000978.000000  
Pre-Launch took (s):,36.409988  
Ascent took (s):,1.354000  
Descent took (s):,0.459000  
Stayed airbourne for (s):,1.813000  
Peaked at altitude(m):,0.000000  
Collected sample of (L):,1005.723633  
Altitude window (m):,-17.620377,to (m):,-19.120430  
Sampling took (s):,30.199997

The following is the contents of the same TestData.csv after importing the CSV file into a spreadsheet.

NEW TEST DATA FOLLOWS							
Pre-Launch Stage:							
Launching from altitude (m):	-16.289337	@CPU time (ms):	2962755	To collect a sample of (L):	1000	At altitude (m):	-100
CPU uptime (ms)	Altitude (m)	Pressure (hPa)	Temperature (C)	Humidity (%)	MAF sensor (kg/s)	VAF (L/s)	Sampled volume (L)
2966143	-17.852776	1015.409912	26.389999	36.331696	0	0	0
Ascent Stage:							
Ascending @CPU time (ms):	2966521						
CPU uptime (ms)	Altitude (m)	Pressure (hPa)	Temperature (C)	Humidity (%)	MAF sensor (kg/s)	VAF (L/s)	Sampled volume (L)
2967958	-18.782394	1015.459961	26.540001	36.40036	0	0	0
Sampling Stage:							
Start sampling @ time (ms):	2968339						
Fast sampling on = no logs.							
2998785	-19.859894	1015.52002	26.949997	35.782379	0	0	1005.723633
Ascent Stage:							
Ascending @CPU time (ms):	2999165						
CPU uptime (ms)	Altitude (m)	Pressure (hPa)	Temperature (C)	Humidity (%)	MAF sensor (kg/s)	VAF (L/s)	Sampled volume (L)
3000603	-19.268326	1015.51001	26.709999	35.751862	0	0	1005.723633
Recovery Stage:							
Started recovery @CPU time (ms):	3000978						
Pre-Launch took (s):	36.409988						
Ascent took (s):	1.354						
Descent took (s):	0.459						
Stayed airborne for (s):	1.813						
Peaked at altitude(m):	0						
Collected sample of (L):	1005.723633						
Altitude window (m):	-17.620377	to (m):	-19.12043				
Sampling took (s):	30.199997						

## Appendix E – Vacuum chamber test data

The following is the contents of TestData.csv, which was produced by the control system software when the sampling altitude was set to 10,000m and the apparatus was inside a vacuum chamber which continuously decreased pressure to a minimum of 68.4hPa. The following is the contents of TestData.csv after importing the CSV file into a spreadsheet.

NEW TEST DATA FOLLOWS

Pre-Launch Stage:

Launching from altitude (m):	78.266998	@CPU time (ms):	536811264	To collect a sample of (L):	1000	At altitude (m):	10000
CPU uptime (ms)	Altitude (m)	Pressure (hPa)	Temperature (C)	Humidity (%)	MAF sensor (kg/s)	VAF (L/s)	Sampled volume (L)
536814624	76.925415	1004	29.75	49.965424	0	0	0
536818272	76.671875	1004.049805	29.769997	49.751801	0	0	0
536821888	76.925415	1004.01001	29.659996	49.416107	0	0	0
536825632	77.506409	1003.949951	29.719994	49.767059	0	0	0
536829280	77.759949	1004	29.699997	50.346893	0	0	0
536832896	78.266998	1003.949951	29.729996	49.644989	0	0	0
536836544	78.266998	1003.959961	29.739998	48.607391	0	0	0
536840160	77.675446	1004.039795	29.709999	48.149628	0	0	0
536843776	77.252869	1004.01001	29.599998	49.63736	0	0	0
536847456	77.590912	1003.969971	29.579994	49.210114	0	0	0
536851040	66.699677	1005.379883	29.529999	49.721283	0	0	0
536854592	66.53064	1005.289795	29.540001	49.118561	0	0	0
536858208	412.832275	963.809814	29.619995	48.256439	0	0	0

Ascent Stage:

Ascending @CPU time (ms):	536858624						
CPU uptime (ms)	Altitude (m)	Pressure (hPa)	Temperature (C)	Humidity (%)	MAF sensor (kg/s)	VAF (L/s)	Sampled volume (L)
536860256	754.676025	925.069824	29.699997	47.073883	0	0	0
536862048	1041.630371	893.449951	29.699997	46.005768	0	0	0
536863840	1305.978516	865.209961	29.649994	44.907135	0	0	0
536865664	1555.61084	839.219971	29.629997	43.915314	0	0	0
536867456	1798.187012	814.47998	29.57	43.030304	0	0	0
536869344	2048.231445	789.619873	29.57	42.046112	0	0	0
251	2282.102539	766.889893	29.529999	41.054291	0	0	0
2044	2509.043945	745.359863	29.509995	40.17691	0	0	0
3839	2731.421875	724.849854	29.5	39.345306	0	0	0
5636	2951.286133	704.899902	29.439995	38.597626	0	0	0
7432	3165.962891	685.899902	29.399994	37.987274	0	0	0
9230	3376.541992	667.629883	29.360001	37.430328	0	0	0
11098	3592.887695	649.389893	29.329994	36.789459	0	0	0
12895	3796.694336	632.369873	29.279999	35.820526	0	0	0
14702	4000.607422	615.909912	29.309998	35.049957	0	0	0
16498	4198.572266	600.169922	29.269997	34.073395	0	0	0
18297	4392.089844	585.099854	29.239998	33.493561	0	0	0
20092	4585.576172	570.25	29.129997	32.74588	0	0	0
22006	4781.259766	555.719971	29.18	31.914276	0	0	0
23805	4967.625	542.079834	29.159996	31.204742	0	0	0
25602	5152.195313	528.849854	29.110001	30.586761	0	0	0
27404	5333.058594	516.119873	29.129997	30.220551	0	0	0
29199	5511.576172	503.809937	29.119995	29.46524	0	0	0
30997	5689.322266	491.469971	28.979996	28.793854	0	0	0
32914	5878.234375	479.269897	29.099998	28.33609	0	0	0
34712	6052.978516	467.919922	29.07	27.763885	0	0	0
36507	6225.951172	456.899902	29.049995	27.092499	0	0	0
38302	6395.066406	446.329956	29.049995	26.672882	0	0	0
40100	6563.833984	435.969971	29.029999	26.169342	0	0	0
41909	6732.515625	425.939941	28.899994	25.772614	0	0	0
43818	6907.90625	415.589966	29.019997	25.116486	0	0	0
45615	7071.507813	406.119995	29.019997	24.834198	0	0	0
47407	7232.720703	396.959961	29	24.269623	0	0	0
49200	7392.804688	388.029907	29.009995	23.956818	0	0	0
50999	7549.824219	379.429932	29.009995	23.407501	0	0	0
52794	7707.679688	370.929932	28.899994	23.140472	0	0	0
54695	7872.884766	362.219971	29.009995	22.713226	0	0	0
56498	8024.191406	354.369995	29	22.263092	0	0	0
58294	8176.255859	346.75	28.989998	21.774811	0	0	0
60088	8324.625	339.219971	28.979996	21.240753	0	0	0
61883	8470.921875	332.029907	28.969994	21.019501	0	0	0
63676	8615.011719	325.059937	28.969994	20.500702	0	0	0
65589	8769.050781	317.859985	28.939995	19.997162	0	0	0
67383	8911.484375	311.119995	28.959999	19.737762	0	0	0
69172	9051.769531	304.789917	28.939995	19.279999	0	0	0
70975	9191.792969	298.47998	28.939995	19.104523	0	0	0

72775	9326.945313	292.380005	28.949997	18.829865	0	0	0
74573	9461.8125	286.5	28.939995	18.425507	0	0	0
76484	9604.351563	280.5	28.93	17.822784	0	0	0
78280	9736.261719	274.939941	28.919998	17.281097	0	0	0
80075	9867.675781	269.599976	28.899994	16.937775	0	0	0
81870	9995.570313	264.269897	28.919998	17.029327	0	0	0
83665	10122.76953	259.159912	28.93	16.87674	0	0	0

Sampling Stage:

Start sampling @ time (ms): 84058

Fast sampling on = no logs.

95823	10943.12891	228.089966	29.119995	15.121979	0	0	1000.286133
-------	-------------	------------	-----------	-----------	---	---	-------------

Ascent Stage:

Ascending @CPU time (ms): 96203

CPU uptime (ms)

Altitude (m)	Pressure (hPa)	Temperature (C)	Humidity (%)	MAF sensor (kg/s)	VAF (L/s)	Sampled volume (L)
97725	11066.51563	223.690002	29.019997	14.908356	0	0
99514	11181.25781	219.769958	29.009995	14.664215	0	0
101308	11295.39063	215.709961	28.93	14.343781	0	0
103215	11409.75391	211.809998	28.979996	13.908905	0	0
105005	11519.82813	208.209961	28.969994	13.504547	0	0
106803	11625.12891	204.619995	28.979996	13.153595	0	0
108599	11731.57813	201.130005	28.979996	13.031525	0	0
110392	11836.40234	197.869995	28.979996	12.787384	0	0
112293	11942.69531	194.380005	28.959999	12.344879	0	0
114083	12046.94922	191.219971	28.969994	12.138885	0	0
115869	12148.26563	188.169983	28.969994	12.047333	0	0
117659	12248.98438	185.119995	28.969994	11.871857	0	0
119456	12352	182.289978	28.949997	11.604828	0	0
121249	12452.40625	179.459961	28.969994	11.459869	0	0
123157	12558.42188	176.519958	28.969994	11.353058	0	0
124948	12654	173.799988	28.979996	11.253876	0	0
126745	12751.04688	171.179993	28.979996	11.116547	0	0
128542	12845.78125	168.679993	28.979996	10.902924	0	0
130332	12941.53125	166.169983	28.979996	10.582489	0	0
132125	13034.82031	163.759949	28.869995	10.300201	0	0
134042	13133.46875	161.380005	28.959999	10.017914	0	0
135832	13225.17969	159.089966	28.969994	9.789032	0	0
137622	13318.22656	156.809998	28.969994	9.68985	0	0
139412	13407.64844	154.630005	28.979996	9.613556	0	0
141204	13494.13281	152.559998	28.979996	9.544891	0	0
142995	13582.64844	150.369995	28.909996	9.476227	0	0
144900	13675.4375	148.309998	28.979996	9.399933	0	0
146695	13755.98438	146.349976	28.979996	9.30838	0	0
148493	13842.44531	144.5	28.979996	9.201569	0	0
150284	13920.17188	142.649963	28.989998	9.094757	0	0
152079	14003.88281	140.909973	28.989998	8.942169	0	0
153980	14088.6875	139.059998	28.989998	8.789581	0	0
155780	14169.48438	137.320007	29	8.644623	0	0
157574	14246.57813	135.579956	29	8.431	0	0
159371	14324.14063	133.940002	29	8.217377	0	0
161172	14403.13281	132.309998	29	8.064789	0	0
162983	14477.22656	130.779968	28.989998	7.92746	0	0
164880	14557.64063	129.259949	28.989998	7.782501	0	0
166679	14628.52344	127.72998	29	7.67569	0	0
168483	14705.27344	126.319977	29	7.584137	0	0
170282	14777.82031	124.899994	29	7.477325	0	0
172083	14845.47656	123.599976	29	7.339996	0	0
173882	14913.33594	122.289978	29	7.187408	0	0
175795	14988.28906	120.869995	28.979996	7.019562	0	0
177602	15058.21094	119.569977	29	6.889862	0	0
179403	15122.39844	118.369995	29	6.775421	0	0
181200	15187.78906	117.169983	29	6.683868	0	0
183001	15247.76563	115.970001	29	6.592316	0	0
184799	15314.44531	114.879974	28.879997	6.485504	0	0
186722	15381.82813	113.690002	29	6.40921	0	0
188520	15443.64844	112.599976	29.009995	6.348175	0	0
190327	15506.07813	111.509979	28.989998	6.302399	0	0
192131	15562.14844	110.419983	29.009995	6.271881	0	0
193930	15619.29688	109.440002	29.019997	6.241364	0	0
195730	15676.95313	108.449982	28.919998	6.195587	0	0
197646	15741.71094	107.47998	29.019997	6.126923	0	0
199444	15800.48438	106.5	29.019997	6.050629	0	0
201247	15853.13281	105.629974	29.019997	5.966705	0	0
203051	15906.20313	104.649994	29.019997	5.882782	0	0
204863	15960.32031	103.889984	29.019997	5.798859	0	0
206659	16014.29688	102.910004	29.019997	5.707306	0	0
208537	16068.72656	102.139984	29.019997	5.623383	0	0
210337	16123.60938	101.269989	29.019997	5.562347	0	0
212131	16171.95313	100.509979	29.029999	5.508942	0	0
213932	16220.65625	99.75	29.029999	5.447906	0	0
215737	16270.38281	98.98999	29.029999	5.386871	0	0
217656	16319.84375	98.220001	29.019997	5.333466	0	0
219472	16369.69531	97.459991	29.029999	5.27243	0	0
221276	16419.92969	96.699982	29.040001	5.203766	0	0
223078	16463.875	96.039978	29.040001	5.15036	0	0
224878	16507.46094	95.389984	29.029999	5.089325	0	0

226677	16551.32813	94.73999	29.040001	5.02829	0	0	1000.286133
228597	16603.70313	93.97998	29.040001	4.974884	0	0	1000.286133
230398	16640.67969	93.429993	29.040001	4.921478	0	0	1000.286133
232195	16685.48438	92.779999	29.040001	4.875702	0	0	1000.286133
234004	16723.64063	92.119995	29.040001	4.837555	0	0	1000.286133
235802	16769.02344	91.579987	29.049995	4.799408	0	0	1000.286133
237602	16807.67188	91.039978	29.049995	4.761261	0	0	1000.286133
239505	16845.85156	90.379974	29.049995	4.715485	0	0	1000.286133
241302	16884.26563	89.839996	29.059998	4.684967	0	0	1000.286133
243101	16931.52344	89.289978	29.059998	4.639191	0	0	1000.286133
244898	16962.5	88.75	29.059998	4.601044	0	0	1000.286133
246696	17002.34375	88.309998	29.059998	4.570526	0	0	1000.286133
248500	17033.66406	87.769989	29.059998	4.540009	0	0	1000.286133
250418	17073.94531	87.329987	29.059998	4.501862	0	0	1000.286133
252216	17113.75781	86.789978	29.059998	4.463715	0	0	1000.286133
254010	17146.36719	86.349976	29.059998	4.448456	0	0	1000.286133
255810	17186.61719	85.809998	29.059998	4.417938	0	0	1000.286133
257610	17219.60156	85.369995	29.059998	4.379791	0	0	1000.286133
259412	17252.01563	84.940002	28.959999	4.349274	0	0	1000.286133
261323	17285.34375	84.5	29.040001	4.326385	0	0	1000.286133
263129	17318.84375	84.059998	29.07	4.295868	0	0	1000.286133
264926	17351.75781	83.629974	29.07	4.27298	0	0	1000.286133
266727	17385.59375	83.190002	29.07	4.250092	0	0	1000.286133
268528	17410.32813	82.759979	29.07	4.219574	0	0	1000.286133
270328	17444.49219	82.319977	28.979996	4.196686	0	0	1000.286133
272240	17478.83594	82	29.07	4.173798	0	0	1000.286133
274036	17503.9375	81.669983	29.079994	4.150909	0	0	1000.286133
275844	17538.60156	81.22998	29.07	4.135651	0	0	1000.286133
277640	17563.92188	80.910004	29.079994	4.112762	0	0	1000.286133
279438	17590.15625	80.470001	29.079994	4.097504	0	0	1000.286133
281231	17625.29688	80.139984	29.019997	4.074615	0	0	1000.286133
283138	17650.96875	79.819977	29.079994	4.059357	0	0	1000.286133
284937	17677.54688	79.48999	29.079994	4.036469	0	0	1000.286133
286735	17704.23438	79.160004	29.089996	4.028839	0	0	1000.286133
288531	17730.23438	78.949982	29.089996	4.01358	0	0	1000.286133
290336	17748.16406	78.509979	29.089996	3.998322	0	0	1000.286133
292130	17784.1875	78.179993	29.019997	3.983063	0	0	1000.286133
294033	17801.44531	77.970001	29.099998	3.960175	0	0	1000.286133
295829	17828.66406	77.639984	29.099998	3.952545	0	0	1000.286133
297628	17855.99219	77.309998	29.099998	3.944916	0	0	1000.286133
299425	17874.28125	77.089996	29.099998	3.929657	0	0	1000.286133
301219	17900.97656	76.769989	29.099998	3.914398	0	0	1000.286133
303129	17928.60938	76.549988	29.099998	3.899139	0	0	1000.286133
304934	17947.10938	76.220001	29.099998	3.89151	0	0	1000.286133
306732	17965.66406	75.899994	29.099998	3.883881	0	0	1000.286133
308522	17992.73438	75.789978	29.099998	3.868622	0	0	1000.286133
310317	18011.41406	75.459991	29.099998	3.853363	0	0	1000.286133
312113	18039.53906	75.129974	29.099998	3.853363	0	0	1000.286133
314025	18057.50781	74.919983	29.099998	3.838104	0	0	1000.286133
315824	18085.82813	74.589996	29.110001	3.822845	0	0	1000.286133
317622	18104.78906	74.369995	29.110001	3.815216	0	0	1000.286133
319425	18123.79688	74.149994	29.119995	3.799957	0	0	1000.286133
321225	18141.99219	73.829987	29.110001	3.792328	0	0	1000.286133
323021	18170.69531	73.609985	29.119995	3.792328	0	0	1000.286133
324932	18189.91406	73.389984	29.119995	3.777069	0	0	1000.286133
326730	18209.17969	73.169983	29.119995	3.76944	0	0	1000.286133
328525	18228.49219	72.949982	29.119995	3.76181	0	0	1000.286133
330316	18247	72.73999	29.119995	3.76181	0	0	1000.286133
332112	18266.42969	72.519989	29.099998	3.754181	0	0	1000.286133
333918	18285.92969	72.299988	29.129997	3.746552	0	0	1000.286133
335836	18305.49219	71.970001	29.119995	3.738922	0	0	1000.286133
337632	18324.20313	71.869995	29.129997	3.746552	0	0	1000.286133
339432	18343.88281	71.759979	29.129997	3.731293	0	0	1000.286133
341229	18353.73438	71.539978	29.139999	3.731293	0	0	1000.286133
343025	18373.50781	71.319977	29.139999	3.723663	0	0	1000.286133
344819	18393.33594	71.099976	29.139999	3.723663	0	0	1000.286133
346744	18413.21875	70.889984	29.119995	3.716034	0	0	1000.286133
348542	18432.25	70.669983	29.139999	3.716034	0	0	1000.286133
350340	18442.25	70.559998	29.139999	3.708405	0	0	1000.286133
352138	18462.28906	70.339996	29.139999	3.708405	0	0	1000.286133
353937	18482.38281	70.119995	29.149994	3.708405	0	0	1000.286133
355736	18492.46875	70.009979	29.029999	3.708405	0	0	1000.286133
357658	18511.73438	69.799988	29.139999	3.700775	0	0	1000.286133
359459	18532.00781	69.579987	29.149994	3.700775	0	0	1000.286133
361265	18552.32031	69.359985	29.149994	3.693146	0	0	1000.286133
363066	18572.71094	69.25	29.149994	3.685516	0	0	1000.286133
364872	18582.92188	69.029999	29.149994	3.685516	0	0	1000.286133
366671	18602.47656	68.819977	29.049995	3.685516	0	0	1000.286133
368590	18612.73438	68.709991	29.149994	3.677887	0	0	1000.286133
370389	18633.32031	68.48999	29.149994	3.677887	0	0	1000.286133
372192	18653.96875	68.379974	29.149994	3.670258	0	0	1000.286133
373912	18502.55469	70.009979	29.110001	3.540558	0	0	1000.286133
375642	18161.10938	73.829987	29.089996	3.471893	0	0	1000.286133
377360	17739.19531	78.72998	29.059998	3.63974	0	0	1000.286133
379197	17775.16406	78.289978	29.119995	3.990692	0	0	1000.286133
380917	17775.16406	78.289978	29.129997	4.227203	0	0	1000.286133
382634	17418.85938	82.759979	29.119995	4.27298	0	0	1000.286133



## Descent Stage:

Started descent @CPU time (ms): 383023

CPU uptime (ms)	Altitude (m)	Pressure (hPa)	Temperature (C)	Humidity (%)	MAF sensor (kg/s)	VAF (L/s)	Sampled volume (L)
384422	15587.74219	111.619995	29.07		5.02066	0	0
385978	13073.3125	164.440002	29.57		7.210297	0	0
387533	11305.00391	217.029968	30		9.865326	0	0
389086	9895.191406	269.909912	30.369995		12.619537	0	0
390643	8696.214844	322.849976	30.689995		15.266937	0	0
392200	7647.275391	375.719971	30.969994		17.815155	0	0
393889	6650.677734	432.51001	31.269997		20.40152	0	0
395449	5839.433594	483.5	31.489998		22.690338	0	0
397006	5115.517578	533.139893	31.739998		24.742645	0	0
398564	4467.251953	580.899902	31.93		26.649994	0	0
400119	3885.556641	626.689941	32.080002		28.412384	0	0
401680	3365.34375	670.169922	32.209991		30.067963	0	0
403357	3022.506836	700.030029	32.309998		31.509918	0	0
404916	2570.334961	741.219971	32.339996		33.012909	0	0
406474	2175.239258	778.679932	32.449997		34.332794	0	0
408034	1824.723145	813.299805	32.539993		35.622162	0	0
409592	1520.211914	844.26001	32.610001		36.758942	0	0
411151	1252.821289	872.219971	32.580002		37.834686	0	0
412839	1002.491699	899.079834	32.699997		38.864655	0	0
414398	809.333008	920.22998	32.73999		39.67337	0	0
415954	648.161621	938.159912	32.759995		40.398163	0	0
417506	514.899414	953.209961	32.75		41.100067	0	0
419063	406.082031	965.719971	32.75		41.748566	0	0
420618	318.191406	976.069824	32.669998		42.274994	0	0
422298	245.259155	984.339844	32.729996		42.694611	0	0
423855	195.112915	990.219971	32.709991		43.053192	0	0
425405	158.12915	994.599854	32.689987		43.381256	0	0

## Recovery Stage:

Started recovery @CPU time (ms): 425797

Pre-Launch took (s): -536715

Ascent took (s): 286.819946

Descent took (s): 42.773987

Stayed airborne for (s): 329.593994

Peaked at altitude(m): 18643.64063

Collected sample of (L): 1000.286133

Altitude window (m): 10151.92578 to (m): 10923.683594

Sampling took (s): 11.438999

## Appendix F – Calibration output

The following is the contents of Calibration.txt, which was produced by the control system software when set to calibration mode.

```
MAF calibration offset = MAFcalos = 16470.781250,  
MAF value = MAFcalVal = 36022.625000,  
Pressure (hPa/mbar) = MAFcalP = 1015.137695,  
Temperature (C) = MAFcalT = 28.042061,  
Vol flow rate (L/min) = MAFcaldvdt = Flow bench data  
No flow bench? Fan rated flow is kinda ok
```

## REFERENCES

Adafruit 2023a, *Adafruit MS8607 Pressure Humidity Temperature PHT Sensor*, New York, New York, USA, viewed 4 July 2023, <<https://www.adafruit.com/product/4716>>.

Adafruit 2023b, *Adafruit\_CircuitPython\_MS8607*, Adafruit, New York, New York, USA, viewed 5 July 2023, <[https://github.com/adafruit/Adafruit\\_CircuitPython\\_MS8607](https://github.com/adafruit/Adafruit_CircuitPython_MS8607)>.

Aerogel Technologies 2023, *Thermal Wrap 6mm Blanket*, Aerogel Technologies, Boston, Massachusetts, USA, viewed 4 July 2023, <<http://www.buyaerogel.com/product/thermal-wrap/>>.

Anderson, K, Brander, C, Carrasco, E, Courville, R, de la Guardia, A, Fensler, J, Giron, L, Oberlies, J, Pankiewicz, W & Wachter, G, 2023, “Compact Lightweight Aerial Sensor System (CLASSy)”, contractor or grantee report, NASA, USA, viewed 6 October 2023 <<https://ntrs.nasa.gov/citations/20230011971>>.

Beaurepaire, M, Rachid, D, Gasperi, J & Tassin, B, “Microplastics in the atmospheric compartment: a comprehensive review on methods, results on their occurrence and determining factors”, *Atmospheric Pollution Research*, volume 13, issue 10, Oct 2022, 101550, viewed 28 September 2023 <<https://doi.org/10.1016/j.cofs.2021.04.010>>.

Beckingham, B, Apintiloaiei, A, Moore, C & Brandes, J, “Hot or not: systematic review and laboratory evaluation of the hot needle test for microplastic identification”, *Microplastics and Nanoplastics*, volume 3, April 2023, Article number 8, viewed 2 October 2023 <<https://doi.org/10.1186/s43591-023-00056-4>>.

BlueKitchen 2023, *btstack*, BlueKitchen, Zurich, Switzerland, viewed 15 August 2023, <<https://github.com/bluekitchen/btstack#supported-protocols-and-profiles>>.

Bryan, NC, Stewart, M, Granger, D, Guzik, TG & Christner, BC, “A method for sampling microbial aerosols using high altitude balloons”, *Journal of Microbiological Methods*, volume 107, December 2014, Pages 161-168, viewed 6 October 2023 <<https://doi.org/10.1016/j.mimet.2014.10.007>>.

Cadex 2023, *Types of Lithium-ion*, Cadex, Vancouver, British Colombia, Canada, viewed 5 July 2023, <<https://batteryuniversity.com/article/bu-205-types-of-lithium-ion>>.

CircuitPython 2023, *MicroPython & CircuitPython License*, CircuitPython, viewed 14 August 2023, <<https://docs.circuitpython.org/en/latest/docs/LICENSE.html>>.

Conkle, JL, Baez Del Valle, CD & Turner, JW, “Are We Underestimating Microplastic Contamination in Aquatic Environments?”, *Environmental Management*, volume 61, January 2018, viewed 2 October 2023 <<https://doi.org/10.1007/s00267-017-0947-8>>.

Derwent, R, Simmonds, P, O’Doherty, S, Manning, A, Collins, W & Stevenson, D, “Global environmental impacts of the hydrogen economy”, *International Journal of Nuclear Hydrogen Production and Applications*, volume 1, issue 1, May 2006, Pages 55-67, viewed 29 September 2023 <<https://doi.org/10.1504/IJNHPA.2006.009869>>.

Dris, R, Gasperi, J, Saad, M, Mirande, C & Tassin, B, “Synthetic fibers in atmospheric fallout: A source of microplastics in the environment?”, *Marine Pollution Bulletin*, volume 104, issues 1–2, 15 March 2016, Pages 290-293, viewed 28 September 2023 <<https://doi.org/10.1016/j.marpolbul.2016.01.006>>.

The Engineering ToolBox 2003, *U.S. Standard Atmosphere vs. Altitude*, The Engineering ToolBox, viewed 4 July 2023, <[https://www.engineeringtoolbox.com/standard-atmosphere-d\\_604.html](https://www.engineeringtoolbox.com/standard-atmosphere-d_604.html)>.

Finetech Research and Innovation 2023, *Glass fiber filter – MGA*, Finetech Research and Innovation, New Taipei City, Taiwan, viewed 4 July 2023, <<https://en.finetech-filter.com/product-detail-1781196.html>>.

Fonseca, J, Carreiro, F, Silva, V, Evtiouguina, M, Marques, M 2003, “ $\mu$ VOC – A lightweight environmental data and air samples acquisition system to install in captive balloons”, IEEE International Symposium on Industrial Electronics, 9-11 June 2003, Rio de Janeiro, Brazil, viewed 6 October 2023 <<https://doi.org/10.1109/ISIE.2003.1267316>>.

Gigault, J, Ter Halle, A, Baudrimont, M, Pascal, P, Gauffre, F, Phi, TL, El Hadri, H, Grassl, B & Reynaud, S, “Current opinion: What is a nanoplastic?”, *Environmental Pollution*, volume 235, April 2018, Pages 1030-1034, viewed 3 October 2023 <<https://doi.org/10.1016/j.envpol.2018.01.024>>.

Gree Altairnano New Energy 2023, *Yinlong LTO Batteries*, Gree Altairnano New Energy, Singapore, viewed 5 July 2023, <<https://www.yinlong.energy/yinlong-battery>>.

Habeck, JB, Flaten, JA & Candler, GV 2020, “High-altitude balloon measurements of atmospheric particulates”, AIAA Scitech 2020 Forum, 6-10 January 2020, Orlando, Florida, USA, viewed 6 October 2023 <<https://doi.org/10.2514/6.2020-1794>>.

Kelly, A, Lannuzel, D, Rodemann, T, Meiners, KM & Auman, HJ, “Microplastic contamination in east Antarctic sea ice”, *Marine Pollution Bulletin*, volume 154, May 2020, viewed 2 October 2023 <<https://doi.org/10.1016/j.marpolbul.2020.111130>>.

Lateran, S, Sedan, MF, Harithuddin, ASM & Azrad, S 2016, “Development of unmanned aerial vehicle (UAV) based high altitude balloon (HAB) platform for active aerosol sampling”, AEROTECH VI – Innovation in Aerospace Engineering and Technology, 8-9 November 2016, Kuala Lumpur, Malaysia, viewed 6 October 2023 <<https://doi.org/10.1088/1757-899X/152/1/012018>>.

Leslie, HA, Van Velzen, MJM, Brandsma, SH, Vethaak, AD, Garcia-Vallejo, JJ & Lamoree, MH, “Discovery and quantification of plastic particle pollution in human blood”, *Environment International*, volume 163, May 2022, 107199, viewed 29 September 2023 <<https://doi.org/10.1016/j.envint.2022.107199>>.

Ma, S, Jiang, M, Tao, P, Song, C, Wu, J, Wang, J, Deng, T & Shang, W, “Temperature effect and thermal impact in lithium-ion batteries: A review”, *Progress in Natural Science: Materials International*, volume 28, issue 6, December 2018, Pages 653-666, viewed 2 October 2023 <<https://doi.org/10.1016/j.pnsc.2018.11.002>>.

National Weather Service 2023, *Pressure Altitude*, viewed 1 July 2023, <<https://www.weather.gov/media/epz/wxcalc/pressureAltitude.pdf>>.

O’Brien, S, Rauert, C, Ribeiro, F, Okoffo, ED, Burrows, SD, O’Brien, JW, Wang, X, Wright, SL & Thomas, KV, “There's something in the air: A review of sources, prevalence and behaviour of microplastics in the atmosphere”, *Science of The Total Environment*, volume 874, 20 May 2023, 162193, viewed 28 September 2023 <<https://doi.org/10.1016/j.scitotenv.2023.162193>>.

Postma, JV, “An inexpensive atmospheric microplastic collector for use in remote areas”, *Atmospheric Pollution Research*, volume 13, issue 10, Oct 2022, 101550, viewed 28 September 2023 <<https://doi.org/10.1016/j.apr.2022.101550>>.

Raspberry Pi Foundation 2023, *Raspberry Pi Pico*, Raspberry Pi Foundation, Cambridge, England, UK, viewed 1 July 2023, <<https://www.raspberrypi.com/products/raspberry-pi-pico/>>.

Renesas 2023, *Temperature Ranges*, Renesas, Tokyo, Japan, viewed 4 July 2023, <<https://www.renesas.com/us/en/support/technical-resources/temperature-ranges>>.

Rosso, B, Corami, F, Barbante, C & Gambaro, A, “Quantification and identification of airborne small microplastics (<100 µm) and other microlitter components in atmospheric aerosol via a novel elutriation and oleo-extraction method”, *Environmental Pollution*, volume 318, 1 Feb 2023, 120889, viewed 28 September 2023 <<https://doi.org/10.1016/j.envpol.2022.120889>>.

RS Group 2023a, *SMC PFMB7 Series Digital Flow Switch Flow Switch for Dry Air*, RS Group, London, England, UK, viewed 2 July 2023, <<https://au.rs-online.com/web/p/flow-sensors/2704310>>.

RS Group 2023b, *Everything You Need To Know About Flow Switches*, RS Group, London, England, UK, viewed 8 July 2023, <<https://au.rs-online.com/web/generalDisplay.html?id=ideas-and-advice/flow-switches-guide>>.

SilverStone 2023, *FHS 80X*, SilverStone, New Taipei City, Taiwan, viewed 1 July 2023, <[https://www.silverstonetek.com/en/product/info/fans/FHS\\_80X/](https://www.silverstonetek.com/en/product/info/fans/FHS_80X/)>.

Song, Z, Liu, K, Wang, X, Wei, N, Zong, C, Li, C, Jiang, C, He, Y, & Li, D, “To what extent are we really free from airborne microplastics?”, *Science of The Total Environment*, volume 754, 1 February 2021, 142118, viewed 2 October 2023 <<https://doi.org/10.1016/j.scitotenv.2020.142118>>.

Sparkfun 2023, *SparkFun Solder-able Breadboard*, Sparkfun, Niwot, Colorado, USA, viewed 14 August 2023, <<https://www.sparkfun.com/products/12070>>.

StratoStar 2022, *How Much Weight Can a High Altitude Weather Balloon Carry?*, StratoStar, Noblesville, Indiana, USA, viewed 6 July 2023, <<https://stratostar.com/how-much-weight-can-a-high-altitude-weather-balloon-carry/>>.

TE Connectivity 2023, *MS8607-02BA01 PHT Combination Sensor*, viewed 1 July 2023, <[https://www.te.com/commerce/DocumentDelivery/DDEController?Action=showdoc&DocId=Data+Sheet%7FMS8607-02BA01%7FB3%7Fpdf%7FEnglish%7FENG\\_DS\\_MS8607-02BA01\\_B3.pdf](https://www.te.com/commerce/DocumentDelivery/DDEController?Action=showdoc&DocId=Data+Sheet%7FMS8607-02BA01%7FB3%7Fpdf%7FEnglish%7FENG_DS_MS8607-02BA01_B3.pdf)>.

Wilson, C 2022, “The replication crisis has spread through science – can it be fixed?”, *NewScientist*, viewed 30 August 2023, <<https://www.newscientist.com/article/mg25433810-400-the-replication-crisis-has-spread-through-science-can-it-be-fixed/>>.