2023 年臺灣國際科學展覽會
優勝作品專輯

作品編號　**100042**

參展科別　工程學

作品名稱　**Automated Inflation and Pressure Regulation for Recreational and Professional Cyclists**

得獎獎項　二等獎

國　　家　**South Africa**

就讀學校　**Outeniqua High School**

指導教師

作者姓名　**Johannes Jaco Jacobs**

關鍵詞

作者照片

Introduction

Cycling is a very popular mode of transport as well as a famous sport around the world. Many people enjoy this sport either professionally or recreationally. Cycling in the UK alone has grown up to 200% since lockdown in 2020. (Chandler, 2020)

Cyclists make use of a broad selection of products to enhance their performance. Those products range from wireless gear shifting, advanced geometry, smart suspension. This project is aimed to indicate the importance of tire pressure and to introduce a product which will be able to adjust tire pressure while cycling.

This product will give cyclist an advantage on different terrains as well as eliminate some common problems amongst cyclists.

Flat tires are one of these problems. It occurs commonly amongst cyclists and can happen due to a variety of reasons. Another problem is wrongly inflated tires. This causes unnecessary loss in a cyclist's power and speeds due to the high rolling resistance between the tires and the surface. This then results in losing time whether racing or commuting. In an article published in 2014 in Velonews.com, Lennard Zinn states:

"Whether on tarmac or singletrack, a tire with lower rolling resistance reduces the power required to move forward while also providing a better quality ride. The tire absorbs small bumps by not transferring them into the bicycle and rider, resulting in a smoother ride, faster speeds, and better cornering." (Zinn, 2014)

Taking this in consideration it becomes clear that it is important to develop a system which is able to control tire pressure.

With technology advancing every day, this project raises the question: "Can technology enhance the effectiveness of tire pressure in cycling and thus enhance a cyclist's performance?"

Background

This project is aimed to indicate the importance of tire pressure and to introduce a product which will be able to adjust tire pressure while cycling as well as to eliminate tire related problems such as punctures and wrongly inflated tires.

In the past attempts were made to create a system for a bicycle which could reflate the tires while cycling. For example, in 2012 ADAPTRAC created a system which could deflate and inflate bicycle tires (Coxworth, ADAPTRAC changes mountain bikes' tire pressure on the fly, 2012). This system had a large canister of air mounted on the bicycle with a connecting tube routed through the hub to the tire. Another example comes from WhiteCrow which consisted of a hub that uses a vane pump to move air to and from the tire. (Coxworth, Tire-inflating bike hub could be something to crow about, 2017)

These attempts were very complex and required cyclists to replace existing parts on their bicycles. Furthermore, these attempts used a lot of space in addition to adding a lot of weight.

Recently both Gravaa and Scope have introduced their products which can regulate tire pressure. Similar to WhiteCrow's concept, Gravaa makes use of a pump driven by the rotation of the wheel. [1] This means that a potential buyer will have to replace their existing hub. (Benson, 2021) Scope however makes use of a reservoir mounted on the hub which releases air to the tires on demand. The reservoir has a limited capacity and will need to be reflated once depleted. (Bromley, 2022) Both these products come at an expensive price which makes it unappealing to most cyclists that do not compete on a highly competitive or international level.

What makes this project's prototype and concept different from the aforementioned products is the simplicity, compatibility, cost and ease of reloading.

Reasons for deflating tires:

There are a number of reasons why cyclist's tires lose pressure, this includes punctures by sharp objects, ripped tires, leaks, wear, pinching and burping (Which is caused by compromising the seal of a tubeless tire.) (Taylor, n/a) Most of these can be solved by just reflating the tires. However, this still requires the cyclist to climb of the bike and reflate it by hand. These problems commonly occur when a cyclist is riding rough terrain like roots, rocks or foliage that could contain thorns.

This could be prevented just by avoiding riding over any parts that include these features. According to (Jess, 2020)through the different mountain biking disciplines, professional cyclist tends to average at speeds from 15km/h up to 32km/h. At these speeds it's hard for cyclists to identify some of these features and to avoid them.

Reasons for wrongly inflated tires:

This usually occurs when a cyclist has adjusted their pressure some time before to accommodate new mass, pressure and different terrain. This also shows an effect when a cyclist crosses onto different terrain (example: gravel onto tar roads) that requires a different amount of pressure to perform at its maximum capability.

Both deflated and wrongly inflated tires introduce rolling resistance which is the factor aimed to reduce. Rolling resistance is a factor that slows down a cyclist, it is caused by the type of terrain and its roughness and hardness. This is also the factor which the tire pressure is based on.

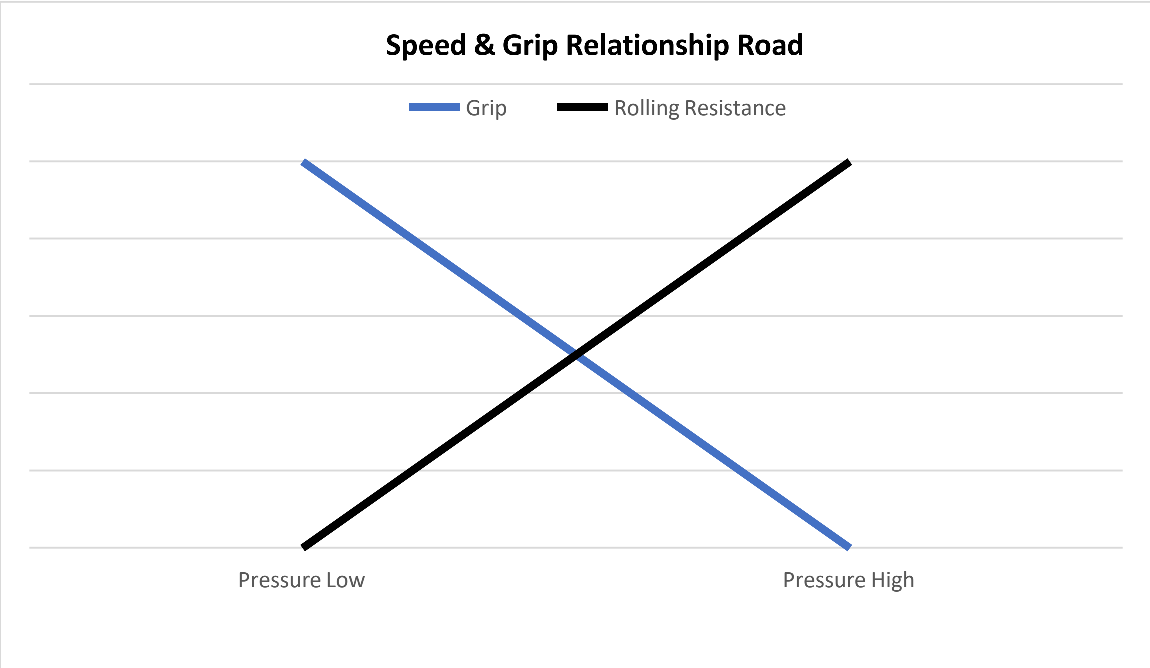
"On a completely smooth surface the following applies: The higher the inflation pressure, the inferior the tire deformation and thus rolling resistance.

[1] Scope and Gravaa's recent introduction still needs to be proven and accepted due to a lot of factors including compatibility, weight, cost and ease of use.

Off road it is exactly the reverse: The lower the inflation pressure, the lower the rolling resistance. This applies equally on hard gravel roads and soft forest tracks.

Explanation: A tire with low inflation pressure can adapt better to a rugged surface. It sinks into the ground less and the whole rotational mass is held back much less by the uneven surface." (Schwalbe, 2020)

The factors that affect rolling resistance are: tire pressure, tire diameter, tire width, tire construction, tire tread (Schwalbe, 2020)



In the diagram these factors have a direct influence on each other. For example: if you want high speed you need higher pressure and therefore will have less grip. Also if you wanted more grip then your pressure would be lower, this then decreases speed.

This is in the case of a flat smooth surface. As Schwalbe stated the exact opposite is true for rough terrain and it's important to find the balance between speed and grip.

Problem Statement and Aim

With a number of contributing factors that affect rolling resistance there is no proven method that is widely used to can control these factors while a cyclist is riding and crossing onto different terrains. Cyclist also commonly experience tire related problems like deflation during training or racing which can lead to them having to get off their bicycle to fix them and waste time that could cost them a position or race lost. Reasons for these problems are described in the above part.

Therefore, cyclists are in need of a method to control these factors and problems to prevent them from causing them to lose time.

The aim of this project is to introduce a new method to advance a cyclist's performance by accessing the relationship between a cyclist's performance and their tire pressure as well serve a dual purpose to prevent tire deflation while not requiring a cyclist to climb off and manually to fix or change their tire pressure or problems.

Another question was what system to use: The system should be able to effectively and accurately control a tires pressure as well as being user friendly. An electrical system was decided on of its wide variety of usage.

Reasons for choosing an electrical system:

- Provides and effective way to monitor and display pressure levels to the user
- Can make use of microcontrollers which can determine when a valve should be opened and for how long
- Can be cross-connected to other devices like phones or other cycling computers
- Provides a way to customize pressure levels according to weight or the cyclist's liking

Engineering and Design goal:

The engineering goal is to design a system to enhance a cyclist's performance through their tire pressure and provide measures to prevent tire deflation. The design will make use of an electrical system which is able to effectively and accurately control a cyclists tire pressure without the need to climb of their bicycle.

The system should take up minimal space on a bicycle while being lightweight and inexpensive.

The system should include:

- Microcontroller
- Measure for Displaying
- Pressure sensor
- CO^2 Cartridge(s)
- Valve(s)

The materials that should be used:

- Recyclable plastics
- Rubber Tubing

Past Concepts:

Swivel Cuff:

Using a Swivel Cuff is a good solution to controlling the air pressure however it will be complicated redesigning the wheel's hub with pipes for redirecting the air through the axle to the wheel's valve. This will also take away the ability to take off the wheel during racing and commuting as taking the wheel off will result in the whole wheel deflating. This will also require a user to replace their current hub which can be inconvenient.

Integrated Valve:

Use of an upside down valve facing inside the tire with a CO2 cartridge connected to it. If the tire gets to deflated the ground will press the valve open resulting in air flowing into the tire. Thus inflating it again. This will work to an extent but if the tire has hit something it will most likely result in the system breaking. Using this will also not give the cyclist control over their pressure as they are unable to adjust the valve.

Prototypes:

Prototype 1 -2020

Goal:

To create a simple mock-up of the basic components for testing the basic features. This mock-up was used to set up the basic design on which new prototypes was expanded on. This aim of this Prototype was to get the display working with the Arduino as well as to incorporate the LED and potentiometer. (LED and potentiometer mimics the valve and pressure sensor)

Components:

- Arduino
- Breadboard
- Potentiometer
- LED
- 2x Buttons
- 2x 1k resistors
- 9v Battery
- 9v Connector

Creating the circuit:

A basic electronic circuit was created to connect all the components to the Arduino. The components were connected as shown in the diagram. The LED was connected to pin 13 and ground of the Arduino. A potentiometer was connected to the Analog 1 pin, ground and 5v. Then both the buttons were connected to the 5v and their grounds to the ground of the Arduino with a 1k ohm resistor separating them. The same pin also supplies an input to the Analog 2 and 3 pins.

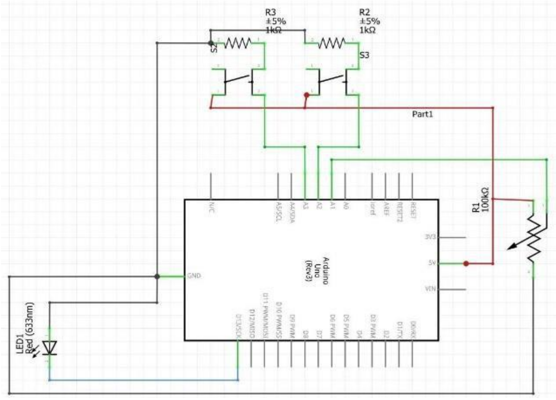


Figure 1 Prototype 1 by Jaco Jacobs 2020

Code:

Firstly, all the variables were declared at the top of the code. Then in the setup the Serial monitor is started at 9600 baud rate. The LED was declared as an output and the buttons as inputs.

A potentiometer then sends its data to the Arduino which is then converted to a range between 0 and 5.

Then in the loop there are if statements that check if and what button was pressed. They then either decrease or increase the pressure range. After this there are another series of if statements to check whether the pressure exceeds the range. If the pressure exceeds the range a LED activates.

(Code in Appendix 1)

What this prototype achieved:

This prototype successfully read the pressure data (represented by the potentiometer's value mapped to a value between 0 and 5) and displayed it on a display for the user. When the buttons were pressed the range was changed up or down as programmed and the LED turned on when exceeding that range. Other than technical achievements, this prototype served as a successful prototype and base to be improved on.

Prototype 2 -2020

Goal:

To integrate Bluetooth capability to allow wireless communication from the cockpit of the bicycle to the wheel. This prototype should also have a display to present the user with live pressure readings from the tires. This prototype also uses a potentiometer and LED instead of the pressure sensor and solenoid valve.

Components Astral:

- •Arduino
- •HC05
- •Mini Breadboard
- •OLED 0,96"
- •2x Buttons
- •2x 1k resistors
- •9v Battery
- •9v Connector

Components Slave:

- •Arduino
- •HC05
- •Mini Breadboard
- •Potentiometer
- •LED
- •9v Battery
- •9v Connector

Creating the circuit:

Prototype 1 was split into two different circuits, Astral and Slave. The Astral circuit was used as the display to provide active feedback of the pressure in the tires and to control the pressure with the use of two buttons. The Slave circuit monitors the pressure in the tires and then adjusts it according to what the Astral circuit has stated.

Astral Circuit:

The Astral circuit's components were connected like the following: Two buttons were connected to the Arduino by the Analog1 and 2 pins, then they were also connected to the ground with a 1k ohm resistor in between. The Bluetooth module was connected to the 5v, ground, TXD and RXD pins of the Arduino. Lastly the OLED was connected to the 5v, ground, analog 4 and analog 5 pins.

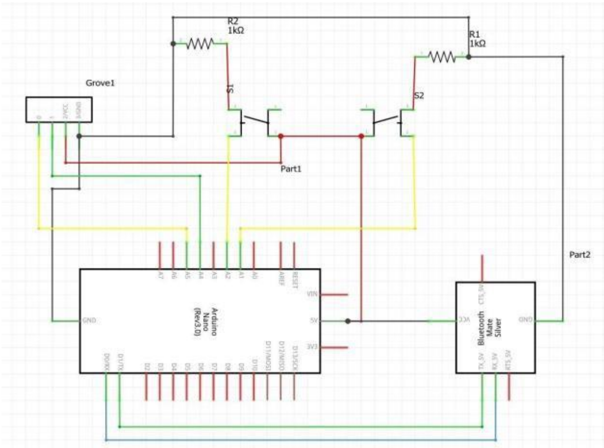


Figure 2 Prototype 2 Astral Circuit by Jaco Jacobs 2020

Slave Circuit:

The Slave circuit was connected as seen below: A LED was connected to the digital 3 and ground pins of the Arduino. The potentiometer was connected to the ground, 5v and analog 1 pins. Lastly the Bluetooth module was connected to the 5V, ground, RXD and TXD pins.

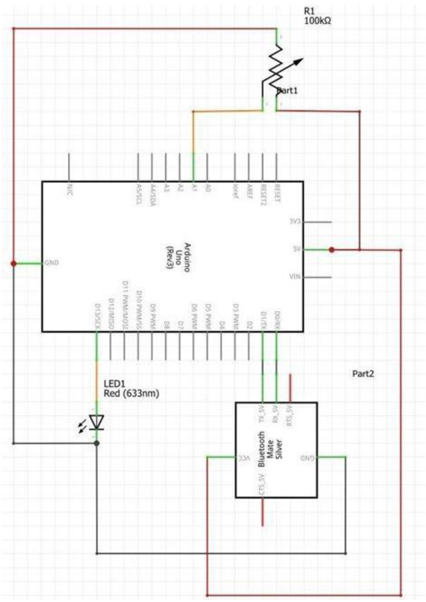


Figure 3 Prototype 2 Slave Circuit by Jaco Jacobs 2020

Code: Astral

Code:

At the top all the different variables were declared:

- dataNumber- stores received data
- buttonPin1- the pin button 1 is connected at
- buttonPin2- the pin button 2 is connected at
- ButtonIsPressed- records if button is pressed
- decreasePressure- sent to Slave to decrease pressure
- increasePressure- sent to Astral to increase pressure
- lastButtonState- records last state of button 1
- lastButtonState1- record last state of button 2
- NumChars- max amount characters
- recievedChars- received characters over Bluetooth
- newData- tests if there is new incoming data

In the setup the serial monitor is started at 9600 baud rate. Then the OLED's parameters are declared. This includes the pins, screen size, font and scroll mode.

In the loop the Arduino tests if there are buttons pressed and then it sends the data to the slave circuit though the Bluetooth module. Then the pressure is received from the Slave through the Bluetooth module and displayed on the OLED.

(Code in Appendix 2.1)

Slave Code:

At the beginning the variables were declared:

- ledPin- pin the LED is connected at
- val- value from the potentiometer
- lowPres- lowest pressure before LED turns on
- highPres- highest pressure before LED turns on
- ledState- LED state on or off

- dataNumber- stores received data
- NumChars- max amount characters
- recievedChars- received characters over Bluetooth
- newData- tests if there is new incoming data

At the setup the pin mode of the LED was declared and the serial monitor was started, at 9600 baud rate. Then in the loop it starts by reading the value from the potentiometer. After this a few if statements check the incoming data from the Astral circuit to the adjust the pressure range. Then there is another if statement to check whether the pressure exceeds the range and then will turn on a LED. The last piece of code checks the incoming data from the Bluetooth module.

(Code in Appendix 2.2)

What this prototype achieved:

The two circuit established a successful connection and were able to send and receive data. The Astral circuit could send data to change the range and the Slave circuit would change the range as intended. The Slave circuit could send the pressure data to the Astral circuit which then successfully displayed it for the user.

This prototype proved that a Bluetooth connection could be made to wirelessly send and receive data to and from the wheels to the cockpit while maintaining full functionality.

Prototype 3 -2021

Goal:

To design a prototype that makes use of solenoid valves. The final prototype is also aimed to be compatible with an Android application This design will remove the Astral, Slave system and only rely on a single electronic system. Doing this will remove the display functionality but opens an option to be connected to other brand cycling computers, mobile phones (as mentioned above) or even fitness watches.

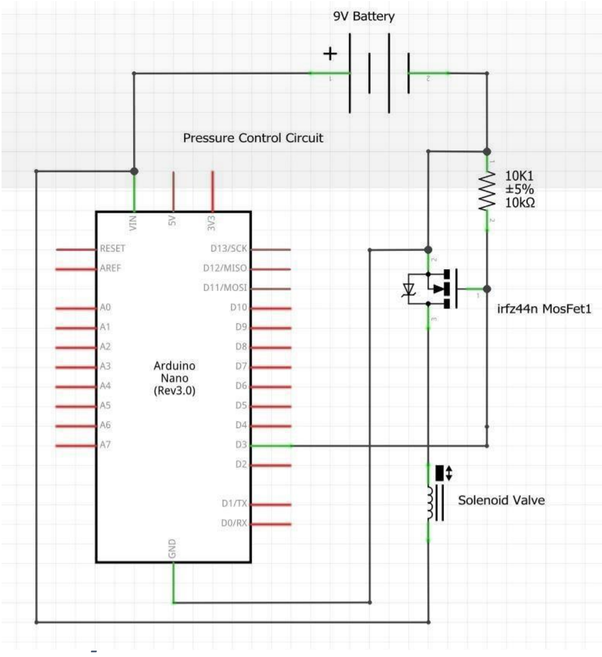
Components:

- Arduino microcontroller
- 9v Battery
- 9v Connector
- 2x MOSFET switches
- 2x Solenoid valves
- 2x 10k ohm resistors
- Potentiometer
- HC05 Bluetooth module

Creating the circuit:

Firstly, a basic circuit similar to prototype 1 was created. This circuit consisted of the Arduino, 9v battery, MOSFET, solenoid valve and 10k ohm resistor.

The components were then connected as seen in the circuit below.



In this circuit a 9v power supply was connected to the Arduino and Solenoid valve. The ground was then also connected to the Arduino. A MOSFET switch was connected between the ground of the power supply, Arduino. The Solenoid valve was then connected to the ground output of the MOSFET.

This circuit was used as a base to expand on with the other components.

Figure 4 Prototype 3 Circuit by Jaco Jacobs 2021

Assembling the Final Circuit:

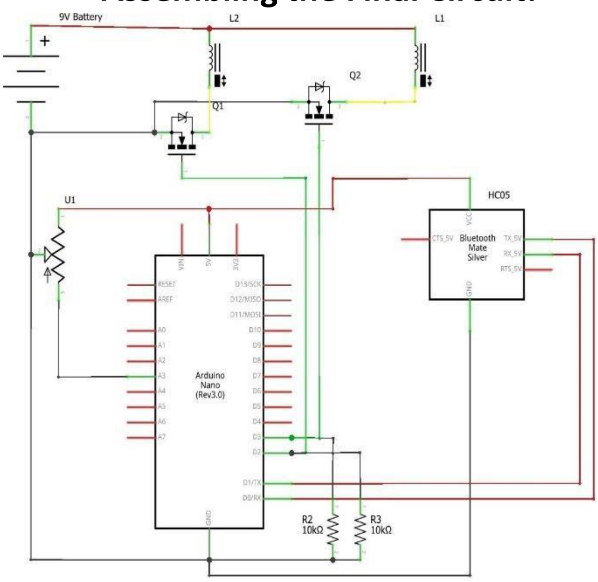


Figure 5 Prototype 3 Final Circuit by Jaco Jacobs 2021

This final circuit consisted of all the before mentioned components. The 9v battery separately powered the solenoid valves while the Arduino and the rest of the components (not related to the solenoid valves) were powered by USB.

The potentiometer was connected to the 5v supply, ground and the analog 3 input pin of the Arduino. The analog pin reads the value from the potentiometer and stores it in a variable. (PressureSensor- Code included in **Appendix 1**)

The MOSFET's source pins were connected to both the Arduino and 9v power supply. A 10k ohm resistor was connected between the gate and source pins.

One of the MOSFET's gate pins was connected to the D2 pin of the Arduino and the other to the D3 pin. The drain pins were then connected to the ground of the solenoid valves. Then the Bluetooth module's ground and VCC pins were connected to the ground and

5v pins of the Arduino. Lastly the Bluetooth module's TXD was connected to the Arduino's RXD and the RXD to the TXD.

Code:

Refer to **Appendix 3**

Firstly, variables had to be declared. These variables included the following:

- PressureSensor- reads the value from the potentiometer
- PressureVal- Converts the value from the potentiometer to a range of 1-5
- Solenoid1- Declares the pin of the solenoid
- Solenoid2- Declares the pin of the solenoid
- Incoming_value- Reads the incoming value from Bluetooth module
- MaxHighPressure- Maximum value the pressure can go
- MaxLowPressure- Minimum value the pressure can go

In the setup the potentiometer was declared as an input and the solenoid valves were declared as outputs. The maximum and minimum pressure was declared as 4 and 2 bar. Then the serial monitor was started at 9600 baud rate. This is to monitor incoming data and outgoing data.

In the loop the Arduino firstly reads the value from the potentiometer and stores it in a variable. Then there is a nested if statement that checks if there is any incoming data. If there is incoming data, it checks whether the date is equal to 'I' or 'D'. When the incoming data is equal to 'I' it increases the MaxHighPressure and MaxLowPressure by 1 bar otherwise if it is equal to 'D' it decreases those values by 1 bar.

After this nested if statement the value from the potentiometer is divided by 204.6 and stored in PressureVal. (The division converts the 0 to 1023 range to 0 to 5) After this there are a series of if and else if statements to check whether the pressure exceeds either the MaxHighPressure or MaxLowPressure. One solenoid opens when the pressure exceeds the MaxHighPressure and the other when it exceeds the MaxLowPressure. If the pressure is in the range both the solenoids are closed.

Lastly the Arduino prints the PressureVal to the serial monitor for an android application to receive.

Creating the Android Application:

The application was designed to have three buttons for connecting the device and increasing or decreasing the pressure, a display for the pressure and range. The user interface can be seen below. To connect the device, the user presses the “Connect Device” button. It opens another screen that display all connected Bluetooth devices. Then the user will be presented with the options. The user the selects the device which is named “HC05”. To increase or decrease the pressure the user simply selects the corresponding button.

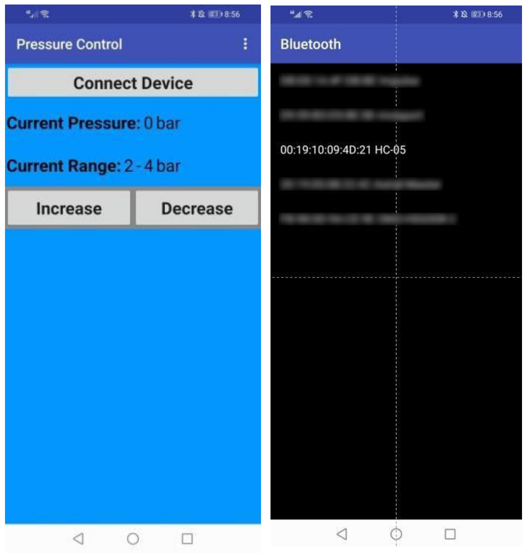


Figure 6 App user interface by Jaco Jacobs 2021

Code:

The first part of the code is a list picker that when pressed displays the available Bluetooth devices. Then when the device is selected the Android phone connects to the device. The buttons when pressed send either “I” or “D”. “I” increases the pressure and “D” decreases it while also changing the range.

Lastly the code reads incoming data, the pressure, from the device and displays it.

What this prototype achieved:

This prototype proved that this circuit is able to incorporate solenoid valves to add or remove pressure from the tires. Furthermore, this prototype also successfully demonstrated that the circuit could be paired with an Android application to expand the compatibility the circuit has to external devices.

Prototype 4 -2022

Goal:

To design and build a prototype that can be mounted onto the bicycle to add air into the tire from a canister of CO_2.

Components

- Raspberry Pi microcontroller
- HC05 Bluetooth Module
- Servo motor
- 5V power supply(Battery)
- CO_2 cartridge
- CO_2 connector
- 3D printed gears and housing

Design:

Air release valve:

The air release valve was designed to work with a servo motor. This was used because the servo and valve are able to resist the pressure from the canister better than a solenoid valve of the same size. A 3D printed assembly was used to connect the valve and the servo together with the gears used to rotate the valve. The servo was then directly connected to the Raspberry Pi.

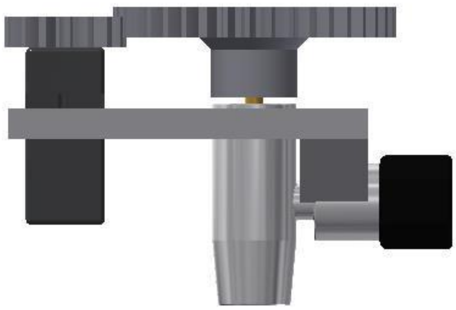


Figure 7 Prototype 4 Release Valve by Jaco Jacobs 2022

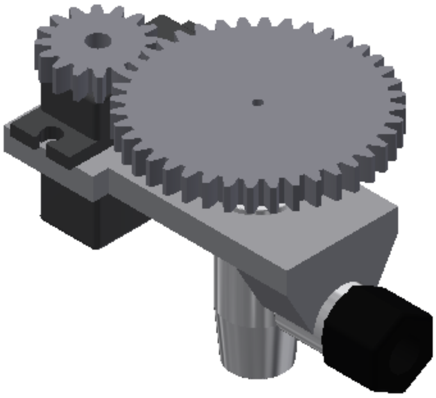


Figure 8 Prototype 4 Release Valve by Jaco Jacobs 2022

Electronics and Battery Housing:

The housing was designed with a low profile in mind. The housing fits the Raspberry Pi microcontroller, 3.7v lipo, charging circuit, Bluetooth module and pins to connect the servo motor.

The housing then fits onto the hub mount which is later fitted to the hub with cable ties.

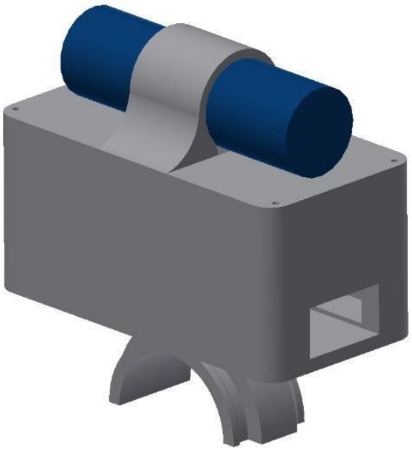


Figure 9 Prototype 4 Housing by Jaco Jacobs 2022

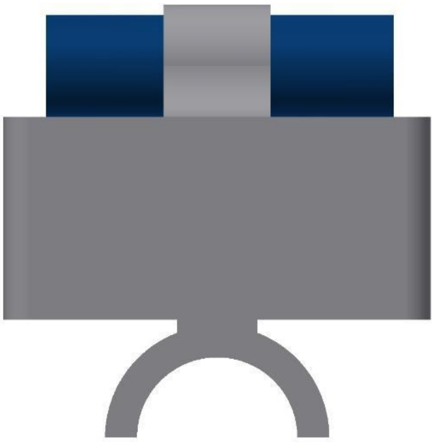


Figure 10 Prototype 4 Housing by Jaco Jacobs 2022

The Assembly:

The assembly uses a clamp to secure it to the hub. It then serves as a mounting point for all the components.

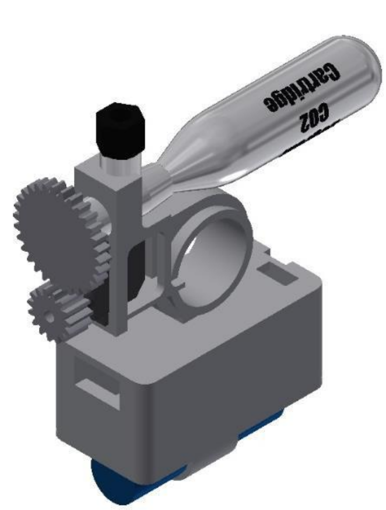


Figure 11 Prototype 4 Assembly by Jaco Jacobs 2022

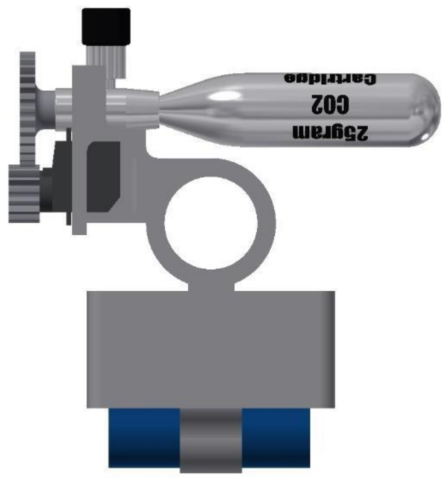


Figure 12 Prototype 4 Assembly by Jaco Jacobs 2022

The Circuit:

All the components are connected to a PCB where the Raspberry Pi is situated. The Raspberry Pi is powered by the power supply which then distributes power towards the rest of the components. The rest of the components include the Bluetooth module, connected to the RX and TX pins, and the servo connected to the GP2 output pin.

Android Application:

The same application mentioned in the previous prototype is still being used for this prototype.

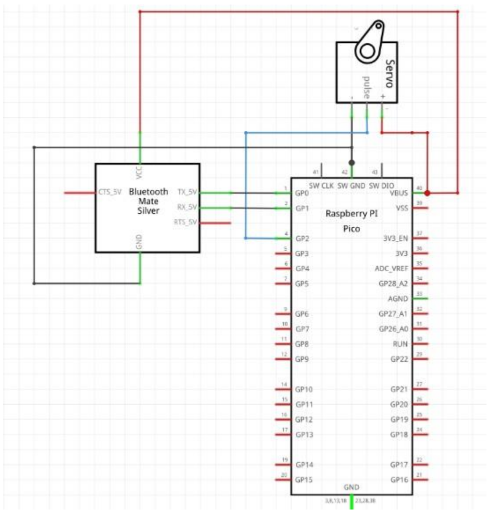


Figure 9 Prototype 4 Circuit by Jaco Jacobs 2022

Code:

The basic code consists of if statement that continuously looks through the UART for incoming data. If the incoming data matches the statement, for instance 'I' for increase, it then opens the valve through the servo.

```
from machine import Pin, PWM, ADC, UART
import utime

uart = UART(0,9600)

servo = PWM(Pin(2))

servo.freq(50)

servo.duty_u16(8200)

while True:


    if uart.any():
        command = uart.readline()
        print(command)
        if command == b'I':
            servo.duty_u16(1350)
            utime.sleep(0.5)
            servo.duty_u16(8200)
```

Figure 10 Prototype 4 Raspberry Pi Code by Jaco Jacobs 2022

Limitations with Prototype 4:

One of the main limitations with this prototype is the amount of CO_2 it can contain. The current design can fit a 25g cartridge that could inflate an average bicycle tire while having some CO_2 to spare. As this will be used to change the pressure more than once it is more suitable to use a larger capacity cartridge or more than one cartridge. This is however limited to the space available on the hub as electronics and a battery also needs to be fitted.

Prototype 5.1 -2022

Goal:

Build and design a prototype that is mountable to the bicycle hub and that can add, remove and monitor pressure in the tire.

Components

- Arduino Nano
- HC05 Bluetooth Module
- Servo motor
- Solenoid Valve
- Pressure Sensor
- 9V power supply(Battery)
- CO_2 cartridge
- CO_2 connector
- 3D printed gears and housing
- Rubber tubing and connectors

Design:

The Holder:

The holder is designed by utilizing a central skeleton where the electronics, gears and valves are fitted. The components are held together by fitting two shells over the components. The shells are then held together by two plates on the sides.

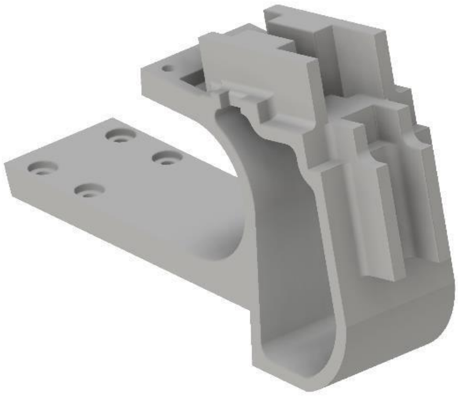


Figure 11- Prototype 5 by Jaco Jacobs 2022

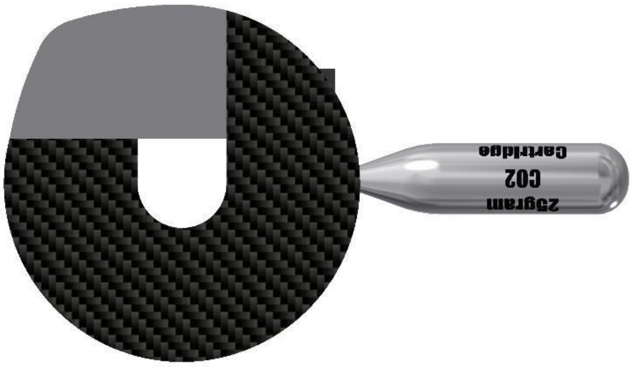


Figure 12- Prototype 5 by Jaco Jacobs 2022

The pneumatic assembly:

For the pneumatics the tire, solenoid and CO2 connector were fitted to a T shaped connector.

The Circuit:

The prototype is designed around the Arduino microcontroller which distributes the power and controls the inputs and outputs.

Android Application:

The same application mentioned in the previous prototypes is still being used for this prototype.

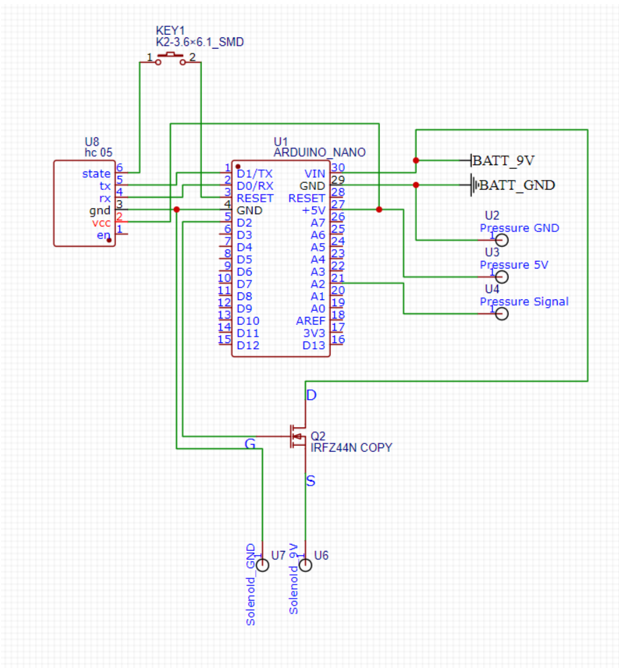


Figure 13- Prototype 5 Circuit by Jaco Jacobs 2022

Code:
The basis of the code consists of a Bluetooth serial monitor which is able to detect incoming data. When incoming data is detected it compares the data against specifications to process if an action will be taken. The current code is written to demonstrate the prototypes functionality. Thus it only sends pressure data to the phone and receives data to open the servo or solenoid valve. **Code included in appendix 4.**

Limitations with Prototype 5.1:
This prototype has similar limitations to prototype 4. The amount of CO2 is still limited by the amount of cartridges that can be fitted onto the prototype. Furthermore, there are other limitations with regards to the size of the components, electronics and connectors used.

Errors with Prototype 5.1
A problematic valve caused air to leak out when a cartridge was fitted.

Figure 14- Prototype 5 by Jaco Jacobs 2022

Prototype 5.2 -2022

Goal:

Fix the error in Prototype 5.1 and update the application and programming.

Components

All the components of the previous version of the prototype was used.

Android Application:

The android application was updated to supply a more simplified design.

Figure 16- Main Screen by Jaco Jacobs 2022

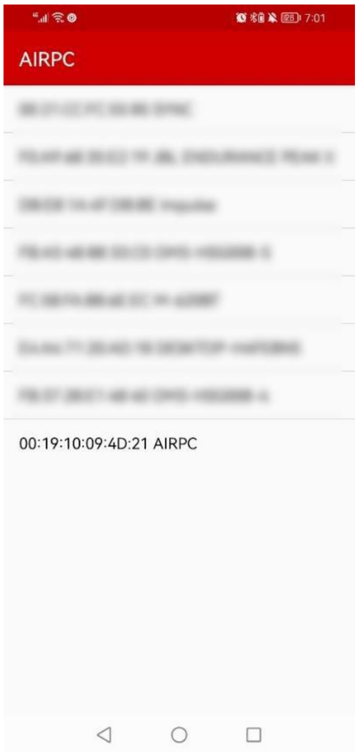


Figure 15- Bluetooth selection Screen by Jaco Jacobs 2022

Code:

Prototype 5.1's code was updated to be used while cycling. This included adding the function to inflate the tires when they are deflating. The pressure is read from the pressure sensor. When the pressure changes to lower than the limit the servo controlled valve opens. When the pressure changes to above the limit the solenoid valve opens.

Code included in appendix 5.

Limitations with Prototype 5.2:

This prototype shares the same limitations that were presented with prototype 5.1.

Proposed Future Design:

Purpose of the Design:

The purpose of the design is to propose a possible future product. This design is designed with modern manufacturing and production in mind. Thus this design will use custom parts such as CO2 adaptors and motor driven valves. The recommendations for future research ties in directly with this as most of them can be incorporated to create a feature full product for a cyclist's every ride. The limitations could also be taken in reference here as some of them are able to be exploited by professionals in their field.

Design Idea:

The proposed design will consist of a clamp that clamps onto the hub of a bicycle wheel. The clamp will serve as an enclosure for the electronics. This includes the microcontroller, battery, valves etc. Two or more CO2 cartridges can be screwed onto the mount to provide air when needed. On the valve of the wheel there will be a pressure sensor mounted. A release valve could optionally be mounted there if there is sufficient space available and it does not add too much weight to offset the wheel's balance.

Figure 17 Proposed Design by Jaco Jacobs 2022

Design Process:

Components:

- Custom circuit board
- Battery
- 2x Motor Driven Valves
- 2x CO2 Adaptors
- 2x CO2 Cartridges
- Rubber Inserts
- Pressure sensor

Figure 16 Proposed Design by Jaco Jacobs 2022

CO2 Network

The CO2 network consists of all the components that have to do with controlling and monitoring the airflow. This includes the CO2-Cartridges, -adaptors, valves, pressure sensor and pipes connecting everything. The 2 CO2 Cartridges are connected through the same pipe towards the inlet and outlet valve. The inlet valves allow the air to pass into the tire and the outlet valves allow the air to escape from the tire. The CO2 is then connected near the valve to allow pressure data to be read.

Electronics:

The electronics consists of a custom circuit board that contains the microcontroller. The motors are then connected to the circuit board. The battery is mounted on an external slider which will allow it to be removed and replaced by a charged battery or just to charge it. The battery's terminals then pass through the other half when connected to connect to the circuit board. The circuit board will contain the microcontroller, motor controller, Bluetooth module etc.

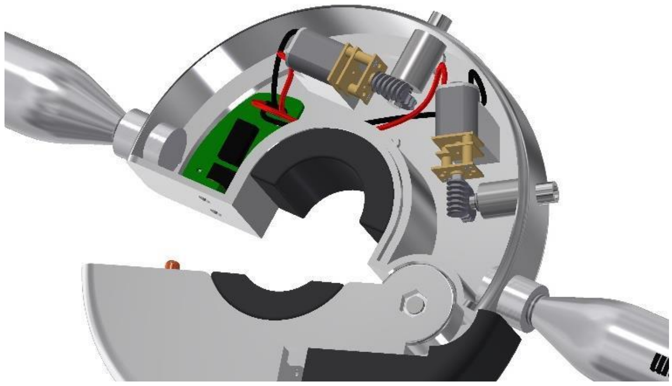


Figure 17 Proposed Design by Jaco Jacobs 2022

Compatibility:

This design is designed with compatibility in mind. This means it should be able to fit on any bicycle's hub. Rubber inserts can be found where the device is mounted onto the hub. These inserts will come in different sizes: wide, narrow and tapered.

Figure 18 Proposed Design Rubber Inserts by Jaco Jacobs 2022

Variables:

The variables in this project that affect the prototypes and proposed design are the surface type and the pressure in the tires.

Surface Type

Surface type is a variable that changes regularly as a cyclist is riding. The surface changes while riding as a cyclist crosses between different types of terrain. For instance, a cyclist crosses from a dirt road onto a paved road and then onto a gravel road. All these terrain's surfaces and resistance differ dramatically from each other.

Tire Pressure

Tire pressure is a very important variable in the cycling sport. Different pressure on terrain affects the rolling resistance of a tire over that surface.

Riding on a smooth, hard terrain with low pressure will result in lower speed, whereas riding with a higher pressure will result in higher speeds. But then riding on a soft rocky terrain with high pressure will result in slower speed rather than lower pressure. Too low pressure on rocky terrain can also result in rocks denting the wheel frame causing leaks that negatively affect the tire pressure.

These variables all come to play while cycling and being able to control it will give a cyclist the upper hand while competing or commuting.

Limitations:

The following limitations were identified:

- Size of electrical and mechanical components will heavily influence the overall size.
- The size of the CO cartridge will affect the amount of CO_2 able to use to inflate the tire.
- Compatibleness with other brand cycling computers will be dependent on their manufacturers choice to collaborate with the product.

Recommendations for Future Product:

Features:

1. A single axis acceleration sensor could be added to the product to measure the vertical movement of the bicycle. This could then be used to detect the roughness of the terrain to then adjust the pressure to compensate for a smoother ride.
2. By including use of a satellite service the product can predetermine when and where the pressure can be adjusted and also when is the best time to do so. If only a certain amount of CO_2 is left the product can calculate when to use it with the most benefit.

Conclusion:

In all cycling disciplines what separates all the cyclists are their strategy and advantages. A top cyclist will always leverage any factor to be better than the other. These factors range from their bicycle and gear to their eating habits. Thus it’s crucial in the cycling industry to create products to advance a cyclist’s performance.

This project introduced a method to improve a cyclist’s performance that will significantly benefit any cyclist while training or racing. By taking the factors that negatively affect a cyclist’s performance and creating new ways to bypass or solve them has already and will still revolutionize cycling as a whole moving forward. In this project it’s shown that by just controlling a cyclist’s tire pressure a cyclist’s speed can significantly better to give them the edge over a competitor. And giving a cyclist full control over their tire pressure while riding and the ability to add or remove pressure other problems like tire deflation also get solved.

The reason this project was initiated was to see problems that slow us down and to solve them while paving a path to further advance cycling as a whole. Thus it’s important for a person to see these opportunities for improvement and to act on it.

Acknowledgements

Funding: Johan Jacobs; JDS Design Studio
Overseeing and Advice: Yvette Pitts: Outeniqua High School

References

Benson, C. (2021, August 9). *Gravaa KAPS teases lightweight, on-the-fly tire pressure adjustment for gravel & XC*. Retrieved from Bikerumor!: https://bikerumor.com/gravaa-kaps-teases-lightweight-on-the-fly-tire-pressure-adjustment-for-gravel-xc/

Bromley, S. v. (2022, April 12). *Wireless €3,998 tyre pressure control system to be used at Paris-Roubaix*. Retrieved from Bikeradar: https://www.bikeradar.com/news/wireless-e3998-tyre-pressure-control-system-to-be-used-at-paris-roubaix/

Chandler, M. (2020, November 25). *The 2020 Bike Boom: How Much Did Cycling Numbers Increase + Will it Continue?* Retrieved from Discerning Cyclist : https://discerningcyclist.com/bike-boom-numbers-statistics/

Coxworth, B. (2012, April 17). *ADAPTRAC changes mountain bikes' tire pressure on the fly*. Retrieved from New Atlas: https://newatlas.com/adaptrac-mountain-bike-tire-pressure/22190/?itm_source=newatlas&itm_medium=article-body

Coxworth, B. (2017, January 05). *Tire-inflating bike hub could be something to crow about*. Retrieved from New Atlas: https://newatlas.com/whitecrow-tire-inflating-hub/47250/

Jess. (2020, February 5). *HOW FAST DOES A MOUNTAIN BIKE GO?* Retrieved from Pedal Chile: https://pedalchile.com/blog/mtb-speed

Schwalbe. (2020, n.d n.d). *Rolling Resistance*. Retrieved from Schwalbe: https://www.schwalbe.com/en/rollwiderstand

Taylor, P. (n/a, n/a n/a). *Why Does My Bike Tire Keep Going Flat?* Retrieved from Bicycle Universe: https://bicycleuniverse.com/why-does-my-bike-tire-keep-going-flat/

Zinn, L. (2014, December 5). *Why Does My Bike Tire Keep Goeing Flat>*. Retrieved from Bicycle Universe: https://www.velonews.com/gear/resistance-futile-tire-pressure-width-affect-rolling-resistance/

Appendix:

Appendix 1:

```
#include<LiquidCrystal_I2C.h>
#include <Wire.h>



LiquidCrystal_I2C lcd(0x3f, 16, 2);

int potpin = A0; int val; int ledPin = 9; int buttonPin1
= 2; int buttonPin2 = 3;
int decreasePressure = 0; int increasePressure = 0;
int lastButtonState = 0; int lastButtonState1 = 0;
int lowPres = 72; int highPres = 144;

void setup() {
  Serial.begin(9600);   pinMode(ledPin, OUTPUT);
pinMode(buttonPin1, INPUT);
pinMode(buttonPin2, INPUT);
}

void loop() {
  decreasePressure = digitalRead(buttonPin1);

  if (decreasePressure != lastButtonState) {
    if (decreasePressure == HIGH) {
Serial.println("Worked");          lowPres = lowPres - 36;
highPres = highPres -36;
    } else {
      Serial.println("Waiting...");
    }
    delay(5);       lastButtonState = decreasePressure;

  }

  increasePressure = digitalRead(buttonPin2);

if (increasePressure != lastButtonState1) {
    if (increasePressure == HIGH) {
Serial.println("Worked");          lowPres = lowPres + 36;
highPres = highPres + 36;
    } else {
      Serial.println("Waiting...");
    }
    delay(5);
      lastButtonState1 = increasePressure;
  }
  val = analogRead(potpin);   val = map(val, 0, 1023,
0, 180);



  if ((val >= lowPres ) && (val <= highPres ))
  {
    digitalWrite(ledPin, LOW);
  }
  if ((val >= 0 ) && (val <= lowPres ))
  {
    digitalWrite(ledPin, HIGH);
  }
  if ((val >= highPres ) && (val <= 180 ))
  {
    digitalWrite(ledPin, HIGH);
  }

  int sensorValue = analogRead(A0);
Serial.println(sensorValue);   delay(1);
  float pressure = sensorValue * (5.0 / 1023.0);
lcd.begin();   lcd.backlight();   lcd.setCursor(3, 0);
lcd.print("A.I.R.P.C");   lcd.setCursor(0, 1);
lcd.print("Pressure=");   lcd.setCursor(9, 1);
lcd.print(pressure);
  lcd.setCursor(13, 1);   lcd.print("bar");

  delay(10);
}
```

Appendix 2:

2.1:

```
#include <Wire.h>
#include "SSD1306Ascii.h"
#include "SSD1306AsciiWire.h"

#define I2C_ADDRESS 0x3C
#define RST_PIN -1

int dataNumber;
int buttonPin1 = 2; int buttonPin2 = 3;
int ButtonisPressed; int decreasePressure = 0; int
increasePressure = 0;
int lastButtonState = 0; int lastButtonState1 = 0;

SSD1306AsciiWire oled;

const byte numChars = 32; char
receivedChars[numChars];

boolean newData = false;

int blank = 0; // Count of blank lines. int count = 0; //
Count of displayed lines. int dir = 1;   // Scroll
direction. uint32_t scrollTime = 0;

void setup() { Wire.begin();
   Wire.setClock(400000L);
   Serial.begin(9600);
   Serial.println("<Arduino is ready>");

   #if RST_PIN >= 0
  oled.begin(&Adafruit128x64,    I2C_ADDRESS,
RST_PIN);
#else // RST_PIN >= 0   oled.begin(&Adafruit128x64,
I2C_ADDRESS);
#endif // RST_PIN >= 0

  oled.setFont(System5x7);
  oled.setScrollMode(SCROLL_MODE_AUTO);
  oled.print("Airpc");  delay(100);
  oled.setRow(oled.displayRows() - oled.fontRows());
  }
void loop() {     recvWithEndMarker();
showNewNumber();

   decreasePressure = digitalRead(buttonPin1);

  if (decreasePressure != lastButtonState) {
    if (decreasePressure == HIGH) {
     Serial.print("1");

  if (increasePressure != lastButtonState1) {
    if (increasePressure == HIGH) {
    Serial.print("2");
     Serial.println(',');
    }     else {
    }
    delay(5);     lastButtonState1 = increasePressure;
  }
delay(1);

     if (!oled.scrollIsSynced()) {     uint32_t now =
millis();     if ((now - scrollTime) >= 15) {      // Scroll
display window.      oled.scrollDisplay(dir);
scrollTime = now;
    }
  } else if ((millis() - scrollTime) > 15) {     if
(blank*oled.fontRows() > oled.displayRows()) {
blank = 0;
     count = 0;
     dir = -dir;      if (dir > 0) {        if (oled.magFactor()
== 1) {          oled.set2X();         } else {
oled.set1X();
       }
     }
     oled.setCursor(0, dir < 0 ? 0 : oled.displayRows() -
oled.fontRows());
    }
    oled.scrollMemory(dir*oled.fontRows());
oled.setCol(0);
    if (count*oled.fontRows() <= oled.displayRows()) {
oled.print(dataNumber);
    } else {      blank++;
    }
    oled.clearToEOL();
  }
}

void recvWithEndMarker() {     static byte ndx = 0;
char endMarker = '\n';     char rc;

    if (Serial.available() > 0) {        rc = Serial.read();

      if (rc != endMarker) {
         receivedChars[ndx] = rc;            ndx++;
if (ndx >= numChars) {               ndx = numChars - 1;
          }        }        else {           receivedChars[ndx] =
'\0';            ndx = 0;            newData = true;
       }
    }
```

```
    Serial.println(',');
  }   else {
  }
  delay(5);
  lastButtonState = decreasePressure;
 }
 increasePressure = digitalRead(buttonPin2);
```

```
}

void showNewNumber() {     if (newData == true) {
dataNumber = 0;          dataNumber =
atoi(receivedChars);          newData = false;
   }
}
```

2.2

```
int state; int ledPin = 13; int val; int lowPres
= 72; int highPres = 144; int sensorValue;
int ledstate;
int dataNumber;

const byte numChars = 32; char
receivedChars[numChars];
boolean newData = false;


void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  Serial.begin(9600);
  Serial.println("<Arduino is ready>");
}

void loop() {   recvWithEndMarker();
showNewNumber();
  val = analogRead(A0);   val = map(val, 0,
1023, 0, 180);

  if (dataNumber == 1) {     lowPres =
lowPres - 36;     highPres = highPres - 36;
  }
  if (dataNumber == 2) {     lowPres =
lowPres + 36;     highPres = highPres + 36;
  }
  if ((val >= lowPres ) && (val <= highPres ))
  {

    digitalWrite(ledPin, LOW);
  }
 else{ digitalWrite(ledPin, HIGH); }
  Serial.print(val);
  Serial.println(",");
```

```
  dataNumber = 0;
}

void recvWithEndMarker() {   static byte
ndx = 0;   char endMarker = '\n';   char rc;

  if (Serial.available() > 0) {     rc =
Serial.read();

    if (rc != endMarker) {
receivedChars[ndx] = rc;       ndx++;       if
(ndx >= numChars) {         ndx = numChars -
1;
      }    }    else {      receivedChars[ndx] =
'\0';       ndx = 0;       newData = true;
    }
  }
}
void showNewNumber() {   if (newData ==
true) {     dataNumber =
atoi(receivedChars);       newData = false;
  }
}
```

Appendix 3:

```
int PressureSensor = 0;
int PressureVal = 0;
int Solenoid1 = 4;
int Solenoid2 = 5;
char Incoming_value = 0;

float MaxHighPressure = 0;
float MaxLowPressure = 0;

void setup() {
pinMode(PressureSensor, INPUT);
   pinMode(Solenoid1, OUTPUT);
  pinMode(Solenoid2, OUTPUT);

  MaxHighPressure = 4;
  MaxLowPressure = 2;


  Serial.begin(9600);
}

void loop() {
  PressureSensor = analogRead(A3);

  if(Serial.available() > 0)
  {
    Incoming_value = Serial.read();
    //Serial.print(Incoming_value);
    //Serial.print("\n");
 if(Incoming_value == 'I'){
      MaxHighPressure = MaxHighPressure +
1;
      MaxLowPressure = MaxLowPressure +
1;
    }
    else if(Incoming_value == 'D'){
      MaxHighPressure = MaxHighPressure -
1;
      MaxLowPressure = MaxLowPressure - 1;
    }
  }

  PressureVal = PressureSensor/204,6;
if (PressureVal >= MaxHighPressure){
digitalWrite(Solenoid1, HIGH);
digitalWrite(Solenoid2, LOW);
delay(100);
  }
  else if (PressureVal <= MaxLowPressure){
 digitalWrite(Solenoid2, HIGH);
 digitalWrite(Solenoid1, LOW);
delay(100);
  }   else {
digitalWrite(Solenoid2, LOW);
digitalWrite(Solenoid1, LOW);
  }
  Serial.print(" ");
  Serial.print(PressureVal);   delay(1000);
}
```

Appendix 4:

```
#include <Servo.h>

int PressurePin = A0;
float Pressure = 0;
int Solenoid = 3;
Servo ServoMotor;
int pos = 180;
char Incoming_value = 0;
float SetPressure = 100;

void setup() {
  pinMode(PressurePin, INPUT);
  pinMode(Solenoid, OUTPUT);
  ServoMotor.attach(4);
  ServoMotor.write(pos);

  Pressure = 100;
  Serial.begin(115200);
}
```

```
void loop() {
  Pressure =
map(analogRead(PressurePin),0,1024,-
100,300);

  if(Serial.available() > 0){
    Incoming_value = Serial.read();
    if(Incoming_value == 'I'){
    SetPressure = SetPressure + 50;
    digitalWrite(Solenoid,HIGH);
    delay(100);
    digitalWrite(Solenoid,LOW);
    }
    else if(Incoming_value == 'D'){
      SetPressure = SetPressure - 50;
      ServoMotor.write(100);
      delay(100);
      ServoMotor.write(pos);
    }
  }
  Serial.println(Pressure);
  delay(100);
}
```

Appendix 5:

```
#include <Servo.h>

Servo valve;

int PressurePin = A5;
int SolenoidPin = 8;

float Pressure = 0;

float HighPressure = 0;
float LowPressure = 0;

char Incoming_value = 0;

void setup() {
  valve.attach(4);
  valve.write(180);

  pinMode(PressurePin, INPUT);
  pinMode(LED_BUILTIN, OUTPUT);

  SetPressure =
map(analogRead(PressurePin),0,1024,-
100,300);
  HighPressure =
map(analogRead(PressurePin),0,1024,-
100,300) + 500;
  LowPressure =
map(analogRead(PressurePin),0,1024,-
100,300) - 500;


  Serial.begin(115200);
}

void loop() {
  Pressure =
map(analogRead(PressurePin),0,1024,-
100,300);

  if(Serial.available() > 0){
    Incoming_value = Serial.read();
    if(Incoming_value == 'I'){
      delay(200);
      valve.write(180);
      HighPressure = HighPressure + 50;
      LowPressure = LowPressure + 50;
    }
    else if(Incoming_value == 'D'){
      digitalWrite(PressurePin, HIGH);
      delay(200);
      digitalWrite(PressurePin, LOW);
      HighPressure = HighPressure + 50;
      LowPressure = LowPressure + 50;
    }
  }

  if (Pressure < (LowPressure - 15)){
        digitalWrite(PressurePin, HIGH);
        delay(200);
        digitalWrite(PressurePin, LOW);
  }else{
    digitalWrite(PressurePin, LOW);
  }



  Serial.println(Pressure/100);
  delay(500);
}
```

```
valve.write(90)
```

【評語】100042

1. The work presents a comprehensive and complete design for the adjustment of the tire pressure for a better adaptation to road friction and roughness in bicycles. This project aims to indicate the importance of tire pressure and to introduce a product which will be able to adjust tire pressure while cycling as well as to eliminate tire related problems such as punctures and wrongly inflated tires.
2. In all cycling disciplines what separates all the cyclists are their strategy and advantages. A top cyclist will always leverage any factor to be better than the other. These factors range from their bicycle and gear to their eating habits. Thus it's crucial in the cycling industry to create products to advance a cyclist's performance.
3. This project introduced a method to improve a cyclist's performance that will significantly benefit any cyclist while training or racing. By taking the factors that negatively affect a cyclist's performance and creating new ways to bypass or solve them has already and will still revolutionize cycling as a whole moving forward. In this project it's shown that by just controlling a cyclist's tire pressure a cyclist's speed can significantly better to give

them the edge over a competitor. And giving a cyclist full control over their tire pressure while riding and the ability to add or remove pressure other problems like tire deflation also get solved.

4. The principle，design，and implementation are well presented. It is recommended to provide some application data in practice to verify the design. Also，a friendly human interface/interaction is recommended.