March 14, 2019

Dr. Craig Scratchley
School of Engineering Science
Simon Fraser University
Burnaby, BC, V5A 1S6

Re: ENSC 405 Design Specification for *DeepBreath*

Dear Dr. Scratchley,

This document proposes our design specifications for *DeepBreath*, a smart training mask. *DeepBreath* will measure fitness metrics through a user's breath, and present them in a meaningful way on their devices to help them visualize and meet their fitness goals. We are striving to create this product with low costs to significantly improve the accessibility of this type of information to regular consumers.

The detailed design specifications for the different subsystems of *DeepBreath* are outlined in this document, including circuit schematics and background theory for measuring metrics about a user's breath, the hardware used for Bluetooth communication with a microcontroller, the design of our companion phone application, and the physical mask design. This document also explains the flow of data and communication across these subsystems and how they interact with each other.

Our diligent team consists of 4 senior engineering students in the SFU program - Alex Boswell, Austin Phillips, Jim Park, and Sulaiman Omar Marouf. We plan to meet consistently and hold each other accountable for this project, and are confident we will produce an impressive product.

Thank you for reviewing this design specifications document and future documents. If you have any questions or concerns, please contact us through GitLab or contact Jim Park, our Chief Communications Officer, at jim_park@sfu.ca.

Sincerely,
Austin Phillips
Chief Executive Officer
Exotic

Exotic

DESIGN SPECIFICATIONS

For

DeepBreath

Get more out of each breath

Team Members	Austin Phillips	CEO
	Jim Park	CCO
	Sulaiman Omar	CIO
	Alex Boswell	CTO
Submitted To	Dr. Craig Scratchley	ENSC405W
	Dr. Andrew Rawicz	ENSC440
	School of Engineering Science	
	Simon Fraser University	
Contact Person	Jim Park	
	jim_park@sfu.ca	
	604-880-3682	

Issue Date	March 14, 2019
Revision Number	1.0.0

Abstract

This document will provide detailed specifications for the design of Exotic's smart mask project, *DeepBreath*. The system consists of 3 main components:

1. Gas sensors that sample air inhaled and exhaled by the user, and a microcontroller to receive and communicate this data.
2. The physical mask that forms a comfortable and safe housing for the electronics within.
3. A companion app that provides analytics, data display, and user interface for the system.

The *DeepBreath* project intends to use fitness tests that are commonly administered in labs (such as VO_2 max max measurements), and create a product cable of replicating these tests that is both portable and affordable. The main objectives, portability and affordability, leave our team with some easily identifiable constraints for our design choices; our parts need to be small and inexpensive. We kept those goals in mind while selecting our components, and were able to reach a design that we are satisfied with for our prototype. The details of this design, and those of the components within it, are described in the remainder of this document.

Table of Contents

List of Tables

List of Figures

Glossary

API An application programming interface is a subset of rules that software and applications follow in order to communicate with each other.

APK Android Package is the file format in Android OS for distribution and installation of its applications.

BLE Bluetooth Low Energy is a wireless communication standard that has lower power consumption compared to its regular Bluetooth counterpart.

combine A sports combine is a combination of various sport specific activities to determine the strength, speed, endurance, skill of an athlete. It is usually done in front of coaches who wish to evaluate athletes.

CPU Central processing unit is a electrical component that deals with computation, log and other basic operations.

GATT General Attributes, abbreviated GATT, is the data structure interface exposed to devices communicating over BLE.

memory leak Failure to properly manage memory in such a way that memory which is unneeded is accidentally released.

QA Quality Assurance refers to the process of testing, debugging and assuring that the software/product works as intended.

respiratory exchange ratio (RER) The ratio of oxygen to carbon dioxide in a person's exhale breath, which is used to determine information about the body's internal processes.

VO_2 max Commonly known as the maximum oxygen consumption, which measure the amount maximum amount of oxygen a person may use during a defined period of physical activity.

1 Introduction

1.1 Scope

This document intends to outline the technical design specifications for the *DeepBreath* training mask. These specifications will describe the various design choices that were made in accordance to the functional requirements for the product, as outlined in the *DeepBreath* Functional Requirements Document.[1] Each major system and subsystem sections will provide detailed justifications for these design decisions, and the specific design processes on how the corresponding systems shall be implemented.

The document also includes a User Interface Appearance Appendix that summarizes the requirements for *DeepBreath*'s appearance and usability. Also in the appendix are supporting test plans to verify detailed design specifications for the subsystems and components of *DeepBreath*.

1.2 Intended Audience

The intended audience of this document is for all developers responsible for implementation and design of the *DeepBreath* product. The design specifications outlined here will serve as a guideline during the development phases of the project, to be referenced by Exotic's engineers to provide a detailed overview of our design choices. This process shall ensure the correctness and quality of the final product.

1.3 Design Specification Classification

For the purposes of clarification and prioritization, the following convention will be used to label the design specifications throughout this document:

Des [Section].[Subsection].[Specification Number]-[Design Stage]

For the sake of clarity and readability, the different design stages will be abbreviated as follows:

PC - Proof-of-concept Prototype
EP - Engineering Prototype
PV - Production Version

Req 3.1.3-EP

2 System Overview

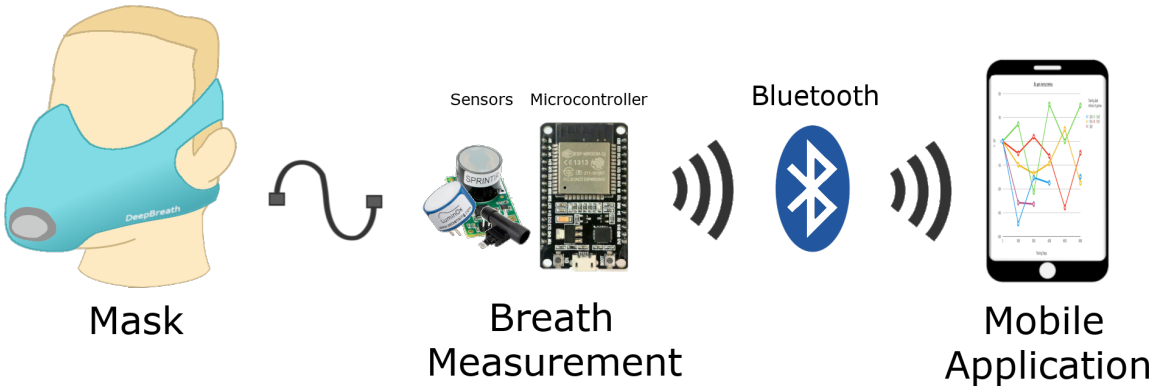

Figure 1: System overview

DeepBreath consists of three main components: the companion mobile application, the breath measurement circuits, and the physical mask. The detailed specifications for each system will be outlined in their corresponding sections throughout the document, whereas the general system and performance specifications of *DeepBreath* are listed in the table below.

Table 2.1: General Design Specifications

[Des 2.1.1-EP]	*DeepBreath* shall be able to calculate and measure the user's **VO_2 max**.
[Des 2.1.2-EP]	*DeepBreath* shall be able to calculate and measure the user's **respiratory exchange ratio (RER)**.
[Des 2.1.3-EP]	The device shall be easy to assemble and disassemble.
[Des 2.2.4-EP]	All systems shall be thoroughly and consistently tested prior to being commercially shipped, according to the established Acceptance Test Procedure in the Requirements Specification document [1] as well as the Supporting Test Documents outlined in the Appendix section of this document.

3 Mobile Application Design

The companion mobile application for *DeepBreath* is the main software component for our system. The user shall be able to view a graphical display of their fitness metrics through the app, where the raw measurements taken from the sensors on the mask are processed into meaningful data to be displayed as statistics and figures on screen.

There are three primary components within the mobile application software: the Bluetooth module responsible for receiving sensor data from the microcontroller; the data processing module responsible for transforming raw measurement data into meaningful fitness metrics; and the graphical interface module that displays these metrics as graphs and figures. The data flow between the mask/sensors and the phone application is shown in the figure below.

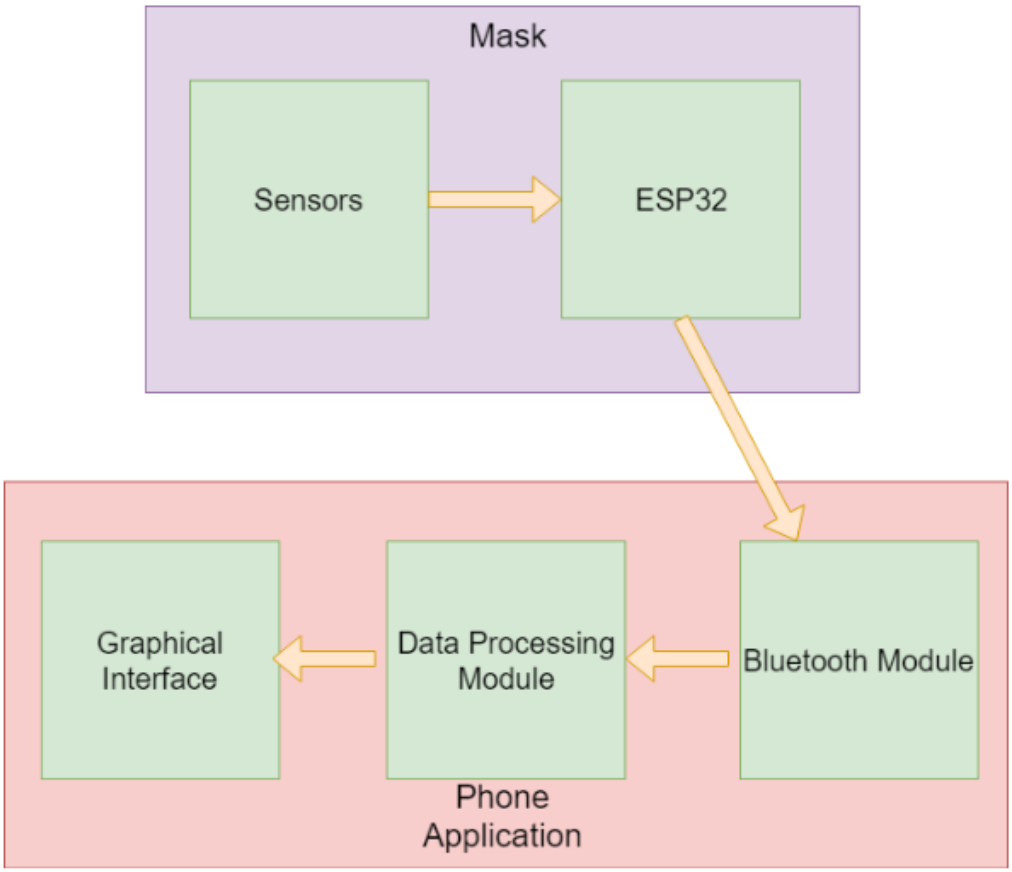

Figure 2: Data flow diagram between the microcontroller and the phone application

The app will be available for Android mobile devices. The phone application shall at minimum be compatible with devices that are of version number Android 9.0 (**API** level 28) in order to adhere to Google Play's target API level requirement for uploading the app's **APK** for commercial release. [2]

3.1 Bluetooth Module

The Bluetooth module is the subsystem within the mobile application software responsible for receiving data transmitted from the ESP32 microcontroller. The communication between the phone application and the microcontroller can be viewed as a client-server architecture, with the phone acting as the client and the microcontroller acting as the server.

Table 3.1: Bluetooth Module Design Specifications

[Des 3.1.1-PC]	The Bluetooth module shall be able to receive incoming data packets being streamed from the mask via wireless **BLE**.
[Des 3.1.2-PC]	The Bluetooth module shall act as a Bluetooth client and will listen to data packets being broadcast by the microcontroller on the mask, which acts as the Bluetooth server.
[Des 3.1.3-PC]	The Bluetooth module shall be able to explicitly ask for user permissions as well as send notifications for the various Bluetooth operations.

For our system, the Bluetooth standard that will be used is Bluetooth Low Energy (**BLE**) for ease of implementation and widely available documentation, especially for ESP32 applications. For all connected BLE devices, they follow the **GATT** (Generic Attributes) hierarchical data structure. The GATT standard defines the abstract interface over which two BLE devices can communicate with each other. The `BluetoothGatt` class from the Android API handles interactions with BLE devices from the phone, such as searching and connecting to a BLE GATT server. [3]

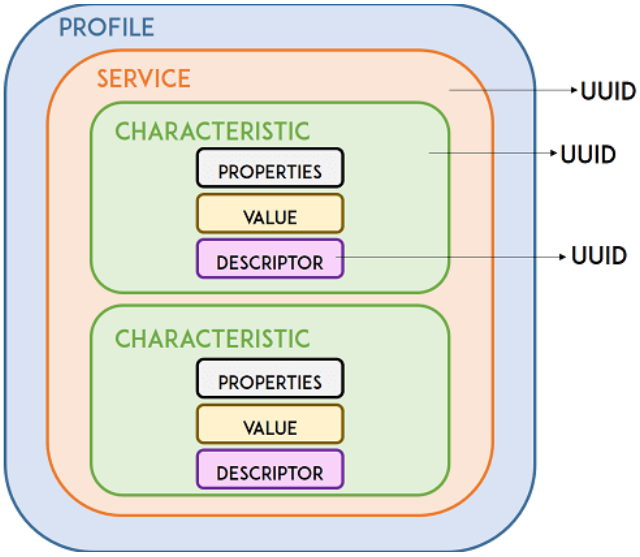

Figure 3: GATT data structure specifications [4]

The `BluetoothAdapter` class in the Android API handles all Bluetooth-pairing related operations. When activating the mobile device to be discoverable by the microcontroller, the module shall ask for explicit user permission when attempting to turn Bluetooth on for the phone. Its built-in dialog features will ensure that the user is always notified of any Bluetooth activity during the pairing process. [5]

The measurement data streamed from the microcontroller will be sent in packets. For each data packet, the relevant payload will be approximately 20 bytes. [6] The implementation for BLE packet deliverance and collection will adhere to the official specifications for BLE data packets to ensure compatibility with the underlying communication hardware.

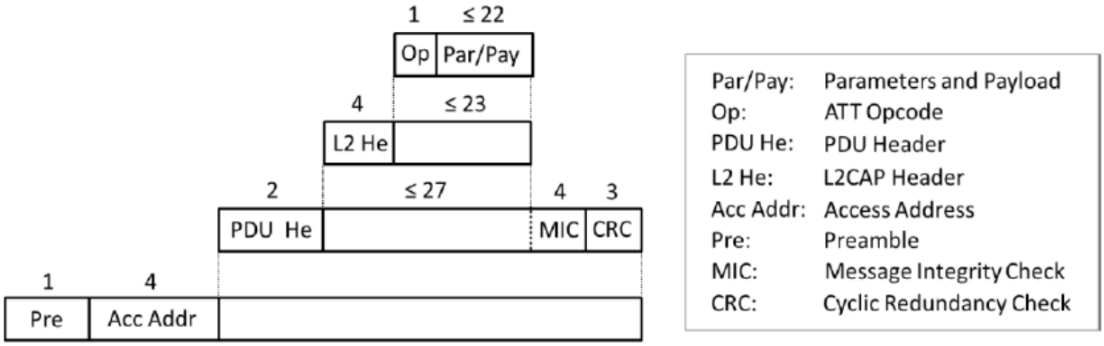

Figure 4: Detailed specifications for a BLE data packet [7]

3.2 Data Processing Module

The data processing module performs calculations on the raw breath measurement data to transform it into meaningful fitness metrics. The two primary metrics that *DeepBreath* aims to analyze are **VO_2 max** and **respiratory exchange ratio (RER)**. The `Math` library of the Android API will be sufficient to perform the operations necessary for the calculations. [8]

Table 3.2: Data Processing Module Specifications

[Des 3.2.1-EP]	The Data Processing Module shall efficiently calculate a user's VO_2 max by summing and multiplying volumes and percentages received from the microcontroller.
[Des 3.2.2-EP]	The Data Processing Module shall efficiently calculate a user's RER by summing and multiplying volumes and percentages received from the microcontroller.
[Des 3.2.3-EP]	The Data Processing Module shall provide storage and analytics for VO_2 max progress reports and resting respiratory exchange ratio trends.

The VO_2 max is calculated using summation of breaths over period of time: [9]

$$V_{O_2}\ Max = \frac{Total\ Volume\ \cdot\ (V_{O_2\_in} - V_{O_2\_out})}{Weight} \tag{3.1}$$

Where:

Total Volume is the sum of the volumes of every inhalation in the time interval;
VO_2_in is the average percentage of oxygen in all inhalations in the interval;
VO_2_out is the average percentage of oxygen exhaled;
$Total\ Volume\ \cdot\ (VO_2\_in\ -\ VO_2\_out)$ is the average percentage of oxygen in all inhalations in the interval;
Weight is simply the weight of the user in kilograms.

The RER can also be calculated using summation of breaths over period of time: [10]

$$RER = \frac{V_{CO_2}}{V_{O_2}} \tag{3.2}$$

Where:

$$V_{CO_2} = \frac{Total\ volume\ of\ air\ exhaled}{\%CO_2\ in\ exhale\ breath} \tag{3.3}$$

$$V_{O_2} = \frac{Total\ volume\ of\ air\ inhaled}{\%O_2\ in\ inhale\ breath - \%O_2\ in\ exhale\ breath} \tag{3.4}$$

Both values are calculated in a short time interval (approximately 10 seconds).

The Data Processing Module will also handle the storage of data that will be useful to watch for trends. For example, VO_2 max test results will be stored so that the user can track their progress over time. Additionally, a user might wish to track their resting RER to see if a low-carb diet is successfully encouraging fat burn.

3.3 Graphical Display Module

The graphical display module is primarily concerned with displaying the relevant fitness data about the user in terms of figures, graphs, and statistics. The main focus of this module is to be able to display fitness metrics over time. As described in the previous section, the Data Processing module saves test measurements and calculations to the device's storage. The display module should be able to aggregate that data taken in the past and draw a relevant scatter or line plot. This way, the user can track their athletic progress over time.

The module wraps the underlying `MPAndroidChart` library, a third-party, open-source tool allowing for easy creation of graphical data. While the framework's support for real-time data updates is limited at this time, for the scope of this project where we will only be processing already saved data it is more than sufficient.

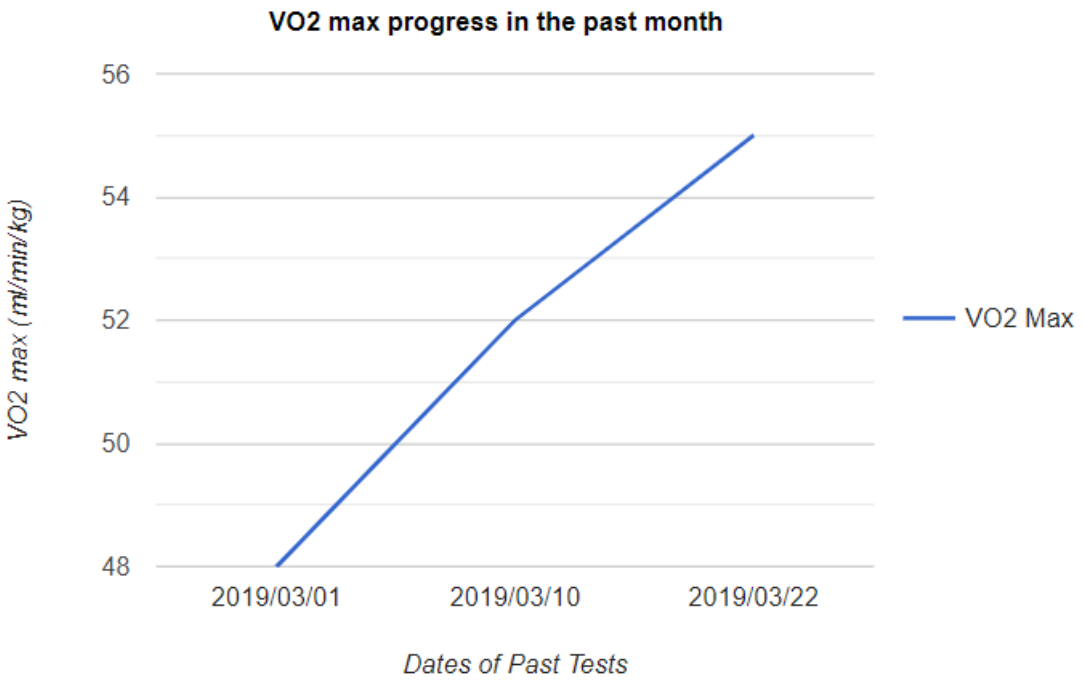

Figure 5: Example illustration of a possible graph for tracking user progress

The scope of this module only covers the implementation specifications for plotting and drawing the collected data. Specifications regarding the general GUI details of the app will be covered in the User Interface Appendix.

4 Breath Measurement System

The Breath Measurement System is made up of the sensors, microcontroller and BLE that work together to collect data from the user's breath, and transmit that data to the companion app. This section will detail the attributes and design choices involved with each component of the system.

4.1 Sensor Specifications

DeepBreath employs various sensors for measuring the user's breaths. There will be 3 sensors that are utilized in conjunction to obtain the data: the oxygen sensor, the carbon dioxide sensor, and the mass airflow sensor.

Table 4.1: Sensors Design Specifications

[**Des 4.1.1-PC**]	All sensors will communicate on a protocol compatible with the microcontroller.
[**Des 4.1.2-PC**]	Oxygen sensor shall be capable of measuring up to at least 25% oxygen content.
[**Des 4.1.3-PC**]	Carbon dioxide sensor shall be capable of measuring up to at least 30% carbon dioxide content.
[**Des 4.2.4-PC**]	Flow sensor will be able to sense inward and outward flow of air.
[**Des 4.2.5-PC**]	The oxygen and carbon dioxide sensors shall have a maximum dimension less than 5 cubic centimeters.
[**Des 4.2.6-PC**]	The sensors' total power dissipation shall be less than 200mW.

Aside from the specifications listed above, the primary goal in selecting sensors for *DeepBreath* was to find options that were inexpensive. Given that one of the main objectives in this project is to produce an affordable product, this was a very important criterion. For the proof of concept and engineering prototype, the sensors we purchased have satisfied our requirements. However, we hope to lower the cost of our sensors even further in future iterations.

After that, we looked at sensor performance, size, and compatibility with our microcontroller. Performance requirements included low power consumption, and a suitable measurement range. The frequency of our data will be relatively low (the rate of human breath) with respect to electronic components, therefore that was not much of a concern.

For the remainder of this section the detailed technical specs of each sensor will be outlined.

4.2 Oxygen Sensor : LOX-02

The LOX-02 oxygen sensor will be used to measure the percentage oxygen content of the air that the user inhales and exhales.

Figure 6: LOX-O2 sensor [11]

Range:	0-25%
Supply Voltage:	5V
Supply Current:	7.5mA (20mA peak)
Power Consumption:	37.5mW Avg
Supply Current:	7.5mA (20mA peak)
Pins:	(1) VS (5V) (2) GND (3) Sensor Transmit (4) Sensor Receive
Dimensions:	20mm diameter 12.5mm height

Table 4.2: LOX-O2 Specifications

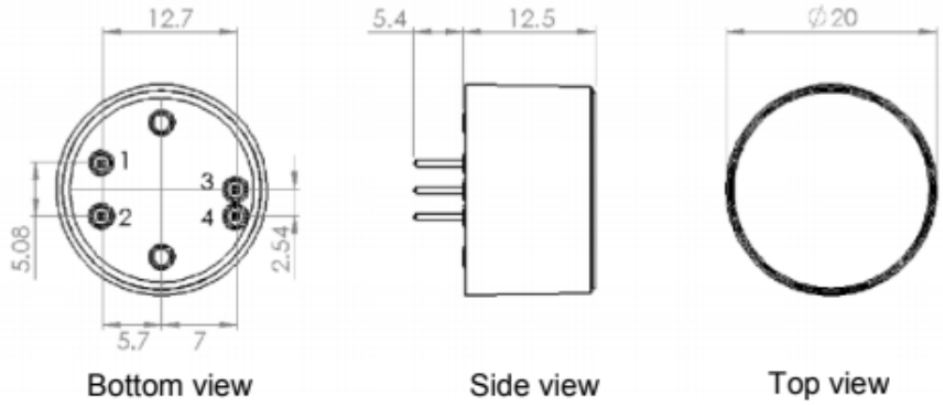

Figure 7: LOX-O2 oxygen sensor dimensions [12]

4.3 Carbon Dioxide Sensor : SprintIR

The SprintIR sensor will be used to measure the percentage carbon dioxide content of the air that the user inhales and exhales.

Figure 8: SprintIR sensor [13]

Range:	0-100%
Supply Voltage:	3.3V
Supply Current:	<15mA (100mA peak)
Power Consumption:	37.5mW Avg
Supply Current:	7.5mA (20mA peak)
Pins:	(1) GND (3) VS (3.3V) (5) Sensor Receive (7) Sensor Trensmit
Dimensions:	25mm x 40mm x 30mm

Table 4.3: SprintIR Specifications

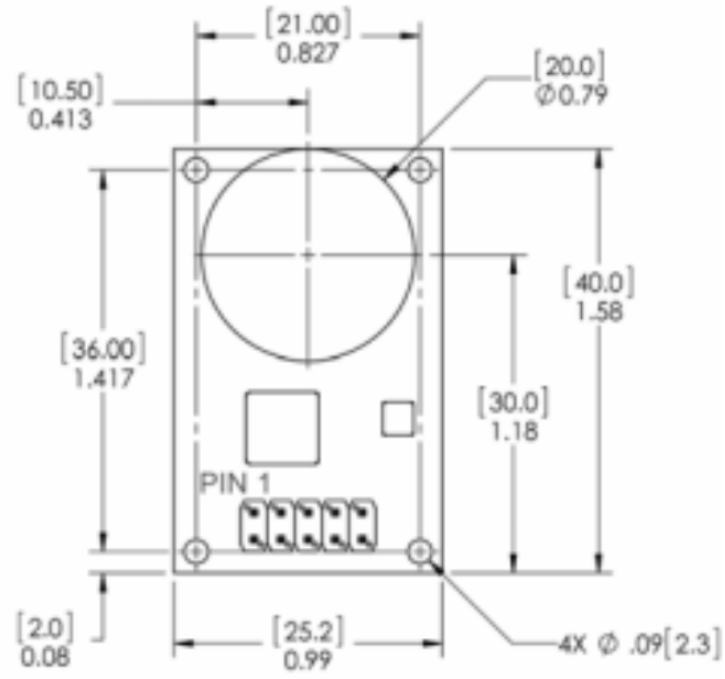

Figure 9: SprintIR carbon dioxide sensor dimensions [14]

4.4 Mass Airflow Sensor : Honeywell AWM730B5

The mass airflow sensor will be used to measure the amount of air that the user inhales and exhales.

Figure 10: AWM730B5 sensor [15]

Range:	± 300L/min
Supply Voltage:	15V
Supply Current:	4mA
Power Consumption:	60mW Avg
Supply Current:	7.5mA (20mA peak)
Pins:	(1) 15 Vdc (3) GND (5) 5 Vdc (7) Vout
Dimensions:	46.23 x 32.5 x 46.5mm

Table 4.4: AWM730B5 Specifications

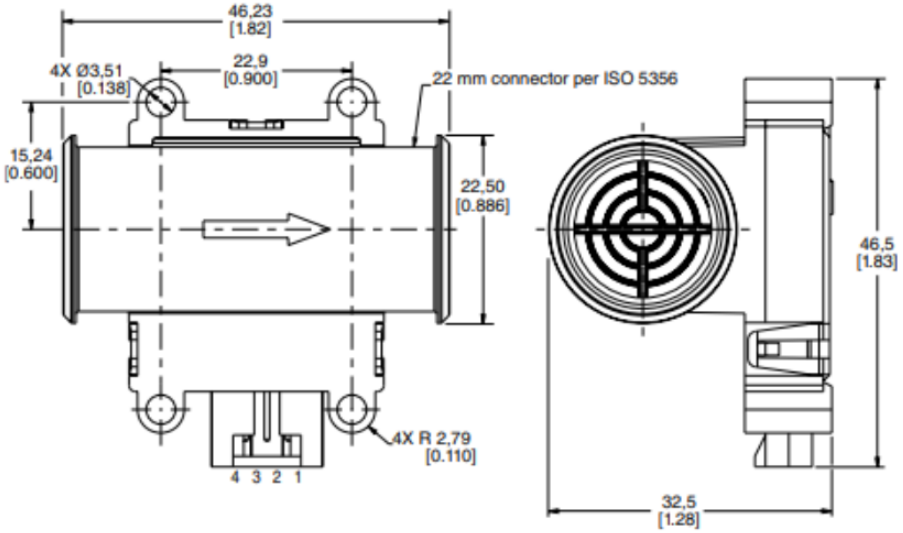

Figure 11: AWM730B5 flow sensor dimensions [16]

5 Hardware design

5.1 ESP32 Microcontroller

The microcontroller we will be using is the ESP32. It is a small, yet extremely effective programmable microcontroller that supports WiFi and BLE. It includes a 32 bit Xtensa LX6 CPU and 416KB of SRAM,128KB of ROM, and 64MB of flash memory. The ESP32 includes a multitude of GPIO pins, which can be assigned to all sorts of peripheral duties, including(17) :

- **Analog-to-Digital Converter (ADC)** Up to 16 channels of 12-bit SAR ADCs. The ADC range can be set, in firmware, to either 0-1V, 0-1.4V, 0-2V, or 0-4V.
- **UART** 2 UART interfaces.
- **GPIO** 34 GPIO Pins.
- **I²C, SPI, I²S** There are two I²C, four SPI and two I²S interfaces.

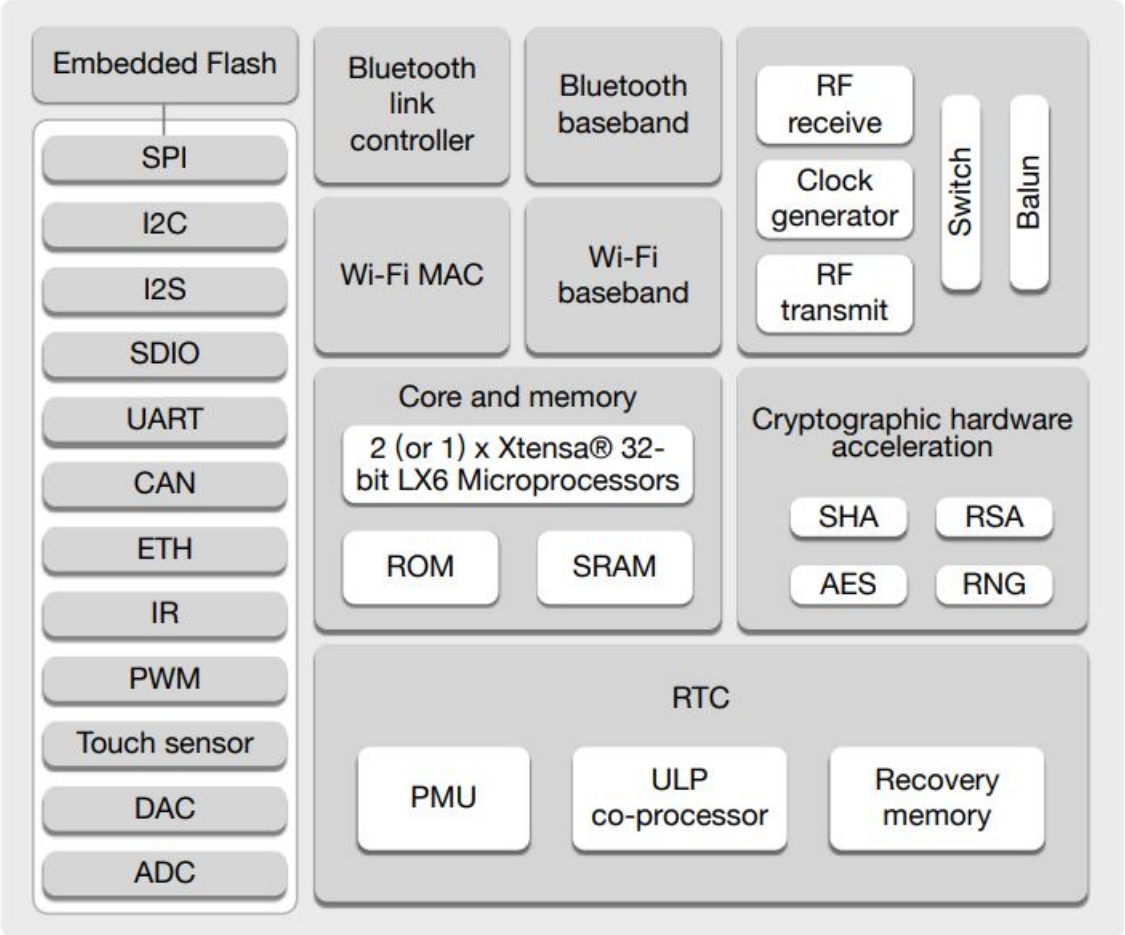

Figure 12: Block Diagram of ESP32

Table 5.1: Hardware Design Specifications

[**Des 5.2.1-PC**]	The microcontroller shall consume 240mA under maximum load.
[**Des 5.3.1-PC**]	A total of five GPIO pins on the microcontroller will be used.
[**Des 5.4.1-PC**]	The gas sensors will send data using RS232 protocol.
[**Des 5.5.1-PC**]	The gas sensors will communicate with the microcontroller through the UART.
[**Des 5.10.1-PC**]	The circuit shall be powered by two power supplies.
[**Des 5.11.1-PC**]	The circuit will use a 15V 1800mAh battery and a 5v 5200mAh battery.
[**Des 5.11.2-PC**]	The microcontroller and all other electrical components will last a combined 34 hours under maximum load.

The ESP32 has far more functionality than we need, however as a PC the microcontroller fits our needs perfectly. With two UARTs, an ADC and BLE, we will be able to easily read the output of our sensors and send it to the app. The additional GPIOs are extremely good to have during the PC phase as it allows for quick prototyping if we require more functionality.

5.2 ESP32 Power Consumption

The power consumption of the ESP32 varies based on the situation. Without our sensor usage, the rough estimates based different scenarios can be seen below. Exact power consumption including our sensors will be determined for the EP.

Scenario	**CPU 80Mhz**	**CPU 160 Mhz**	**CPU 240 Mhz**
CPU + BLE + Electronic systems	110 mA	125 mA	140 mA
CPU + BLE	40mA	50 mA	240 mA
CPU (in sleep mode)	3.5 mA	3.5 mA	3.5 mA

Table 5.2: ESP32 current draw under various conditions

5.3 ESP32 Pin Setup

The ESP32 has various GPIO's that all have specific uses. Using the pin layout and GPIO layout in figures 13 and 14 we mapped the various pins to our sensors. The pins which we will be used for both the sensors and ESP32 are outlined in the table below (18).

Sensor	Sensor Function	Name	Type	GPIO	ADC	UART	PIN
SprintIR	TX RX	TXD0 RXD0	I/O I/O	GPIO01 GPIO03	N/A N/A	U0TXD U0RXD	PIN4 PIN5
LOX-02	TX RX	IO17 IO16	I/O I/O	GPIO17 GPIO16	N/A N/A	U2TXD U2RXD	PIN11 PIN12
AWM730B5	Vout	IO04	I/O	GPIO04	ADC2_CH0	N/A	PIN13

Table 5.3: Pin connections between the microcontroller and sensors

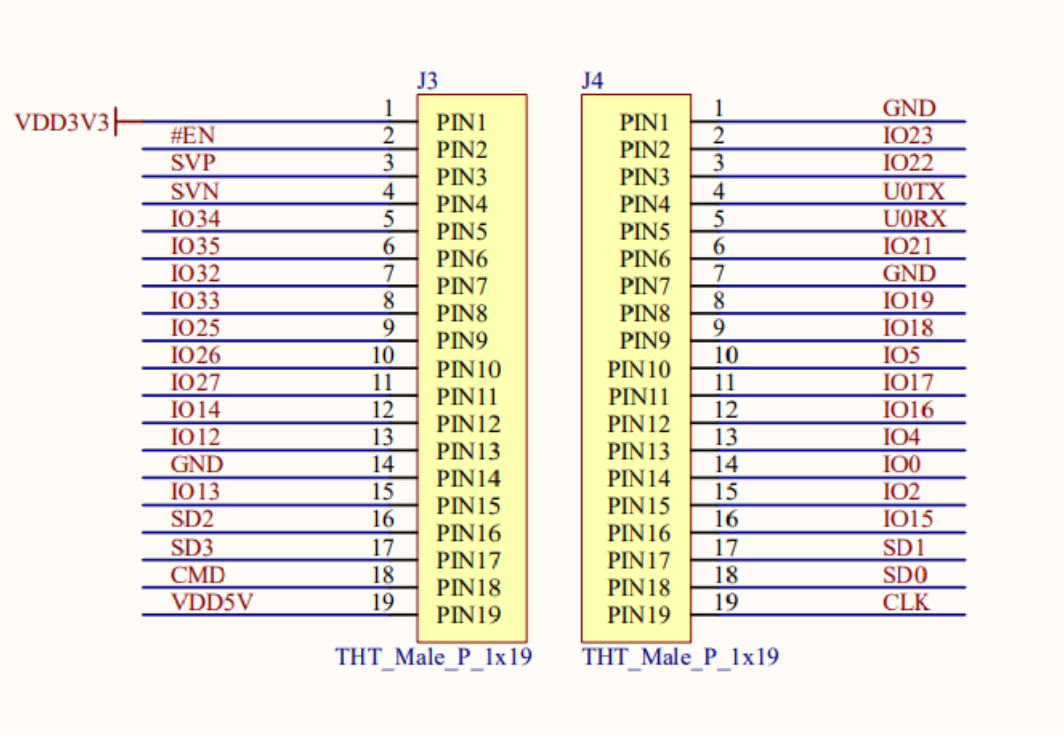

Figure 13: Pin layout of ESP32

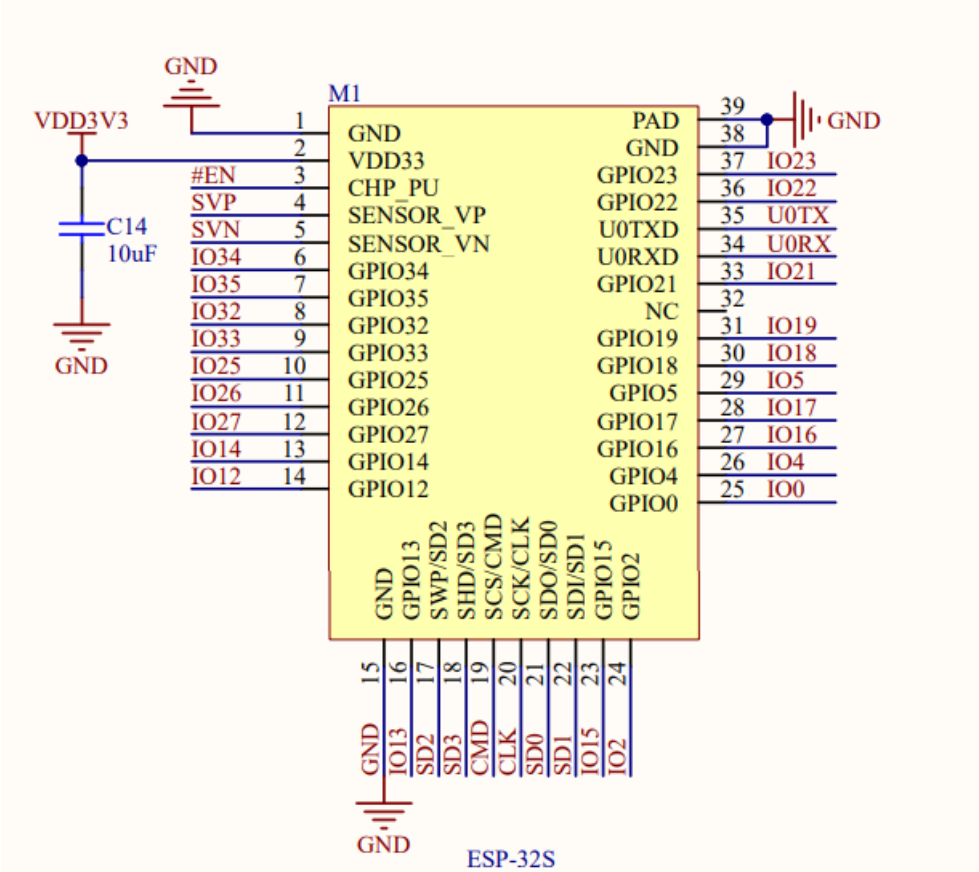

Figure 14: GPIO layout of ESP32 (Maps to pin layout)

5.4 Communication Method : RS232

Both the SprintIR and LOX-02 use RS232 to send and receive data. RS232 is a protocol that uses serial communication in which data is sent one bit at a time along with one data line. This is done by specifying voltage levels that correspond to logical ones and zeros. Compare this to parallel communication, where multiple bits of data are sent using multiple data lines (19)

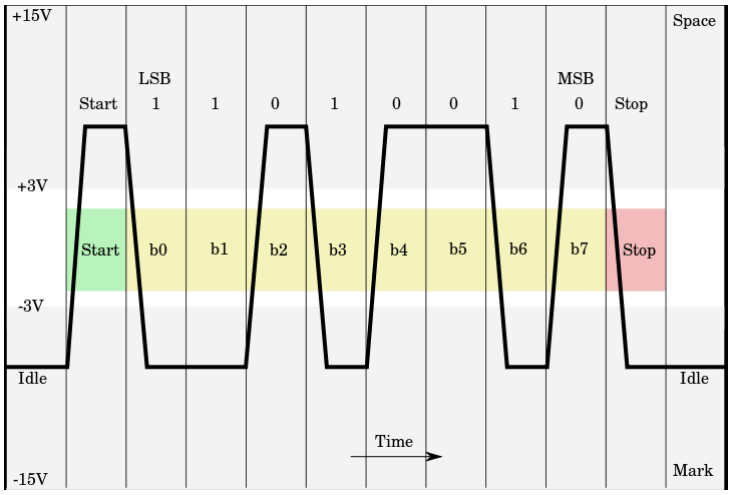

Figure 15: Oscilloscope of voltage levels for an ASCII 'K' using RS232

5.5 Communication Method : UART

A UART is essentially a serial communication interface. Due to its 'universal' property, it can communicate with a multitude of different serial protocols. This means the two onboard UARTs can receive/tranceive data between the two gas sensors (20).

5.6 General Workflow Overview

Figure 16 creates the guidelines for a general workflow of our sensors communicating with our microcontroller. Through the UART, both the SprintIR and LOX-02 will send data to PIN4 and PIN11 while receiving data from PIN5 and PIN12. With the on-board ADC the AWM730B5 will send an output voltage to PIN13 which will then be converted to a digital signal.

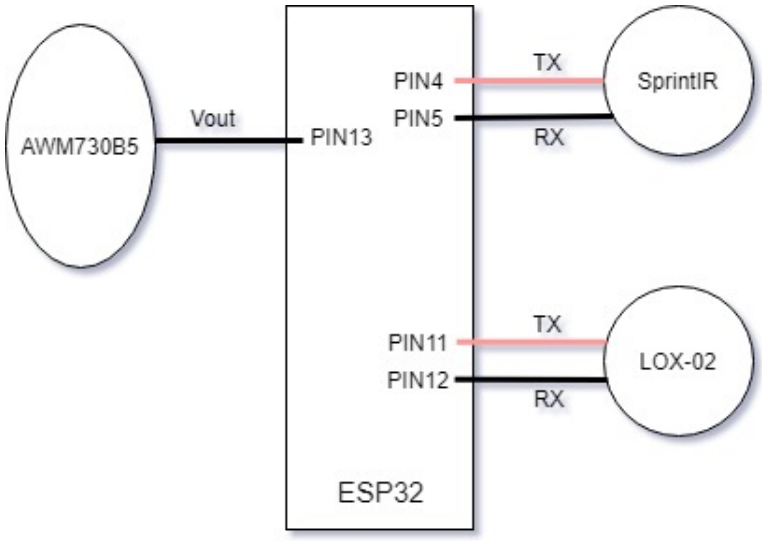

Figure 16: Simple block diagram for communication between the sensors and microcontroller

5.7 Microcontroller Prototype

For the final prototype it is undecided whether or not we will continue using an ESP32. There are many considerations to make, many of which will be decided when we have a working PC. One extremely beneficial points to continuing with the ESP32 is that an additional UART can be "hacked" in to allow for more sensors if required.

5.8 Design Alternatives

Other microcontrollers that are under consideration are the PSoC 5 and PSoC 4. These devices come at a higher price point for a few reasons - they operate in a larger voltage range (1.71V to 5.5V), have many GPIO pins (however many would be unused), and can operate in lower power ranges. A big reason these devices are more expensive is because they come with proprietary software that allows you

to drag and drop modules to create your design. This comes in handy if you're a beginner programmer or it's your first time using a microcontroller.

5.9 Future Considerations

Inevitably, production of microcontroller will become extremely difficult due to reasons such as size, cost and excess of unneeded functionality. For the future we hope to create our own proprietary microcontroller with functionality that is specific to our needs. This will allow us to optimize power usage, mask size, and many other things which isn't possible when using an out-of-the-box microcontroller.

5.10 Circuit Design

For our PC our goal was to make the circuit as simple as possible. Our circuit consists of a breadboard, two power supplies, a voltage divider and our three sensors.

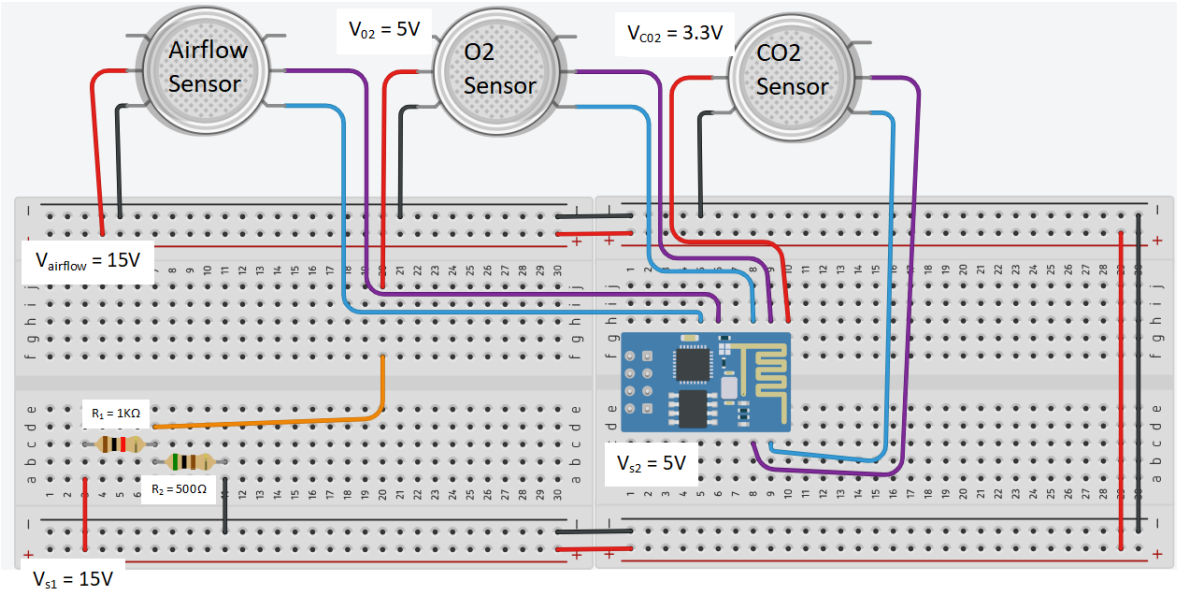

Figure 17: Circuit design with supply voltages and resistors

In Figure 17 we provide a source voltage of $V_{S1} = 15V$. With the use of a voltage divider, we split the voltages into two branches, $V_{airflow}$ and V_{02}, providing them 15V and 5V respectively. We can determine V_{02} with the following:

$$\frac{V_{S1} * R_2}{R_1 + R_2} \qquad \text{(Eq. 1.1)}$$

Substituting $V_{S1} = 15V$, $R_1 = 1K\Omega$, $R_2 = 500\Omega$ in Eq. 1.1 :

$$V_{02} = \frac{15 * 500}{1000 + 500} = 5V$$

The ESP32 is able to provide 3.3V from one of its pin, which directly connects to the CO2 sensor. Lastly we have $V_{S2} = 5V$ which is used to power the microcontroller.

5.11 Power supplies

In order for our device meet our design specifications we determined the two power supplies by the following:

Power Supply V_{S1}

We combined 10 1.5V Energizer E91 in series to give a supply voltage of 15V. Using a worse-case scenario of 500mA discharge, these batteries can provide roughly 15V at 1800mA. The AWM730B5 airflow sensors uses 60mW at 15V. We can determine the mA with the following, under the assumption that the sensor is on at all times: (21)

$$Q_{(mAh)} = \frac{1000 * E_{(Wh)}}{V} \quad \text{(Eq. 2.1)}$$

Which translates to:

$$mAh = \frac{mWh}{V} \quad \text{(Eq. 2.2)}$$

Substituting the given values for the AWM730B5 into Eq 2.2 we get:

$$\frac{60mWh}{15V} = 4mAh$$

Following the same procedure for the L0X-02 oxygen sensor gives us a hourly current draw of 7.5mAh Combining the two we get a total worse case power consumption of 11.5mAh. This means that with our 1800mAh battery, we are able to power the two sensors for over 156 hours or 6 days.

Power Supply V_{S2}

To power the ESP32 we will use a portable battery back which can provide 5V at 5200mAH. This is a cheap and simple option, as it can give USB power to the ESP32 and is portable. To find the current draw of the SprintIR C02 sensor we can simply use eq. 2.2 again to get:

$$\frac{37.5mWh}{3.3V} = 11.4mAh$$

Looking at the worst case current draw scenario for the ESP32 we see that it uses 140mA. Combined, these draw a total of 151.36mAh, which means we are able to power the microcontroller and sensor for well over 34 hours under maximum load and constant use. The tabulated results for both power supplies can be seen below in table 5.4

Power supply	Combined Power draw	Estimated battery life
V_{S1}	11.5mAh	156 hours
V_{S2}	151.4mAh	34 hours

Table 5.4: Power draw and estimated battery life

During EP phase we aim to remove the AA power supply, and only use one battery pack to power our device. This will allow for a cleaner, and less bulky design. To achieve this, a singular power supply will have a USB to DC voltage converter. This supply voltage will be routed to the various sensors, and be stepped up or stepped down when needed.

5.12 Bluetooth

The 2-way communication between the ESP32 (Microcontroller) inside the mask and the phone application will happen over Bluetooth. The ESP32 has 2 Bluetooth settings - Bluetooth Low Energy and Bluetooth Basic Rate. Bluetooth Basic Rate, also known as Enhanced Data Rate, is a communication protocol optimized for continuous data streaming. It operates in a frequency band of 2.4GHz and can transmit up to 3Mbps (4). Bluetooth Low Energy is the protocol more optimized for short bursts of data transmission and operates within the same frequency band. The benefit of using this protocol over the Basic Rate is that this consumes 100x less power, however it can only transmit up to 2 Mb/s (4). The Low Energy Protocol most likely requires more development time and integration due to the increased complexity. Due to the nature of our application, which involves transmitting frequent data, we will first create the product using the Bluetooth Low Energy for efficiency and low power usage.

To begin the Bluetooth connection there are initialization steps that are required. From now on we will refer to the mask as the BLE Server and the phone application as the BLE Client. The following figure shows the steps the ESP32 will take when the mask is powered on. To create the BLE server from the mask, the ESP32 will make use of several libraries that are necessary for BLE communication - BLEDevice, BLEUtils, and BLEServer. A universally unique identifier (UUID) will also need to be created to label the Service and Descriptor (5).

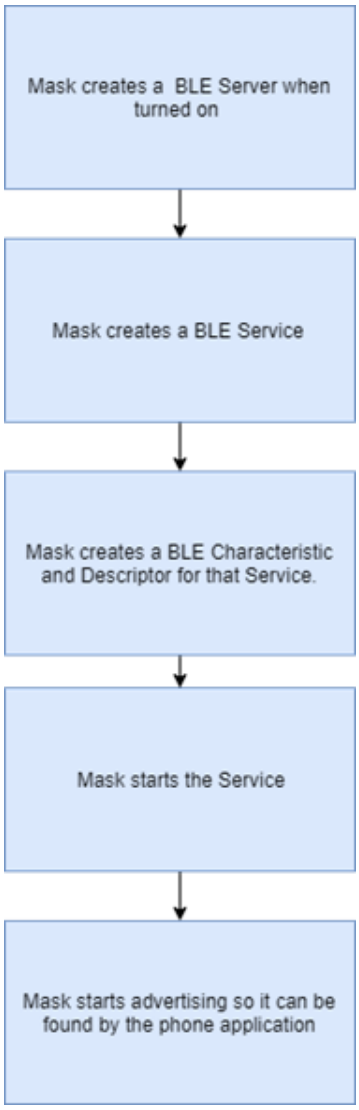

Figure 18: Steps for Mask to pair with the Phone Application

6 Conclusion

This document serves as a reference for the design of *DeepBreath*'s subsystems and the scientific theory behind our product. These subsystems include the specifications and usage of the sensors, the industry standard and implementation for BLE communication, the implementation and choice of microcontroller, the theory behind the metrics that can be analyzed from a user's breath, and the design of the physical mask itself. With these design specifications we are confident that we fully can realize our product.

The following is a summary of the mentioned design specifications:

Mobile Application Design Specifications

- The app shall provide easy-to-understand results and metrics that are accurate in a meaningful way

- The app shall be able to run multiple threads that communicate with the mask, process incoming data, and display the processed data in a timely manner
- The app and mask shall be able to connect to each other within a reasonable amount of time and reliably communication with each other until the end of the session

Breath Measurement System Specifications

- The sensors shall accurately and reliably measure metrics about a user's breath and relay them to the microcontroller in a reasonable amount of time
- The sensors shall not dissipate a lot of heat into the mask, making it uncomfortable for the user
- The microcontroller shall be able to house and power our sensors reliably
- The microcontroller shall be able to host a BLE server during the userâĂŹs session and be discoverable by other devices

Mask Design Specifications

- The mask shall be comfortable and a reasonable weight
- The mask shall look aesthetically pleasing

During the prototyping phase, if Exotic discovers that an aspect of our design is no longer suitable or does not work, we will revise this Design Specifications document appropriately. We will measure our success based on these design requirements.

7 References

[1] Exotic, "Requirements Specification for DeepBreath", Exotic, Burnaby, 2019.

[2] "Meet Google Play's target API level requirement | Android Developers", Android Developers, 2019. [Online]. Available: https://developer.android.com/distribute/best-practices/develop/target-sdk. [Accessed: 12- Mar- 2019].

[3] "GATT Overview | Bluetooth Technology Website", Bluetooth.com, 2019. [Online]. Available: https://www.bluetooth.com/specifications/gatt/generic-attributes-overview. [Accessed: 13- Mar- 2019].

[4] "ESP32 Bluetooth Low Energy (BLE) on Arduino IDE", Random Nerd Tutorials, 2019. [Online]. Available: https://randomnerdtutorials.com/esp32-bluetooth-low-energy-ble-arduino-ide/. [Accessed: 11- Mar- 2019].

[5] "Bluetooth overview | Android Developers", Android Developers, 2019. [Online]. Available: https://developer.android.com/guide/topics/connectivity/bluetooth. [Accessed: 12- Mar- 2019].

[6] "Bluetooth Low Energy Packet Types - Developer Help", Microchipdeveloper.com, 2019. [Online]. Available: http://microchipdeveloper.com/wireless:ble-link-layer-packet-types. [Accessed: 12- Mar- 2019].

[7] "Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology", mdpi.com, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/12/9/11734/htm. [Accessed: 12- Mar- 2019].

[8] "Math | Android Developers", Android Developers, 2019. [Online]. Available: https://developer.android.com/reference/java/lang/Math. [Accessed: 11- Mar- 2019].

[9] "VO2 max", En.wikipedia.org, 2019. [Online]. Available: https://en.wikipedia.org/wiki/VO2_max. [Accessed: 09- Mar- 2019].

[10] "Respiratory exchange ratio", En.wikipedia.org, 2019. [Online]. Available: https://en.wikipedia.org/wiki/Respiratory_exchange_ratio. [Accessed: 09- Mar- 2019].

[11] "Optical Oxygen Sensor for 0-25% O2 Measuring LOX-02 purchasing, souring agent | ECVV.com purchasing service platform", Ecvv.com, 2019. [Online]. Available: https://www.ecvv.com/product/4745592.html. [Accessed: 10- Mar- 2019].

[12] "LuminOx Optical Oxygen Sensor - LOX-02", industrial product, 2019. [Online]. Available: https://industrialproduct.typepad.com/blog/2016/11/luminox-optical-oxygen-sensor-lox-02.html. [Accessed: 10- Mar- 2019].

[13] "SprintIR-W 100% CO2 Sensor", CO2 Meter, 2019. [Online]. Available: https://www.co2meter.com/products/sprintir-100-percent-co2-sensor. [Accessed: 10- Mar- 2019].

[14] "SprintIR-W 100% CO2 Sensor", CO2 Meter, 2019. [Online]. Available: https://www.co2meter.com/products/sprintir-100-percent-co2-sensor. [Accessed: 10- Mar- 2019].

[15] H. Solutions, "AWM730B5 Honeywell Sensing and Productivity Solutions | Sensors, Transducers | DigiKey", Digikey.com, 2019. [Online]. Available: https://www.digikey.com/product-detail/en/honeywell-sensing-and-productivity-solutions/AWM730B5/480-6166-ND/3538583. [Accessed: 10- Mar- 2019].

[16] H. Solutions, "AWM730B5 Honeywell Sensing and Productivity Solutions | Sensors, Transducers | DigiKey", Digikey.com, 2019. [Online]. Available: https://www.digikey.com/product-detail/en/honeywell-sensing-and-productivity-solutions/AWM730B5/480-6166-ND/3538583. [Accessed: 15- Mar- 2019].

[17] Espressif.com, 2019. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf. [Accessed: 13- Mar- 2019].

[18] "E-Paper ESP32 Driver Board - Waveshare Wiki", Waveshare.com, 2019. [Online]. Available: https://www.waveshare.com/wiki/E-Paper_ESP32_Driver_Board. [Accessed: 13- Mar- 2019].

[19] "RS-232", En.wikipedia.org, 2019. [Online]. Available: https://en.wikipedia.org/wiki/RS-232. [Accessed: 13- Mar- 2019].

[20] "Universal asynchronous receiver-transmitter", En.wikipedia.org, 2019. [Online]. Available: https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter. [Accessed: 13- Mar- 2019].

[21] Data.energizer.com, 2019. [Online]. Available: http://data.energizer.com/pdfs/e91.pdf. [Accessed: 13- Mar- 2019].

[22] I. 9241-161:2016, "ISO 9241-161:2016", ISO, 2019. [Online]. Available: https://www.iso.org/standard/60476.html. [Accessed: 13- Mar- 2019].

[23] "IEEE 1621-2004 - IEEE Standard for User Interface Elements in Power Control of Electronic Devices Employed in Office/Consumer Environments", Standards.ieee.org, 2019. [Online]. Available: https://standards.ieee.org/standard/1621-2004.html. [Accessed: 13- Mar- 2019].

[24] I. 11073-10418:2014, "ISO/IEEE 11073-10418:2014", ISO, 2019. [Online]. Available: https://www.iso.org/standard/61897.html. [Accessed: 13- Mar- 2019].

[25] "2019 RoHS Compliance Guide: Regulations, 10 Substances, Exemptions, WEEE", Rohsguide.com, 2019. [Online]. Available: https://www.rohsguide.com/. [Accessed: 13- Mar- 2019].

[26] "CSA C22.2 No. 0.23-15 | Standards Council of Canada - Conseil canadien des normes", Scc.ca, 2019. [Online]. Available: https://www.scc.ca/en/standardsdb/standards/28121. [Accessed: 13- Mar- 2019].

A User Interface and Appearance Design

A.1 Introduction

In the design of *DeepBreath*, Exotic's engineers strive to create a product that offers a simple and elegant user interface that is intuitive to everyone. Our top priorities are to ensure safety and comfort without sacrificing ease of use, especially for the physical mask component. It is imperative that the product is easy to operate since the safety of the users depends on how easily the mask is able to be put on or be taken off.

Purpose

The purpose of this appendix is to explain the design of *Deepbreath*'s user interface. It will provide justifications for our design decisions, and how they take into consideration the critical elements of UI interaction as seen both in lectures and in *The Design of Everyday Things*.

Scope

This document will outline 5 major topics in our UI design process:

1. **User Analysis**
   User analysis section will discuss the required user knowledge and restrictions when using *DeepBreath*.

2. **Technical Analysis**
   Technical analysis describes how our design process takes into account the "Seven Elements of UI Interaction" (discoverability, feedback, conceptual models, affordances, signifiers, mappings, constraints).

3. **Engineering Standards** The various engineering standards that are relevant to the user interfaces for our system will be outlined in this section.

4. **Analytical Usability Testing** Analytical usability testing details the series of testing procedures that will be undergone from an analytical perspective, to discover any possible inconveniences that may be present in the system.

5. **Empirical Usability Testing** Empirical usability testing details the methods of testing that will be required for future iterations of *DeepBreath*, with actual users of our product.

A.2 Graphical Presentation

The 2 main components of *DeepBreath* that the user primarily interacts with are the physical mask and the mobile application. For diagrams describing the structure and look of the mask, please refer to the mask design section (Section 6) of the Design Specifications document.

In this section some we will describe some graphical representations of what our mobile app may look like. We hope to create a friendly, aesthetically pleasing look that also comes with an interface that is intuitive to all users.

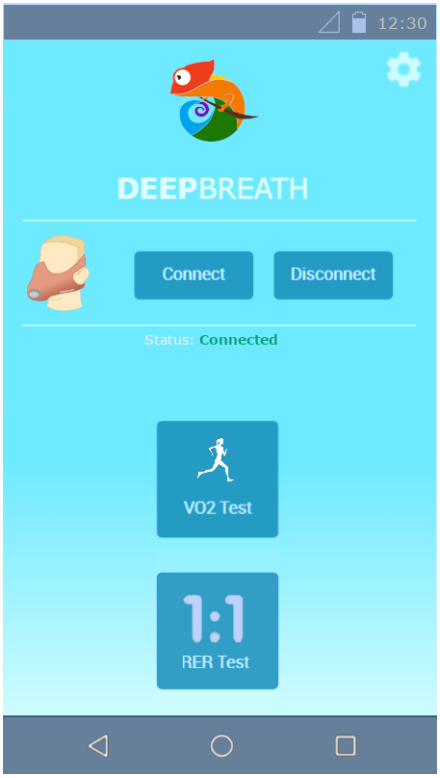

Figure 19: Illustration of the main page of the mobile application

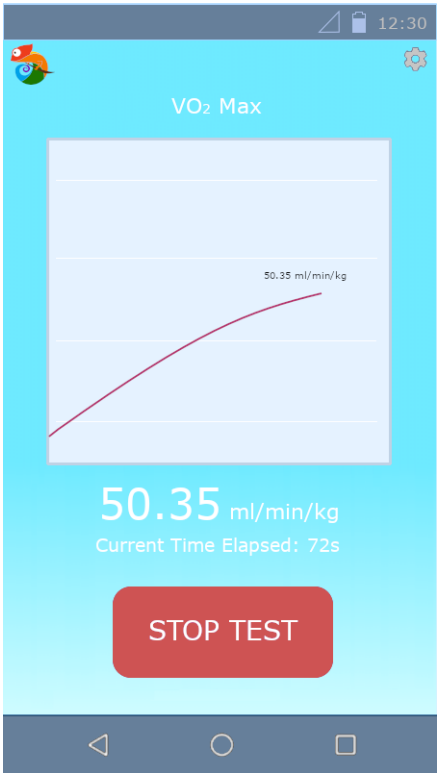

Figure 20: Illustration of the app displaying an ongoing fitness test

A.3 User Analysis

The targeted users of this product are individuals who are interested in fitness and cardio who want to know more about their cardiovascular health outside of your traditional mile-time or speed. This type of equipment is usually very expensive and we are aiming to provide a cost-effective solution for those enthusiastic about fitness but don't want to spend thousands of dollars.

The mask itself shall be very easy to put on and have it stable on the user's face, which will be accomplished by straps/velcro for adjustments. There will be some responsibility from the user to ensure the mask is secure before starting a fitness test.

Users must also download the companion app for the mask. The purpose of our app is relatively basic, and we are confident it will be easy to understand and use. We expect the user to only take a few minutes to connect their device and get used to the application UI layout before being able to start their first test. The app will have buttons to navigate layouts, easily connect to the mask, and to begin their fitness tests and see the results in a graphical display.

A.4 Technical Analysis

The main criteria from Don Norman's book, "The Design of Everyday Things", are Discoverability, Feedback, Conceptual Models, Affordance, Signifiers, Mapping, and Constraints.

Discoverability refers to the visual display being easy-to-understand and intuitive. Essentially we want the user to quickly learn the ropes of the app. To accomplish this, our application is not going to have lots of buttons and options to avoid any confusion. Perhaps some users are not super familiar with Bluetooth connection and pairing devices, our app will aim to make this process seamless and easy. For instance, we should always create user dialog notifications when trying to pair with the microcontroller over Bluetooth. We will also avoid fancy vernacular without defining it at the very least.

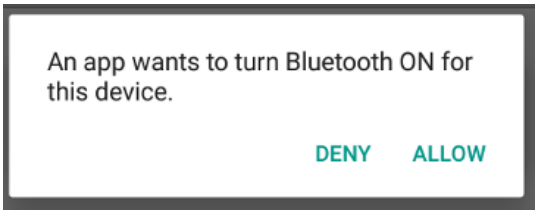

Figure 21: Example dialog of asking user permission for Bluetooth

Feedback refers to signalling to the user to notify them of changes or the status of the system. To ensure this, we are going to try and have some LED on the mask that will notify of its state. For example, it should obvious whether or not the two devices are paired - if they are, the LED will be green, otherwise it will be red. On the app, there will be an indicator or animation so the user knows exactly when

the fitness test has started and whether or not it's currently in progress. The view during the fitness test itself will be different.

Conceptual Models are when users are able to extrapolate their experiences to another task. The most common conceptual model would be a user tracking their calorie intake of their steps in a pedometer in an app on their phone. There is potentially a market where users who are trying to lose weight who be interested in using our product.

Affordance relates to how intuitive an object in the interface is to manipulate/use. Our companion app will have a low amount of buttons that are well-labeled. Any graphical displays will be displaying only important information. The main buttons refer to connecting to the mask, picking/configuring a fitness test, and starting/terminating a fitness test.

Signifiers are similar to Feedback. In the case with the LED on the mask, the LED itself would be a signifier. The user will be able to look at the LED and see the color to determine information about the state of the Bluetooth connection. The color of the LED would be the feedback, the actual information relayed back to the user.

Mapping describes for the user to easily understand the purpose or usage or a button or functionality. Since our application is a "companion" application, the general usage of it will be simple. There will not be that many buttons or areas for user input, and the buttons that we will have will be well-labeled and perform simple tasks (Eg. connect the app to a device, begin test).

Constraints are limitations to the user's access. Our mask will require power so therefore the user must remember to charge the mask or replace the batteries. To use the mobile application, the user of course must have a mobile device. Any information that the user wants to view or setting that they want to change, it must be done through the app. Connecting the mask must also be done with the companion app.

A.5 Engineering Standards

DeepBreath shall adhere to various well-established engineering standards in the industry that pertain to user interface design. They are as follows:

1. **ISO 9241-161:2016**
   Ergonomics of Human-System Interaction
   Describes recommendations and requirements on when and how to use visual user-interface elements in software products. [22]

2. **IEEE 1621-2004**
   User Interface Elements in Power Control of Electronic Devices
   Describes recommendations on user interface symbols and indicators for indicating power control in consumer electronic devices. [23]

3. **ISO/IEEE 11073-10418:2014**
   Health Informatics - Personal Health Device Communication
   Addresses communication standards between health and wellness monitoring devices defined as "agents", and "managers" (e.g. cell phones, computers). Defines various transport standards, data modeling, and interoperability requirements. [24]

In addition, safety for the users is an important design consideration for *DeepBreath*'s engineers as well. There are many potential hazards that could occur when a user is operating the product. To ensure safe usage of *DeepBreath*, our product will adhere to the following safety standards:

1. **RoHS Compliancy**
   Restriction of Hazardous Substances
   The electrical components shall be compliant with the RoHS directive, to ensure that hazardous or toxic materials are not used in the manufacturing of our device. [25]

2. **C22.2 NO. 0.23-15**
   General Requirements for Battery-Powered Appliances
   Describes general standards specifications for battery powered electronic appliances. [26]

A.6 Analytical Usability Testing

In this section are outlined the analytical testing procedures that Exotic engineers will perform. This can help us evaluate overlooked problems present in our interface design. The two main UI-related components of our product are the physical mask and the mobile application. The test steps are listed below:

Mask Buttons

1. The power button for the mask gives a physical, mechanical "feel" of being pressed down.

2. Once the power button is pressed, the LED light indicating status should be turned on or off

3. Once the power button is pressed, the mask should be ready, and can be turned on for testing from the app.

Mask Straps

1. The mask can be worn over the head.

2. The face seal forms an airtight seal, and the wearer should not be able to feel any discernible amounts of air escaping.

3. The straps should not irritate the user in any way.

Mobile Application

1. The mobile application should visibly show the logo of the company as well as the *DeepBreath* title.

2. The main menu of the app has a intuitive "Settings" icon in the form of a gear, where the user can change settings regarding the application.

3. Purposefully not having our mask turned on, and attempting to pair with it from the app displays an alert saying that a pairable device has not been found.

4. The main menu has visible buttons with large letters indicating the types of fitness tests that the user can perform.

A.7 Empirical Usability Testing

In this section are outlined the empirical testing procedures that Exotic engineers will perform. These tests will be performed by end users who have no prior knowledge of *DeepBreath*, and we will be indiscriminate of their athletic background. After the test steps are performed by the users, feedback will be taken about what their thoughts were regarding the usability and friendless of our interface design. The following are the list of interactions that the end users will be asked to carry out:

End User Activity 1
Put on the mask
Feedback Questions

1. Did you experience any discomfort when attempting to wear the mask?

2. Are the straps easy to manipulate?

3. Are you feeling any irritation or significant discomfort?

4. Can you breathe comfortably through the face seal?

End User Activity 2
Perform light exercise, but not too intense that you have breathing difficulties
Feedback Questions

1. Is the mask bouncing up and down and being a nuisance?

2. Can you still breathe comfortably?

3. Does the mask feel heavy during exercise?

End User Activity 3
Attempt to pair the mobile application with the phone
Feedback Questions

1. Is the graphical user interface of the app pages intuitive and friendly?

2. Did you have any difficulties locating the pairing buttons?

3. Did you receive status alerts for all Bluetooth activities that you have manually started?

End User Activity 4
Start a fitness test
Feedback Questions

1. Was there a visual indicator of the test starting?
2. Can you see the measurement data changing in real-time?
3. Is the stop button of a large enough size?

A.8 Conclusion

User Interface Design requirements are key to having your product accepted by the users. During the implementation phase it's very easy for developers to overlook the fact that many users may not have the same technical skills as others, so it's important to design an interface that caters to those people. With this goal in mind, along with Don Norman's text, we were able to identify key qualities and components necessary for an intuitive and powerful interface design. Exotic plans to demo *DeepBreath* to fellow students and fitness enthusiasts at Simon Fraser for feedback on ease-of-use since those people are our target audience. We plan to abstract away all information about what sensors are in the mask, how they work, etc. For our prototype demo, we plan to meet our goal of having an intuitive and easy-to-use display while delivering useful and simple results.

B Supporting Test Procedures

Tester Name:		Date:	
Mobile Application System			
Relevant Specification(s)	Procedure	Result	Comments
Bluetooth Module			
Des. 3.1.1-PC Des. 3.1.2-PC	The app is able to read and process wirelessly BLE data packets transmitted from the mask. All data collected are correct and without errors.	Pass ☐ Fail ☐	
Des. 3.1.3-PC	The app displays a dialog notification asking for user's permission to turn on Bluetooth features.	Pass ☐ Fail ☐	
Des. 3.1.3-PC	The phone displays a dialog indication when the application pairs with the microcontroller.	Pass ☐ Fail ☐	
Data Processing, Calculations, and Storage			
Des 3.2.1-EP	The app should be able to calculate and display accurate measurements of the user's VO_2 metrics after a fitness test.	Pass ☐ Fail ☐	
Des 3.2.2-EP	The app should be able to calculate and display accurate measurements of the user's RER metrics in real-time.	Pass ☐ Fail ☐	
Des 3.2.3-EP	The measurement results after a fitness test is stored in the internal storage of the phone, and can be retrieved at a later date in the app.	Pass ☐ Fail ☐	

Tester Name:		Date:	
Breath Measurement System			
Relevant Specification(s)	Procedure	Result	Comments
Sensor Specifications			
Des 4.1.2-PC	During the fitness test, the oxygen sensor is capable of correctly measuring up to at least 25% oxygen content.	Pass ☐ Fail ☐	
Des 4.1.3-PC	During the fitness test, the carbon dioxide sensor is capable of correctly measuring up to at least 30% CO_2 content, without any errors.	Pass ☐ Fail ☐	
Des. 4.1.5-PC	The oxygen and carbon dioxide sensors will have a maximum dimension less than 5 cubic centimeters.	Pass ☐ Fail ☐	
Des. 4.1.6-PC	When the system is operating, measuring the total power output of the sensors should yield a measurement less than 200mW.	Pass ☐ Fail ☐	

Tester Name:		Date:	
Hardware Design			
Relevant Specification(s)	Procedure	Result	Comments
Microcontroller Specifications			
Des 5.2.1-PC	Under maximum load, the microcontroller is at most drawing 240mA.	Pass ☐ Fail ☐	
Des 5.3.1-PC	When all components are connected to the microcontroller, only the 5 GPIO pins outlined will be used.	Pass ☐ Fail ☐	
Des. 5.4.1-PC	The microcontroller is able to receive the incoming data from the gas sensors.	Pass ☐ Fail ☐	
Des. 5.5.1-PC	The microcontroller is able to asynchronously send and receive data from the gas sensors.	Pass ☐ Fail ☐	
Circuit Specifications			
Des 5.10.1-EP	The circuit shall be powered by two power supplies.	Pass ☐ Fail ☐	
Des 5.11.1-EP	The circuit will use a 15V 1800mAh battery and a 5V 5200mAh battery.	Pass ☐ Fail ☐	
Des 5.11.2-EP	The microcontroller and all other electrical components will last a combined 34 hours under maximum load.	Pass ☐ Fail ☐	

Tester Name:		Date:	
Mask Design Specifications			
Relevant Specification(s)	Procedure	Result	Comments
Wearability and Comfort			
Des. 6.1.1-PC	The mask shall have a hard shell rugged enough to protect the electronic components from regular wear and tear.	Pass ☐ Fail ☐	
Des. 6.1.2-PC	The mask shall form an airtight seal with the user's face to ensure accurate measurements.	Pass ☐ Fail ☐	
Des. 6.1.3-PC	The mask shall be as compact and lightweight as possible, and made with light materials such as plastic, rubber, and fabric.	Pass ☐ Fail ☐	
Des 6.1.4-EP	The mask will be secured sturdily to the users face with adjustable straps that do not cause irritation to the users ears, skin, or scalp.	Pass ☐ Fail ☐	
Des 6.1.5-EP	The mask shall be comfortable to wear in general, and will inhibit physical activity as little as possible.	Pass ☐ Fail ☐	