PORTABLE OXYGEN CONCENTRATOR WITH FAULT DETECTION

By

Ahsan Shadab Patel

LIST OF FIGURES

FIGURE NAME	PAGE NO.
1.1 Process Flow	2
1.2 Control of proposed system	3
2.1 Process flow chart.	16
2.2 Adsorbate loading	16
2.3 Compressed Air input system	17
2.4 Representative example of Zeolite 5A molecular structure	18
2.5 Commercially available artificial Zeolite 5A spheres	19
2.6 Commercial synthesized Zeolite 13X cylinders	21
2.7 Oxygen generation in zeolite canister Z1	22
2.8 Nitrogen purge from Z1	23
2.9 Oxygen generation zeolite canister Z2	24
2.10 Nitrogen purge from Z2	25
2.11 Oxygen Sensor Circuit for calibration measurements	25
3.12 System Flow	22
3.13 Process Flow	25
3.14 Saturate the readings for 100 % Oxygen	29
4.15 Helium Gas where Oxygen is 0 %	30
4.16 Proposed model of oxygen analyzer with fault detection	58
4.17 Prototype of the proposed Oxygen Concentrator model	113
4.18 Prototype of the proposed Oxygen analyzer model	114

ABSTRACT

Long-term oxygen therapy (LTOT) is one of the several methods increasing the duration of survival in chronic obstructive pulmonary disease (COPD), with the oxygen concentrators being the most appropriate and economical choice for this treatment. There are many possibilities where there is no stable electricity or no access to compressed oxygen cylinders, these problems in real life are the most important factors causing delays in the treatment process because the patients are unable to receive oxygen at sufficient purity levels during the scheduled period.

I have designed and constructed a portable oxygen concentrator based on pressure swing adsorption that operates on a battery. Millions with ailments treatable by concentrated oxygen, like hypoxemia, live in countries without access to compressed oxygen or stable electricity for oxygen generation. Although compressed oxygen cylinders are preferred, each 300 cu ft-cylinder lasts only four days at 1.5 L/min flow. Portable electrical oxygen concentrators can be charged by solar power. My system generates oxygen concentrations ranging from 30% to 80%, typical for use in hospitals, with flow rate sufficient to supply one patient. My system compresses air into pores in zeolite (an alumina-silicate ceramic lattice with uniform pores 5x10-10 m diameter), preferentially adsorbing nitrogen based on molecular size and chemical affinity. Because nitrogen is retained, oxygen passes through at higher concentrations. The cycle is completed by purging nitrogen from the zeolite allowing repetition. Last year, I learned about Zeolite 13X which absorbs nitrogen based on pore size, requiring more pressure to achieve higher oxygen concentrations.

TABLE OF CONTENTS

CONTENTS	PAGE NO.
CERTIFICATE	ii
DECLARATION	iii
ACKNOWLEDGEMENT	iv
LIST OF FIGURES	v
ABSTRACT	viii
CHAPTER 1: INTRODUCTION	
1.1 Introduction	1
1.2 Literature Survey	1
1.3 Design of the Proposed System	2
1.4 Oxygen concentrator	2
1.5 Control of proposed system	5
CHAPTER 2: O2 SYSTEM DESIGN	
2.1 Oxygen Concentrator - Process Flow	6
2.2 Compressed Air and Filtration	7
2.3 Pressure Swing Adsorption	11
2.4 Oxygen Sensors and Calibrations	16
CHAPTER 3: O2 ANALYZER WITH FAULT DETECTION	
3.1 Introduction	21
3.2 Design of the Proposed System	21
3.3 Oxygen Analyzer	22
3.4 Block Diagram	25
3.5 Prototype of proposed	57
CHAPTER 4: LITERATURE SURVEY	
4.1 Pressure Swing Absorption Oxygen Concentratorequipped with Remote Monitoring Pulse	58
4.2 A Controller Design for Oxygen Concentrator	67
4.3 Design, implementation and testing of a portable device for multiparametric activity and SpO2 monitoring	72
CHAPTER 5: SOFTWARE	
5.1 Arduino	80
5.2 Code	81
CHAPTER 6: RESULTS AND OBSERVATIONS	112
CHAPTER 7: CONCLUSIONS AND FUTURE SCOPE	116
REFERENCES	118

1.1 Introduction

In recent years, the life quality of the patients with chronic obstructive pulmonary disease (COPD) has not only improved their health expenses but also seriously decreased due to the widespread use of long-term oxygen therapy (LTOT) and the developments in the monitoring opportunities of treatment.

When it is considered Journal of Mechanics in Medicine and Biology Vol. 12, No. 4 (2012) 1250060 (21 pages) °c World Scientific Publishing Company DOI: 10.1142/S0219519412005071 1250060-1 that the costs of a pulmonary blood gas measurement is equivalent to the costs of 2-day oxygen therapy and hospital stay expense is equivalent to the costs of 1-month oxygen therapy, the use of LTOT in the treatment of this disease can be clearly seen to be fairly economic.

A few studies published on LTOT have proven to be unique treatment method which extends the survival and improves the life quality due to the fact that it minimizes the decreases in the respiration functions during the treatment of the patients with COPDs.36 A research carried out on the 8487 patients in Denmark shows that the average life expectancy of a patient who receives the LTOT for about 1524 h/day increased from 1.07 to 1.40 years.

1.2 Literature Survey

In the literature, oxygen concentrators are reported to be the most suitable and economic choice for LTOT.1.8 Today, although the utilization rate of oxygen concentrators used in LTOT is 90% in France and 80% in America, 8.9 this treatment first launched in Turkey in 1986.10 The study made by Kurtar et al.6 showed that the rate of the problems in LTOT was 39%. These problems was identified respectively as the device fault, device maintenance, device expenses and no periodical maintenance, low oxygen purity, power cuts, power consumption and insufficient training given to the patient. Whereas the malfunction rate of all devices was 45%, the yearly maintenance rate of these devices was found to be 16%.

Technical service necessity is inevitable for the periodic maintenance of oxygen concentrators. The oxygen purity and oxygen °ow velocity should be checked ideally once in a month. In a similar study, Tuckered al.2 observed that almost all of those devices were wrongly used by the patients, technical services were insufficient, the yearly maintenance rate of these devices was highly low, and no regular maintenance was fulfilled on a certain number of patient devices.

In addition, the study showed that the hypoxemia in the patients could not be adequately improved by some oxygen concentrator because they were not supply oxygen at the expected purity level during long-term use.

All of the results indicate the fact that a significant part of the problems results from the structural deficiencies of oxygen concentrators and non-periodic device maintenance done by firm. The recent developments on wireless communication technology have made a major contribution to biomedical field as well as other fields.

Thanks to the use of this technology to monitor both patient and device parameters during the treatment process, it has become possible to develop low-cost and portable health monitoring systems.

1.3 Design of the Proposed System

The proposed system consists of an oxygen concentrator containing several equipment's to obtain the required oxygen, the controller unit providing the control of whole system, the sensor circuits used to sense the pressure in the product tank, purity of oxygen given to the patient, and internal ambient temperature.

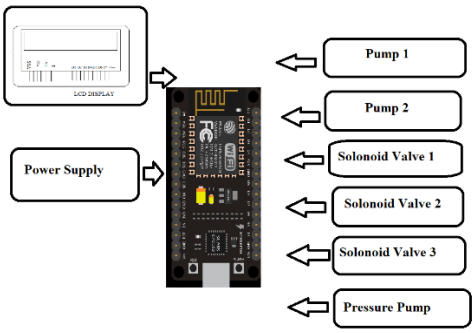


Figure 1: Process Flow

1.4 Oxygen concentrator

Oxygen concentrators are devices increasing oxygen in the air to a purity rate of 90% to 97%. The air which we breathe consists of approximately 78% nitrogen, 21% oxygen and 1% other gases. The easiest way to separate the oxygen from the air mixture is to use the pressure swing absorption technology developed by NASA. The process of obtaining oxygen using the pressure swing absorption technology is based on the air being filtered through the aluminosilicate minerals which are known as zeolite. For this purpose, the minerals are placed into a container known as molecular or zeolite bed. When the ambient air is applied to this structure with a specific pressure, the oxygen passes through it into the output with the applied pressure while the nitrogen molecules in the air are absorbed with the minerals in the bed. Block diagram of the designed oxygen concentrator using the pressure swing absorption technology is shown in Fig. 2. The concentrator device consists of an air filter, a compressor, a three solenoid valve, a molecular sieve, a product tank, a pressure-regulator, a water container and an exhaust component. In this system, the air taken from the atmosphere is passed through the air filter, and then sent to the first molecular sieve with the pressure provided by compressor. While the pressurized nitrogen

which enters into the molecular sieve is being held in the zeolite bed, the oxygen is allowed to pass through unrestricted into the product tank. When the pressure in the product tank reaches 23 PSI, the zeolite in the first molecular sieve is completely saturated with nitrogen. Thus, the compressed air is given to the second sieve by changing the valve position so that oxygen production can continue. Meanwhile, the first molecular sieve is depressurized and regenerated by the removal of the absorbed nitrogen, carbon dioxide and water vapor. When the zeolite in the second sieve is saturated with the nitrogen, the pressurized air is given to the first sieve again and the nitrogen in the second sieve starts to be thrown out by the egzost system. The pressurization and depressurization cycle proceeds alternately during the system operation. The oxygen which is brought into a suitable pressure and a purity of 90%-95% in the product tank is passed through into the pressure regulator and the °O2 meter respectively. Then, it is given to the patient with a nasal canola or an oxygen mask. The molecular sieves which contain a chemical called \5A molecular sieve" are one of the most important components that affect the device performance.

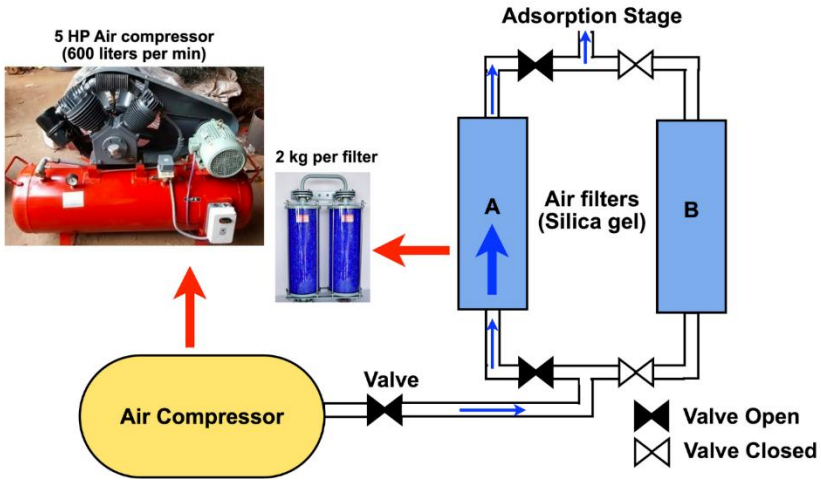
Two molecular sieves are used to supply the oxygen continuity in the designed system, because this chemical is quickly saturated with nitrogen. The design of each molecular sieve depends on basic parameters such as the bed length, the bed diameter, the package type of the bed and the air pressure fed to the bed. Two types of packaging are used in the production of molecular sieves: conventional packaging and multilayered packaging. The conventional one is a uniform packaging type made by using only one kind of molecular sieve.

In other words, one kind of molecular sieve with the same diameter is uniformly distributed in the adsorption column. One of the most significant problems faced with this type packaging is that the °ow velocity in the central core of the column is much higher than the velocity in the outside of the central core. As a result, while the molecular sieve in central core is firstly saturated by adsorbing enough nitrogen, the outside of the central core is still not saturated. Thus, the molecular sieves in the whole column are not sufficiently used in this type packaging.

The drawbacks of the conventional packaging can be eliminated by reducing the °o2 velocity in the central core. This function can be achieved by using a multi-layer arrangement called multi-layered packaging instead of using a single type molecular sieve in the whole column. In the multi-layered packaging, the small diameter of molecular sieves is packed in the central core while the larger diameter of sieve lies in the outside of the central core.25 Two-layered packaging containing central core and outer core are used in this study. The dimensions of molecular sieves in the central and outer ones are 1.6 mm and 2.1 mm, respectively. The diameter of each cylinder is 82 mm, with heights of 650 mm each. The compressor used to provide the necessary pressure to the molecular sieves is a dry-air compressor which comprises a single electric motor and two reciprocating piston mechanisms driven from opposite ends of the motor shaft. Each mechanism contains a piston which is reciprocated in a cylinder by the motor. The two cylinders are connected together to provide the required air °ow and pressure used to produce oxygen at a desired purity. The experimental studies show that an electric motor of 330 W/1.5 A can obtain the

pressure of 3 bar, which provides oxygen at a purity of about 9582% from the oxygen concentrator output at a °ow velocity of 15 L/min. Another factor that affects the oxygen purity in the system output is the volume of the product tank used for storing the oxygen from molecular sieves. The use of a large volume tank in the oxygen concentrator causes the decrease of the oxygen purity in the output of device while a small volume tank increases the fluctuations in oxygen °ow. Therefore, the tank volume in the design of oxygen concentrator is determined in accordance with the dimension of molecular sieves and the compressor power. An air filter, which filters all particles including harmful substances such as dust, humidity and pollen caused by the environmental conditions until a minimum of 0.3 microns, is used in concentrator input. A pressure regulator that decreases the pressure of approximately 23 PSI in product tank to 510 PSI is connected to the tank output to minimize the °ow imbalances in °ow meter input. Then, the oxygen is given to the patient over the water container with a 9 PSI-valve security by setting the °ow meter to the °ow velocity desired.

1.5 Control of proposed system:



2. System Design

2.1 Oxygen Concentrator - Process Flow

The design below is scalable from portable units for individual use that are commercially sold to larger installations that some hospitals use to produce their own medical grade oxygen. What changes from the portable to institution-scale systems is the compressor size and the quantity of zeolite needed for concentrated oxygen production. Otherwise, the process, which is shown in the flowchart below (Figure 1) is identical between the two scales.

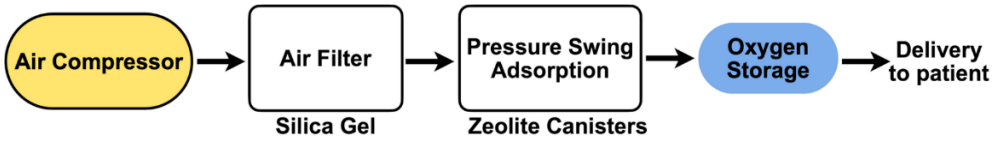


Figure 1: Process flow chart.

The basic process requires an air compressor that pumps air at a pressure above the standard atmospheric pressure through an air filter. The air then passes through a chamber containing the zeolite crystals that remove nitrogen, and out comes medical grade oxygen that is stored in a tank before being supplied to patients. Now we get into the specifics of each step in the above process.

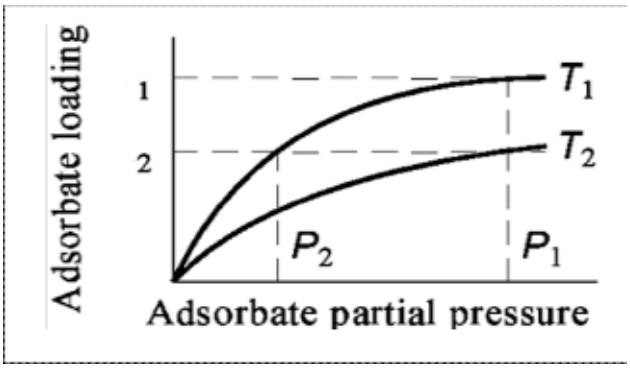


Figure 2: Adsorbate loading, i.e. Nitrogen adsorbed as a function of Adsorbate partial pressure, i.e. external pressure applied at two different temperatures.

We start by shooting down our earlier assumption -- that all nitrogen in the pumped air is removed by the zeolite crystals: **it is not**. I will not get into an involved discussion of the thermodynamics of sorption kinetics and adsorption equilibria, the interested reader is referred to this study. Simply put, the amount of nitrogen removed by zeolite depends on both temperature and pressure of air (see Figure 2 for a representative pressure curve). The fraction of nitrogen removed by zeolite from incoming air increases with decreasing temperature ($T_1 < T_2$ in Figure 2 above) as well with increasing pressure ($P_1 > P_2$ in Figure 2 above).

2.2. Compressed Air and Filtration

Although temperature is more readily manipulated in domestic settings, higher pressure does better for nitrogen adsorption in the present context, and pressurized air can be more easily achieved than we realize. Every gas station (petrol pump, if you're in India), welding shop, and auto shop has an air compressor in India. Portable car tire inflators are basically the same, just in a compact form.

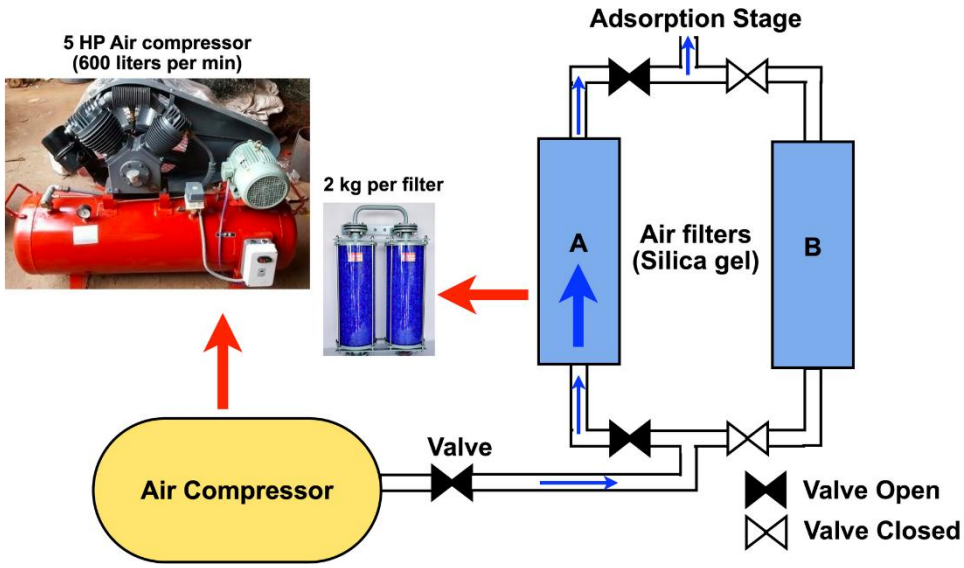


Figure 3: Compressed Air input system.

Whether you pick an industrial size air compressor as I did (see inset in Figure 3 above), or a portable car tire inflator, make sure you select one capable of going up to 6-8 bars of pressure (if you're more comfortable with PSI, pounds per square inch, 1Bar = 14.5 psi). The compressor I used is an Ingerrsol Rand 5 HP air compressor (purchased second-hand for INR 14000, approx. JPY 20000) capable of processing 600 liters per minute of air. If purchasing a new one, try to pick one that is oil free, but it's not important.

As shown in Figure 3 above, we connect the output from the air compressor to a HEPA filter or an air filter with silica gel. Air contains water vapor, plus if you use an old compressor as I did, the pressurized air from the compressor also contains tiny oil droplets. Zeolite crystals adsorb both vapor molecules and oil, and any zeolite surface occupied by vapor or oil is then unavailable for nitrogen adsorption, which means that the efficiency goes down. In fact, zeolite adsorbs vapor molecules far more readily (even at atmospheric pressure) than nitrogen, the reason being that water is polar and more readily attracted to zeolite whereas nitrogen interacts via a quadrupole moment. For this reason, it’s essential to pass the pressurized air exiting the compressor through a HEPA filter or a silica gel desiccant that can absorb vapor and oil.

In the design above (see Figure 3), I used two silica gel desiccant air filters in parallel. I use filter A initially, and once it saturates with vapor and/or oil, I shut off the valve and re-direct incoming pressurized air to filter B and hot-swap a new filter in place of A to ensure continuous operation. About 2 kg of silica gel per filter can last a few hours, depending on the liters per minute of air processed by the system. The swapped out air filter is then opened and the moisture in the silica gel can be removed by heating it at 110ºC. You can use a convection oven or gently heat the silica gel on a stove and it's ready for reuse. If you have a convection oven, it's advisable to pre-heat the oven and bake the silica gel for 20 minutes at 150ºC. If heating on a stove, use a baking/oven thermometer to take it up to 110ºC and keep gently turning over the silica gel particles for 1 hour.

Word of caution: In a cheap early version, I tried to bubble the compressed air through a bottle of glycerin since it is both hygroscopic and oleophilic. But the bubbling causes huge pressure fluctuations at the next stage and the process was a disaster. HEPA or Silica gel air filters cut out these fluctuations, and in fact, owing to Darcy's law, they even smooth out the fluctuations, although this also causes a pressure drop at the output end of the filter.

Before we proceed to the next and most crucial stage of the concentrated oxygen generation process, I will take a small digression to explain a bit about zeolites -- the material used to adsorb nitrogen. Zeolites are a class of microporous aluminosilicate materials commonly used as adsorbents and catalysts because they act as molecular sieves. This property arises from removal of their "water of crystallization" which leave behind a porous structure or cage at the molecular scale. These pores or cages can accommodate a wide variety of cations (positive ions) including sodium, potassium, calcium, magnesium, which are loosely held. These cations can be readily exchanged with other molecules by external application of pressure, or thermal or electrical forces.

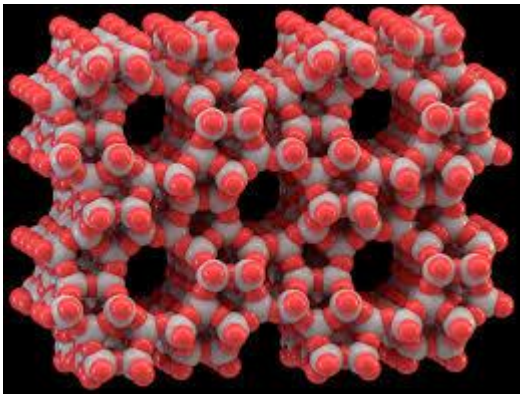


Figure 4: A representative example of Zeolite 5A molecular structure showing the five pores or cages at the center of the unit cell. These pores allow molecules smaller than the cage size to pass through, provided they are not attracted to the superstructure.

Zeolites are a family of such molecular sieves comprised of various compounds, some of which occur naturally, but many are synthesized artificially for industrial applications. For

reasons unknown to me, it turns out India is a major producer of zeolites. They are used as catalysts in fractionation of hydrocarbons, as desiccants for water removal, and as in our present case, for separation of gases.

In particular, for our purpose, two types of artificial zeolites, known as Zeolite 5A and 13X are suitable for nitrogen adsorption from air at low temperatures and/or high pressures. Now, all gases that make up air are non-polar; we know from elementary chemistry that oxygen and nitrogen molecules are electrically neutral and the existence of the molecule itself is due to electronic stability. Then why does nitrogen adsorb to Zeolite 5A or 13X whereas oxygen does not?

The reason is owed to a higher order interaction between zeolite and air. The positive charge on the zeolite molecule induces a quadrupole moment in both nitrogen and oxygen, but it is three times stronger in nitrogen than it is in oxygen. For this reason, far more nitrogen gets attracted to the zeolite crystal surface, whereas oxygen passes through the pore or cage. However, we know quadrupole interactions are much weaker than dipolar interactions, and for this reason desorption (i.e. removal) of nitrogen from the zeolite matrix is also easily achieved. Much of the process described below in the subsection 2.3 on Pressure Swing Adsorption concerns the initial adsorption, followed by subsequent desorption of nitrogen in a cyclic fashion to achieve continuous concentrated oxygen generation. Between Zeolite 5A and 13X, Zeolite 13X exhibits much higher selectivity towards nitrogen.

Commercially, Zeolite 5A and 13X are manufactured as spheres or cylinders at macroscopic scales, as shown in Figure 5 and 6 below.

Figure 5: Commercially available artificial Zeolite 5A spheres.

Figure 6: Commercial synthesized Zeolite 13X cylinders.

2.3 Pressure Swing Adsorption

Once the pressurized air passes through the HEPA or desiccant filter, it enters the crucial stage of nitrogen removal from air. This stage is commonly known as Pressure Swing Adsorption (PSA) because it involves a series of four steps repeated in a cycle for continuous generation of pure medical grade oxygen. The schematic is shown together with step 1 of this 4-step procedure. Although the schematic looks rather complicated, it is quite simple and mostly involves plumbing.

First, a brief description of the structure. I took two aluminum cylinders and filled the bottom with a bed of glass marbles and poured Zeolite 13X into the cylinders. After jiggling the cylinders to compact the zeolite beads, a second bed of glass marbles formed the top layer. These two cylinders form the zeolite canisters Z1 and Z2 in the schematic below (see Figure 7).

Word of caution: An early version of zeolite canister made with PVC pipe was a disaster. I used a pipe that wasn't thick enough to handle the pressure. Secondly, PVC pipes have to be glued to seal them airtight and if anything goes wrong they have to totally discarded. Instead, metal canisters (see Figure 7 inset and Figure 10 inset below) can be sealed with gaskets or O-rings and can be held together with screws. This makes it easier to replace the zeolite if necessary. This is particularly true if water vapor invades the zeolite canisters. As it takes much higher pressures and/or temperatures to remove the water, instead it's easier to simply replace it with a fresh batch of zeolite beads.

A second problem comes from the tubing. If you use cheap, collapsible PVC tubes, they easily kink and cause pressure blow outs. Use either thick PVC tubing used in vacuum applications as I did for the portable version or copper tubing as I did for the larger version. It is more stable.

The canisters were sealed with gaskets. The filtered compressed air enters the bottom of a pipe that bifurcates in two.

In step 1, we open valve V2 and close valve V4 to operate canister Z1. Valve V1 should be closed, else the compressed air escapes directly back into the room. Keeping V2 open and V1 shut allows the filtered air to pass through the zeolite bed at 6-8 bars of pressure and purified oxygen is released from the top end. By keeping valve V5 closed and V6 open, the purified medical grade oxygen can exit to a storage cylinder.

The inset in Figure 7 shows a portable working version of the two aluminum zeolite canisters and the associated plumbing. This was run with a portable car tire inflator and produced 9 liters of purified medical grade oxygen per minute.

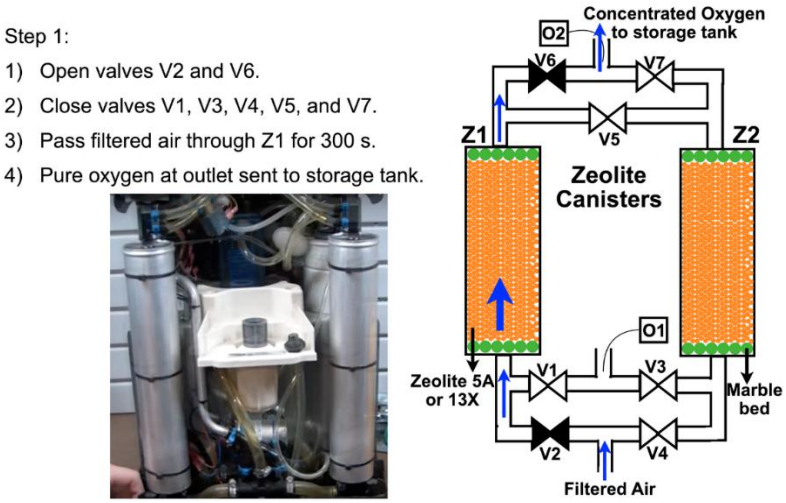


Figure 7: Pressure Swing Adsorption
Step 1 -- Oxygen generation in zeolite canister Z1:

The zeolite is soon saturated with nitrogen, and the process must stop before this happens. How long one can produce oxygen with a single canister depends on the pressure and quantity of zeolite used. The portable version had canisters of 5 cm in diameter and 20 cm in height. But the larger version shown in inset of figure 10 using the larger 5HP air compressor used canisters of 25 cm in diameter and 1 m in height. These large canisters could run 450 - 470 seconds before the zeolite was saturated with nitrogen. To be on the safer side, we stopped this step 1 after 300 seconds. Later, I will explain how the calibrations were performed to work out the saturation time, so you can perform them on your own system.

The step 1 process for pressure swing adsorption is described in Figure 7 above.

Step 2:

1) Close valves V2, V3, V6, and V7.
2) Open valves V1, V4 and V5.
3) Pass filtered air through Z2 for 100 s.
4) Expunge Nitrogen purged from Z1.

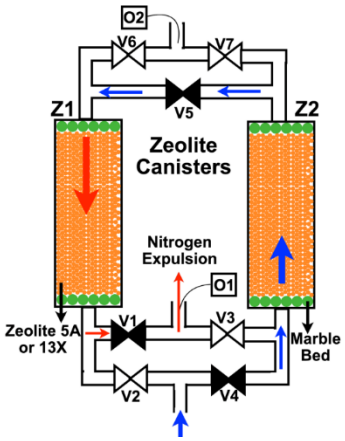


Figure 8: Pressure Swing Adsorption
Step 2 -- Nitrogen purge from Z1.

After step 1 is completes, in step 2 we close valve V2 and open valves V4 and V5, but keep V3 and V7 closed and pass the compressed air through canister Z2. The purified oxygen coming out of Z2 passes through V5 into canister V1 and purges its zeolite bed of the adsorbed nitrogen. Opening valve V1 then releases this nitrogen into open air. See schematic and the outlined steps in Figure 8 above.

Step 3:

1) Close valves V1, V2, V3, V5, and V6.
2) Open valves V4, and V7.
3) Pass filtered air through Z2 for 300 s.
4) Pure oxygen at outlet sent to storage tank.

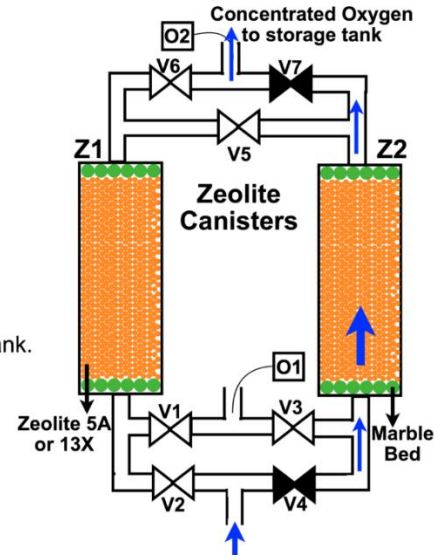


Figure 9: Pressure Swing Adsorption
Step 3 -- Oxygen generation zeolite canister Z2.

After purging nitrogen from Z1 for 100 seconds in step 2, in step 3 we shut off valve V5 and open valve V7 to let pure oxygen into the storage cylinder for 300 seconds. By now canister Z2 has run for a total of 400 seconds and is saturated with adsorbed nitrogen.

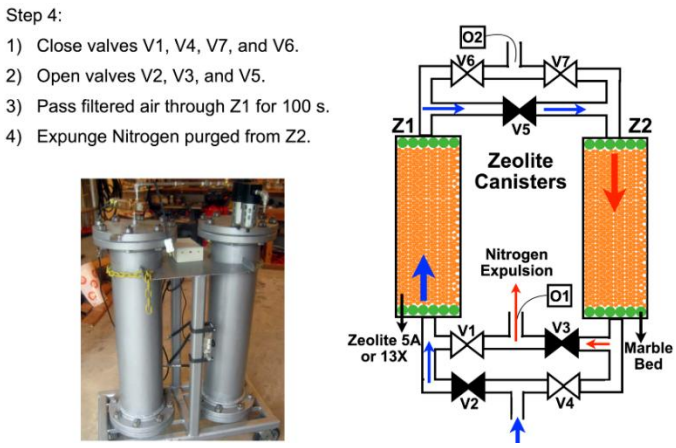


Figure 10: Pressure Swing Adsorption

Step 4 -- Nitrogen purge from Z2.

Now in step 4, we close valve V4 and re-open valve V2. But we do not open valve V6. Instead we open valves V5 and V3 so the compressed air enters canister Z1 and the purified oxygen passes through V5 to enter canister Z2 to desorb nitrogen from its zeolite bed. This process runs for 100 seconds before valve V5 is shut off.

We then repeat steps 1 - 4 and cycle through in order to get continuous oxygen output.

This four step cyclic process is standard for both portable and industrial scale oxygen concentrators and is known as Pressure Swing Adsorption, because the pressure is applied in step 1 to adsorb nitrogen and produce oxygen, then it is swung in step 2 to purge the saturated nitrogen from canister Z1 using the pure oxygen produced by Z2. In step 3, pure oxygen is collected again from Z2 before it gets saturated at the 400 second mark. Finally in step 4, the re-generated zeolite bed in canister Z1 is used to purge saturated nitrogen from canister Z2 for 100 seconds.

Doing so is beneficial because in older processes, the nitrogen purge was performed by shutting of air supply and applying a vacuum to suck out nitrogen, but this would involve a second external line which would introduce short quantities of water vapor between line switches and would eventually degrade the zeolite canisters beyond use.

The pressure swing adsorption process allows near continuous oxygen output, except for the 100 second durations in steps 2 and 4 when it is used to purge nitrogen from zeolite canisters. For this reason, the last stage involves a storage tank. The tank provides continuous supply of medical grade oxygen to patients in need so that they do not notice the oxygen output cut off for 100 second duration in steps 2 and 4. Furthermore, the storage tank also acts like a capacitor does in electrical circuits to smooth fluctuations. Just as a capacitor stores charge, the tank stores oxygen, and if there are any pressure fluctuations upstream, they are smoothed out by the stored oxygen.

In my implementation, I used a 20 liter cylinder for oxygen storage in the portable version and a 250 liter gas tank in the larger version. Whereas the portable version was useful for individual use, the large version provided a central generation line that could continually supply up to 10 patients self-isolating in a house.

2.4 Oxygen Sensors and Calibrations

The 100 and 300-second durations applied in the above steps were determined from calibrations I performed on these home-built canisters. In order to do so, I had to embed oxygen sensors to measure both the purity and the duration for which each canister would output purified oxygen before its zeolite bed was saturated with nitrogen.

Ideally, I would have liked to use both oxygen and nitrogen sensors, but I was unable to source nitrogen sensors locally. So as I explain below, I used oxygen sensors as a proxy.

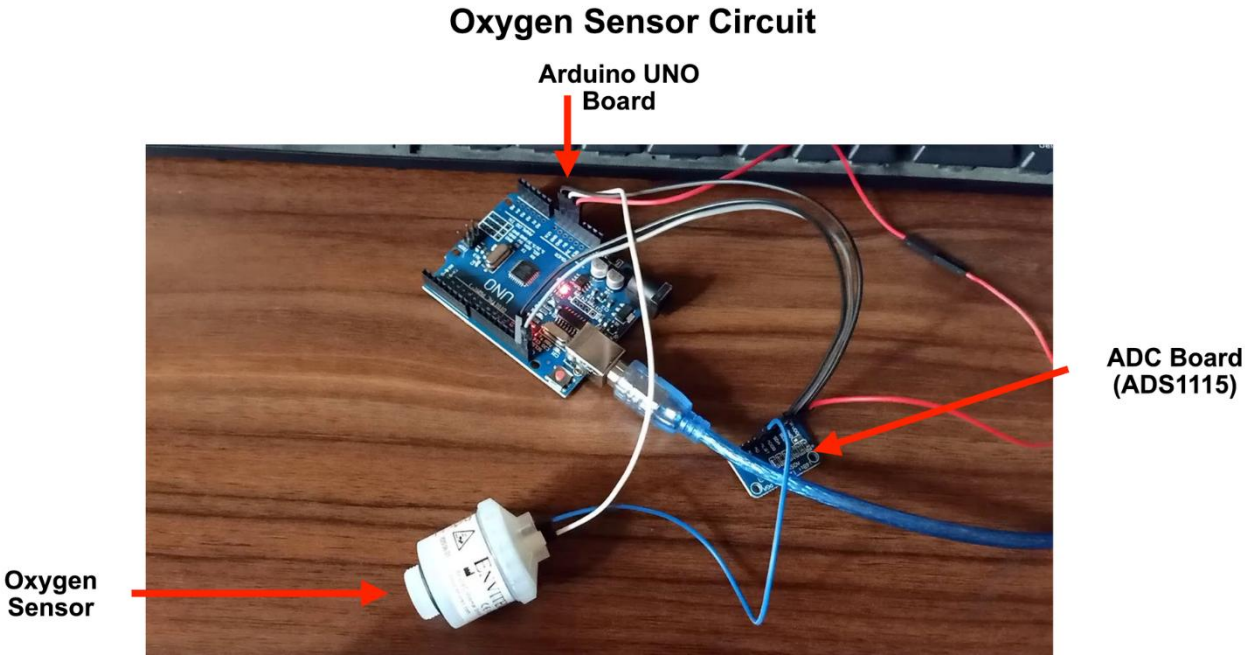


Figure 11: Oxygen Sensor Circuit for calibration measurements

The schematic shown in Figures 7 - 10, has two sensors O1 and O2 at the nitrogen and oxygen outlets. The sensor O2 operates as a normal oxygen sensor, but O1 sensor placed near the nitrogen expulsion outlet detects for end of nitrogen purge when pure oxygen starts being expelled. This sets the duration for nitrogen purge.

I used two medical grade OOM202 oxygen sensors from Envitec. This point is important -- since air contains approximately 21% oxygen, most cheap oxygen sensors can only detect oxygen in the range 0 - 26%. Medical grade oxygen sensors used in ventilators can detect oxygen over the entire range of 0 - 100% oxygen level.

The OOM202 is an active sensor that does not require an external power source and provides output in the range of 13 - 16 mV. Since this voltage level is low for Arduino detection, I used an analog-to-digital convertor (ADC) with a gain before interfacing it with an Arduino UNO target board. The ADC I used is ADS1115 that mates easily with the Arduino.

The OOM202 sensor had three pins, of which the ground pin (white wire in Figure 11 above) and analog output (blue wire in Figure 11) are used. The analog output is connected to A0 terminal of the ADS (ADS1115) whereas the white cable is connected to GND port of the Arduino UNO target board. The ADS1115 has two outputs, the SCL and SDA pins which connect to the corresponding SCL and SDA pins on the Arduino UNO target. The ADS1115 takes a 5 V power input and a ground connection, both of which are drawn from the UNO target. This completes the circuit.

The sensor was inserted in the outlet pipe wall. Its 13 -16 mV output is converted by the ADC to a digital output in the range 70 - 350 (70 being 0% oxygen and 350 being 100% oxygen reading). In total each sensor (Rs. 4000), UNO board (Rs. 500) and ADS 1115 (Rs. 500) system cost Rs. 5000 (approx. 7200 yen).

The arduino target was run from a laptop using serial interface. The program I used is pasted below for your reference.

Oxygen Analyzer and Fault Detection and Alert System

3.1 Introduction

An oxygen analyzer is a device used to measure the concentration of oxygen (O2) in a gas mixture. This measurement is crucial in various industries and applications where maintaining the right oxygen levels is important for safety, quality, and efficiency. Oxygen analyzers are utilized in fields such as healthcare, industrial manufacturing, food and beverage production, environmental monitoring, and research.

The present invention relates to an advanced oxygen concentrator and analyzer, combining Arduino Uno with OOM202 sensor, GPS, GSM module, OLED display, and Max30100 sensor. This innovative medical device offers automated error detection and alert capabilities, ensuring patient safety and efficient communication with service authorities. The oxygen concentrator delivers medical-grade oxygen while monitoring the purity of oxygen, blood oxygen saturation (SPO2), and heart rate, providing valuable health data to healthcare professionals.

The oxygen concentrator and analyzer described in this patent application represent a significant advancement in the field of medical devices. It addresses the need for a portable, reliable, and feature-rich oxygen delivery system that ensures precise oxygen concentration levels while simultaneously monitoring vital health parameters, such as SPO2 and heart rate. The integration of Arduino Uno, OOM202 sensor, GPS, GSM module, OLED display, and Max30100 sensor ensures efficient and accurate operation, making it a valuable device in various medical settings.

Background: Oxygen concentrators are widely used in medical practice to deliver supplemental oxygen to patients with respiratory conditions. Conventional concentrators are limited in their functionalities, often lacking the ability to monitor oxygen purity and alert healthcare providers in case of errors. Furthermore, the need for portable oxygen concentrators with remote tracking capabilities has become increasingly crucial, particularly for patients in remote areas or during emergency situations. The proposed oxygen concentrator and analyzer aim to address these limitations and provide a comprehensive solution.

3.2 Design of the Proposed System

The proposed system consists of an oxygen analyzer containing several equipment’s to obtain the required oxygen purity, SpO2, the controller unit providing the control of whole system, the sensor circuits used to sense the SpO2, purity of oxygen given to the patient, and sends an SMS alert to the service provider with the GPS coordinates of the system to be serviced.

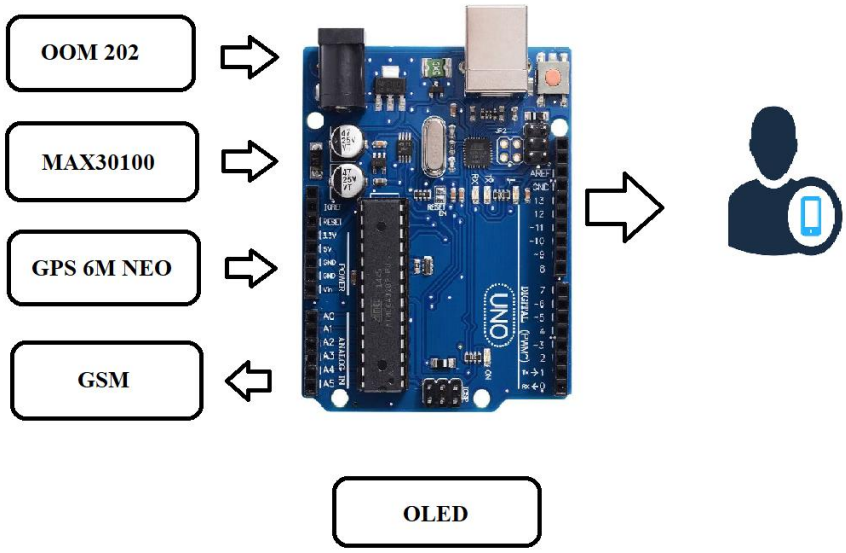


Figure 12: System Flow

3.3 Oxygen Analyzer

IEEE Transactions on Information Technology in Biomedicine 8(4):439–47, 2004.

An oxygen analyzer, also known as an oxygen sensor or oxygen meter, is a device used to measure the concentration of oxygen (O2) in a gas mixture. These devices are widely used in various industries and applications, including medical settings, industrial processes, environmental monitoring, and research.

Oxygen analyzers work on the principle of oxygen partial pressure or oxygen concentration. They typically use electrochemical, optical, or paramagnetic sensors to measure oxygen levels. Here's a brief overview of these sensor types:

1. **Electrochemical Sensor**: This is the most common type of oxygen sensor. It consists of an electrolyte solution, two electrodes, and a gas-permeable membrane. When oxygen comes into contact with the sensor, a chemical reaction occurs in the electrolyte, generating an electrical current proportional to the oxygen concentration. This current is then used to calculate the oxygen level in the gas mixture.
2. **Optical Sensor**: Optical oxygen sensors use the luminescence quenching phenomenon. A special dye is embedded in a material, and when this dye interacts with oxygen, its luminescence changes. By measuring the change in luminescence, the oxygen concentration can be determined.
3. **Paramagnetic Sensor**: Paramagnetic oxygen sensors rely on the fact that oxygen is paramagnetic, meaning it is weakly attracted to a magnetic field. When a gas mixture flows through a magnetic field, the oxygen in the mixture will cause a disturbance in the field. This disturbance can be measured and used to calculate the oxygen concentration.

Oxygen analyzers are crucial in various contexts. In medical settings, they are used to monitor oxygen levels in patients' blood and help manage conditions such as respiratory diseases. In industrial applications, oxygen analyzers are used to monitor and control processes where oxygen levels are critical, such as in the production of metals, food and beverage, and chemical manufacturing. Additionally, they are used in environmental monitoring to assess air quality and ensure workplace safety.

3.3.1 Electrochemical oxygen sensor:

An electrochemical oxygen sensor, also known as a galvanic cell oxygen sensor, operates based on the electrochemical reactions that occur when oxygen interacts with specific materials in the sensor. The sensor consists of several key components:

1. **Electrolyte Solution**: The sensor contains an electrolyte solution that facilitates the electrochemical reactions between the oxygen and the sensor's electrodes.
2. **Anode (Positive Electrode):** The anode is typically made of a noble metal, such as gold or platinum, coated with a porous layer of a metal oxide. The anode is exposed to the gas mixture being measured.
3. **Cathode (Negative Electrode):** The cathode is also coated with a metal oxide and is typically made of the same material as the anode. It is isolated from the gas mixture and separated by a gas-permeable membrane that allows oxygen to pass through.
4. **Gas-Permeable Membrane**: This membrane separates the cathode from the gas mixture and allows oxygen molecules to diffuse through. It prevents other gases and contaminants from interfering with the measurement.

Here's how the electrochemical oxygen sensor works:

1. **Oxygen Interaction**: Oxygen molecules from the gas mixture diffuse through the gas-permeable membrane and reach the cathode side of the sensor.
2. **Cathode Reaction**: At the cathode, oxygen molecules undergo a reduction reaction. This reaction involves the oxygen molecules reacting with the metal oxide on the cathode surface and taking electrons from the cathode, resulting in the formation of oxygen ions (O2-).
   $O2 + 4e- \rightarrow 2O2.$
3. **Electron Flow**: The electrons released during the cathode reaction flow through an external circuit to the anode.
4. **Anode Reaction**: At the anode, oxygen ions and electrons combine to form oxygen molecules.
   $O2- + 4e- \rightarrow O2$
5. **Current Generation**: The flow of electrons from the cathode to the anode generates an electrical current that is proportional to the concentration of oxygen in the gas mixture.
6. **Measurement and Calibration**: The generated current is measured and converted into an oxygen concentration reading. The sensor is calibrated based on known oxygen concentrations to ensure accurate measurements.

It's important to note that the electrochemical reaction is sensitive to temperature, humidity, and other environmental factors. Therefore, proper calibration and compensation for these factors are necessary to obtain accurate oxygen concentration measurements. Electrochemical oxygen sensors are widely used due to their relatively simple design, reasonable cost, and reliability in various applications where oxygen monitoring is essential.

3.3.2 Optical oxygen Sensor:

An optical oxygen sensor, also known as an optical oxygen analyzer or luminescent oxygen sensor, operates based on the principle of luminescence quenching. These sensors use a special luminescent material that responds to the presence of oxygen molecules by changing its luminescence properties. Here's how an optical oxygen sensor works:

1. **Luminescent Material**: The sensor contains a luminescent material or dye that emits light (luminescence) when excited by a light source. The luminescent material is chosen for its ability to change its luminescence properties in response to the presence of oxygen.

2. **Light Source**: The sensor is equipped with a light source, often an LED (light-emitting diode) that emits light of a specific wavelength to excite the luminescent material.
3. **Gas-Permeable Membrane**: The luminescent material is typically embedded in a gas-permeable membrane that separates it from the gas mixture being measured. This membrane allows oxygen molecules to diffuse through while keeping out other gases and contaminants.
4. **Luminescence Quenching**: When the luminescent material is excited by the light source, it emits luminescent light. However, when oxygen molecules interact with the luminescent material, they cause a phenomenon called luminescence quenching. Oxygen molecules quench (reduce) the luminescence intensity of the material.
5. **Measurement of Luminescence**: A photodetector is used to measure the luminescent light emitted by the material. The intensity of the luminescence is proportional to the oxygen concentration in the gas mixture.
6. **Calculation of Oxygen Concentration**: By comparing the luminescence intensity of the material in the presence of the gas mixture with a reference value obtained in a known oxygen-free environment, the oxygen concentration can be calculated.
7. **Temperature and Pressure Compensation**: The optical oxygen sensor's accuracy can be affected by temperature and pressure variations. Therefore, many sensors include mechanisms to compensate for these factors and provide accurate measurements.

The main advantage of optical oxygen sensors is their ability to provide non-intrusive and non-consumptive measurements, meaning they do not consume the oxygen they measure and do not interfere with the gas mixture. They are often used in applications where the gas mixture needs to remain uncontaminated or where other types of sensors might be impractical.

Optical oxygen sensors are commonly used in medical applications (such as blood oxygen level monitoring), industrial processes, environmental monitoring, and research, where accurate and reliable oxygen concentration measurements are crucial.

3.3.3 Paramagnetic oxygen Sensor:

A paramagnetic oxygen sensor operates based on the principle that oxygen is weakly attracted to a magnetic field due to its paramagnetic properties. These sensors use a magnetic field to measure the oxygen concentration in a gas mixture. Here's how a paramagnetic oxygen sensor works:

1. **Magnetic Field**: The sensor generates a magnetic field using a permanent magnet or an electromagnet. This magnetic field creates a force that attracts or repels paramagnetic substances, like oxygen, when placed in its presence.
2. **Gas Chamber**: The sensor has a gas chamber or cell that is exposed to the gas mixture being measured. This chamber typically contains a reference gas, which is a known oxygen-free gas that provides a baseline for comparison.
3. **Gas Mixture Interaction**: When the gas mixture flows into the gas chamber, the oxygen molecules within the gas mixture experience a force due to the magnetic field. Oxygen is paramagnetic, meaning it is slightly attracted to the magnetic field. As a result, oxygen molecules will be drawn toward the region of the magnetic field.
4. **Gas Movement**: The movement of the gas molecules in response to the magnetic field causes a change in the pressure within the gas chamber.
5. **Pressure Differential Measurement**: The sensor measures the pressure differential between the reference gas (with negligible oxygen content) and the gas mixture being measured (with varying oxygen content).
6. **Calculation of Oxygen Concentration**: The pressure differential is proportional to the concentration of oxygen in the gas mixture. By comparing the pressure differential to the reference gas pressure, the oxygen concentration can be calculated.
7. **Temperature and Pressure Compensation**: Similar to other types of sensors, paramagnetic oxygen sensors require compensation for changes in temperature and pressure to ensure accurate measurements.

It's important to note that paramagnetic oxygen sensors are sensitive to other factors that can affect the gas movement, such as gas flow rates, gas viscosity, and sensor geometry. Careful calibration and corrections are necessary to account for these factors and ensure accurate oxygen concentration measurements.

Paramagnetic oxygen sensors are commonly used in various industrial and laboratory settings, especially where accurate and continuous monitoring of oxygen levels is required, such as in industrial gas production, combustion control, and environmental monitoring applications.

3.4 Block Diagram

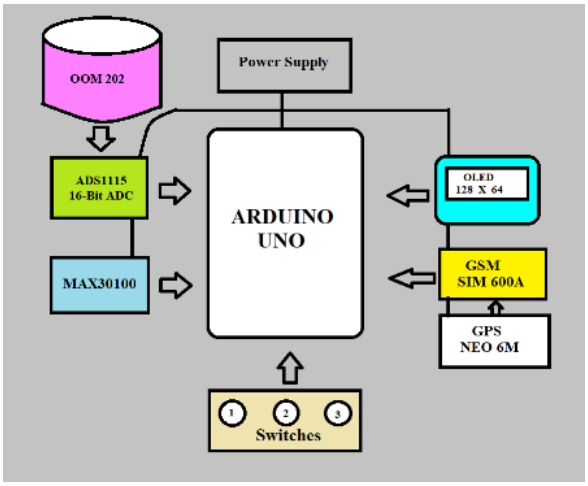


Figure 13: Process Flow

In this development we will be using the following components:

1. MCU or MCU Board : **Arduino UNO**
2. Oxygen Sensor : **OOM202 Oxygen Sensor (Envitech Series by HoneyWell)**
3. 16-ADC : **ADS1115**
4. Display : OLED **128 x 64**
5. Temperature Sensor: **MAX30100**
6. Switches : **Push Buttons**
7. GSM Module: **GSM SIM600A**
8. GPS Module: **GPS NEO-6M**

Hardware:

3.4.1 OOM202 Oxygen Sensor:

OOM202 is a medical grade oxygen sensor made in Germany by Honeywell.

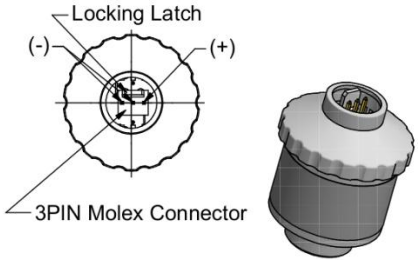
OOM202

Some important parameters of OOM202 Oxygen Sensor to be consider.

- Output in Ambient Air : 13 mV to 16 mV
- Response Time : < 12 s to 90% final value
- Linearity error : < 3 % relative
- Built in Temperature Compensation

OOM202 Sensor Pin Map and Interfacing

The Sensor has 3 pin to interface it with ADC, out of which 2-pins are marked '-' (negative) and 1-pin is marked '+' (positive).



The Positive will connect to any of the four Analog Channel of ADS1115 (In this design it is connected A0) and any one negative pin is connected to GND.

Understanding OOM202 Oxygen Sensor Output Range

The datasheet clearly mention for ambient Air it measures between 13 mV to 16 mV.

Ambient Air in Atmosphere contains 20.95% of Oxygen

Therefore we can say

Sensor measure 13 mV to 16 mV for 20.95% of oxygen

If we consider Base value that is 16 mV for 20.95% oxygen then

The upper range as per linear behavior of OOM202 Sensor will be

- Upper Range = 16 mV x (100 / 20.95)
- **Upper Range = 76.37 mV**

So therefore

For 0 to 100 % of Oxygen, Sensor will output 0 mV to 76.37 mV.

Obviously there might be some variation as per Linear Error mentioned in datasheet, that is < +/-3%.

Converting Sensor Output in mV to Oxygen Concentration in %
The sensor outputs the mV (millivolt) as per Oxygen Concentration and to convert it into Oxygen percent we can do the simple maths

For Ambient Air my sensor is showing 15.4 mV.

So for 20.95% Oxygen sensor outputs 15.4 mV

- Here 15.4 mV is baseline reading
- Now my calculation for Oxygen % of sensor output in mV will be

Oxygen Concentration in % = ((Sensor Output in mV / Baseline Volatge in mV) * Oxygen % in Ambient Air)

Thus

Oxygen Concentration in % = ((Sensor Output in mV / 15.4 mV) * 20.95 %)

For Example

- if Sensor is exposed to unknown percent of Oxygen and it Output 69 mV.

Then

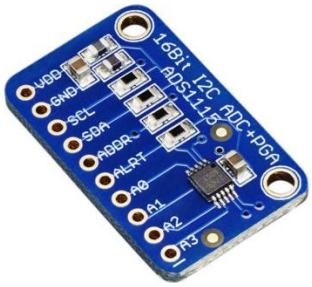
- ***Oxygen = ((69 mV / 15.4 mV) * 20.95 %)***
- ***Oxygen = 93.86 %***

Therefore the device measures 93.86 % Oxygen Concentration.

The Sensor is straight forward to use and require external high resolution ADC to read measurements.

3.4.2 ADS1115 External 16-Bit ADC

ADS1115 has High Precision 16-Bit ADC and also includes Programmable Gain Amplifier upto 16x, to boost up smaller signal/differential signal to full range.



ADS1115

Technical Specification

1. Supply Range: 2.0V to 5.5V
2. Low Current Consumption: Continuous Mode: Only 150µA Single-Shot Mode: Auto Shut-Down
3. Sampling Rate: 8SPS to 860SPS
4. Internal Low-Drift Voltage Reference
5. Internal Oscillator
6. Internal PGA
7. I2C Interface: Pin-Selectable Addresses
8. Four Single-Ended OR Two Differential Inputs

This board/chip uses I2C 7-bit addresses between 0x48-0x4B, selectable with jumpers.

Understanding ADS1115 Programmable Gain Amplifier

The Gain of the ADS can be programmed which decides the ADS1115 input voltage range for analog to digital conversion and resolution.

The table below relates the Gain with Input Voltage Range and Resolution

Gain	ADC Input Volatge Range	1-Bit Resolution	16-Bit Resolution
2/3x gain	+/- 6.144V	1 bit = 3mV	0.1875mV
1x gain	+/- 4.096V	1 bit = 2mV	0.125mV
2x gain	+/- 2.048V	1 bit = 1mV	0.0625mV
4x gain	+/- 1.024V	1 bit = 0.5mV	0.03125mV
8x gain	+/- 0.512V	1 bit = 0.25mV	0.015625mV
16x gain	+/- 0.256V	1 bit = 0.125mV	0.0078125mV

In this development we will be using **16x PGA** which can **measure the input voltage in the range 0 mV to 256 mV** and gives **resolution of 0.0078125 mV**.

The Output Range (0 mV to 76.38 mV with +/- 3% linearity error) of OOM202 Oxygen Sensor fits perfectly into the Input Voltage Range for ADC at 16x PGA with resolution of 0.0078125 mV.

How this DIY Oxygen Analyzer will Work?

For the time being MCU board will read ADS1115 over I2C channel every 1 seconds to get reading from OOM202 sensor connected to the Analog A0 channel of ADS1115.

These readings are in mV which will be converted to Oxygen percent and displayed to OLED over I2C channel.

Other features will be added as we progress with development.

Breadboard prototype Testing

The Breadboard prototype is exposed to known sample of oxygen and other gases. The readings are observed on Serial Monitor.

Testing with Known 99.7 % Oxygen

For 99.7% Oxygen of Known Sample and considering the baseline sensor voltage 15.4 mV for 20.9% oxygen the readings Screenshot is given below:

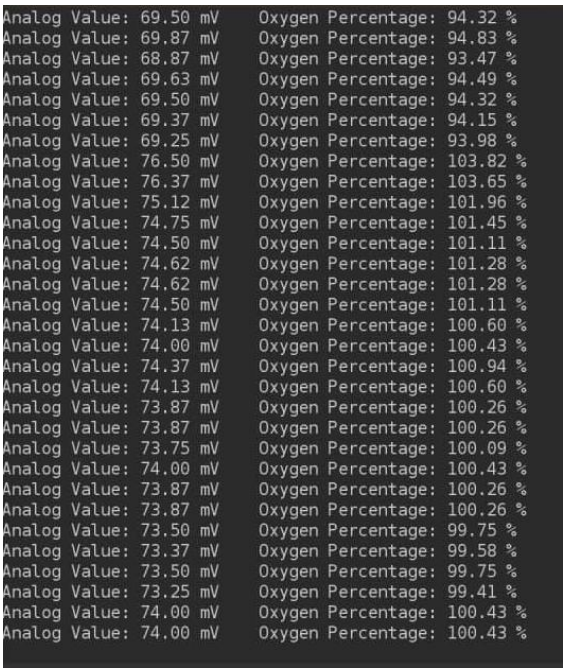
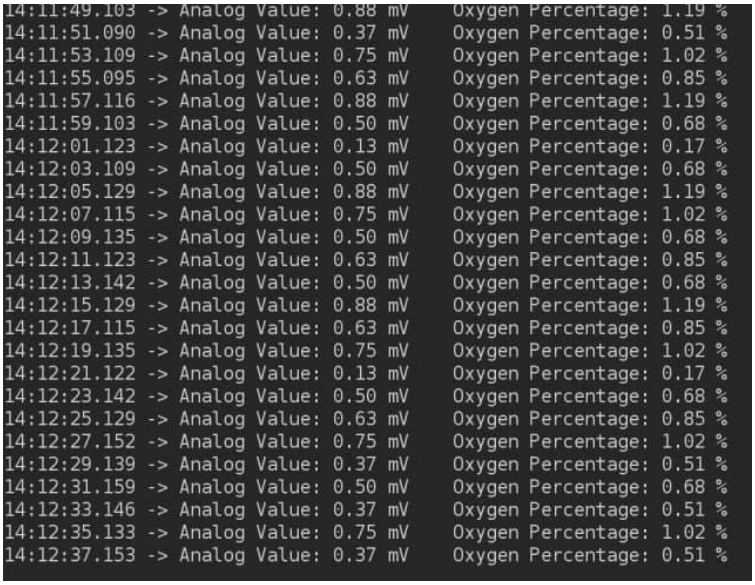


Figure 14: Here the prototype is reading 100 % to 103.82 % for Known Oxygen of 99.7%, we need to write the calibration code and fixing the error as well as saturate the readings for 100 %.

Testing with Helium Gas

For Helium Gas where Oxygen is 0 %



The breadboard prototype here also working fine in the range with negligible error and thus can be corrected with calibration.

Schematic Design

Proof of Performance Our oxygen analyzer must be capable of accurately testing the concentration of oxygen in the air, which is approximately between 20-21%. The Grove oxygen sensor we utilized in our device needed to be initially calibrated. Thus, the oxygen sensor was allowed to equilibrate with atmospheric oxygen in the air, and the Arduino code (see Appendix A) used by the device to sense oxygen was modified so that the LCD readout was 20.5% oxygen concentration. To ensure that this reading was reliable, we tested the oxygen concentration of expired air of 2 different subjects. The oxygen concentration of expired air is known to be around 15-16% [4]. Two subjects flattened a balloon to remove all atmospheric air and breathed into it. The balloon was attached to the oxygen sensor, and the expired air and the sensor were allowed to equilibrate. The oxygen concentration reading before and after exposure to the expired air was recorded. This data for 2 subjects is seen below in Table 1

Table 1: Oxygen concentration data for two subjects breathing expired air

Subject	Trial	Atmospheric O_2 (%)	Expired Air O_2 (%)
Subject 1	Trial 1	20.45	15.22
	Trial 2	20.45	15.07
	Trial 3	20.43	15.32
Subject 2	Trial 1	20.45	17.60
	Trial 2	20.50	17.50
	Trial 3	20.50	17.42

For both subjects, the average atmospheric oxygen concentration detected before exposing the device to expired air was 20.45%, proving that the analyzer could consistently and accurately detect atmospheric oxygen even after several perturbations such as analyzing the oxygen content of expired air. Looking at the expired air trials, Subject 1 had an average expired air oxygen concentration of 15.21% and standard deviation of 0.13%, while Subject 2 had an average of 17.51% and a standard deviation of 0.07%. The differences in these two average values can likely be attributed to differing respiratory quotients for each subject, which depend on factors such as time spent awake and type of food eaten / metabolized. The more revealing data is the similarity of expired air oxygen concentrations over multiple trials detected by the 6 analyzer, as both subjects had very small standard deviation values (0.13% and 0.07%, respectively). Additionally, both average expired air oxygen concentrations values are in or close to the range of 15-16%, which is the expected value for expired air. Thus, we can conclude that our sensor calibration was performed correctly and we can thus accurately detect atmospheric oxygen levels and oxygen levels that are less than 20%. Oxygen readings from the analyzer should not significantly change even when the battery is not at full charge. A voltage divider was built to decrease the amount of voltage supplied from the battery that reached the analyzer, and atmospheric oxygen readings were recorded at different voltage values, seen in Table 2

Table 2: Atmospheric oxygen concentration output at different battery voltages

Voltage Received by Arduino (V)	**Atmospheric O_2 (%)**
9	20.45
7	20.41
5	20.32
4	No readout

As seen by the atmospheric O2 readings at decreasing voltage values, the oxygen readout does not significantly change as long as the voltage supplied is above the minimum voltage needed by the Ardunio to turn on the LCD (5V). As seen by the 4V data point, the LCD display has no readout when the input voltage is less than 5V, which is desirable as this indicates the battery needs recharging without producing an erroneous oxygen concentration value. Given the desire to use the analyzer weekly for 2 years and for a maximum of 3 hours at a time, the components of the device that are subject to failure over time (the Grove oxygen sensor and the 9V lithium ion batteries) must be able to operate over 2 years and a total of 324 hours. The Grove oxygen sensor has a lifetime of at least 2 years, and thus is capable of accurately detecting oxygen over this time frame. Additionally, a fully-charged 9V battery was able to power our analyzer for around 3 hours, thus meeting the requirement of operating the device for a maximum of 3 hours at a time. Additionally, the BONAI 9V lithium-ion batteries utilized in our device are capable of being recharged 1200 times before losing charging capability [5]. Thus, 1200 uses at 3 hours maximum per use yields 3600 hours of oxygen sensing capability, well exceeding the 324 hours we plan for the analyzer to be implemented.

3.4.3 MAX30100

MAX30100 is an integrated pulse oximeter and heart-rate monitor sensor solution. It's an optical sensor that derives its readings from emitting two wavelengths of light from two LEDs – a red and an infrared one – then measuring the absorbance of pulsing blood through a photodetector. This particular LED colour combination is optimized for reading the data through the tip of one's finger. It is fully configurable through software registers and the digital output data is stored in a 16-deep FIFO within the device.It has an I2C digital interface to communicate with a host microcontroller.

The pulse oximetry subsystem in MAX30100 consists of ambient light cancellation (ALC), 16-bit sigma delta ADC, and proprietary discrete time filter. It has an ultra-low-power operation which makes it ideal for battery operated systems. MAX30100 operates on a supply in the range of 1.8 to 3.3V.It can be used in wearable devices, fitness assistant devices, medical monitoring devices, etc.The MAX30100 operates from 1.8V and 3.3V power supplies and can be powered down through software with negligible standby current, permitting the power supply to remain connected at all times.

Pin Configuration of MAX30100 Pulse Oximeter Heart Rate Sensor Module:-

SN	PINS	DEFINITION OF PINS
1	VIN	Input voltage (1.8V to 5.5V)
2	SCL	IIC-SCL
3	SDA	IIC-SDA
4	INT	MAX30100INT
5	IRD	MAX30100 IR_DRV
6	RD	MAX30100 R_DRV
7	GND	Ground

Specifications and Features of MAX30100 Pulse Oximeter Heart Rate Sensor Module:-

- It is an integrated pulse oximetry and heart rate monitor sensor solution.
- Integrated LEDs, Photo Sensor, and High-Performance Analog Front -End
- Complete Pulse Oximeter and Heart-Rate Sensor Solution Simplifies Design
- Measures absorbance of pulsing blood
- I2C interface plus INT pin
- Tiny 5.6mm x 2.8mm x 1.2mm 14-Pin Optically Enhanced System-in-Package
- Ultra-Low-Power Operation Increases Battery Life for Wearable Devices
- Programmable Sample Rate and LED Current for Power Savings
- Ultra-Low Shutdown Current (0.7µA, typ)
- Advanced Functionality Improves Measurement Performance
- High SNR Provides Robust Motion Artifact Resilience
- Integrated Ambient Light Cancellation
- High Sample Rate Capability
- Fast Data Output Capability

MAX30100:

Heart Rate click carries Maxim's **MAX30100** integrated pulse oximetry and a heart-rate sensor. It's an **optical sensor that derives its readings from emitting two wavelengths of light from two LED**s – a red and an infrared one – then **measuring the absorbance of pulsing blood through a photodetector**. This particular LED color combination is optimized for reading the data through the tip of one's finger.
The signal is processed by a **low-noise analog signal processing unit and communicated to the target MCU through the mikroBUS I2C interface**. Developers of end-user applications should note that the readings can be negatively impacted by excess motion and changes in temperature. Also, too much pressure can constrict capillary blood flow and therefore diminish the reliability of the data. A programmable **INT pin** is also available. The **operates at the 3.3V power supply**.

Pin Configuration:

SN	PINS	DEFINITION OF PINS
1	VIN	Input voltage (1.8V to 5.5V)
2	SCL	IIC-SCL
3	SDA	IIC-SDA
4	INT	MAX30100INT
5	IRD	MAX30100 IR_DRV
6	RD	MAX30100 R_DRV
7	GND	Ground

Specifications and Features:

1. Optical sensor: IR and red LED combined with a photodetector
2. Measures absorbance of pulsing blood
3. I2C interface plus INT pin
4. 3.3V power supply complete pulse oximeter and heart rate sensor solution, simplifies design, integrated LEDs, photo sensor, and
5. high-performance analog front
6. Ultra low power operation increases battery life for wearable devices
7. Advanced functionality improves measurement performance, high SNR provides robust motion artifact resilience integrated
8. ambient, light cancellation high sample rate capability fast data output capability
9. It is an integrated pulse oximetry and heart rate monitor sensor solution.

3.4.4 GPS:

GY-NEO6MV2 board is a cost-efficient and flexible gadget which features the u-box NEO-6M GPS module. NEO-6 GPS boasts an excellent navigation performance, making it a great choice for personal navigation projects and location services.

The module also features an antenna for strong signal and built-in EEPROM for saving the configuration when powered off. It has a 25mm x 35mm PCB and a separate 25mm x 25mm ceramic antenna connected by a small cable.

The board has four connectors: VCC, GND, TX (transmit) and RX (receive) and can be powered by a 3V-5V power supply.

Also, on the board is a small button-cell battery for GPS back up and a LED signal indicator.
The module is very easy to hook up to a microcontroller using RX and TX and is compatible with Raspberry Pi, Arduino or with various flight controller designed to work with a GPS module.

Features:

- Ceramic antenna for strong signal
- EEPROM for saving the configuration data when powered off
- Backup battery
- LED signal indicator
- Applicable for Raspberry Pi and Arduino
- Compatible with various flight-control modules

Specifications:

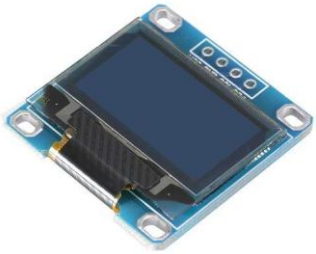
- Model: GY-GPS6MV2
- Antenna Size: 25mm x 25mm
- Module Size: 25mm x 35mm
- Cable: 20mm
- Mounting Hole Diameter: 3 mm
- Default Baud Rate: 9600
- Power Supply Range: 3 V to 5 V
-

Package:

- 1 x GY-NEO6MV2 Flight Controller GPS Module with Antenna

3.4.5 Oled:

- **OLED is Organic Light Emitting Diode** that emits light in response to an electric current. OLED display works with no backlight so it can display deep black levels. It is small in size and light in weight than Liquid Crystal Displays.
- 128x64 OLED display is simple dot matrix graphic display. It has 128 columns and 64 rows which make it display of total **128x64 = 8192 pixels**. By just turning on/off these pixel's led we can display a graphical image of any shape on it.



OLED displays driven by SSD1306 driver IC. SSD1306 is a CMOS OLED driver with controller for OLED dot-matrix graphic display system. Due to use of SSD1306 driver, number of external components required and power consumption has reduced.

- OLED display is used for displaying text, images and various patterns. It is also suitable for mobile phone sub-display, MP3 player, calculators etc.
- OLED display has 256 steps for brightness control.
- OLED display also available with different resolution like 128x32, 128x64. OLED display in above image has resolution of 128x64 pixels.

Available Interfaces for OLED

OLED display module can be interfaced with microcontrollers using three interfaces given below:

6800/8000 series compatible Parallel Interface

In this interface 8-bit data send/receive could be done through parallel lines i.e. D0-D7.

I2C interface

In this interface, data send/receive could be done serially through SDA line.

Serial Peripheral Interface

In this interface, data send/receive could be done serially through SDI and SDO lines.

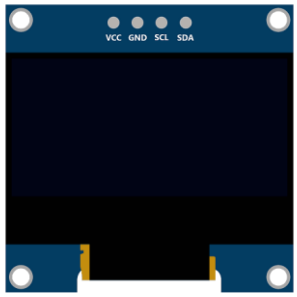
Note: The module illustrated here has I2C interface pins so all the discussion below is considering I2C as interfacing standard.
There are different types of OLED modules are available in market, having different resolutions, communication protocol (as discussed in above **Available Interfaces for OLED** section) and pixel colours (e.g. blue, yellow, white). Some modules support multi-colours as well.

Specification of ssd1306 128x64 OLED

- Display Type: OLED (Organic Light Emitting Diode)
- Display Size: 128x64 pixels
- Display Driver: ssd1306
- Display Colors: Monochrome (White), Yellow, and Blue
- Operating Voltage: 3.3V to 5V
- Interface: I2C

- Operating Current: ~20mA

OLED Display Pins (I2C interface)



OLED Display Pins

SDA (Serial Data):

SDA is used to transmit data between master and slave. The data and acknowledgment are sent through SDA.

SCL (Serial Clock):

It is a clock signal. This pin transmits clocks to slave, SCL. Data will be sent to other devices on clock tick event. Only master device has control over this SCL line

VCC:

This is power supply pin. +3.3V supply is required. More than 3.3 V supply can damage the display.

GND:

This is Ground pin. Connect ground of supply to this pin.

I2C Address of OLED display

In I2C interface devices are recognized by their slave address. OLED display has slave address format as shown in below image.

7	6	5	4	3	2	1	0
0	1	1	1	1	0	SA0 bit	R/W bit

SA0 (Slave Address) bit:

- This bit provides two slave address option to choose from.

R/W (Read/Write) bit:

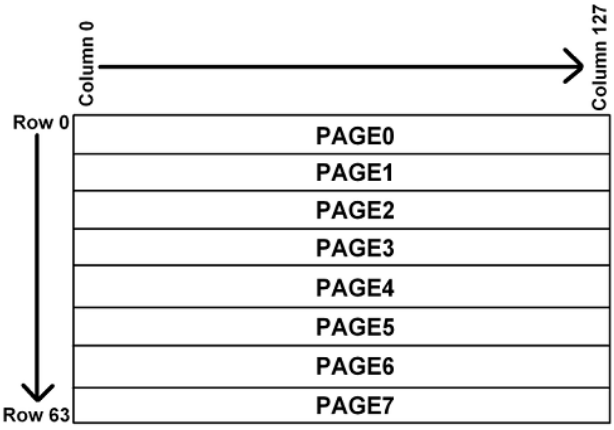
- This bit is used to determine mode of operation i.e. I2C write or I2C read operation. 1 for read operation and 0 for write operation.

Now if **SA0** bit is **0**, then microcontroller can do read/write operation with OLED display using below I2C address.

- I2C write address is 0x78
- I2C read address is 0x79

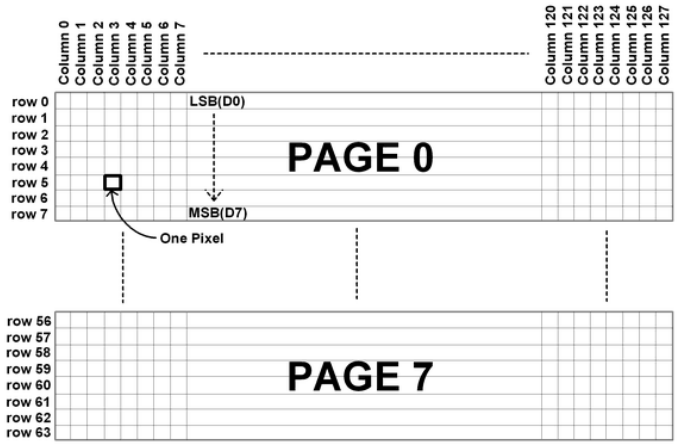
Display Structure

OLED Display is mapped with GDDRAM of SSD1306.



GDDRAM pages structure of SSD1306

The GDDRAM (Graphics Display Data RAM) is a bit mapped static RAM. It holds the bit pattern to be displayed.



Pixelwise structure of OLED Display

- The size of GDDRAM is 128 x 64 bits and it is divided into eight pages, from PAGE0 to PAGE7, which are used for monochrome 128x64 dot matrix display as shown in above figure.
- When a data byte (D0-D7) is written into GDDRAM, all the rows data pixels of the same page of the current column gets filled. Data bit D0 is written on the top row, while data bit D7 is written into bottom row as shown in Figure

Row and Column Mapping on OLED

- Display has total 8 pages, 64 rows and 128 Columns.
- Each page contains of 8 rows and 128 columns as shown in above figure.
- Display has total 128 columns called as segments.
- For displaying data at first location set page address to 0 and column address to 0. We should also select the end address of page and column.
- Maximum end address of page is 07H and Maximum column address is 7F H.

Commands for SSD1306

Note: Commands and initialization sequence given here is generalized and it worked for us.

Set Display Clock Divide Ratio/ Oscillator Frequency (0xD5 h):

- Bit 3 to 0: Display Clock Divide Ratio (D) Set the divide ratio to generate DCLK (Display Clock) from CLK. The divide ratio is from 1 to 16, with reset value = 1.
- Bit 7 to 4: Set oscillator frequency. The 4-bit value results in 16 different frequency settings. The default setting is 1000b.

Set Multiplex Ratio (0xA8 h):

- This command switches the default 63 multiplex mode to any multiplex ratio, ranging from 16 to 63.

Set Display Start Line (0x40 H~0x7F H):

- This command sets the Display Start Line register to determine starting address of display RAM, by selecting a value from 0 to 63. In our program we set this to zero and map RAM row 0 to COM 0.

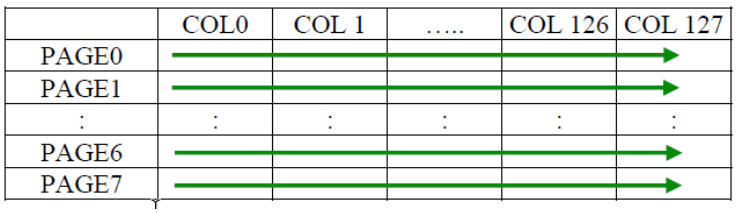
Set Memory Addressing Mode (0x20 H):

There are three different addressing modes in SSD1306:

1. Page addressing mode
2. Horizontal addressing mode
3. Vertical addressing mode

1. Page Addressing Mode:

- In page addressing mode, after the display RAM is read/written, the column address pointer is increased automatically by 1.
- If the column address pointer reaches column end address, the column address pointer is reset to column start address but page address pointer not points to next page.
- Hence, we need to set the new page and column addresses in order to access the next page RAM content.
- We need to set lower two bits to '1' and '0' for **Page Addressing Mode.**

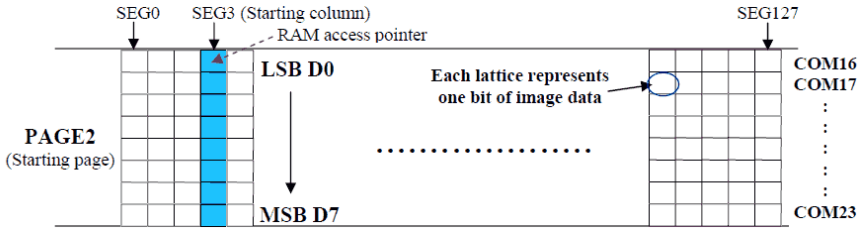


Address Pointer Movement of Page addressing mode

In page addressing mode, the following steps are required to define the starting RAM access pointer location:

- Set the page start address of the target display location by command B0h to B7h.
- Set the lower start column address of pointer by command 00h~0Fh.
- Set the upper start column address of pointer by command 10h~1Fh.

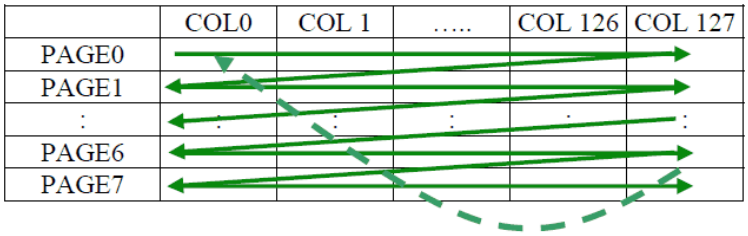
For example, if the page address is set to B2h, lower column address is 03h and upper column address is 10h, then that means the starting column is SEG3 of PAGE2. The RAM access pointer is located as shown in Figure. The input data byte will be written into RAM position of column 3.



Example of GDDRAM access pointer setting in Page Addressing Mode

2. Horizontal Addressing Mode:

- In horizontal addressing mode, after the display RAM is read/written, the column address pointer is increased automatically by 1.
- If the column address pointer reaches column end address, the column address pointer is reset to column start address and page address pointer is increased by 1.
- When both column and page address pointers reach the end address, the pointers are reset to column start address and page start address
- We need to set last two digits to '0' and '0' for **horizontal addressing mode**.

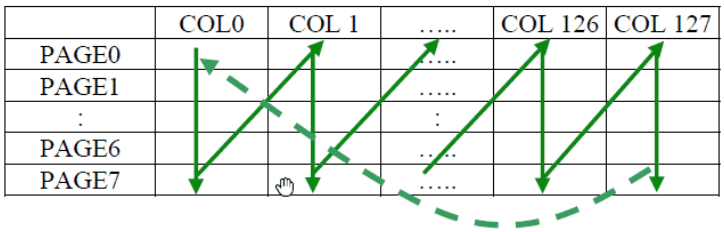


Address Pointer Movement of Horizontal addressing mode

Vertical Addressing Mode:

- In vertical addressing mode, after the display RAM is read/written, the page address pointer is increased automatically by 1.
- If the page address pointer reaches the page end address, the page address pointer is reset to page start address and column address pointer is increased by 1.
- When both column and page address pointers reach the end address, the pointers are reset to column start address and page start address.
- We need to set last two digits to '0' and '1' for **vertical addressing mode**.

Figure 10-4 : Address Pointer Movement of Vertical addressing mode



Address Pointer Movement of Vertical addressing mode

In normal display data RAM read or write and **horizontal/vertical** addressing mode, the following steps are required to define the RAM access pointer location:

- Set the column start and end address of the target display location by command 21h.
- Set the page start and end address of the target display location by command 22h.

Set Column Address (0x21 H):

- This is a triple byte command. First byte specifies the command for setting column address (0x21 H).
- Second byte specifies the column start address and third byte specifies column end address.
- This command also sets the column address pointer to column start address.

Set Page Address (0x22 H):

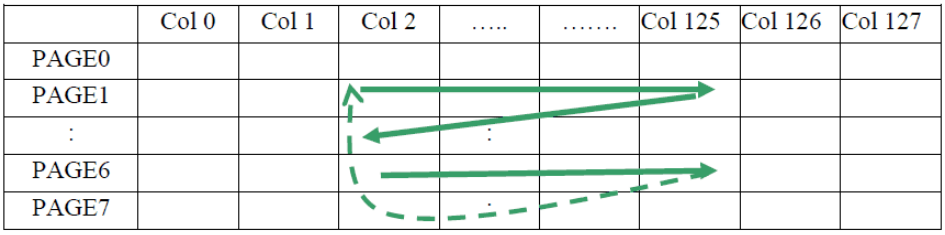
- This is a triple byte command. First byte specifies the command for setting page address (0x22 H).
- Second byte specifies page start address and third byte is page end address.
- This command also sets the page address pointer to page start address.

Example of column and row address pointer movement:

In the following example, Horizontal addressing mode is used. Column start address is set to 2 and column end address is set to 125.

- Page start address is set to 1 and end address is set to 6. In this case, the graphic display data RAM column accessible range is from column 2 to column 125 and from page 1 to page 6 only.
- In addition, the column address pointer is set to 2 and page address pointer is set to 1. After finishing read/write one pixel of data, the column address is increased automatically by 1 to access the next RAM location for next read/write operation.

- Whenever the column address pointer finishes accessing the end column 125, it gets reset back to the column 2 and page address is automatically increased by 1.
- While the end page 6 and end column 125 RAM location is accessed, the page address gets reset back to 1 and the column address is reset back to 2.



Example of Column and Row Address Pointer Movement

Set Contrast Control (0x81 H):

- This command sets the Contrast Setting of the display. The chip has 256 contrast steps from 00h to FF H. The segment output current increases as the contrast step value increases.

Set Pre-charge period (0xD9 H):

- This command is used to set the duration of the pre-charge period. The interval is counted in number of DCLK, where RESET equals 2 DCLKs.

Set VCOMH Deselect Level (0xDB H):

- This command adjusts the VCOMH regulator output.

Entire display on (0xA4 H/0xA5 H):

- A4 h command resumes the display from entire display "ON" stage.
- A5 h command forces the entire display to be "ON", regardless of the contents of the display data RAM.

Set Normal/Inverse Display (0xA6h/0xA7h):

- A6 h command is for normal display.
- A7 h command is for inverse display.

Set Display ON/OFF (0xAE H/0xAF H):

- AE h: Set display OFF.
- AF h: Set display ON

Note: Clear the OLED display as it can print garbage value if left uncleared.
Warning:

1. The data bit, which is transmitted during each SCL pulse, must keep at a stable state within the "HIGH" period of the clock pulse.
2. Except in start or stop conditions, the data line can be switched only when the SCL is LOW.
3. Both the data line (SDA) and the clock line (SCL) should be pulled up by external resistors.

Alternate options for 128x64 OLED

- OLED 128x32
- GLCD 128x64
- LCD16x2
- Nokia 5110 display

3.4.6 GSM:

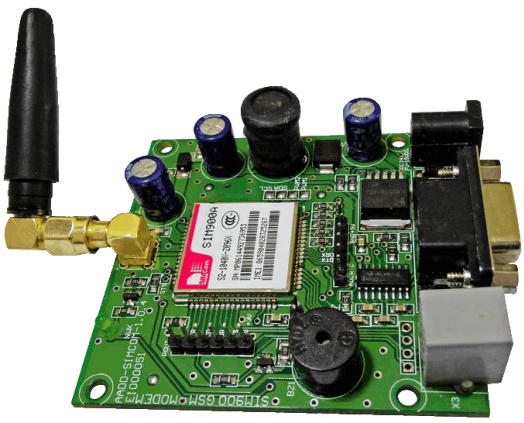
SIM900A Modem can work with any GSM network operator SIM card just like a mobile phone with its own unique phone number.

SIM900A GSM/GPRS modem is plug and play modem with RS232 serial communication supported. Hence Advantage of using this modem will be that its RS232 port can be used to communicate and develop embedded applications.

Applications like SMS Control, data transfer, remote control and logging can be developed. SIM900 modem supports features like voice call, SMS, Data/Fax, GPRS etc.

SIM900A modem uses AT commands to work with supported features.

Note that to be connected to a cellular network, the modem requires a SIM card provided by a network provider.



Power Requirement

This board requires external power supply of ~12V and can draw up to ~2A of current at its peak.

Indicators

It has two LED indicators as,

ON: It shows that the Modem is getting powered and is switched on.

NET: This network LED blinks when the modem is communicating with the radio network.

Network LED

When modem is powered up, network LED blink every second and after network registration it will start to blink after every 3 seconds. This shows that the modem is registered with the network.

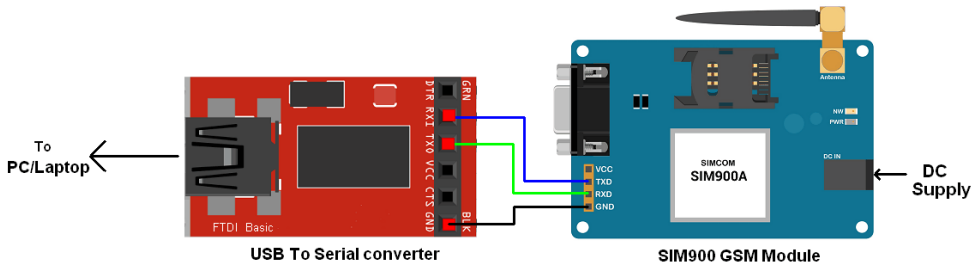
To test modem, connect board serially to PC and send "ATE0" or "AT" through serial terminal. If "OK" response is received from the modem, then it means all is well.

AT Command Reference

There are many AT commands for SIM900A modem, kindly go through 'SIM900 AT Command Manual'.

Interfacing GSM with PC/Laptop

- Before interfacing SIM900A GSM module with PIC18F4550 microcontroller, we can check GSM module output for various AT commands on PC/Laptop.
- To do this, connect GSM module to the PC/laptop via USB to Serial converter as shown below.



Sim900A Serial Interface

- Insert SIM Card

- Connect power supply to the SIM900A GSM module.
- Now, there is one Network LED on GSM module which blinks continuously every second on power up. When the SIMregisters in network then this LED blinks after every 3 sec. It takes 10-60 second to register in network.
- Now send AT commands and check the response.
- Every time we have to write command and send it by Enter key<CR>. You will always see the response in form of

<CR><LF><Response><CR><LF>

In following table, only responses are present, <CR><LF> are removed intentionally.

Basic AT Command

There some basic AT commands which are used as follows,

Command	Description	Response
AT	Checking communication	OK
ATE0 or ATE1	Turn off /ON echo	OK
ATI	Product Identification	e.g.: SIM900 R11.0 OK
AT+GSN	Product serial number identification (IMEI)	IMEI no.
AT+GMI	Manufacturer Name	Manufacturer Name E.g. SIMCOM_LTD
AT+GMM	Model Identification	Model no. E.g. SIMCOM_900A
AT+CSMINS?	SIM Inserted Status Reporting	+CSMINS: <n>, <SIM inserted> 0: not inserted 1: inserted
AT+CSPN?	Get Service Provider Name from SIM	Service Provider Name E.g. +CSPN: “!dea”, 1

Calling

To use calling service following AT commands are used.

Command	Description	Response
ATD<Mob. No.>;	Dial / call a number	OK (if successful), BUSY (if busy), NO CARRIER (if no connection)
ATA	Answer a call	OK
ATH	Hang up call	OK
ATDL	Redial Last Telephone Number Used	OK (if successful), BUSY (if busy),NO CARRIER (if no connection)

Message

To use SMS service following AT commands are used.

Command	Description	Response
AT+CMGF=<index> index- 0: PDU 1: Text	Select message format	OK
AT+CMGS="9881xxxxxx"	Send message	> "Type message here" press 'ctrl+z' to end msg or 'ESC' to exit without sending OK
AT+CMGR=<index>	Read message at that index	+CMGR: "Message Header" Message Body OK
AT+CMGD=<index>	Delete message at that index	OK (if present at that index)
AT+CMGDA="DEL ALL"	Delete all SMS	OK

HTTP (Hypertext Transfer Protocol)

To use HTTP function using SIM900 modem follow below AT commands. Two methods are there namely,

GET: to get data from server.

POST: to post data to server.

First we need to connect to GPRS by configure bearer profile using AT command as,

Demonstration	**Syntax**	**ExpectedResponse**
Configure bearer profile 1	AT+SAPBR=3,1,"Contype","GPRS" AT+SAPBR=3,1,"APN","internet"	OK OK
To open a GPRS context.	AT+SAPBR =1,1	OK
	AT+SAPBR=2,1	+SAPBR: 1,1,"10.89.193.1" OK
To close a GPRS context.	AT+SAPBR =0,1	OK

Note: Enter Access Point Name (APN) of corresponding network service provider. E.g. here we have entered “internet” which is APN of IDEA network.

Now let’s see AT commands for GET method,

HTTP GET method

Demonstration	Syntax	ExpectedResponse
Initiate http service	AT+HTTPINIT	OK
Set parameters for HTTP session	AT+HTTPPARA= “CID”,1 AT+HTTPPARA= “URL”,”api.thingspeak.com”	OK OK
GET session start	AT+HTTPACTION=0	OK
GET successfully	AT+SAPBR =0,1	+HTTPACTION:0,200,1000
Read the data of HTTP server	AT+HTTPREAD	+HTTPREAD: 1000 Read data…. OK
Terminate http service	AT+HTTPTERM	OK

Note: Enter http client URL. E.g. here we have entered “api.thingspeak.com”.

Now let's see AT commands for POST method,

HTTP POST method

Demonstration	Syntax	Expected Response
Initiate http service	AT+HTTPINIT	OK
Set parameters for HTTP session	AT+HTTPPARA= "CID",1 AT+HTTPPARA= "URL","www.google.com"	OK OK
POST the data whose size is 100Bytes and the maximum latency time for inputting is 10000ms. It is recommended to set the latency time long enough to download all the Data in the latency time.	AT+HTTPDATA=100,10000	DOWNLOAD Enter data of exact size mentioned in syntax OK All data has been received
POST session start	AT+HTTPACTION=1	OK
POST successfully		+HTTPACTION:1,200,0
Terminate http service	AT+HTTPTERM	OK

TCP (Transmission Control Protocol)

TCP is communication protocol of internet. There are two modes of connection used for SIM900A TCP/IP application,

- Single Connection - Can establish only one connection.
- Multi Connection - Can establish 8 connections.

There are two TCP/IP application modes,

- Transparent Mode
- Non-Transparent (Normal) Mode

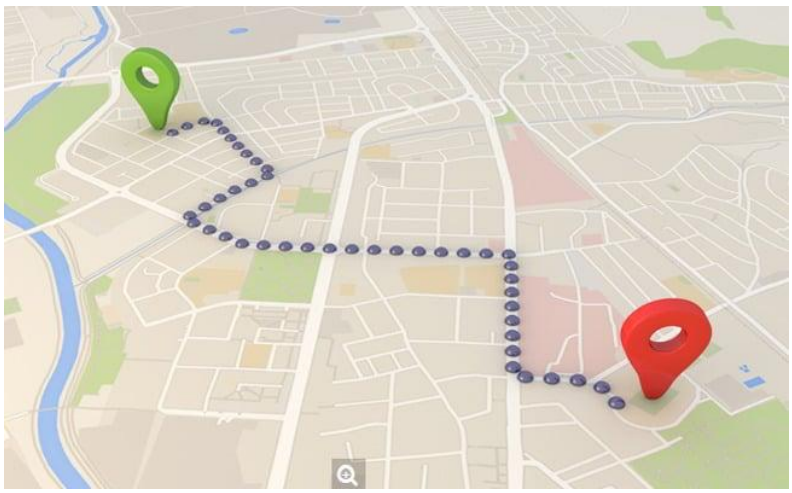
Note, when in multi connection mode SIM900A can work only in Non-transparent mode.

Let's see TCP Client connection AT commands

Demonstration	Syntax	Expected Response
To select connection mode	AT+CIPMUX=0	OK
Sets apn, username and password	For single connection AT+CSTT=”APN",”USERNAME”,”PASSWORD”	OK
Bring up wireless connection	AT+CIICR	OK
Get local IP	AT+CIFSR	Local IP e.g.“10.81.14.12”
Start-up TCP connection	for single connection, AT+CIPSTART=”TCP”,”api.thingspeak.com”,”80”	CONNECT OK
Send data	AT+CIPSEND	> …Enter data and substitute CTRL+Z (0x1A) …. SEND OK
Close the connection	For quick close, AT+CIPCLOSE=1	CLOSE OK

3.4.7 GPS:

Introduction



GPS receivers are generally used in smartphones, fleet management system, military etc. for tracking or finding location.

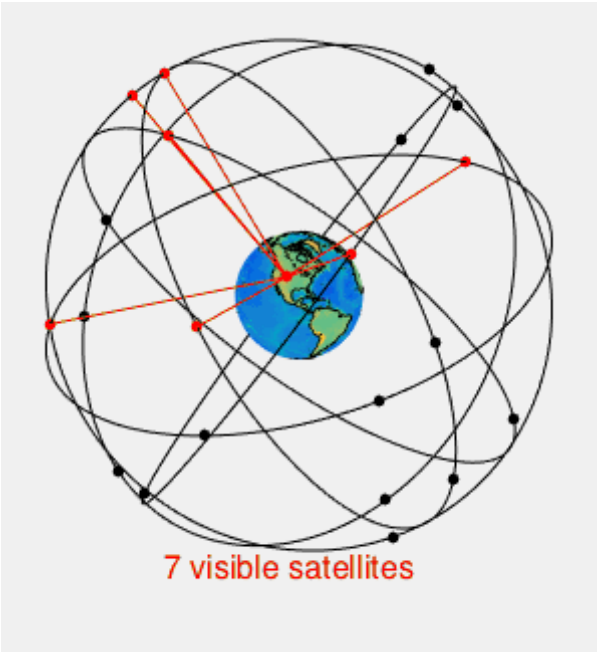
Global **P**ositioning **S**ystem (GPS) is a satellite-based system that uses satellites and ground stations to measure and compute its position on Earth.

GPS is also known as Navigation System with Time and Ranging (NAVSTAR) GPS.

GPS receiver needs to receive data from at least 4 satellites for accuracy purpose. GPS receiver does not transmit any information to the satellites.

This GPS receiver is used in many applications like smartphones, Cabs, Fleet management etc.

How GPS Works

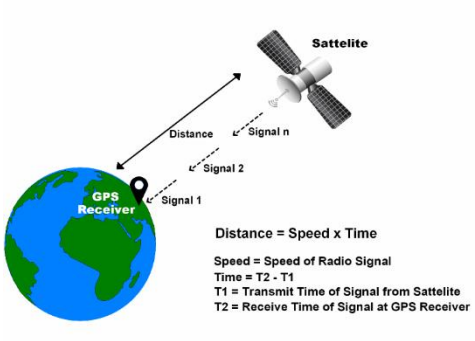


GPS receiver uses a constellation of satellites and ground stations to calculate accurate location wherever it is located.

These GPS satellites transmit information signal over radio frequency (1.1 to 1.5 GHz) to the receiver. With the help of this received information, a ground station or GPS module can compute its position and time.

How GPS Receiver Calculates its Position and Time

GPS receiver receives information signals from GPS satellites and calculates its distance from satellites. This is done by measuring the time required for the signal to travel from satellite to the receiver.



GPS Distance Calculation

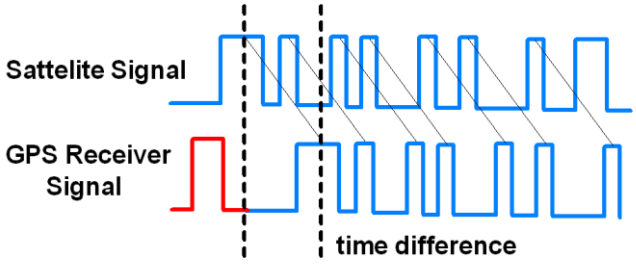
$$\text{Distance} = \text{Speed x Time}$$

Where,

Speed = Speed of Radio signal which is approximately equal to the speed of light i.e.$3 * 10^8$

Time = Time required for a signal to travel from the satellite to the receiver.

By subtracting the sent time from the received time, we can determine the travel time.



GPS Signal Time Difference

To determine distance, both the satellite and GPS receiver generate the same pseudocode signal at the same time.

The satellite transmits the pseudocode; which is received by the GPS receiver.

These two signals are compared and the difference between the signals is the travel time.

Now, if the receiver knows the distance from 3 or more satellites and their location (which is sent by the satellites), then it can calculate its location by using Trilateration method.

GPS Module

GPS Receiver

GPS receiver module gives output in standard (National Marine Electronics Association) NMEA string format. It provides output serially on Tx pin with default 9600 Baud rate.

This NMEA string output from GPS receiver contains different parameters separated by commas like longitude, latitude, altitude, time etc. Each string starts with '$' and ends with carriage return/line feed sequence.

E.g.

$GPGGA,184237.000,1829.9639,N,07347.6174,E,1,05,2.1,607.1,M,-64.7,M,,0000*7D

$GPGSA,A,3,15,25,18,26,12,,,,,,,,5.3,2.1,4.8*36

$GPGSV,3,1,11,15,47,133,46,25,44,226,45,18,37,238,45,26,34,087,40*72

$GPGSV,3,2,11,12,27,184,45,24,02,164,26,29,58,349,,05,26,034,*7F

$GPGSV,3,3,11,21,25,303,,02,11,071,,22,01,228,*40

$GPRMC,184237.000,A,1829.9639,N,07347.6174,E,0.05,180.19,230514,,,A*64

Pin Description

GPS Receiver Module

VCC: Power Supply 3.3 – 6 V

GND: Ground

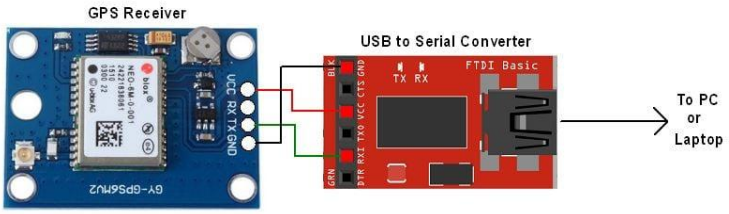
TX: Transmit data serially which gives information about location, time etc.

RX: Receive Data serially. It is required when we want to configure GPS module.

Check GPS module

Before Interfacing the GPS module with the PIC18F4550 microcontroller, we can check the output of GPS module. From that string, we can extract information like longitude, latitude, and time which is helpful to find location and timing information.

To do this, connect this GPS module to the PC via USB to Serial converter or DB9 connector. Also, it is necessary to keep the antenna of GPS module on proper location.



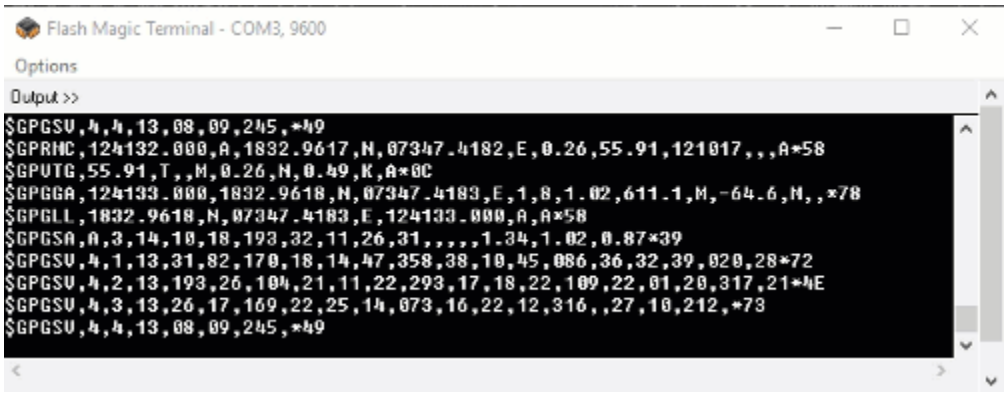
GPS Serial Interface

Now open any serial terminal e.g. Realterm, Hyper terminal, Putty, etc. on a PC/laptop.

Open the PORT with a 9600 baud rate.

The terminal will show data coming from the GPS receiver module.

The output data from the GPS receiver module displaying on a serial terminal as follows.



In the above string, the NMEA string starting with “$GPGGA” is most popularly used. It provides us Time, Longitude, Latitude and Altitude along with directions. This information is helpful to find Time and Location.

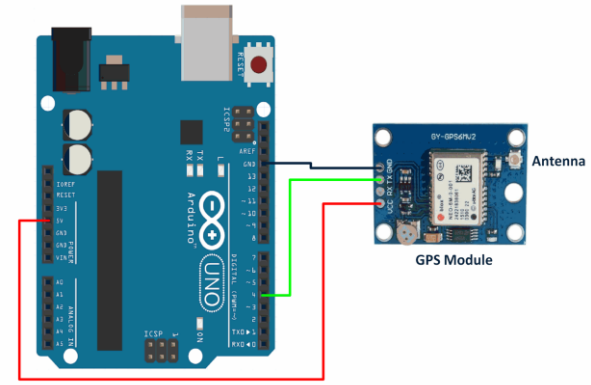
E.g.

$GPGGA,184241.000,1829.9639,N,07347.6174,E,1,05,2.1,607.1,M,-64.7,M,,0000*7C

Name	Example	Units	Description
Message ID	$GPGGA		GGA Protocol Header
UTC Time	184241.000		hhmmss.sss
Latitude	1829.9639		ddmm.mmmm
N/S Indicator	N		N=North, S=South
Longitude	07347.6174		dddmm.mmmm
E/W Indicator	E		E=East, W=West
Position Fix Indicator	1		Fix GPS SPS mode
Satellites Used	05		Range 0 to 12
HDOP	2.1		Horizontal Dilution of Precision
MSL Altitude	607.1	Meters	Mean Sea Level
Units	M	Meters	
Geoid Separation	64.7	Meters	
Units	M	Meters	

Age of Diff. Corr.	-		Null field if DGPS is not used
Diff. Ref Station ID	0000		
Checksum	*7C		
Carriage return Line Feed	<CR><LF>		End of message transmission

Connection Diagram of GPS Module with Arduino Uno



Connection Diagram of GPS Module with Arduino Uno

3.5 Prototype of proposed Oxygen Analyzer

This Arduino project integrates the OOM202 Oxygen Sensor, MAX30100 Pulse Oximeter Sensor, GSM Module, GPS NEO-6M Module, OLED Display, and a push button to create an oxygen purity detection and alert system. It monitors oxygen purity, sends an SMS alert if purity is below 40%, includes GPS coordinates in the alert, and displays SpO2 and pulse rate data on demand. The project combines sensor interfacing, communication, user interaction, and display functionalities to create a comprehensive health monitoring solution.

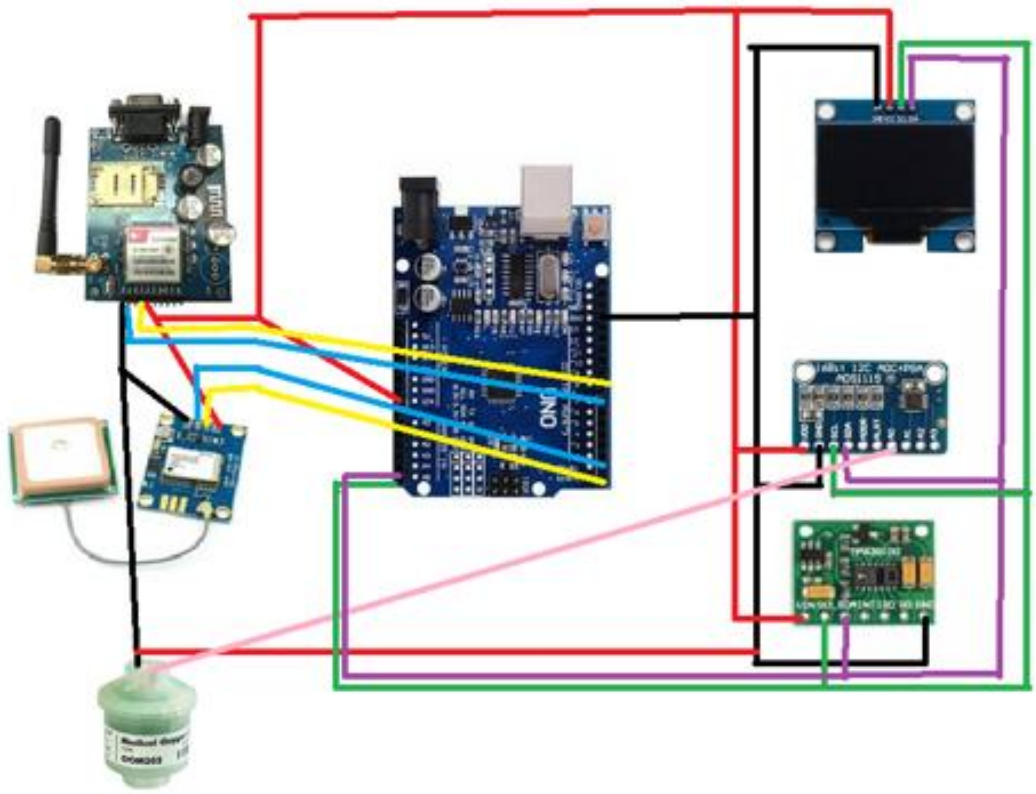


Figure 16: Proposed model of oxygen analyzer with fault detection

Pressure Swing Absorption Oxygen Concentratorequipped with Remote Monitoring Pulse Oximeter

(Treesukon Salila, Ongtrakul, Anyarin, Thitiratannapong,Chuchart Pintavirooj)

***Abstract*— The new COVID-19 disease was first identified in China at the end of 2019 and has spread rapidly all over the world. It has been projected that, by March 2021, the number of infections could reach 300 million cases and over two million deaths. One of the main implications for COVID-19 patients is pneumonia where the lung is infected, hence patients suffering from insufficient oxygen in the blood. As the number of COVID-**

19 cases have significantly increased, the demand for oxygen generators have also skyrocketed. This research concerns the design and construction of emergency low-cost oxygen concentrators used for mild COVID-19 symptoms, of which are forced to be treated at home. Our absorption-based oxygen concentrator uses zeolite packed in a sieve canister. An Oil-free compressor is then used to pump air in. Zeolite will absorb nitrogen from the air leaving oxygen free to travel towards the outlet. To evaluate the treatment, we have equipped the system with a pulse oximeter to measure the percent saturation oxygen, pulse rate and temperature. To prevent COVID-19 infections between patients and caretakers, we have designed an android application to remotely control the oxygen concentrator. Experiment has shown that our emergency low-cost oxygen concentrator can supply oxygen with an 85% purity rate.

***Index Terms*— Oxygen Concentrator, Percent Saturation Oxygen, Zeolite**

INTRODUCTION *(HEADING 1)*

In current society, we all experience some type of health problem. This could be due to numerous factors, but one of the common symptoms that a patient can undergo is insufficient levels of oxygen. This is caused by lack of oxygen intake leading to oxygen starvation. During the past two years, a global pandemic had commenced. The new COVID-19 disease is a severe acute respiratory syndrome that reduces the efficiency of the patient's respiratory system as well as other bodily systems. Patients diagnosed with this disease occasionally has oxygen levels below normal saturation resulting in the need of oxygen concentrators to replace the oxygen they failed to utilize from the environment.

Maintaining healthy levels of oxygen saturation in the patient's blood is crucial, especially in the current COVID-19 pandemic. Numerous approaches have been invented in the history of medical appliance development and the most commonly used types are compressed oxygen tanks, liquid oxygen, and oxygen concentrators. Each type of supplement, despite serving similar purposes, possess distinct advantages and disadvantages. [1,2] Absorption based portable oxygen concentrators have drawn attention from researchers during the past decade. [3,4,5,6,7,8,9,10] The concept is to pump air through the adsorbents adsorbing nitrogen and allow oxygen to pass through. The type of adsorbent widely used is zeolite. To increase the efficiency, two canisters of zeolite have been used. The two canisters will operate alternatively in one of the two phases, (i) adsorption and (ii) de-adsorption phase. In the adsorption phase, compressed air is pumped into the first canister adsorbing nitrogen. While

the de-absorption phase, occurring in the second canister, will be partially purged of nitrogen. Releasing it to room atmosphere by parts of the high-pressured oxygen from canister 1. The process is also known as pressure swing absorption (PSA). There are 3 different variants of adsorption-based air separation processes that most research covers, which are traditional pressure swing adsorption (PSA), vacuum-pressure swing adsorption (VPSA), and rapid cycling pressure swing adsorption (RPSA). In rapid cycling pressure swing adsorption, also called single-bed PSA, only one canister is used. The cycle time of single-bed adsorptions is usually shorter than normal PSA. [11] However, rapid cycling pressure swing adsorption suffers from a large pressure drop. [3,12] For vacuum-pressure swing adsorption (VPSA), rather than using air compressors to pump air in, VPSA uses the vacuum generator at the output end to suck the air through the canister. VPSA is proven to perform superior to the PSA as it can operate under low operating pressures. [8] This paper concentrates on pressure swing absorption (PSA) oxygen concentrator. The salient aspects and/or contributions of this paper are enumerated as follows.

1) This paper presents the development of a low-cost, remotely controlled, remotely monitored and easy-to-implement oxygen concentrator which is very suitable in current situations of the COVID-19 pandemic.

2) What differentiates our research to previous work is that our PSA oxygen concentrator is capable of monitoring four-parameter vital signals simultaneously including Plethysmogram, percent saturation oxygen (SpO2), temperature and heart rate.

ZEOLITE

Zeolite, a microporous crystalline solid used in PSA, is a commonly used adsorbent for separation and purification of fluids. Zeolites have a chemical composition of $M_{2/n}O.Al_2O_3.xSiO_2.yH_2O$ with an elementary structure of an aluminosilicate.[13] It is a tetrahedral arrangement of Si and Al cations surrounded by 4 oxygen molecules each and sharing one oxygen molecule between them. This allows the formation of cavities, thus the term molecular sieve. Other than the basic units of zeolite, it also contains counterions such as alkaline and alkaline earth metals. The presence of these metals is to balance out the negative sites caused by the difference in formal valency. They are mostly found on the surface of the material bounded through weak electrostatic bonds. Due to the diverse ways of arrangement between the molecules, more than 40 kinds of natural zeolites are found. As zeolites can form different structures, developing them in a controlled environment could result in an intended use of the pore size. Frequently used types of synthetic zeolites are type A, X and Y, used in commercial purposes for adsorption and ion exchange. The selectivity of a zeolite is its ability to adsorb one type of atom preferably over another. The selectivity can affect the overall separation by three factors: equilibrium separation, kinetic separation, and molecular sieving. Equilibrium separation occurs when a difference in molecular interactions of the zeolite and adsorbate is present. Kinetic separation is the difference in transport rate or how fast the adsorbate diffuses through. Molecular sieving is the separation due to size of the pore or molecule. Apart from industrial uses, medical fields also apply the use of zeolites in separation and purification of fluids, achieved through PSA.

[14] Zeolites are the main component of oxygen concentrators, and we will be using them in our research as well. In the application of zeolite with oxygen concentrators, the purpose is to extract nitrogen from ambient air. The most commonly used type of commercial zeolite is

zeolite 13X for the reason that the adsorption selectivity between zeolite 13X and nitrogen is much higher than with oxygen. Figure 1. shows the zeolite 13X used in our research.

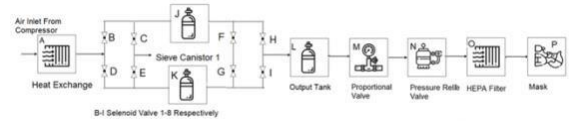
Fig. 1. Zeolite 13X used in this project.

DESIGN OF OXYGEN CONCENTRATOR

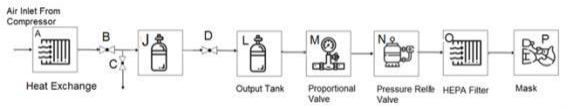
PSA oxygen concentrators can be classified into one-bed and two-bed systems. Figure 2. (a) And (b) illustrates pneumatic diagrams of one-bed and two-bed systems respectively. Computer aided design of one-bed and two-bed systems are shown in Figure 3. (a) and (b) respectively. In one bed systems, as shown in Figure 2. (a), the compressed air will flow in and out of canister J by the control of solenoid valve B, C and D. Concentrated oxygen will be store in the buffer tank L. Proportion valve M will then control the flow rate of the oxygen. Attached to buffer tank, a pressure safety relief valve N will release excessive pressure oxygen to room atmosphere. Before supplying oxygen to patients, a heat filter is used. One-bed PSA system reduces the cycle time; however, it suffers from pressure drop. The two-bed system shown in Figure 2. (b) uses two zeolite canisters depicted as J and K. The two-bed system will alternately adsorb and purge nitrogen to room atmosphere, i.e., first canister adsorbs nitrogen and releases oxygen to the patient, while the second canister removes the adsorbed nitrogen. The two-bed system performs better than the one-bed system. However, the two- bed system is more costly than the one-bed system.

The processing step of one-bed and two-bed systems are shown in Figure 4. (a) and (b) respectively. [15] The one-bed system operates in a three-step cycle, while the two-bed system operates in four step cycles. The first step in the three-step cycle is referred to as the adsorption phase. In this phase, solenoid valves B and D are opened, and high-pressure air passes through to canister J. Nitrogen is adsorbed while oxygen is concentrated and stored in the buffer tank. In the second phase, the delay phase, solenoid valves B and C are closed while solenoid valve D is opened. Oxygen gradually moves to the buffer tank. In the last phase, de-absorption, solenoid valve B is closed while solenoid valves C and D are open. Nitrogen is removed and released to the room. For the two-bed system,[16] the first step is known as the pressurization step. In this step, high-pressured air fills the first canister by opening solenoid valve B. While the second canister undergoes the blowdown phase; nitrogen is desorbed from zeolite by decreasing the pressure and opening solenoid valve E. In the second phase, adsorption phase, the solenoid valve B is still opened to supply the high-pressure feed air to the entire column at constant velocity and enriched oxygen is delivered to the buffer tank through valve H. At this phase, some parts of the oxygen are used to purge the second column through the opening of

solenoid valves E and G. In the third step, canister
1 undergoes a blowdown step by the opening of solenoidvalves C and D to lower the pressure in canister 1. All the remaining valved are closed. By the opening of solenoid valve D, the canister 2 now enters pressurization phase. Finally,canister 1 undergoes the purging step while canister 2 undergoes the adsorption step. In this step, the valves E and C are open to purge column 1 and D and I are open to deliver oxygen-rich air at the column exit of the second bed. A part of product gas from bed 2 is used to purge the first bed.



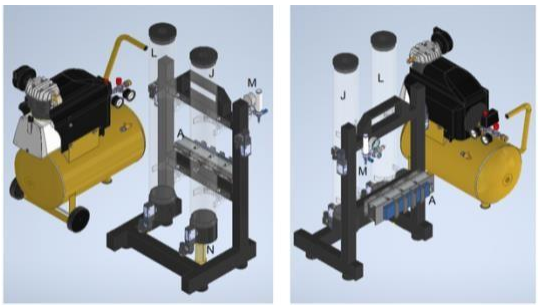
(a)



(b)

Fig. 2. Pneumatic Diagrams of (a) one-bed and (b) two-bed system.

(a)



(b)

Fig. 3. Computer aided design of (a) one-bed and (b) two-bed system.

	B	C	D	E	F	G	H	I
Pressurization	ON	OFF	OFF	ON	OFF	OFF	OFF	OFF
Adsorption	ON	OFF	OFF	ON	ON	ON	ON	OFF
Blow down	OFF	ON	ON	OFF	OFF	OFF	OFF	OFF
Purge	OFF	ON	ON	OFF	ON	ON	ON	ON

(a)

	B	C	D
Adsorption	ON	OFF	ON
Delay	OFF	OFF	ON
Desorption	OFF	ON	ON

(b)

Fig. 4. Processing step of (a) one-bed and (b) two-bed system.[18]

ELECTRONIC AND MICROCONTROLLER

To evaluate the treatment of patients through the oxygen concentrator, we have equipped our PSA oxygen concentrator with a pulse oximeter. Our pulse oximeter model is an OEM YS2000A Spo2 module, shown in Figure 5. (a),[17] which is capable of providing three parameters including Spo2 (%), pulse rate(BPM) and body temperature(℃). The YS2000A Spo2 module is equipped with USB communication Interface allowing the transfer measured data. We have used ESP 32 microcontroller to interface with YS2000A Spo2 module. The serial interfacing protocol via USB port is shown in figure 5(b).

Fig. 5. YS2000A SpO2 module

Byte	Bit	Description
1	0-3	Signal Strength (0-8)
	4	1 = no signal, 0 = OK
	5	1 = probe unplugged, 0 = OK
	6	1 = pulse beep
	7	Sync bit= 1 (Search Package Header)
2	0-6	Pleth invalid = 0
	7	Sync bit = 0
3	0-3	Bar graph invalid = 0
	4	Sensor off = 1, 0 = OK
	5	Pulse research = 1, 0 = OK
	6	Bit 6 of Byte 3 is bit 7 of the Pulse Rate
	7	Sync bit = 0
4	0-6	Bit 0-6 of Byte 3 is bit 7 of the Pulse Rae
	7	Sync bit = 0
5	0-6	SpO_2 0 - 100% invalid = 0x7f
	7	Sync bit = 0
6	0-6	Temperature Integer (0-50) invalid = 0
	7	Sync bit = 0
7	0-6	Temperature decimal invalid = 0
	7	Sync bit = 0

TABLE I. SERIAL INTERFACING PROTOCOL

```
1           len = mySerial.readBytes(buff1,16);
2           if(len > 0)
3           {
4             for (iii=0;iii<16;iii++)
5              if ((uint8_t)buff1[iii]>127 && iii<9){
6                 pso2 = (uint8_t)buff1[iii+4];
7                 oxy = (uint8_t)buff1[iii+1];
8                 hr = (uint8_t)buff1[iii+3];
9                 Temp = (uint8_t)buff1[iii+5];
10                Tempd = (uint8_t)buff1[iii+6];
11                bodytemp = Temp+(0.1*Tempd);
12                iii=iii+6;
13          }
14            mySerial.flush();
15
16           }
17
```

Fig. 6. The code for detecting parameter streams.

TABLE II. CODE EXPLANATION

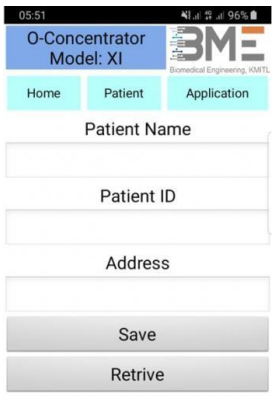
Line number	Explanation
1	16 streaming data is read into buffer
2	Check if length of streaming data ismore than 0
4	Loop through data
5	Find the header (data >127)

6-11 If (header found), acquire parameters

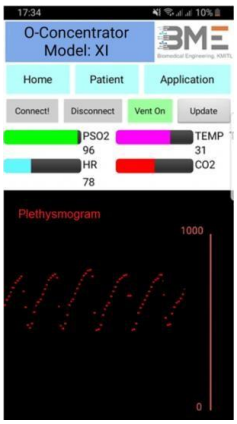
SOFTWARE DESIGN

As COVID-19 is highly contagious, we have designedan application software to remotely monitor and control our PSA oxygen adsorption. MIT App inventor [18] has been used. The application Graphic user interface, shown in Figure 7, consists of three pages, i.e., (i) log-in page, (ii) patient data page and (iii) application page. For the login- page, Firebase cloud database has been used to provide a secured access to the application. Patient data page exploits Firebase cloud database. The application page is used in communication with ESP32 microcontroller to control the operation and also to receive data from pulse oximeter module including PSO2, heartrate, body temperature and Plethysmogram. Only Plethysmogram is plotted on the application canvas. The remaining are displayed with bar charts and numeric display.

(a)



(b)



(c)

Fig. 7. MIT App Inventor; (a) log-in page, (b) patient data page and (c) Application page

I. EXPERIMENTAL RESULTS

The constructed oxygen concentrator is shown in Figure 8. An aluminum rack has been used for the main chassis. The dimension of the rack is 170mm x 300 mm x 448mm. A PC water-cooling cylinder-tank reservoir is used as a canister. The diameter and the height of the reservoir is 50 mm and 400 mm respectively. The ESP32 microcontroller and YS2000A Spo2 module is installed in the aluminum control box with the dimensions 100mm x150mm x80 mm. To control the air from compressor, eight of solenoid valves will be used for the two-bed system. For the one-bed system, only three solenoid valves are used. All solenoid valves are mounted on the main chassis. To calibrate the system, Fluke VT Plus HF Gas Flow Ventilator Analyzer has been used as shown in figure 9. Our system can provide oxygen with 85% concentration and 80% concentration for two-bed PSA system and one-bed PSA system respectively.

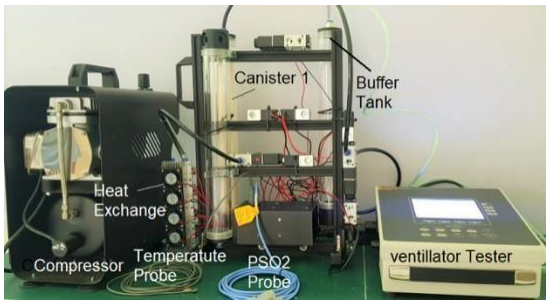


Fig. 8. PSA Oxygen Concentrator in One-Bed Configuration

II. CONCLUSIONS

Under the current pandemic of COVID-19, an increasing demand for oxygen generator is skyrocketed to treat a mild- to-severe patient who suffered from lung infection or pneumonia. An emergency oxygen concentrator based on pressure swing absorption (PSA) has been proposed in this research. The system consists mainly three cylindrical canisters. The first two canisters are filled with zeolite absorber. The remaining canister is the buffer container. Dust-free air from oil-free compressor is controlled with a series of solenoid valves to alternately pump into the first two canisters. Zeolite acted as the molecule filter will absorb nitrogen from the air. The remaining air with enriched oxygen will then store at the buffer container before releasing to the patient. Using two zeolite-filled canisters, the system is called two-bed pressure swing absorption oxygen concentrator. Only one zeolite-filled canister, however, can be used. Obviously, two-bed pressure swing absorption oxygen concentrator is more efficient than the two-bed pressure swing absorption oxygen. To evaluate the oxygen treatment, we have equipped the system with pulse oximeter that can provide saturation oxygen concentration, pulse rate, body temperature and Plethysmogram. As the COVID-19 is an infectious disease, MIT app inventor is designed to remotely control and monitor the oxygen concentrator. The experiment results demonstrate a promising result as the proposed oxygen concentrator can provide an oxygen up to 80% for one-bed system and 85% for two-bed system.

A Controller Design for Oxygen Concentrator

Soohyun Kim, Hansil Kim, Donghyun Moon Key

***Abstract*— In recent years, an improvement of air quality has considered as a big issue in many countries. Fine dust and exhaust have a strong influence on ventilating indoor or outdoor. This makes people close the window to prevent inflow of corrupted air by fine dust from outside. It causes low level of air clearness of people's studying or working. This phenomenon has led some useful devices for better air quality to appear. Furthermore, there are other types of device for air clearness which directly emit oxygen through isolating nitrogen and oxygen. But it is not worthy because of many limited conditions. In this paper, a controller design for an oxygen concentrator is constructed by on-line and performed for multi purposes. PCB board including wi-fi system is equipped by IoT network, relay system an regulator.**

Figure 1 Oxygen Emitting Module from Keynet Corporation

Keywords— Oxygen Concentrator, Air Quality, Fine Dust, IoT

I. INTRODUCTION

In recent year, the problems about air quality have continuously become main issue about environment. Some substances like fine dust, exhaust, or smog raise a problem of indoor ventilation [1][2]. Children or old people who are vulnerable to respiratory disease should be taken care of living well. For these reasons, oxygen concentrator can be useful in many purposes.

Even though the device places a big role of air quality, the problem is very expensive. In this paper, various functions were equipped to oxygen generator for performing multiple objects. There are many additional devices for oxygen emitter, such as wireless charging system, Lamps. A remote-control system with IoT network is also adopted and progressed in this research [3][4].

II. EXPERIMENTAL ENVIRONMENT

A. *Oxygen Emitting machine*

Keynet Corporation provide Oxygen Emitting Module for this study. This module divides oxygen and nitrogen with thin hollow fibers that are inside of the module. Name of this module is "KEYNET_01M". The external diameter of the hollow fiber is 400□m and circumference of it is 1,256□m. There are 3,800 strands of hollow fiber inside of the module. Figure 1 is the module that used in this study. As the picture shows, there are lots of tiny fibers in the pipe. They divide the air and emit the oxygen. Two modules are used in one machine.

MCU

In this paper, Arduino system board is used for controlling the system. It is currently used in a variety of system designs due to its excellent ease of use.

B. *Regulator*

The ATmega2560 MCU requires 5V, but the Wi-Fi module runs at 3.3V. Sometimes, specific systems which perform very high-quality use double-input power system [5]. In this paper, we need a single-input power system run and control whole system with same voltage. We need to convert 5V input to 3.3V with a PAM3110 Regulator.

C. *Relay*

The voltage difference between MCU and Oxygen emitter leads to use relay. The machine needs 220V AC for running choosing simply a relay HF3FD.

D. *Wi-Fi Module*

For accessing system with network, wireless system is chosen. Because the Bluetooth or Zigbee modules are not fit for our purpose [6], Wi-Fi ESP8266 is selected. [7]

METHODS

A. *PCB Board Design*

We design each module for circuit and merge every module in one board. There are several steps for this design.

(1) Design and test the circuit on breadboard briefly. If the circuit works correctly, go on next step.

(2) Design it with OrCAD and make a Gerber Format file for make PCB (Printed Circuit Board).

(3) Make a PCB. Figure 2 is the design-finished circuit made by OrCAD

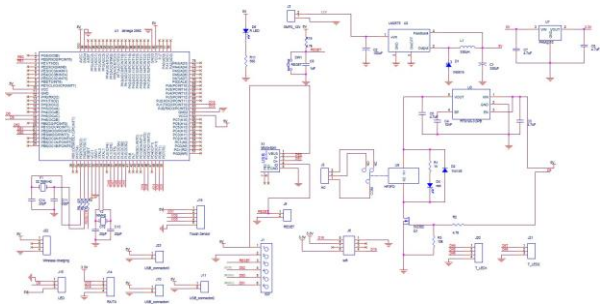


Figure 2 Design Finished Circuit

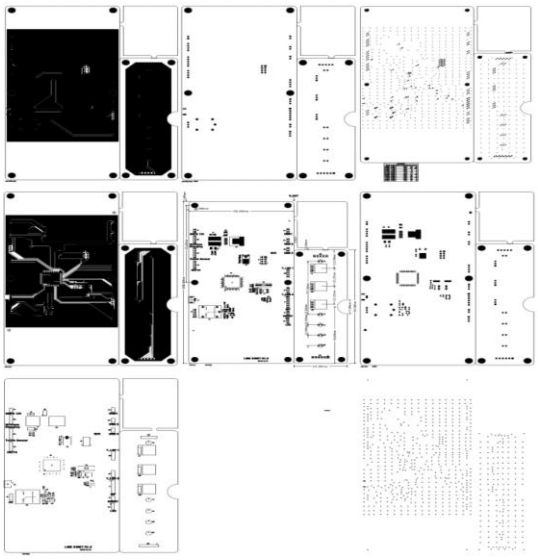


Figure 3 Gerber Format File

Figure 2 shows the merged circuit of this system. This circuit includes MCU, regulator, relay, touch sensor, LED, and I/O pin for USB. Touch sensor, LED and the USB part is added while the research and meeting. Touch sensor and LED is for convenience of use, USB is for providing wired and wireless charging system.

Figure 3 shows the Gerber file. Figure 2 may be just circuit file, Figure 3 has difference from the composition. There are elements and wiring for making PCB. Elements are arranged in set frame, and wires are connected to each elements and pins. When Gerber file be made, it goes to professional company to make PCB.

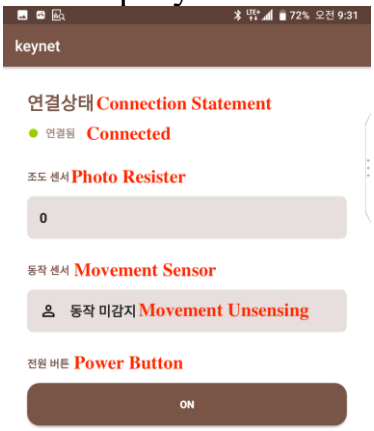


Figure 4 Application for the Oxygen Emitting Machine

B. *Application for Wireless Control*

Figure 4 shows an application for the Oxygen Emitting Machine. We design it and monitor photo resistor and movement sensor state. It can control the machine by remote system. This application can access the server [8]. The server has a table with 4 columns, which are Device ID, On/Off status, photo resistor value and movement sensor status. When it accesses the server, it checks device ID, and read photo resistor value and movement sensor status. It can change the value only the on/off status of the machine. Contrastively, at the machine, it can change three values except the ID of machine.

III. RESULTS

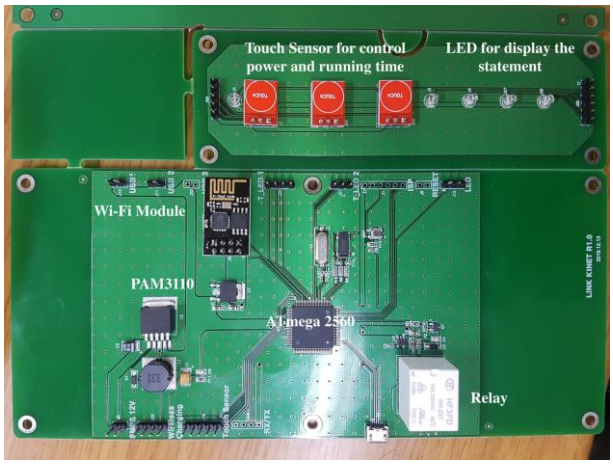


Figure 5 Final PCB Board

Figure 5 shows the final PCB board. We design the system able to control the running time. So, the touch sensors are added for power status and running time and also the LEDs for display the status.

Figure 6 Final designed system

Figure 6 shows the final designed system. Of course, it can emit oxygen. Wireless charging system, mood lamp, and USB port for charging are installed at the machine.

IV. CONCLUSION

This study was conducted with national project with Keynet Corporation. Now the project is finished, but still in progress with the company. The voltage problem and wifi application with server can be further research subject next time. This will enrich our life of our mankind with final product.

Design, implementation and testing of a portable device formultiparametric activity and SpO2 monitoring

S. Shiri and A. Aliverti

Abstract— **Patients with chronic obstructive pulmonary diseases (COPD) are usually prescribed supplemental oxygen, especially during outdoor physical activities. However, the GOLD (Global Initiative for Obstructive Lung Disease) guidelines do not provide specific recommendations about oxygen dosage during exercise. Also, previous studies have demonstrated that using resting arterial blood gases to regulate the flow of portable oxygen concentrator does not effectively prevent desaturation during activities of daily living (ADL). On the other hand, some clinicians prefer to individualize oxygen therapy during exercise using several types of stress tests which are mostly inaccurate comparing to the ADL, expensive and time consuming for both physician and patient.**

Because of the relationship between oxygen desaturation and some activity parameters such as walking velocity and number of steps, we have designed a portable device composed of an Android smartphone to monitor multiple activity parameters, a Pulse Oximeter and a Raspberry pi 3 which communicate with other devices through Bluetooth Low Energy (BLE) in order to find out the relationship between each activity parameter and desaturation during ADL in order to help physicians to prescribe the proper supplementary oxygen therapy.

Thanks to the use of Raspberry pi 3 with a powerful quad- core microprocessor, other additional sensors equipped with BLE can be connected to the device to synchronously collect the more comprehensive data.

I. INTRODUCTION

Chronic obstructive pulmonary disease (COPD) is a progressive disease affecting patients in daily life, both physically and emotionally [1]. There are some factors that can cause COPD symptoms to worsen or flare up, such as air pollution and physical inactivity [2,3]. In addition, inappropriate LTOT (long-term oxygen

therapy) prescriptions can cause significant morbidity as well as might lead to increasing oxygen costs [4].

The benefits of LTOT in patients with COPD have been reported by many investigators [5]. However, there is little information on how clinicians should choose the flow rates of supplemental oxygen that the patients should receive during their activities of daily living (ADL) [6]. In addition, there is not a specific recommendation in GOLD (Global Initiative for Obstructive Lung Disease) guidelines about oxygen dosage during physical activities [7].

Nowadays, physicians mostly use NOTT (Nocturnal Oxygen Therapy Trials) guidelines to prescribe oxygen therapy for COPD patients which recommend increasing the resting flow rate by 1 Lit/min during exercise [8]. However, using oxygen flow rate during the resting conditions to adjust the flow of oxygen does not efficiently correct desaturation during daily physical activities [9]. In some cases, to improve the quality of long-term oxygen therapy during exercise, patients are asked to wear different activity monitoring sensors which help the physician to find out the relationship between desaturation and the level of exertion during physical activity [10]. Aside from being costly, time-consuming and inconveniences which people and especially elderly have for using wearable sensors, they are only used to do certain tests (such as 6-minutes walking) in clinics and not in the real-life daily activities. Such tests do not provide clear information about the required oxygen dosage during the hypoxemic COPD patient's ADL [11]. Moreover, during the walk tests, there are several factors that may be sources of variation that affect SpO2 response [12,13] and prescription of oxygen flow rate.

Taking into account the wide usage of smartphones all around the globe and taking advantage of their built-in sensors, we have designed a portable device which is able to communicate with the smartphone, a pulse oximeter and other additional sensors through Bluetooth Low Energy (BLE) in the same time to collect useful data related to physical activity (such as acceleration, number of steps, velocity of walking and GPS tracking) not only in a short period of time or for a particular test, but during ADL. The output of this device is a text file contains all those data which are arranged in columns. Using MATLAB for data analysis, a graphical user interface has been designed to filter out noises and finally show the graphs in such a way to be comparable to each other. Examining the correlation between desaturation and the number of steps during a certain period, velocity of walking and heart rate individually, a clear relationship between data can be found. These correlations, which always can be measured, will help physicians to personalized the oxygen prescription for each patient according to their severity of COPD. This can be a step toward achieving a patient-centered approach to appropriately prescribe oxygen therapy for COPD patients during their outdoor physical activities.

II. MATERIAL AND METHODS

A. *Hardware and software architecture*

The architecture of the system consists of: one small single-board computer that we have used Raspberry pi 3 model B which includes an ARM compatible central processing unit (CPU) with 1.2 GHz speed and on-board memory 1GB LPDDR2 (900 MHz) RAM, Secure Digital (SD) card to store the operating system and program memory, BLE and 40-pin

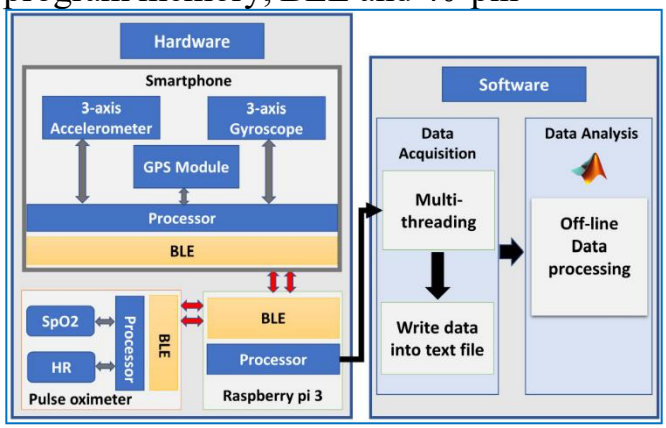


Fig. 1 Hardware and software architecture of the entire system

header populated GPIO; a light weight (296 g) ROMOSS power bank able to provide 5 volts, 2.1 amperes with the capacity of 10400 mAh; a 3-D printed box which is designed in order to place Raspberry pi 3 and the power bank inside it; NONIN Model 3150 Bluetooth Wrist-Worn Pulse Oximeter to measure SpO2 and Heart Rate with the sampling frequency of 1 Hz with the possibility to communicate with Raspberry pi 3 to send data through BLE; a smartphone with an Android operative system (OS) and SensoDuino installed software to read data from built-in sensors (Accelerometer, Gyroscope and GPS) simultaneously.

Using multi-threading technic, data from Nonin pulse- oximeter and smartphone are saved to the Raspberry pi 3 in the same time.

B. *Data acquisition*

In this work, raspberry pi 3 plays a role as the main processor to communicate with a pulse oximeter and the smartphone at the same time thanks to multi-threading which is done in a single process.

Reading data from smartphone and the pulse oximeter is dedicated to 2 separated threads. Smartphone transmits a package of data contains: 3-axis accelerometer with the frequency of 10Hz, 3-axis gyroscope with the frequency of 10Hz as well as latitude, longitude and velocity calculated by

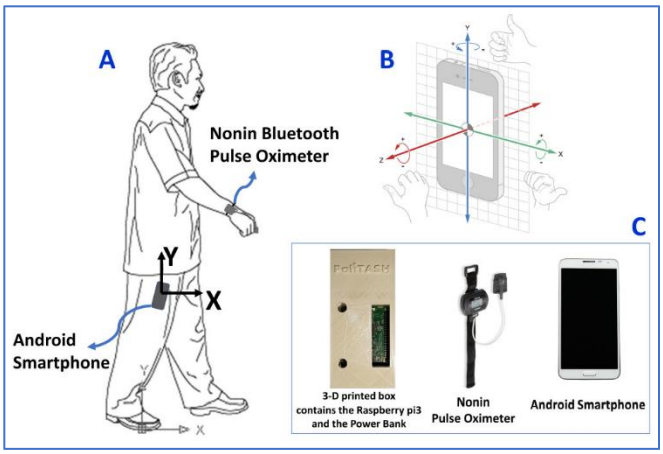


Fig. 2 A, B: Attachment of the pulse oximeter and position of the smartphone inside the subject's pocket. C: Hardware components of the entire system GPS with the frequency of 2Hz to the Raspberry pi 3. In the same time, %SpO2 and heart rate measured by pulse oximeter with the frequency of 1Hz transmitted to the raspberry pi 3. By dedicating a single thread only to write transmitted data with the frequency of 20Hz, we may have some repetitive data in the output text file (which can be corrected in the off-line data processing) but we are sure about the safe recording of all data.

As demonstrated in the Fig.2 by inserting the smartphone in the pocket vertically, as the legs move back and forward during walking consequently smartphone moves around Yaw angle and data from z-axis of the gyroscope demonstrates a signal which the number of steps can be extracted by counting the number of picks. To monitor the measured acceleration, total value has been calculated (1) and filtered by 5^{th} order HPF (high-pass filter) with the cut-off frequency of 0.01 Hz.

C. *Data Analysis*

Using graphical user interface of MATLAB, the user can monitor:

- Acceleration, velocity, the number of steps, heart rate and SpO2 graphs which can be plotted either in the same window with the same x-axis (minute) or separately.
- Tracking of the patient; the user can demonstrate heart rate, %SpO2 and activity information by clicking on a specific point of the tracking path. (in addition, having an air pollution module, the patient is warned about the polluted area which may flare up the COPD symptoms).

All graphs can be plotted in such a way to be comparable with each other to demonstrate the relationship between activity parameters and desaturation (Fig.

3). These relationships will help to achieve comprehensive, personalized and patient-centered information which help physicians to appropriately grade the severity of COPD and consequently prescribe proper oxygen therapy for each patient during ADL.

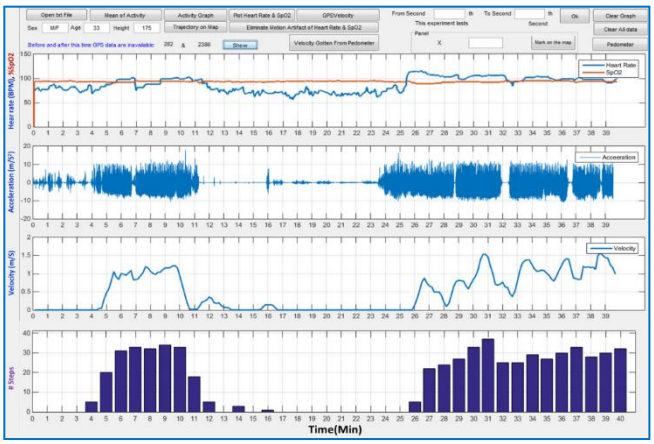
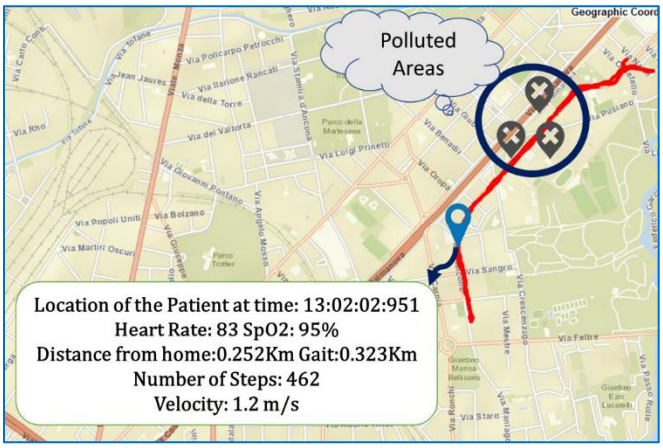


Fig. 3 GUI of MATLAB which shows heart rate, %SpO2, acceleration, velocity and the number of steps graphs which are plotted with respect to the time

1) *Velocity Calculation*

$$\text{Total Acceleration}(m/s^2) = \sqrt{x^2+y^2+z^2} \quad (1)$$



III. RESULTS

Subjects	Age	Height	Sex	Physiological Problem
Subject 1	30	165	Female	Mitral valve prolapses
Subject 2	28	174	Male	Smoker
Subject 3	28	183	Male	No
Subject 4	26	167	Male	No
Subject 5	33	175	Male	No

Fig. 4 In case of using the pollution module, subject can be warned about the polluted area on his path. By selecting a point on the path, all acquired

information will be shown In the validation part, the good reliability of the walking velocity calculated by GPS is discussed. The problem is for sometimes that GPS data are not available (GPS Blind Spots) and the GPS velocity is inaccurate. Considering a specific interval (we have chosen 60 seconds), we have used (2) to get the velocity in case of GPS data unavailability.

number of Steps*Stride Length

Table 1. Information of subjects participated in the tests

Five subjects have participated in the various tests (Table 1). Four subjects were asked to do 6-minute walking tests with the various velocity (slow walking, fast walking and running). The resting time between each test is chosen by the subject. One subject was asked to walk freely for 40 minutes. During all tests information about acceleration, velocity calculated by GPS, velocity calculated in GPS Blind Spots, %SpO2, heart rate, the number of steps, stride length and GPS tracking were recorded.

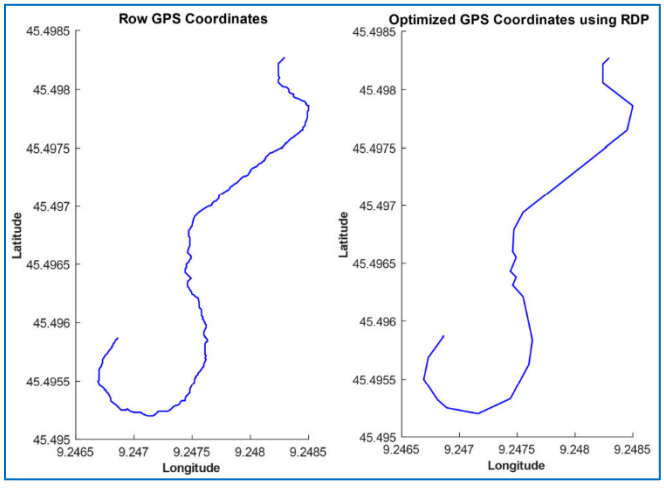
Subject 1 was asked to do 6-minute slow walking test. As it is illustrated in the Fig. 6 by even the low intensity of walking, the subject experienced a decrease in %SpO2 from
%94 to %82 approximately and a gradual increase of the heartrate from 90 to about 98 BPM (beat per minute). Subject 4

Velocity(m/s) = *Stride Length Calculation*

2)

3) however, showed a fast reaction of the heart rate from 60 to 99 BPM as the intensity of walking increased, while %SpO2 remained approximately constant (Fig. 7).

In order to get the accurate stride length, first, we have reduced the noise of GPS tracking using Ramer-Douglas- Peucker (RDP) algorithm.



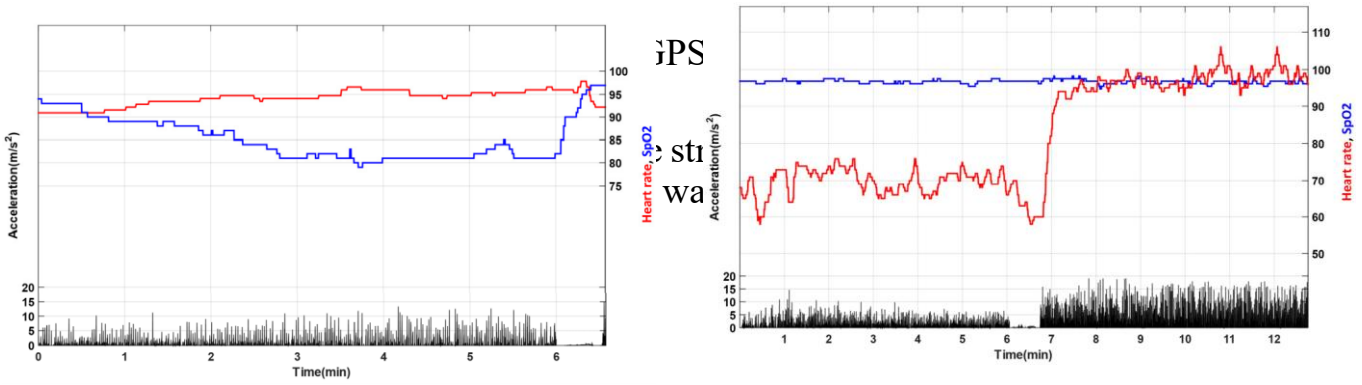


Fig. 6 Acceleration, %SpO2 and heart rate for subject 1 plotted in one graph

$$\text{Distance (AB) Stride Length(m)} = \frac{\text{Number of Steps}}{}$$

Fig. 7 Acceleration, %SpO2 and heart rate for subject 4 plotted in one graph. *Validation*

Because of the significant role of the walking velocity in the changing of the systemic oxygen delivery and arterial oxygen saturation [14] the reliability of the calculated walking velocity in this work is important.

In order to validate the accuracy of the velocity calculated by GPS, a healthy subject (subject 3) was asked to do three walking tests with the various velocity in a running track. Each test included walking for a lap and subject tried to keep walking with the constant velocity. Since the length of each lap is known (ΔX) and the time for each test can accurately be measured (ΔT), it can be considered as the gold standard to validate the velocity calculated by GPS of the android smartphone. As the subject couldn't walk with the constant velocity, the mean of GPS velocity is calculated for each test.

Table 2 shows a good reliability of the velocity calculated by GPS for each three tests comparing the gold standards.

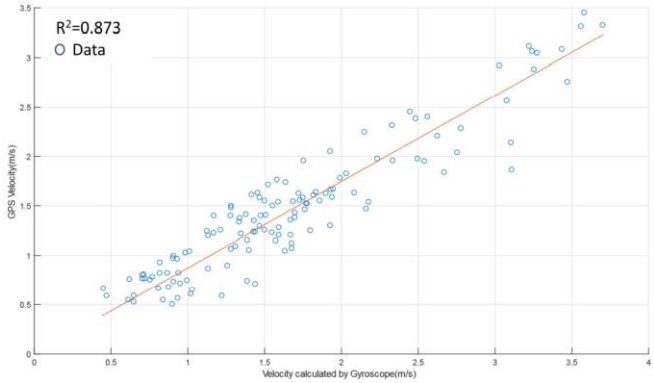


Fig. 8 Bland-Altman graph to measure the accuracy of velocity calculation in the GPS blind spots comparing to the GPS velocity

Fig. 9 Linear Regression of the velocity calculated GPS blind spots and GPS velocity (R^2=0.873)

As far as the validation of calculation of the velocity in the GPS Blind Spots is concerned, the Bland-Altman and linear regression graphs are plotted using the data collected from all the subjects who participated in the tests. Both Bland-Altman and linear regression graphs demonstrate an acceptable agreement between to measurement of the velocity calculation in the GPS Blind Spots and GPS velocity (Fig. 8 and Fig. 9).

SOFTWARE

Arduino

Arduino is an open-source hardware and software company, project, and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices. Its hardware products are licensed under a CC BY-SA license, while software is licensed under the GNU Lesser General Public License (LGPL) or the GNU General Public License (GPL),[1] permitting the manufacture of Arduino boards and software distribution by anyone. Arduino boards are available commercially from the official website or through authorized distributors.

Arduino board designs use a variety of microprocessors and controllers. The boards are equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards ('shields') or breadboards (for prototyping) and other circuits. The boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs. The microcontrollers can be programmed using the C and C++ programming languages, using a standard API which is also known as the **Arduino language**, inspired by the Processing language and used with a modified version of the Processing IDE. In addition to using traditional compiler toolchains, the Arduino project provides an integrated development environment (IDE) and a command line tool developed in Go.

The Arduino project began in 2005 as a tool for students at the Interaction Design Institute Ivrea, Italy,[2] aiming to provide a low-cost and easy way for novices and professionals to create devices that interact with their environment using sensors and actuators. Common examples of such devices intended for beginner hobbyists include simple robots, thermostats and motion detectors.

Code:

1: Oxygen Concentrator:

```
//#include <Servo.LOW>
//Servo myservo;
//void SERVOMOTOR(void);

unsigned char set1=16;   //D0 First set of three pumps
unsigned char set2=4;   // D1 Second set of three pumps
unsigned char oxyvalve=5; //D2

unsigned char oxyvalve2=0; //D3 //Oxygen Valve
unsigned char oxyvalve3=2; //D4 //Oxygen Valve 4

unsigned char pressurepump=14; //D5
unsigned char fan=12; //D6

//int pos = 0;



void setup()
{
  pinMode(set1,OUTPUT);  //Set 1 is set as OUTPUT
  pinMode(set2,OUTPUT);  //Set 2 is set as OUTPUT
  pinMode(oxyvalve,OUTPUT);  //Oxygen Valve is set as OUTPUT
  pinMode(oxyvalve2,OUTPUT);
  pinMode(oxyvalve3,OUTPUT);//Oxygen Valve is set as OUTPUT
  pinMode(pressurepump,OUTPUT);  // pressurepump is set as OUTPUT
```

```
  pinMode(fan,OUTPUT);    //fan is set as OUTPUT
}

void loop()
{
  digitalWrite(set1,LOW);  //Set 1 is ON
  digitalWrite(oxyvalve,HIGH);  //Oxygen Valve is OFF
   delay(9000);   //Delay time 1 minute 20 seconds

 digitalWrite(set1,HIGH);  //Set 1 is ON
digitalWrite(oxyvalve,LOW);//Oxygen Valve is ON
  digitalWrite(pressurepump,LOW);//Pressure Pump is ON
  delay(4500);   //Delay time 3 seconds
  digitalWrite(oxyvalve,HIGH);//Oxygen Valve is OFF
   digitalWrite(pressurepump,HIGH);//Pressure Pump is OFF
  delay(1000);
  OXYVALVE();
  }



void OXYVALVE(void)
 {
  digitalWrite(oxyvalve3,LOW);  //Oxygen Valve is OFF
 digitalWrite(set2,LOW);  //Set 2 is ON
  digitalWrite(oxyvalve2,HIGH);//Oxygen Valve is OFF
 delay(9000);   //Delay time 9 seconds
 digitalWrite(set2,HIGH);  //Set 2 is ON
  digitalWrite(oxyvalve2,LOW);  //Oxygen Valve is ON
```

```
  digitalWrite(pressurepump,LOW);//Pressure Pump is ON

  delay(4500);   //Delay time 4.5 seconds
  digitalWrite(oxyvalve2,HIGH);  //Oxygen Valve is OFF
  digitalWrite(pressurepump,HIGH);//Pressure Pump is OFF

digitalWrite(oxyvalve3,HIGH);//Oxygen Valve is ON



}
```

Code For Purity of Oxygen:

```
//ADS1050 12 bit ADC
#include <Wire.h>
#include <Adafruit_ADS1015.h>
Adafruit_ADS1015 ads;     /* Use thi for the 12-bit version */

//Nokia Display
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Adafruit_PCD8544.h>

// Hardware SPI (faster, but must use certain hardware pins):
// SCK is LCD serial clock (SCLK) - this is pin 13 on Arduino Uno
// MOSI is LCD DIN - this is pin 11 on an Arduino Uno
// pin 5 - Data/Command select (D/C)
// pin 8 - LCD chip select (CS)
// pin 6 - LCD reset (RST)
```

```
 Adafruit_PCD8544 display = Adafruit_PCD8544(5, 8, 6);
// Note with hardware SPI MISO and SS pins aren't used but will still be read
// and written to during SPI transfer.  Be careful sharing these pins!



//Button and backlight pin declaration
int button1pin=3;//Calibration, Enter Button


//General Cariable setup
double  calibrationv; //used to store calibrated value
int sensorcheck=0;//to check health on sensor. If value is 0 sensor works, if value is 1 sensor out of range or not connected
int Sensor_lowrange=58;//When sensor is healthy and new it reads 58 on low
int Sensor_highrange=106;//When sensor is healthy and new it reads 106 on high
int current_function=0;
void setup(void)
{
 Serial.begin(9600);
  pinMode(button1pin,INPUT);
//starts ADS readings
 ads.setGain(GAIN_SIXTEEN);    // 16x gain  +/- 0.256V  1 bit = 0.125mV  0.0078125mV
  ads.begin();
  calibrationv=calibrate();
  //Nokia Display Setup
  display.begin();
  // you can change the contrast around to adapt the display
  // for the best viewing!
  display.setContrast(50);
    if ((calibrationv > Sensor_highrange) || (calibrationv < Sensor_lowrange))
   {
```

```
    sensorcheck=1;
     current_function=1;//Sensor needs to be calibrated
     need_calibrating();//print need calibrating message
   }
}
//Prints need calibrating text
void need_calibrating(){
   display.clearDisplay();
  display.setCursor(0,0);
  display.setTextSize(1);
  display.setTextColor(BLACK);
  display.println("Sensor error");
  display.println("Please");
  display.println("calibrate");
  display.println(calibrationv);
  display.display();

}
//Take 20 readings and avaraging it to exclude miner diviations of hte reading
int calibrate(){
  int16_t adc0=0;
  int result;
  for(int i=0; i<=19; i++)
       {
         adc0=adc0+ads.readADC_SingleEnded(0);
       }

  result=adc0/20;
  return result;
```

```
}


void loop(void) {
  //********  Main Loop variable declaration ***********
     double modr;//Variable to hold mod value in
    int16_t adc0=0;
    double result;//After calculations holds the current O2 percentage
    double currentmv; //the current mv put out by the oxygen sensor;
    double calibratev;

 //***** Function button read section ********
 int button1state=digitalRead(button1pin);



  if(button1state==LOW){
    if(current_function==0){
       current_function=1;//Sensor needs to be calibrated
    }
  }

  switch(current_function){
   case 0://Analyzing O2
     //taking 20 samples. The sensor might spike for a millisecond. After we average the samples into one
value
     for(int i=0; i<=19; i++)
       {
         adc0=adc0+ads.readADC_SingleEnded(0);
       }
```

```
  currentmv = adc0/20;
  calibratev=calibrationv;
  result=(currentmv/calibratev)*20.9;
  //Write to display
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE, BLACK);
  display.setCursor(0,0);
  display.print("O2%");
  display.setTextColor(BLACK);
  display.print(" ");
  display.setTextSize(2);
  display.println(result,1);
  display.display();
      delay(1000);
 break;
case 1:
   display.clearDisplay();
 display.setCursor(0,0);
 display.setTextSize(1);
 display.setTextColor(WHITE,BLACK);
 display.println("Calibrating");
 display.display();

 current_function=0;//O2 analyzing
 calibrationv=calibrate();
 delay(2000);
  if ((calibrationv > Sensor_highrange) || (calibrationv < Sensor_lowrange)){
```

```
        current_function=1;//Sensor needs to be calibrated
        need_calibrating();//print need calibrating message
      }
  break;

 }

}
```

2: Oxygen Analyzer Code:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
// Set the LCD address to 0x27 or 0x3f for a 16 chars and 2 line display
LiquidCrystal_I2C lcd(0x27, 20, 4);
#include <Adafruit_ADS1X15.h>
Adafruit_ADS1115 ads;
#include <SoftwareSerial.h>
#include <TinyGPSPlus.h>
SoftwareSerial SIM900A(2,3);
static const int RXPin = 0, TXPin = 1;
static const uint32_t GPSBaud = 9600;
#define BUTTON_PIN 5
TinyGPSPlus gps;
```

```
SoftwareSerial ss(RXPin, TXPin);


void SendMessage();

void displayInfo();

void RecieveMessage();

void MakeCall();

void RedialCall();

void gsmsetup();

void gsmloop();

void puritysetup();

void purityloop();

#include <SPI.h>

//Adafruit_ADS1115 ads;

#include <Adafruit_GFX.h>

#include <Adafruit_SSD1306.h>/

#define SCREEN_WIDTH 128 // OLED display width, in pixels

#define SCREEN_HEIGHT 32 // OLED display height, in pixels

#define OLED_RESET    -1 // 4 // Reset pin # (or -1 if sharing Arduino reset pin)

#define SCREEN_ADDRESS 0x3C///< See datasheet for Address; 0x3D for 128x64, 0x3C for 128x32

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

float Voltage = 0.0;

float Oxy = 0.0;

#define BUTTON_PIN1 4

void purity();

void drawLine();
```

```
void oxysetuo();

void oxyloop();

void heart_beat();

void display_data();

void drawLine();

void onBeatDetected();
```

```
#include "MAX30100_PulseOximeter.h"

#define ENABLE_MAX30100 1


#if ENABLE_MAX30100

#define REPORTING_PERIOD_MS     5000


PulseOximeter pox;

#endif

uint32_t tsLastReport = 0;

const int buttonPin = 12;

int xPos = 0;

void onBeatDetected()

{

  Serial.println("Beat!");

  heart_beat(&xPos);
```

```
}




void setup(){


  pinMode(BUTTON_PIN1, INPUT_PULLUP);
  pinMode(BUTTON_PIN, INPUT_PULLUP);
  //gsmsetup();
  //puritysetup();
  ads.setGain(GAIN_SIXTEEN);
  Serial.begin(9600);
  ads.begin();


  delay(1000);
  SIM900A.begin(9600);   // Setting the baud rate of GSM Module
  //Serial.begin(9600);    // Setting the baud rate of Serial Monitor (Arduino)
  Serial.println ("SIM900A Ready");
  delay(1000);
  ss.begin(GPSBaud);
  display.begin();
```

```
Serial.println(F("Gps connected"));
Serial.print(F("Testing TinyGPSPlus library v. ")); Serial.println(TinyGPSPlus::libraryVersion());
Serial.println(F("by Ahsan"));
display.clearDisplay();
display.setCursor(0,0);
display.setTextColor(WHITE);
display.print("Oxygen Analyser with");
display.setCursor(0,10);
display.print("fault notification");
display.setCursor(0,20);
display.print("by Ahsan");
Serial.println();
display.display();
delay(3000);
```

```
{
Serial.println("SSD1306 128x64 OLED TEST");
// SSD1306_SWITCHCAPVCC = generate display voltage from 3.3V internally
if (!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
  Serial.println(F("SSD1306 allocation failed"));
  for (;;); // Don't proceed, loop forever
}
```

```
  // Show initial display buffer contents on the screen --
  // the library initializes this with an Adafruit splash screen.
  //display.display();
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(20, 18);
  // Display static text
  display.print("Pulse OxiMeter");
  int temp1 = 0;
  int temp2 = 40;
  int temp3 = 80;
  heart_beat(&temp1);
  heart_beat(&temp2);
  heart_beat(&temp3);
  xPos = 0;
  display.display();
  delay(2000); // Pause for 2 seconds
  display.cp437(true);
  display.clearDisplay();
  Serial.print("Initializing pulse oximeter..");
#if ENABLE_MAX30100
  // Initialize the PulseOximeter instance
  // Failures are generally due to an improper I2C wiring, missing power supply
  // or wrong target chip
```

```
  if (!pox.begin()) {
    Serial.println("FAILED");
    for (;;);
  } else {
    Serial.println("SUCCESS");
  }
  // The default current for the IR LED is 50mA and it could be changed
  //   by uncommenting the following line. Check MAX30100_Registers.h for all the
  //   available options.

  if (digitalRead(buttonPin) == LOW)
  {
  pox.setIRLedCurrent(MAX30100_LED_CURR_7_6MA);
  // Register a callback for the beat detection
  pox.setOnBeatDetectedCallback(onBeatDetected);
  display_data(0, 0);
  }
#endif
}
  }
```

```
void loop(){
  byte buttonState = digitalRead(BUTTON_PIN);
  byte buttonState1 = digitalRead(BUTTON_PIN1);

  if (buttonState == LOW) {
    purityloop();
    delay(2000);
    oxyloop();

  }

  if (buttonState1 == LOW) {
    gsmloop();
    delay(2000);
    oxyloop();
  }

  }
```

```
//void gsmsetup()
//{
 // SIM900A.begin(9600);   // Setting the baud rate of GSM Module
 // Serial.begin(9600);    // Setting the baud rate of Serial Monitor (Arduino)
 // Serial.println ("SIM900A Ready");
 // delay(1000);
 // ss.begin(GPSBaud);

 // Serial.println(F("Gps connected"));
 // Serial.print(F("Testing TinyGPSPlus library v. ")); Serial.println(TinyGPSPlus::libraryVersion());
  //Serial.println(F("by Ahsan"));
 // Serial.println();
 // delay(1000);
//}

void gsmloop()
{

  //byte buttonState = digitalRead(BUTTON_PIN);

 // if (buttonState1 == LOW) {
  while (ss.available() > 0)
    if (gps.encode(ss.read()))
      displayInfo();
```

```
  if (millis() > 5000 && gps.charsProcessed() < 10)
  {
    Serial.println(F("No GPS detected: check wiring."));
    while(true);
  }
  }
//}




void displayInfo()
{
  Serial.println("Fetching Location...");
  Serial.print(F("Location: "));
  if (gps.location.isValid() || (gps.date.isValid() && gps.time.isValid())  )
  {
    {
    Serial.print(gps.location.lat(), 6);
    Serial.print(F(","));
    Serial.println(gps.location.lng(), 6);
    delay(1000);
    SendMessage();
```

```
}


Serial.print(F("  Date/Time: "));
if (gps.date.isValid())
{
  Serial.print(gps.date.month());
  Serial.print(F("/"));
  Serial.print(gps.date.day());
  Serial.print(F("/"));
  Serial.print(gps.date.year());
}



Serial.print(F(" "));
{
  if (gps.time.hour() < 10) Serial.print(F("0"));
  Serial.print(gps.time.hour());
  Serial.print(F(":"));
  if (gps.time.minute() < 10) Serial.print(F("0"));
  Serial.print(gps.time.minute());
  Serial.print(F(":"));
  if (gps.time.second() < 10) Serial.print(F("0"));
  Serial.print(gps.time.second());
  Serial.print(F("."));
  if (gps.time.centisecond() < 10) Serial.print(F("0"));
```

```
    Serial.println(gps.time.centisecond());
  }
}
  else
  {
    Serial.print(F("INVALID"));
  }

  Serial.println();
}

void SendMessage()
{

  SIM900A.println("AT+CMGF=1");    //Sets the GSM Module in Text Mode
  delay(1000);
  //Serial.println ("Set Number to send sms");
  SIM900A.println("AT+CMGS=\"+918801058225\"\r"); //Mobile phone number to send message
  delay(1000);
```

```
 Serial.print("Setting location to send");
 delay(100);
 //Serial.print("Location: Lat: ");
      // Serial.print(gps.location.lat(), 6);
       //Serial.print(", Lon: ");
       //Serial.print(gps.location.lng(), 6);
       SIM900A.print("Location: Lat: ");
       SIM900A.print(gps.location.lat(), 6);
       SIM900A.print(", Lon: ");
       SIM900A.println(gps.location.lng(), 6);
       //SIM900A.println((char)26);// ASCII code of CTRL+Z
// delay(1000);

       SIM900A.print("Date:");
       SIM900A.print(gps.date.month());
       SIM900A.print("/");
       SIM900A.print(gps.date.day());
       SIM900A.print("/");
       SIM900A.println(gps.date.year());
       SIM900A.print("Time:");
       SIM900A.print(gps.time.hour());
       SIM900A.print(":");
       SIM900A.print(gps.time.minute());
       SIM900A.print(":");
       SIM900A.print(gps.time.second());
```

```
      SIM900A.print(":");
      SIM900A.println(gps.time.centisecond());
      int16_t adc0; // 16 bits ADC read of input A0
      adc0 = ads.readADC_SingleEnded(0);
      Voltage = (float(adc0) * 0.1875 /15.4);
      Oxy = float (((Voltage *1.22)/.9));

      SIM900A.print("Oxygen Purity:");SIM900A.print(Oxy);SIM900A.print("%");



 SIM900A.println((char)26);// ASCII code of CTRL+Z
 delay(1000);
 Serial.println ("Location has been sent");
 delay(1000);
}


 void RecieveMessage()
{
 Serial.println ("SIM900A Membaca SMS");
 delay (1000);
 SIM900A.println("AT+CNMI=2,2,0,0,0"); // AT Command to receive a live SMS
 delay(1000);
```

```
 }



void MakeCall()

{

  SIM900A.println("ATD+918801058225;"); // ATDxxxxxxxxxx; -- watch out here for semicolon at the end!!

  Serial.println("Calling  "); // print response over serial port

  delay(1000);

}



void RedialCall()

{

  SIM900A.println("ATDL");

  Serial.println("Redialing");

  delay(1000);

}



//void puritysetup(void)

//{

  //ads.setGain(GAIN_SIXTEEN);

  //Serial.begin(9600);

  //ads.begin();

  //pinMode(BUTTON_PIN, INPUT_PULLUP);
```

```
  //display.begin();




//}




void purityloop()

{

     purity();

  }









void purity(void)

{

int16_t adc0; // 16 bits ADC read of input A0

adc0 = ads.readADC_SingleEnded(0);

Voltage = (float(adc0) * 0.1875 /15.4);

Oxy = float ((Voltage - 21)/0.9);

  Serial.print("AIN0: ");
```

```
  Serial.print(adc0);
  Serial.print("\tVoltage: ");
  Serial.println(Voltage, 2);
  Serial.println();
  display.clearDisplay();
  display.setCursor(0,0);
  //display.setTextSize(2);
  display.setTextColor(WHITE);
  display.println("Oxygen Purity:");display.print(Oxy);display.print("%");
  display.display();
  delay(2000);

}

//void oxysetup()

void oxyloop()
{
#if ENABLE_MAX30100
  // Make sure to call update as fast as possible
  pox.update();
  int bpm = 0;
  int spo2 = 0;
  // Asynchronously dump heart rate and oxidation levels to the serial
  // For both, a value of 0 means "invalid"
```

```
  if (millis() - tsLastReport > REPORTING_PERIOD_MS) {
    //Serial.print("Heart rate:");
    bpm = pox.getHeartRate();
    spo2 = pox.getSpO2();
    Serial.println(bpm);
    //Serial.print("bpm / SpO2:");
    Serial.println(spo2);
    //Serial.println("%");
    tsLastReport = millis();
    display_data(bpm, spo2);
  }
#endif
  drawLine(&xPos);
}
void display_data(int bpm, int spo2) {

display.fillRect(0, 18, 127, 15, BLACK);
  //if(bpm ==0 && spo2==0){
 //    display.setTextSize(1);
 // display.setTextColor(WHITE);
  //display.setCursor(0, 18);
  // Display static text
 // display.print("Fingure Out");
  //}
  display.fillRect(0, 18, 127, 15, BLACK);
```

```
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0, 18);
  // Display static text
  display.print("BPM ");
  display.setTextSize(2);
  display.print(bpm);
  display.display();
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(64, 18);
  // Display static text
  display.print("Spo2% ");
  display.setTextSize(2);
  display.println(spo2);
  display.display();
}
void drawLine(int *x_pos) {
  // Draw a single pixel in white
  display.drawPixel(*x_pos, 8, WHITE);
  display.drawPixel((*x_pos)++, 8, WHITE);
  display.drawPixel((*x_pos)++, 8, WHITE);
  display.drawPixel((*x_pos)++, 8, WHITE);
  display.drawPixel((*x_pos), 8, BLACK);  // -----
  //Serial.println(*x_pos);
```

```
  display.fillRect(*x_pos, 0, 31, 16, BLACK);
  display.display();
  //delay(1);
  if (*x_pos >= SCREEN_WIDTH) {
    *x_pos = 0;
  }
}
void heart_beat(int *x_pos) {
  /*********************************************/
  //display.clearDisplay();
  display.fillRect(*x_pos, 0, 30, 15, BLACK);
  // Draw a single pixel in white
  display.drawPixel(*x_pos + 0, 8, WHITE);
  display.drawPixel(*x_pos + 1, 8, WHITE);
  display.drawPixel(*x_pos + 2, 8, WHITE);
  display.drawPixel(*x_pos + 3, 8, WHITE);
  display.drawPixel(*x_pos + 4, 8, BLACK); // -----
  //display.display();
  //delay(1);
  display.drawPixel(*x_pos + 5, 7, WHITE);
  display.drawPixel(*x_pos + 6, 6, WHITE);
  display.drawPixel(*x_pos + 7, 7, WHITE); // .~.
  //display.display();
  //delay(1);
  display.drawPixel(*x_pos + 8, 8, WHITE);
```

```
display.drawPixel(*x_pos + 9, 8, WHITE); // --
//display.display();
//delay(1);
/*****************************************/
display.drawPixel(*x_pos + 10, 8, WHITE);
display.drawPixel(*x_pos + 10, 9, WHITE);
display.drawPixel(*x_pos + 11, 10, WHITE);
display.drawPixel(*x_pos + 11, 11, WHITE);
//display.display();
//delay(1);
/*****************************************/
display.drawPixel(*x_pos + 12, 10, WHITE);
display.drawPixel(*x_pos + 12, 9, WHITE);
display.drawPixel(*x_pos + 12, 8, WHITE);
display.drawPixel(*x_pos + 12, 7, WHITE);
//display.display();
//delay(1);
display.drawPixel(*x_pos + 13, 6, WHITE);
display.drawPixel(*x_pos + 13, 5, WHITE);
display.drawPixel(*x_pos + 13, 4, WHITE);
display.drawPixel(*x_pos + 13, 3, WHITE);
//display.display();
//delay(1);
display.drawPixel(*x_pos + 14, 2, WHITE);
display.drawPixel(*x_pos + 14, 1, WHITE);
```

```
display.drawPixel(*x_pos + 14, 0, WHITE);

display.drawPixel(*x_pos + 14, 0, WHITE);

//display.display();

//delay(1);

/*****************************************/

display.drawPixel(*x_pos + 15, 0, WHITE);

display.drawPixel(*x_pos + 15, 1, WHITE);

display.drawPixel(*x_pos + 15, 2, WHITE);

display.drawPixel(*x_pos + 15, 3, WHITE);

//display.display();

//delay(1);

display.drawPixel(*x_pos + 15, 4, WHITE);

display.drawPixel(*x_pos + 15, 5, WHITE);

display.drawPixel(*x_pos + 16, 6, WHITE);

display.drawPixel(*x_pos + 16, 7, WHITE);

//display.display();

//delay(1);

display.drawPixel(*x_pos + 16, 8, WHITE);

display.drawPixel(*x_pos + 16, 9, WHITE);

display.drawPixel(*x_pos + 16, 10, WHITE);

display.drawPixel(*x_pos + 16, 11, WHITE);

//display.display();

//delay(1);

display.drawPixel(*x_pos + 17, 12, WHITE);

display.drawPixel(*x_pos + 17, 13, WHITE);
```

```
display.drawPixel(*x_pos + 17, 14, WHITE);

display.drawPixel(*x_pos + 17, 15, WHITE);

//display.display();

//delay(1);

display.drawPixel(*x_pos + 18, 15, WHITE);

display.drawPixel(*x_pos + 18, 14, WHITE);

display.drawPixel(*x_pos + 18, 13, WHITE);

display.drawPixel(*x_pos + 18, 12, WHITE);

//display.display();

//delay(1);

display.drawPixel(*x_pos + 19, 11, WHITE);

display.drawPixel(*x_pos + 19, 10, WHITE);

display.drawPixel(*x_pos + 19, 9, WHITE);

display.drawPixel(*x_pos + 19, 8, WHITE);

//display.display();

//delay(1);

/*************************************************/

display.drawPixel(*x_pos + 20, 8, WHITE);

display.drawPixel(*x_pos + 21, 8, WHITE);

//display.display();

//delay(1);

/*************************************************/

display.drawPixel(*x_pos + 22, 7, WHITE);

display.drawPixel(*x_pos + 23, 6, WHITE);

display.drawPixel(*x_pos + 24, 6, WHITE);
```

```
  display.drawPixel(*x_pos + 25, 7, WHITE);
  //display.display();
  //delay(1);
  /*********************************************/
  display.drawPixel(*x_pos + 26, 8, WHITE);
  display.drawPixel(*x_pos + 27, 8, WHITE);
  display.drawPixel(*x_pos + 28, 8, WHITE);
  display.drawPixel(*x_pos + 29, 8, WHITE);
  display.drawPixel(*x_pos + 30, 8, WHITE); // -----
  *x_pos = *x_pos + 30;
  display.display();
  delay(1);
}
```

RESULTS AND OBSERVATIONS

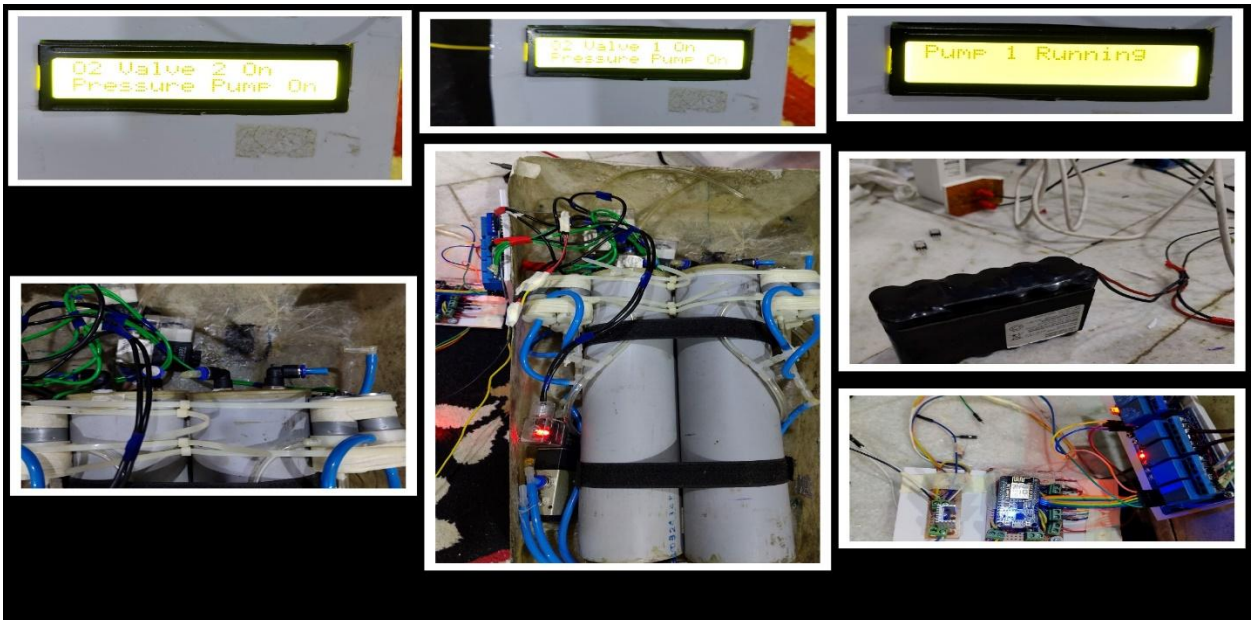


Figure 17: Prototype of the proposed Oxygen Concentrator model

Full Setup:

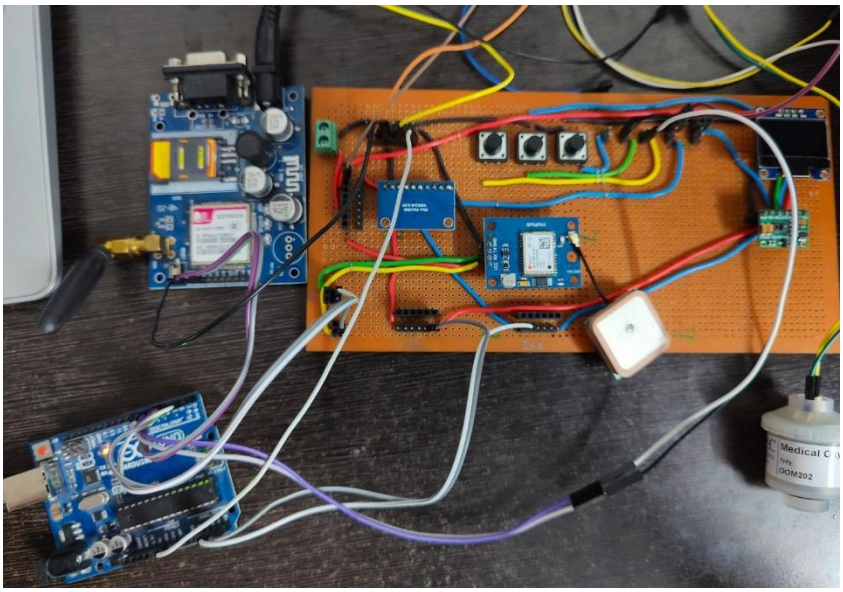
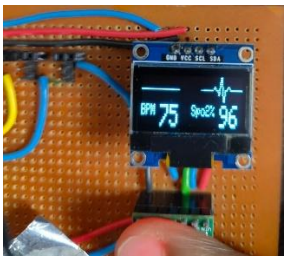
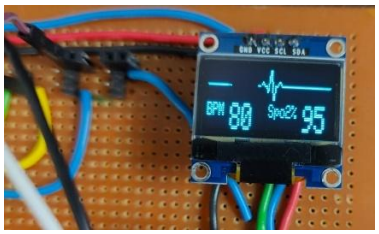


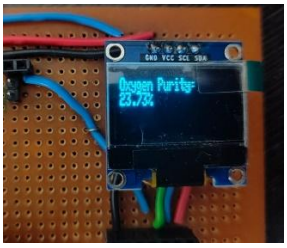
Figure 18: Prototype of the proposed Oxygen Analyzer model

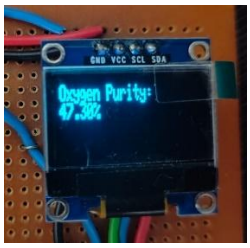
Heart rate using MAX30100 sensor:



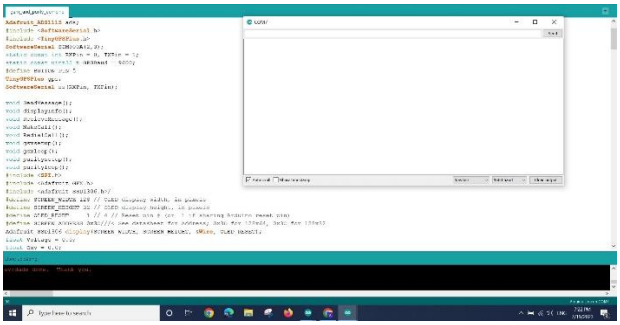
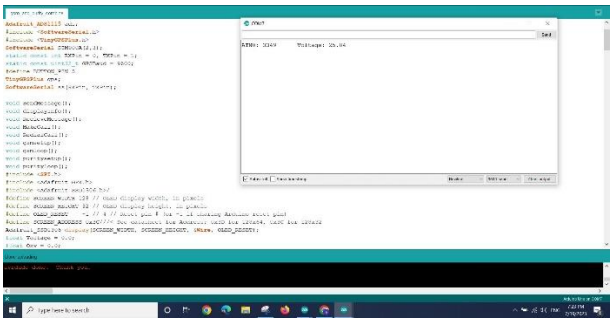
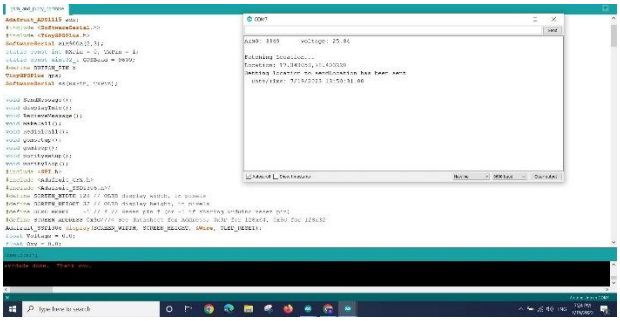
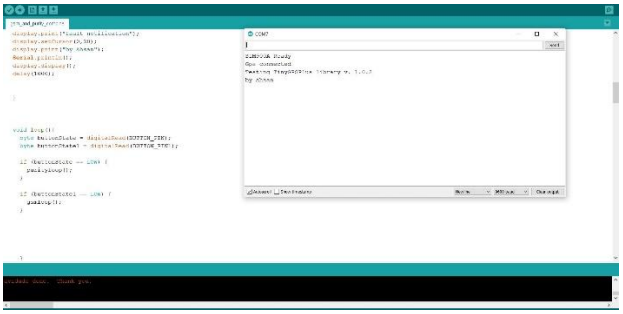


Oxygen purity:

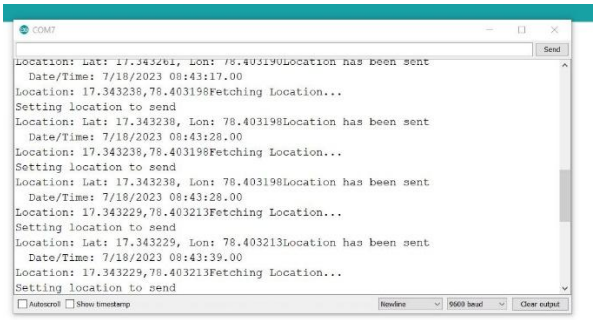


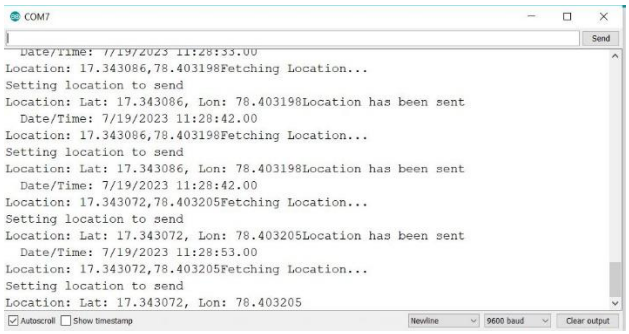


GSM data serial monitor:

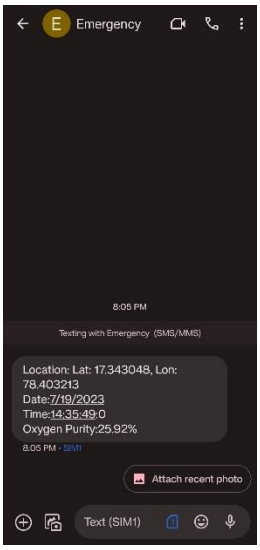





GPS coordinates on serial monitor:





SMS alert receive:



OK

OK

CONCLUSION AND FUTURE SCOPE

CONCLUSION:

The above system is simple, the portable design shown in Figure 2 provided an output of around 9 liters per minute and the larger one provided a little over 100 liters per minute. Both canisters were overdesigned for safety after the blow out of my first PVC canister. Also, I used Zeolite 5A for the portable canister, and it gave me 90.7% pure oxygen -- almost but not medical grade (95% purity). So I switched to Zeolite 13X for the larger implementation, and this got me to 92.5% purity oxygen.

Also, my design uses solenoid valves and designed an automated control system that would provide continuous running at the turn of a switches the pumps on and of continuously.

Finally, please make sure you place the zeolite canisters in a shaded area away from the sun, lest the heat change the purity of the oxygen since temperature also impacts operational efficiency. With the help of 12v/6amps battery the system works about 45 minutes without connecting it to supply directly.

In conclusion, the patent application proposes a remarkable leap forward within the realm of medical devices by introducing an innovative oxygen concentrator and analyzer. The integration of key components such as the Arduino Uno, OOM202 sensor, GPS module, GSM module, OLED display, and Max30100 sensor results in a holistic and dependable solution tailored for oxygen therapy and comprehensive health parameter monitoring. The device's ability to autonomously detect errors and establish remote communication channels signifies a significant stride toward swift maintenance and effective patient management. These attributes collectively position the proposed system as an invaluable asset within contemporary healthcare environments. As technology continues to evolve, this invention embodies a convergence of various fields, fostering enhanced patient care and streamlining healthcare practices.

FUTURE SCOPE:

The new oxygen concentrator and analyzer system, described in the patent application, offers exciting possibilities for the future. As technology advances, there are several ways this system could be improved and expanded. For example, it could learn from past data to predict changes in oxygen levels or health issues, which could help doctors take action early. The system could also connect to faster and more reliable wireless networks, making it easier to share important health information with medical professionals.

Additionally, the device could become smarter by using artificial intelligence to analyze data and give better insights. It might even connect to the internet and store information in the cloud, so doctors can see a complete picture of a patient's health over time. To make things simpler for users, the screen and controls could be made even easier to use. In the future, the device might adapt oxygen therapy based on a patient's needs, or it could connect to virtual doctor visits through a computer or phone.

By adding more sensors, the system could keep track of more health signs, like body temperature or ECG. And as technology gets smaller, the device could become smaller too. But before all these exciting possibilities become reality, the system would need to be tested in real-life situations to make sure it's safe and effective for patients. This might also involve getting approval from medical regulators. Overall, this new system holds great promise for improving healthcare in the years to come.

REFERENCES

1. Salila Ongtrakul , Anyarin Thitiratannapong , Chuchart Pintavirooj , Treesukon Treebupachatsakul “Pressure Swing Absorption Oxygen Concentrator equipped with Remote Monitoring Pulse Oximeter” 13th Biomedical Engineering International Conference (BMEiCON) -2021
2. Soohyun Kim, Hansil Kim, Donghyun Moon “A Controller Design for Oxygen Concentrator” International Conference on Electronics, Information, and Communication (ICEIC) -2020
3. L. Chapman and J. B. Simon, "High-resolution monitoring of weather impacts on infrastructure networks using the Internet of Things", *Bulletin of the Ame* n and J. Suntharalingam, "Long-term oxygen therapy (LTOT)-–Is it always appropriately *rican Meteorological Society*, 2018.
4. S. Shiri, A. Aliverti “Design, implementation and testing of a portable device for multiparametric activity and SpO2 monitoring” IEEE Healthcare Innovations and Point of Care Technologies (HI-POCT)-2017
5. L. Pocero, D. Amaxilatis, G. Mylonas and I. Chatzigiannakis, "Open source IoT meter devices for smart and energy-efficient school buildings", *HardwareX*, vol. 1, pp. 54-67, 2017.
6. R. Galera, R. Casitas, E. Martinez, V. Lores, B. Rojo, C. Carpio, et al., "Exercise oxygen flow titration methods in COPD patients with respiratory failure", *Respiratory Medicine.*, vol. 106, pp. 1544-1550, Jun. 2012.
7. Yuwen Z, Yuyuan W, Jianying G, Jilin Z, The experimental study on the performance of a small-scale oxygen concentration by PSA Separation and Purication Technology. 42(2):123-127, 2005.
8. K.F. Rabe, S. Hurd, A. Anzueto, P.J. Barnes, S.A. Buist and P. Calverley, "Global strategy for the diagnosis management and prevention of chronic obstructive pulmonary disease", *Am J Respir Crit Care Med.*, vol. 176, pp. 532-555, May. 2007.
9. Fauroux B, Howard P, Muir JF, Home treatment for chronic respiratory insuffciency:The situation in Europe in 1992, Eur Respir J 7(9):1721□1726, 1994.
10. https://www.arduino.cc/en/IoT/HomePage
11. https://www.arduino.cc/en/Guide/Nodemcu