

2117071

by Benjamin Naylor

MARK AGREED OF 63%

nd

Submission date: 23-May-2023 12:44PM (UTC+0100)

Submission ID: 207315709

File name: 17371_Benjamin_Naylor_2117071_181809_2128113691.pdf (6.04M)

Word count: 5978

Character count: 40595

Course: HNC Electrical and Electronic Engineering

Module: HNC Electrical and Electronic Engineering Project SGEE4006

Assignment: Final Report

Name: Ben Naylor

University Number: 2117071

Date Submitted: 23/05/23.

Word Count: 4100

Contents

1 - Introduction.	6
2 – Specification	7
3 – Planning	9
4 – Solutions Considered	10
5 - Final Solution and Implementation	13
6 - Testing and Results	43
7 - Discussion, Conclusions and Recommendations	52
8 – Reference list and Bibliography	53
9 – Appendix	54

The contents table should be more detailed

Figure 1 Full system.....	7
Figure 2 Ultrasonic sensor.....	8
Figure 3 Motor.....	8
Figure 4 Gantt chart.....	9
Figure 5 Testing table.....	9
Figure 6 Arduino Mega 2560.....	10
Figure 7 PLC Level sensor.....	11
Figure 8 PLC SIMATIC S7-1200.....	11
Figure 9 System diagram.....	13
Figure 10 prototype 1 Arduino code 1.....	14
Figure 11 prototype 1 Arduino code 2.....	14
Figure 12 Prototype 1 Labview byte=0.....	15
Figure 13 Prototype 1 Labview byte=3.....	16
Figure 14 Prototype 1 Labview block diagram.....	16
Figure 15 Prototype 2 Labview output 1.....	17
Figure 16 Prototype 2 circuit setup.....	17
Figure 17 Labview Prototype 2 Block Diagram.....	17
Figure 18 Prototype 2 Arduino code 1.....	18
Figure 19 Prototype 2 Arduino code 2.....	18
Figure 20 Prototype 2 Labview low setting.....	18
Figure 21 Prototype 2 Labview high setting.....	18
Figure 22 Prototype 3 Labview screen.....	19
Figure 23 Prototype 3 True.....	20
Figure 24 Prototype 3 False.....	20
Figure 25 Main Blocks for Prototype 3 block diagram.....	21
Figure 26 Prototype 3 fault condition.....	21
Figure 27 Setup for prototype 4.....	22
Figure 28 Prototype 5 setup.....	24
Figure 29 Prototype 5 code 1.....	24
Figure 30 Prototype 5 code 2.....	25
Figure 31 Prototype 5 code 3.....	25
Figure 32 Prototype 5 code 4.....	25
Figure 33 Prototype 5 code 5.....	25
Figure 34 Prototype 5 code 6.....	25
Figure 35 Higher PWM test for prototype 5.....	26
Figure 36 Lower PWM test for prototype 5.....	26
Figure 37 Prototype 6 setup.....	27
Figure 38 Prototype 6 code 1.....	27
Figure 39 Prototype 6 code 2.....	27
Figure 40 Prototype 6 code 3.....	28
Figure 41 Prototype 6 PWM 255.....	28
Figure 42 Prototype 6 PWM 0.....	29
Figure 43 Prototype 7 code 1.....	30
Figure 44 Prototype 7 code 2.....	30
Figure 45 Prototype 7 code 3.....	30
Figure 46 Prototype 7 code 4.....	30
Figure 47 Prototype 7 code 5.....	31
Figure 48 Prototype 7 code 6.....	31

Figure 49 Prototype 7 setup.....	32
Figure 50 50%.....	33
Figure 51 90%.....	34
Figure 52 70%.....	34
Figure 53 100% LCD Display.....	35
Figure 54 50% LCD Display.....	35
Figure 55 70% LCD Display.....	36
Figure 56 Main vi user screen.....	37
Figure 57 Final prototype screen.....	37
Figure 58 Com link.....	38
Figure 59 graph data.....	38
Figure 60 conditions.....	39
Figure 61 Significant blocks.....	39
Figure 62 screen explanation 1.....	40
Figure 63 screen explanation 2.....	40
Figure 64 Prototype 8 code 1.....	41
Figure 65 Prototype 8 code 2.....	41
Figure 66 Prototype 8 code 3.....	41
Figure 67 Prototype 8 code 4.....	41
Figure 68 Prototype 8 code 5.....	42
Figure 69 Prototype 8 code 6.....	42
Figure 70 58%.....	43
Figure 71 58% results.....	43
Figure 72 81%.....	43
Figure 73 81% results.....	44
Figure 74 16%.....	44
Figure 75 16% results.....	44
Figure 76 Exceeds.....	44
Figure 77 out reacts.....	45
Figure 78 equals.....	45
Figure 79 2cm below.....	45
Figure 80 front display.....	46
Figure 81 response graph.....	47
Figure 82 steady output response.....	47
Figure 83 below 10cm.....	48
Figure 84 11cm.....	48
Figure 85 5cm.....	48
Figure 86 setpoint, input and output.....	49
Figure 87 results.....	49
Figure 88 setpoint 10cm.....	49
Figure 89 setpoint, input and output.....	49
Figure 90 results.....	49
Figure 91 setpoint 12cm.....	50
Figure 92 setpoint, input and output.....	50
Figure 93 output.....	50
Figure 94 setpoint 15cm.....	50
Figure 95 setpoint, input and output.....	51
Figure 96 output.....	51

Figure 97 Code for Prototype 1	54
Figure 98 Prototype 2 Arduino code	54
Figure 99 Prototype 4 code 1	55
Figure 100 Prototype 4 code 2	56
Figure 101 Prototype 5 code 1	56
Figure 102 prototype 5 code 2	56
Figure 103 Prototype 5 code 3	57
Figure 104 Prototype 6 code 1	57
Figure 105 Prototype 6 code 2	58
Figure 106 Prototype 6 code 3	58
Figure 107 Prototype 7 code 1	59
Figure 108 Prototype 7 code 2	59
Figure 109 Prototype 7 code 3	60
Figure 110 Prototype 8 code 1	61
Figure 111 Prototype 8 code 2	62
Figure 112 Prototype 8 code 3	63

Good extensive list of figures

1 - Introduction.

The Project was to solve the problem of a water tank that has no current way of regulating the level of the tank. The solution was to create a system where the level of the water is read through a sensor and then the system would determine how much water to drive into the tank using the water level as a reference, making sure that the level stays at the level that was chosen. I have given two possible solutions the problem, one using an Arduino and the other using a PLC, it was the Arduino version of the solution that was later used and created. The project took place in G23 in Coleg Sir Gar Graig Campus. The access to G23 was however limited due to being only open school times. I created a Gantt chart to analyse the time I should be spending on the problem to solve the issue of limited college time.

Avoid first person terms

There is a Water tank in G23 that has no prevention of overflow or any way of knowing exactly how much water is in the tank. This tank needs to have an easy-to-use way to prevent overflow and know how full it truly is. After weighing the options on which version of a solution to pick as seen in task 4 of the assignment I did end up choosing the Arduino PID with LabVIEW interface as this would give a good output on the laptop using the LabView interface, and the Arduino connected with the sensor will give a reading of how high the water level is and with the program that I will write on the Arduino it will give a output to the Driver board that will open or close the tank valve to let more water into the tank.

Introduction is informative. There are grammatical errors

2 – Specification

Grammatical errors

The project is a Tank level sensor using Arduino PID with LabVIEW interface and will be able to monitor and control the tank level. The Sensor will give a output which will go to the Arduino and then the Arduino will send a output to the driver with the information the sensor has given drive the water from the bottom tank into the top tank until it reaches the desired level of the tank and then the sensor will notice that the tank is full and the Arduino will give a output to make the driver stop and therefore keeping the tank at the desired level.

The system can be seen below:



Figure 1 Full system.

The system used a sensor, Arduino, Driver board and Labview software to create a system that detects the water level and used PWM to achieve the setpoint I have chosen to keep it as the setpoint no matter if the tank is being drained or not. The Arduino and driver are powered by a 12V power supply. The ultrasonic sensor has an input of 5V which it gets from the Arduino board. The driver board drives the motor attached to the system to provide the tank with the water to keep the level balanced. The tank has a release valve under it to release the water.

The Ultrasonic sensor can be seen inside the tank in the image below:



Good supporting image

Figure 2 Ultrasonic sensor.

The Motor can be seen below:



Good image

Figure 3 Motor.

First person used

3 – Planning

I created a Gantt chart for my project to make sure that I manage my time accordingly, it can be seen below:



Figure 4 Gantt chart.

My testing table can be seen below. It is yet to be filled until the testing for the final project is completed and it is built:

Test No	Set value CM	Expected Value CM	Actual Value CM	Error CM	PWM Duty Cycle Value	LabVIEW Fault Detection	Pass / Fail
1	5	5					
2	10	10					
3	12	12					
4	15	15					

Figure 5 Testing table.

Good test plan

4 – Solutions Considered

I Have considered two solutions to fix the problem with the tank water level. Solution 1 is an Arduino PID with LabVIEW interface. Solution 2 is a PLC Interface that will be used with a compatible sensor to fix the problem with the water tank.

Solution 1

Arduino PID with LabVIEW interface:

Advantages of using Arduino PID with LabVIEW interface:

- The Arduino software is easy to use and there is some personal experience using it.
- The LabVIEW interface gives a useful visual output.
- There is a basic number of physical components needed.
- The sensor will be fairly accurate.

Good evaluation

Disadvantages of using Arduino PID with LabVIEW interface:

- There is currently limited experience when it comes to using LabVIEW.
- It will be difficult to link the LabVIEW with the Arduino for a first-time user.
- The LabVIEW Software is expensive.

Overall using an Arduino with LabVIEW is a great solution to the problem with minimum expense, the budget for the problem is justified. The Arduino Mega 2560 costs around £36. (Arduino Store, 2023)

Good supporting ref source and image



Figure 6 Arduino Mega 2560

Solution 2

PLC Interface:

Advantages of using a PLC:

- There is some personal experience using PLC's.
- Using a PLC would give a very accurate measurement.

Good evaluation

Disadvantages of using a PLC:

- The Software that would be used with the PLC is very expensive.
- The PLC's and Ultrasonic sensors are very expensive.

Overall using a PLC Interface would be overkill for such a simple task of sensing the level of a tank and the expense would be far too large for such a task. Price of Siemens software is £3,131. (PLC City, 2023) The sensor used would be a Siemens ST-H Transducer which is £540, making it a very expensive system and the PLC itself would be a £250. (RS Hydro, 2023) (RS, 2023)

This proves your point



Figure 7 PLC Level sensor



Figure 8 PLC SIMATIC S7-1200

Chosen solution:

The chosen solution was solution 1 as it is the most cost-effective way to solve the problem of the water tank and I have previous knowledge in C++ Programming that will help when writing out the program. The solution 2 with PLC software and Siemens PLC devices is a very expensive way of solving the problem with the only benefit being that the accuracy would be slightly better. The accuracy of the ultrasonic sensor is accurate enough for what I need it to do, the accuracy will be off by around 10mm but for the task and the way it's being implemented this will definitely be accurate enough to get the outcome needed. The PLC solution would be more reliable over time but as long as the sensor and Arduino are in a suitable box then they should last a long time. Overall, it was the costs and the experience of using the different software's that made me choose option 1 over option 2.

Good argument for chosen solution

Components used in Solution 1 (Chosen Solution):

The Arduino Board used is a "Arduino mega 2560" costing around £36, this Arduino has a suitable number of inputs and outputs to use for when connecting the other components. The Arduino software is free to download and use which makes it a very good option. The driver board used is a MD10C Motor Driver PCB with an input value of 3.3 and 5V and can operate with a current of 13A. (Cytron Technologies, 2022)

The HC-SR04 Ultrasonic Distance Sensor measures a distance to an object using ultrasound sonar without making contact with it. It has an input of 5V and draws 20mA with a working temperature of -15°C to 70°C and a range of 2cm – 400cm which is definitely plenty for the application it will be used for. (The Pi Hut, 2023)

The Labview software is going to be the most expensive part of the project. The base price for a user licence for the Labview software is around £455 making it by far the only relatively expensive part of the project. (NATIONAL INSTRUMENTS CORP, 2023)

*Good effort here.
Components are identified
and explained with ref
sources*

PWM setting for Arduino Mega 2560

On an Arduino there are many PWM pins and have all have pre-set default frequencies. These can be changes using by accessing the internal registers shown in the text. This project is using pin 45 which runs at 3.9KHZ, which is suitable for a dc motor.

Any DC motor PWM will require a frequency of above 1.5KHZ for ideal operation. Otherwise, the motor will have a stop start continuous characteristic. (eTechnophiles, 2023)

*Good point made
supported with ref
source*

5 - Final Solution and Implementation

This should be deliverables

Provide a full explanation of the final project solution including detailed designs and costs. Note that the developmental stages and prototypes will be found in the logbook. Photographic evidence must also be provided of the final constructed system or product and installation if appropriate. Explain the operation and maintenance requirements and the steps taken to ensure environmental protection and sustainability.

Grammatical errors

The project is a Tank level sensor using Arduino PID with LabVIEW interface and will be able to monitor and control the tank level. The Sensor will give a output which will go to the Arduino and then the Arduino will send a output to the driver with the information the sensor has given drive the water from the bottom tank into the top tank until it reaches the desired level of the tank and then the sensor will notice that the tank is full and the Arduino will give a output to make the driver stop and therefore keeping the tank at the desired level.

The diagram below shows the system:

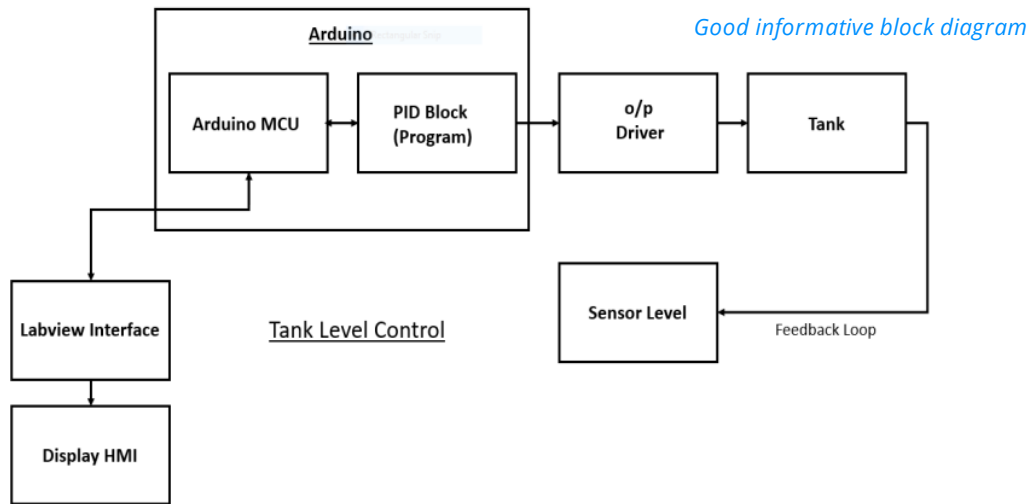


Figure 9 System diagram

The project consisted of 8 different prototypes that were created to perfect the system until I had my final prototype which was the entire working system.

Prototype 1:

For prototype 1 I created a simple read data program on Arduino that links to the Labview. The reason for creating this simplistic byte count and return count prototype was to prove that the Labview will work with the Arduino software with a read data program. The entire code for Prototype 1 can be found in figure 70 in the appendix.

Below shows the int for the Labview is set to zero, initializing it to 0:

```
read_data.ino
1  int diginputpin = 8;
2  int labview = 0;  //initialise to 0
```

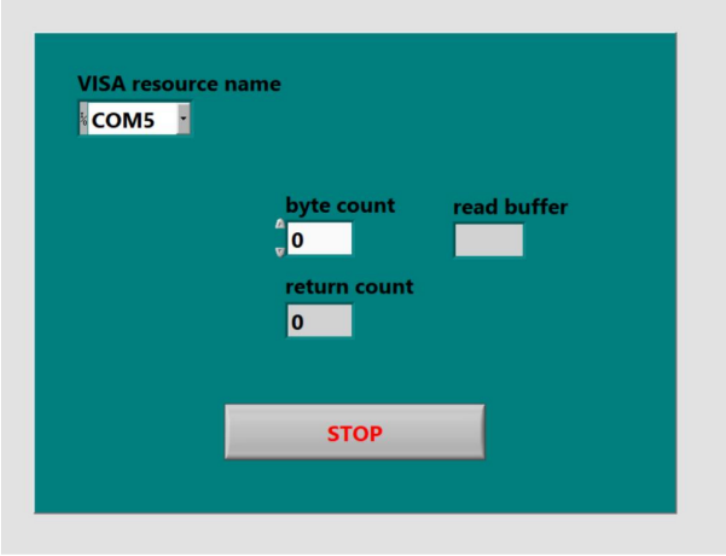
Figure 10 prototype 1 Arduino code 1.

As seen below the baud rate is set to 9600 and the “diginputpin” is declared as the input which the labview software reads and then the Labview is printed into the Labview screen.

```
5  Serial.begin(9600);  //baud rate for serial comms
6  pinMode(diginputpin, INPUT);  //dig input pin declared
7
8  }
9
10 void loop() {
11     labview = digitalRead(diginputpin);
12     Serial.println(labview);
```

Figure 11 prototype 1 Arduino code 2.

Below is the LabVIEW interface reacting with the Arduino program when the byte count is set to Zero:



Good evidence of prototype test outcome

Figure 12 Prototype 1 Laview byte=0.

Below is the LabVIEW interface reacting with the Arduino program when the byte count is set to Three:

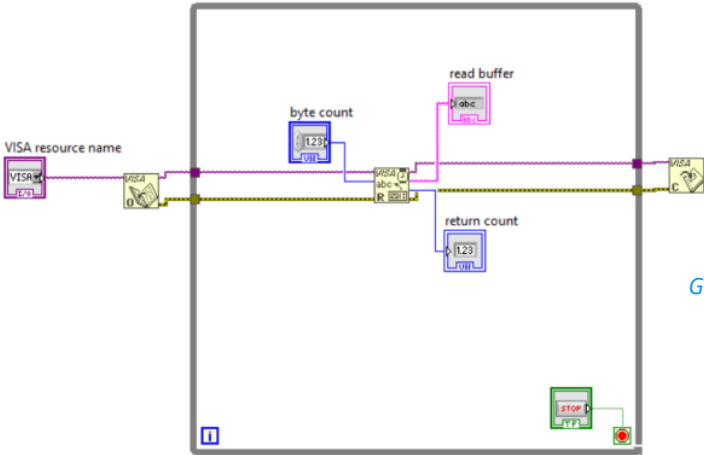


Proof of concept here with serial link operational

Figure 13 Prototype 1 Labview byte=3.

First person

Below is the Block Diagram for the Labview program I created for prototype 1:



Good evidence

Figure 14 Prototype 1 Labview block diagram

The Byte count, read buffer and return count all connect up to the VISA Read function and then to the VISA close function and as seen at the bottom is the stop button.

Prototype 2:

The second prototype uses new Arduino code and a new Labview setup using a potentiometer to move the dial on the Labview screen to prove that the code works with an input from an outside source. The point of this prototype is to prove that the Arduino and Labview will work together to perform an analog read from the potentiometer. The Labview output and the setup of the circuit can be seen below with the middle cable of the potentiometer going to A0 as specified in the code as the input:



Figure 16 Prototype 2 circuit setup

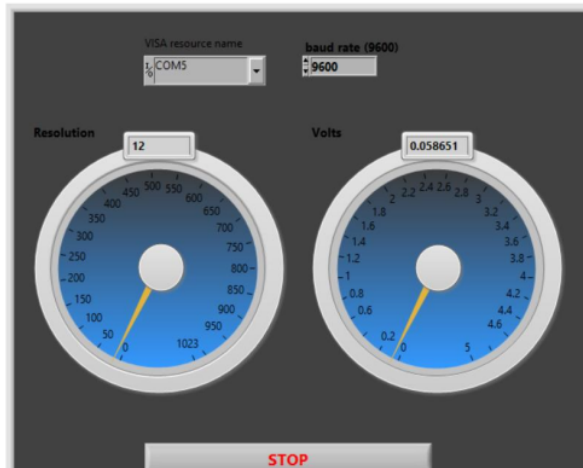


Figure 15 Prototype 2 Labview output 1

A very good prototype progression that is significant for your final outcome.

The Block diagram for the Labview screen in prototype 2 can be seen below:

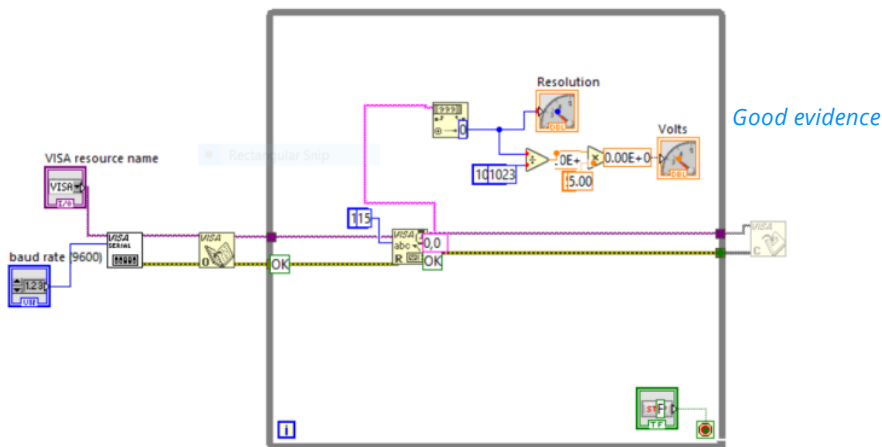


Figure 17 Labview Prototype 2 Block Diagram.

The code below shows that the sensor value has been set to a designated pin number of 0 and the digital input is set to pin 8.

```
Analogue_read.ino
1  int sensorValue = 0; //declare analogue sensor variable with designated pin no
2  int diginputpin = 8; //declare dig input variable with designated pin no
3  int readinputpin8 = 0; //initialise to 0
```

Figure 18 Prototype 2 Arduino code 1.

As seen below the void loop which will constantly run over and over reads the digital input of pin 8 and the sensor value (analog pin 0) is read and then later printed.

```
10 void loop() {
11     readinputpin8 = digitalRead(diginputpin);
12     sensorValue = analogRead(A0);
```

Figure 19 Prototype 2 Arduino code 2.

The entire Arduino code for prototype 2 can be found in figure 71 showing the entire Analog read prototype.

I tested the system with the potentiometer, as seen in the left screenshot I have moved the dial on the potentiometer to give an output on the Labview screen with a resolution of 903 and 4.4V and then as seen on the right-hand side screenshot I have put the potentiometer all the way down to zero giving a resolution on 0 and a voltage of 0. This proves that the Arduino code and the Labview will work with an outside source of information and the analog read is indeed working.

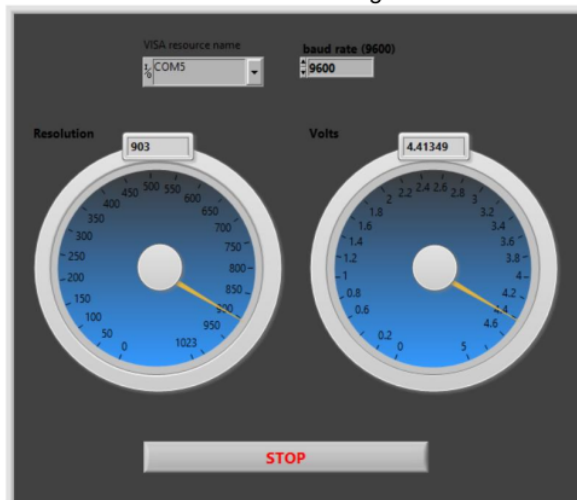


Figure 21 Prototype 2 Labview high setting.



Figure 20 Prototype 2 Labview low setting.

Good evidence and explanation

Prototype 3:

Prototype 3 was a vi.front panel to from the Labview. I created a Labview front panel to display the analog input on an analog input scale. The user screen can be seen below:

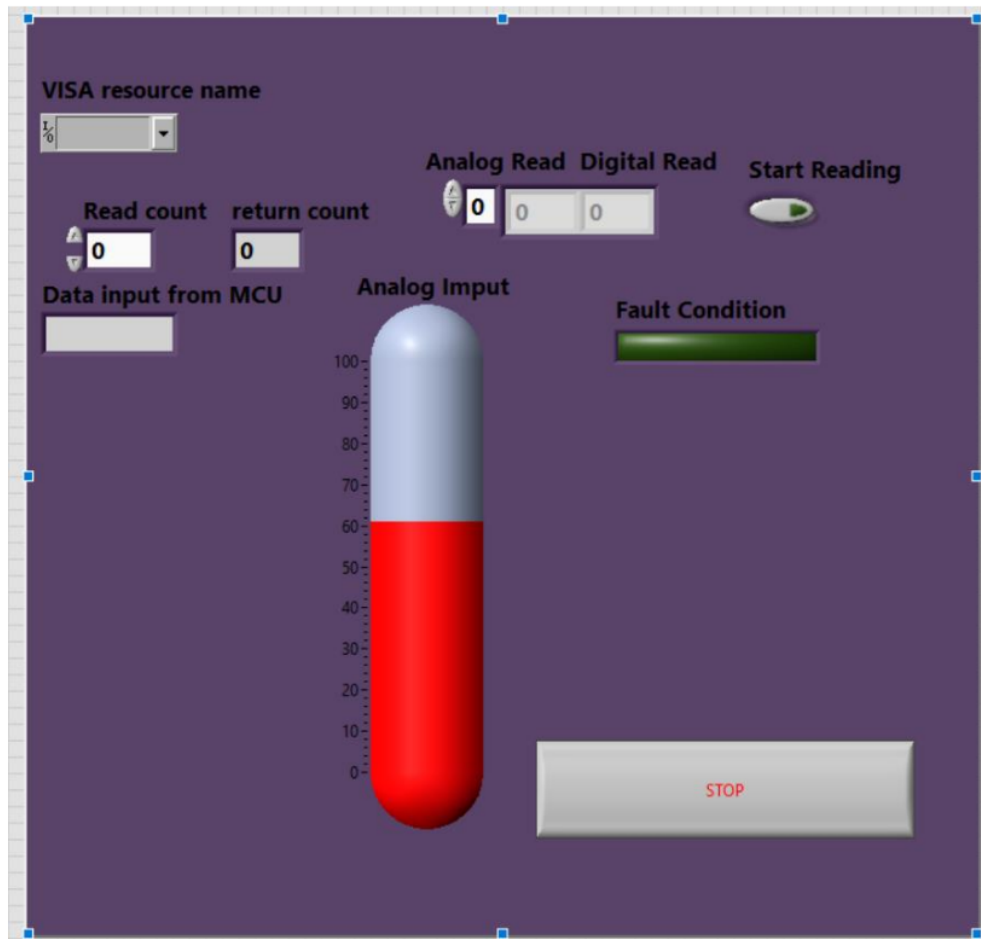
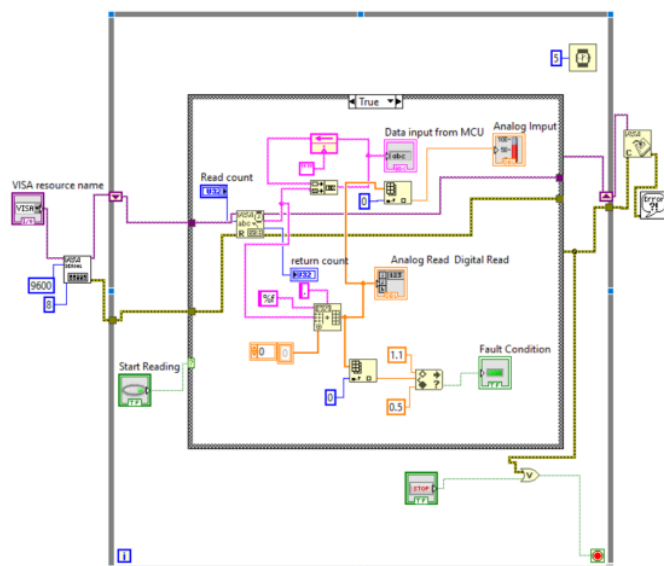


Figure 22 Prototype 3 Labview screen.

The Visa resource will be selected with the correct COM port to allow it to connect to the Correct input. The read count is selected, and the analog read, and digital read are displayed and so is the return count and the data input from the MCU. The Analog input can be displayed on the analog input scale from 0-100. If a fault is found, then the fault condition will illuminate green. The system also has a "stop" button to stop the analog read at any time.

Prototype 3 demonstrates good progress with complexity

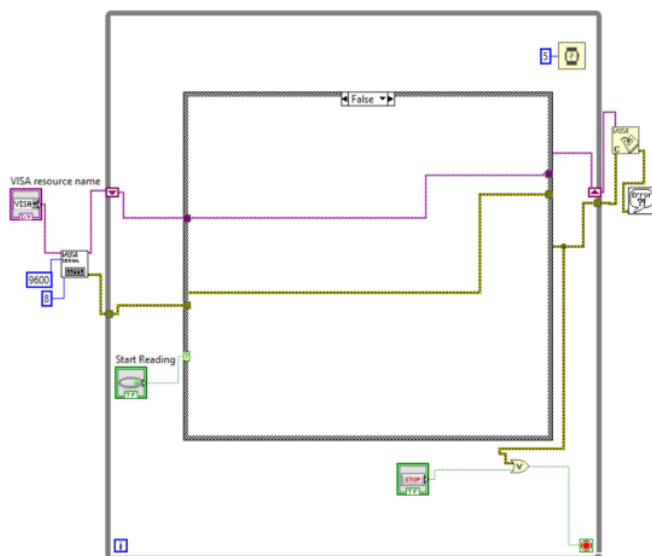
The Block Diagram for prototype 3 for when “true” can be seen below:



Good evidence

Figure 23 Prototype 3 True.

The Block Diagram for prototype 3 for when “False” can be seen below:



Good explanation and evidence

Figure 24 Prototype 3 False.

Figure 1 displays four context help panels for LabVIEW functions:

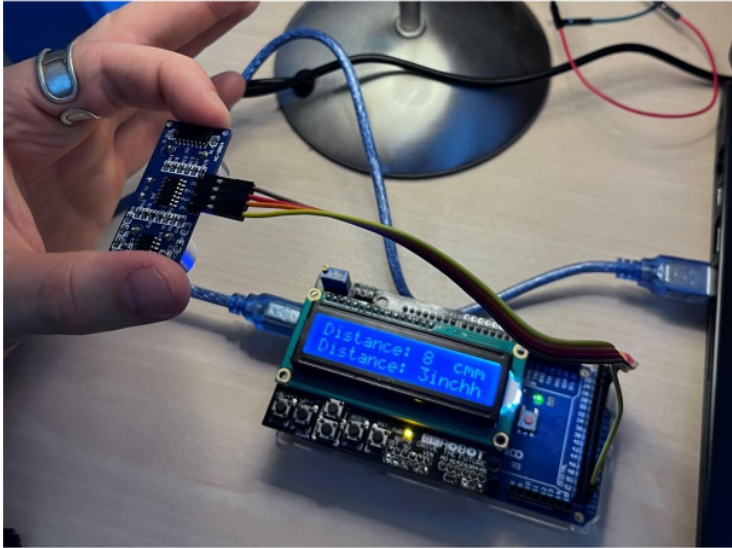
- VISA Read:** Shows inputs for VISA resource name, byte count, and error in (no error). The output is read buffer. A detailed help link is provided.
- VISA Configure Serial Port:** Shows inputs for VISA resource name, baud rate (9600), data bits (8), parity (0none), error in (no error), and flow control (0none). The output is error out. A detailed help link is provided.
- Spreadsheet String To Array:** Shows inputs for delimiter (Tab), format string, spreadsheet string, and array type (2D Dbl). The output is array. A detailed help link is provided.
- VISA Close:** Shows input for VISA resource name and error in (no error). The output is error out. A detailed help link is provided.

Main serial function blocks are identified

Figure 26 Prototype 3 fault condition.

Prototype 4:

Prototype 4 was created to test the system with the ultrasonic sensor input added. The goal was to get the sensor to give the Arduino software a readable and useful input that can be used to measure the water in the tank in the final prototype. The sensor will simply give the distance in cm to the Arduino board in a readable way. The setup for the system can be seen below:



Good evidence and explanation

Figure 27 Setup for prototype 4.

Below you can see that I have included the “LiquidCrystal.h” library to be used for the LCD output:

```
1 #include <LiquidCrystal.h> //Please replace the single quote characters (') with the parenthesis character (<>)
2
3 LiquidCrystal lcd(8, 9, 4, 5, 6, 7); // Creates an LCD object. Parameters: (rs, enable, d4, d5, d6, d7)
4
```

The integers have been declared and the setup has been included as seen below:

```
5 const int trigPin = 23; //10
6 const int echoPin = 25; //9
7 long duration;
8 int distanceCm, distanceInch;
9
10 void setup() {
11
12   lcd.begin(16,2); // Initializes the interface to the LCD screen, and specifies the dimensions (width and height) of the display
13   pinMode(trigPin, OUTPUT);
14   pinMode(echoPin, INPUT);
15 }
```

The code on line 12 starts the interface to the LCD screen and specifies the dimensions, the 16 is the width and the 2 is the height of the display.

Good explanation of program sample

The code below shows the set location where the text will be displayed on the LCD and prints the words "distance:" followed by the distance that the sensor picks up.

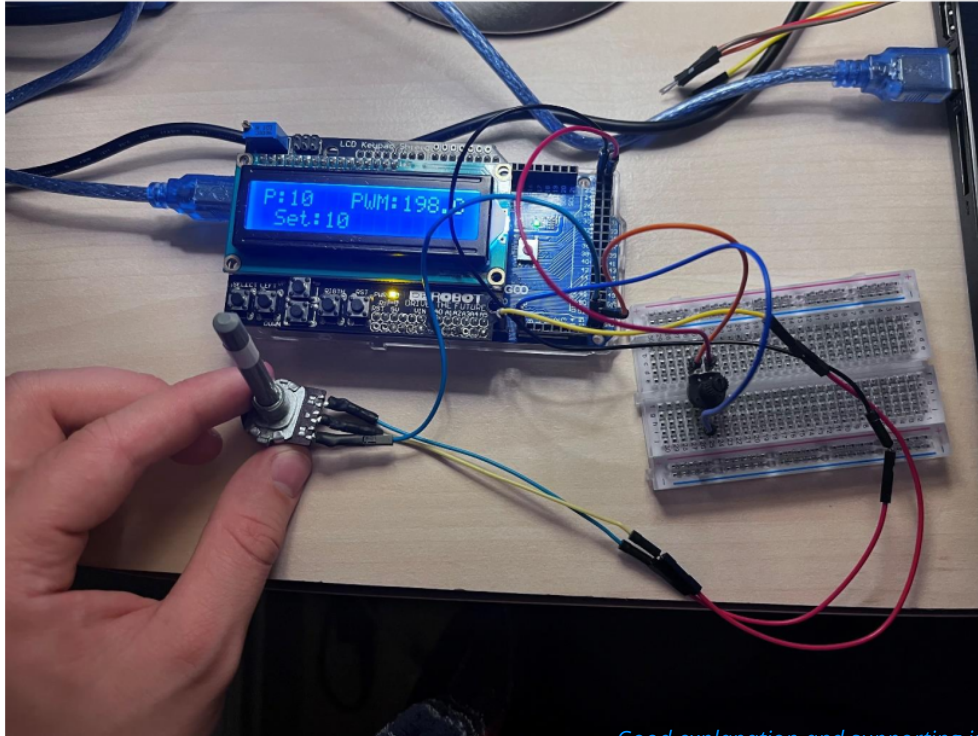
```
30 lcd.setCursor(0,0); // Sets the location at which subsequent text written to the LCD will be displayed
31 lcd.print("Distance: "); // Prints string "Distance" on the LCD
32 lcd.print(distanceCm); // Prints the distance value from the sensor
```

I have uploaded the code onto the Arduino and aimed the sensor at the table below to prove that it gives the output expected. I then measured the distance with a ruler along side the sensor and found that the sensor had around 1cm shorter of a distance reading. The sensor will be adjusted but is accurate enough for the tank level as the water in the tank will be flowing when the system is active so there will be small amount of error but as it's a small tank it doesn't need to be perfectly tuned. The entire code for prototype 4 can be seen in figure 72 and figure 73 in the appendix.

Prototype 5:

First person

In prototype 5 I'm using a potentiometer to set the P which will later be the Height and then I have used another potentiometer to be the set value that I want the tank to be. The "P" simulates the actual distance of the sensor and then the "Set" is what I'm hoping for it to drive the water level to by making the code detect the difference between them and either stop driving the driver by setting the PWM to 0 or fill the tank up with the max PWM being 225. The setup for the prototype 5 can be seen below:



Good explanation and supporting image

Figure 28 Prototype 5 setup.

The code below shows that A7 has been chosen to be the input pin of the potentiometer for the "Set" value and A8 is the potentiometer input for the simulated sensor.

```
7 | int analogInPin2 = A8;
8 | const int analogInPin = A7; // Analog input pin that the potentiometer is attached to
9 |
10 | int sensorValue = 0; // value read from the pot
11 | int sensorValue2 = 0;
12 | LiquidCrystal lcd(8, 9, 4, 5, 6, 7); //set up LCD
13 |
```

Figure 29 Prototype 5 code 1.

The code below shows the setup for the PID:

```
15 //Setup PID
16 double Setpoint, Input, Output; //I/O for PID
17 double Kp=10, Ki=5, Kd=2; //original consKp=1, consKi=0.05, consKd=0.25, Conservative Turning,20,10,10
18 PID myPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, DIRECT); //Initialize PID
```

Figure 30 Prototype 5 code 2.

The code below shows the code for reading the analog input values for the system:

```
29 // read the analog in value:
30 sensorValue = analogRead(analogInPin); //reads value from potentiometer
31 sensorValue2 = analogRead(analogInPin2);
32 sensorValue = map(sensorValue, 0, 1023, 100, 0);
33 sensorValue2 = map(sensorValue2, 0, 1023, 100, 0);
34 Output = map(Output, 0, 1023, 255, 0);
```

Figure 31 Prototype 5 code 3.

I have shown the PID setup and PID pin setup in the code below:

```
36 //PID Setup
37 myPID.SetMode(AUTOMATIC);
38 //TCCR2B = TCCR2B & 0b1111000 | 0x01; //adjust the PWM Frequency, note: this changes timing like delay()
39
40 //Setup Pins
41 pinMode(PUMP, OUTPUT); // Output for pump speed, 0 to 255
42
43 //Setup LCD 16x2 and display startup message
44 lcd.begin(16, 2);
45 lcd.setCursor(0,1);
46 lcd.print(" Starting Up");
47 delay(3000);
48 lcd.clear();
49 }
```

Figure 32 Prototype 5 code 4.

Below is the code of the start of the Loop function where I get the distance and give it to the PID input:

```
53 delay(200);
54 //Get distance and give it to the PID input
55 sensorValue = analogRead(analogInPin);
56 sensorValue2 = analogRead(analogInPin2);
57 sensorValue = map(sensorValue, 0, 1023, 0, 50);
58 sensorValue2 = map(sensorValue2, 0, 1023, 0, 50);
59 Setpoint=(sensorValue2);
60 Input = (sensorValue);
..
```

Figure 33 Prototype 5 code 5.

I have created the code for computing/calculating the PID value and writing the results to the output:

```
78 //Compute PID value
79 myPID.Compute();
80 //write PWM to output
81 analogWrite(PUMP,Output);
```

Good explanation of sample code

Figure 34 Prototype 5 code 6.

Below are the outcomes of the prototype 5, one with the height of the sensor giving a output higher than the setpoint causing the PWM to decrease and one with the sensor input being lower than the setpoint causing the PWM to increase:

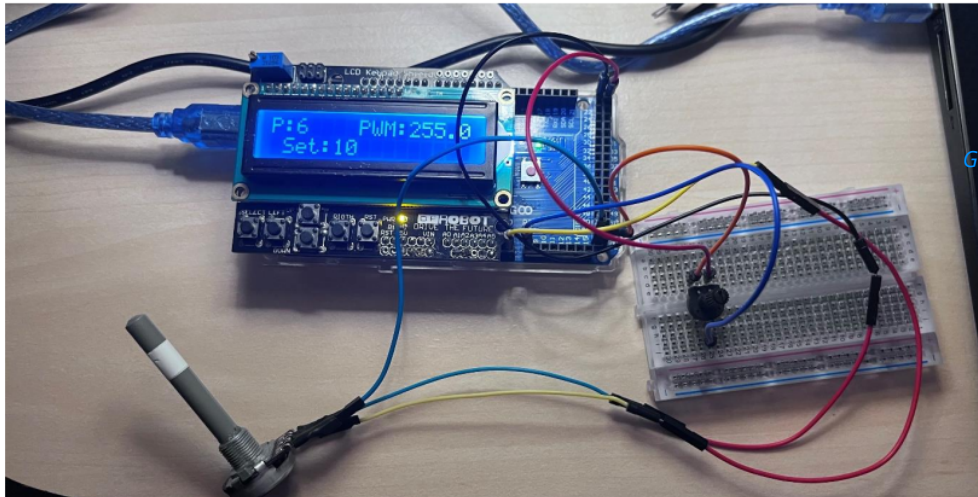


Figure 35 Higher PWM test for prototype 5.



Figure 36 Lower PWM test for prototype 5.

The full code for Prototype 5 can be found in figure 74, 75 and figure 76 in the appendix.

Prototype 6

Prototype 6 carries on from prototype 5 doing the exact same thing but now instead used the actual ultrasonic sensor as an input instead of using another potentiometer as a simulated sensor input.

The setup for prototype 6 can be seen below with the potentiometer and sensor attached to the Arduino board:

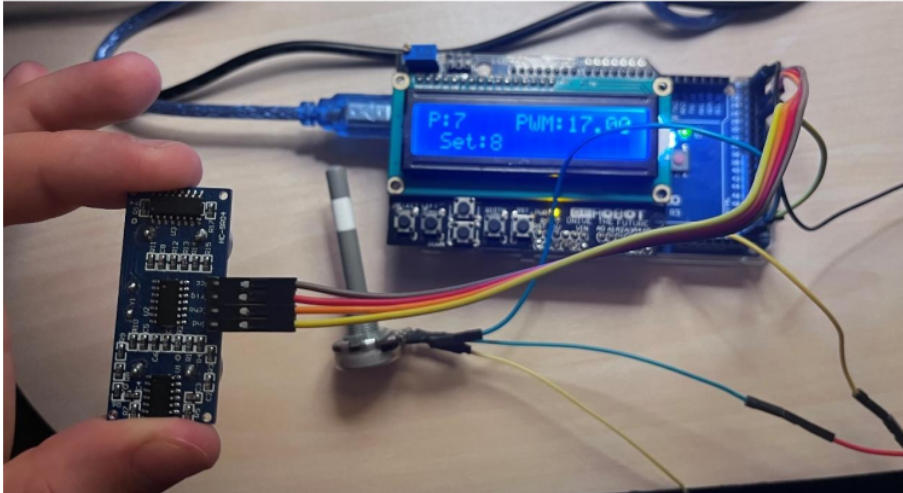


Figure 37 Prototype 6 setup.

The code for prototype 6 is mostly the same as prototype 5 but is has been adjusted to use the sensor as the input. The code below shows the pins I have set the “Trig” and “Echo” pin to:

```
4 //Definitions
5 #define PUMP 45 // Output pin for pump
6 #define echoPin 29 // attach pin D2 Arduino to pin Echo of HC-SR04
7 #define trigPin 27 //attach pin D3 Arduino to pin Trig of HC-SR04
8
9 int analogInPin2 = A8;
```

Figure 38 Prototype 6 code 1.

As seen in the code above the Potentiometer int is set to pin A8.

The code below shows the trig pin is set to an output and the echo pin is a input:

```
28 pinMode(trigPin, OUTPUT); // Sets the trigPin as an OUTPUT
29 pinMode(echoPin, INPUT); // Sets the echoPin as an INPUT
30 Serial.begin(9600); // // Serial Communication is starting with 9600 of baudrate speed
31 Serial.println("Ultrasonic Sensor HC-SR04 Test"); // print some text in Serial Monitor
32 Serial.println("with Arduino UNO R3");
33
```

Figure 39 Prototype 6 code 2.

The code below in the Loop function shows the trigpin has been cleared and its set to “LOW” and “HIGH” after 10 microseconds. On line 68 it reads the echo pin and returns the sound wave travel time in microseconds. Line 70 calculates the distance and then it gets displayed on the serial monitor.

```
62 // Clears the trigPin condition
63 digitalWrite(trigPin, LOW);
64 delayMicroseconds(2);
65 // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
66 digitalWrite(trigPin, HIGH);
67 delayMicroseconds(10);
68 digitalWrite(trigPin, LOW);
69 // Reads the echoPin, returns the sound wave travel time in microseconds
70 duration = pulseIn(echoPin, HIGH);
71 // Calculating the distance
72 distance = duration * 0.034 / 2; // Speed of sound wave divided by 2 (go and back)
73 // Displays the distance on the Serial Monitor
```

Figure 40 Prototype 6 code 3.

I tested the system by setting the setpoint to 8 and then moved the sensor higher and lower than this to give the PWM output difference. Again, when it's below the setpoint the PWM goes up to try and drive the motor and when its above it, does the opposite.

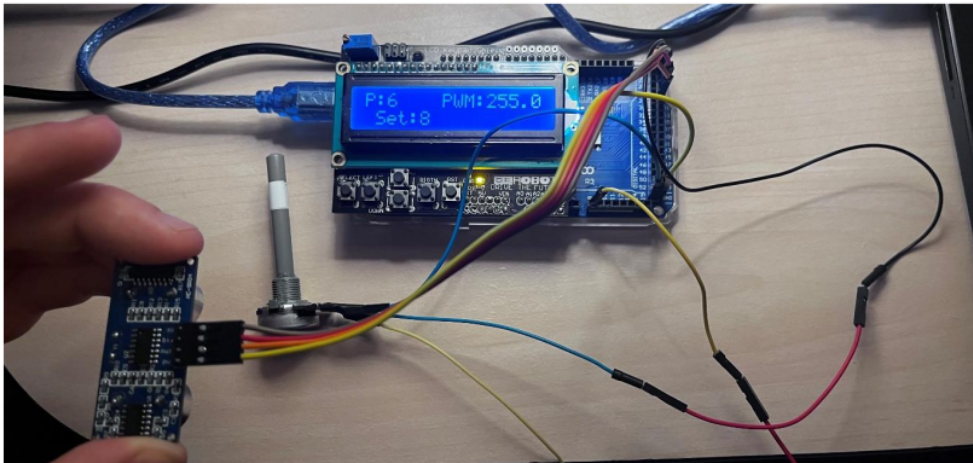


Figure 41 Prototype 6 PWM 255.

Good progression demonstrated in this prototype

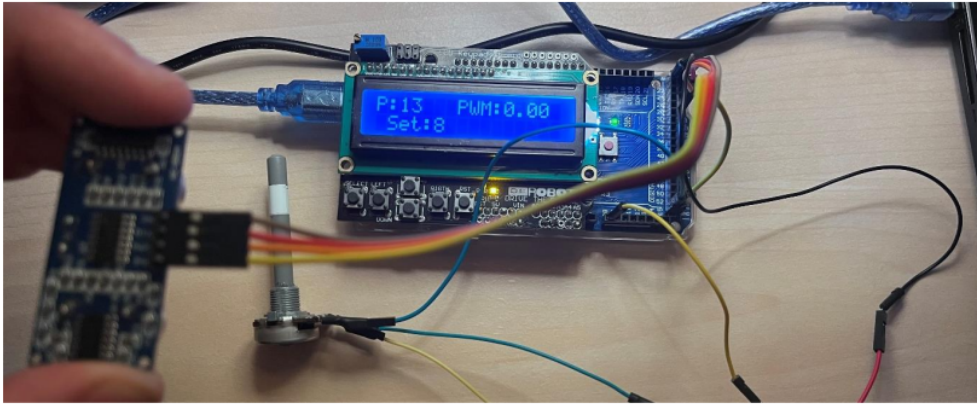


Figure 42 Prototype 6 PWM 0.

Good evidence

The entire code for prototype 6 can be found in figure 77, 78 and 79 in the appendix.

Prototype 7

In prototype 7 I adjusted the Arduino code to be able to attach a Pico scope and read the PWM output so that I can test and analyse it.

I have changed the echo and trig pins to pin 23 and 22. All my defined variables can be seen below:

```
5  #define echoPin 22 // attach pin D2 Arduino to pin Echo of HC-SR04
6  #define trigPin 23 //attach pin D3 Arduino to pin Trig of HC-SR04
7  #define PUMP 45      // Output pin for pump to Cytron board
8
9  // defines variables
10 long duration; // variable for the duration of sound wave travel
11 int distance; // variable for the distance measurement
12 int analogInPin2 = A8; //set point value
13 //const int analogInPin = A7; // Analog input pin that the potentiometer is attached to
..
```

Figure 43 Prototype 7 code 1

My PID is setup as seen below:

```
20 //Setup PID
21 double Setpoint, Input, Output; //I/O for PID
22 double Kp=30, Ki=10, Kd=2; //original consKp=1, consKi=0.05, consKd=0.25, Conservative Turning,20,10,10
23 PID myPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, REVERSE); //Initialize PID
```

Figure 44 Prototype 7 code 2.

My Trigpin was set to an output and the echopin was set to an input:

```
29 pinMode(trigPin, OUTPUT); // Sets the trigPin as an OUTPUT
30 pinMode(echoPin, INPUT); // Sets the echoPin as an INPUT
31 // start serial port for distance readings
32 Serial.begin(9600);
33 Serial.println("Start");
```

Figure 45 Prototype 7 code 3.

The PUMP is defined as an output and will be outputting 0 to 255. The LCD startup is also displayed:

```
47 //Setup Pins
48 pinMode(PUMP, OUTPUT); // Output for motor speed, 0 to 255
49
50 //Setup LCD 16x2 and display startup message
51 lcd.begin(16, 2);
52 lcd.setCursor(0,1);
53 lcd.print(" Starting Up");
54 delay(3000);
55 lcd.clear();
```

Figure 46 Prototype 7 code 4.

The Loop function starts as seen below the same as prototype 6:

```
58 void loop()
59 {
60     // Clears the trigPin condition
61     digitalWrite(trigPin, LOW);
62     delayMicroseconds(2);
63     // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
64     digitalWrite(trigPin, HIGH);
65     delayMicroseconds(10);
66     digitalWrite(trigPin, LOW);
67     // Reads the echoPin, returns the sound wave travel time in microseconds
68     duration = pulseIn(echoPin, HIGH);
69     // Calculating the distance
70     distance = duration * 0.034 / 2; // Speed of sound wave divided by 2 (go and back)
```

Figure 47 Prototype 7 code 5.

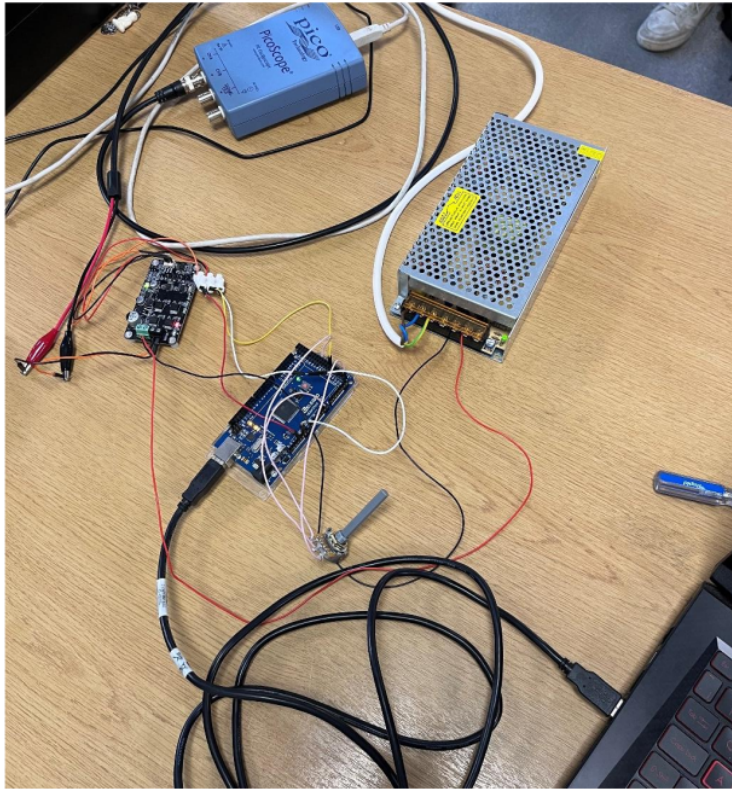
I have removed unneeded code as seen below:

```
72 //sensorValue = analogRead(analogInPin);
73 sensorValue2 = analogRead(analogInPin2);
74 // sensorValue = map(sensorValue, 0, 1023, 255, 0); //map 10 bit to 8 bit
75 sensorValue2 = map(sensorValue2, 0, 1023, 255, 0);
76 Setpoint=sensorValue2;
77 Input = (distance);
```

Figure 48 Prototype 7 code 6.

The code is mostly the same but with added comments to make it easier for me to read and I will be testing the prototype using a Pico scope to analyse the PWM output of the Arduino and driver board combined.

I tested the prototype by setting up the system as seen below:



Good supporting image of test setup

Figure 49 Prototype 7 setup.

The Pico scope connects to the black terminal block on the driver board and the orientation of which cable goes in which slot chooses which way the motor runs. The driver connects to the Arduino is that the Arduino can tell it how hard to drive its output. The potentiometer is again used as the setpoint and the 12V power supply powers the driver board.

The results from the testing:

Test and analysis of the PWM output

Testing the output of using a Pico scope.

50% using the Potentiometer:

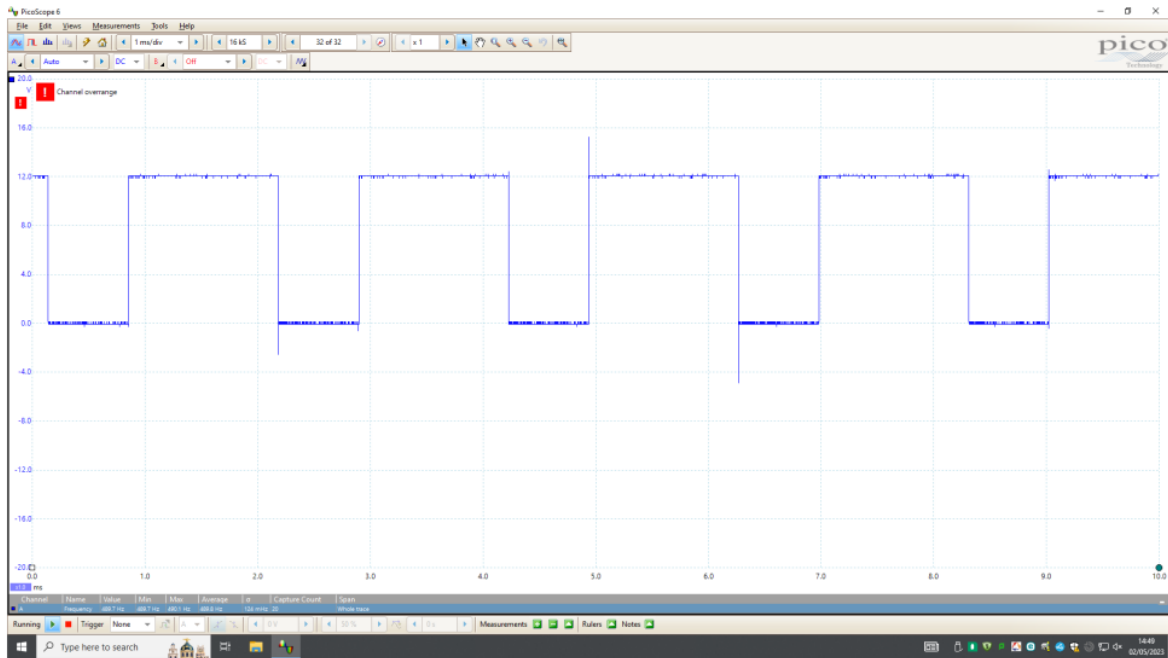


Figure 50 50%.

90% using the Potentiometer:

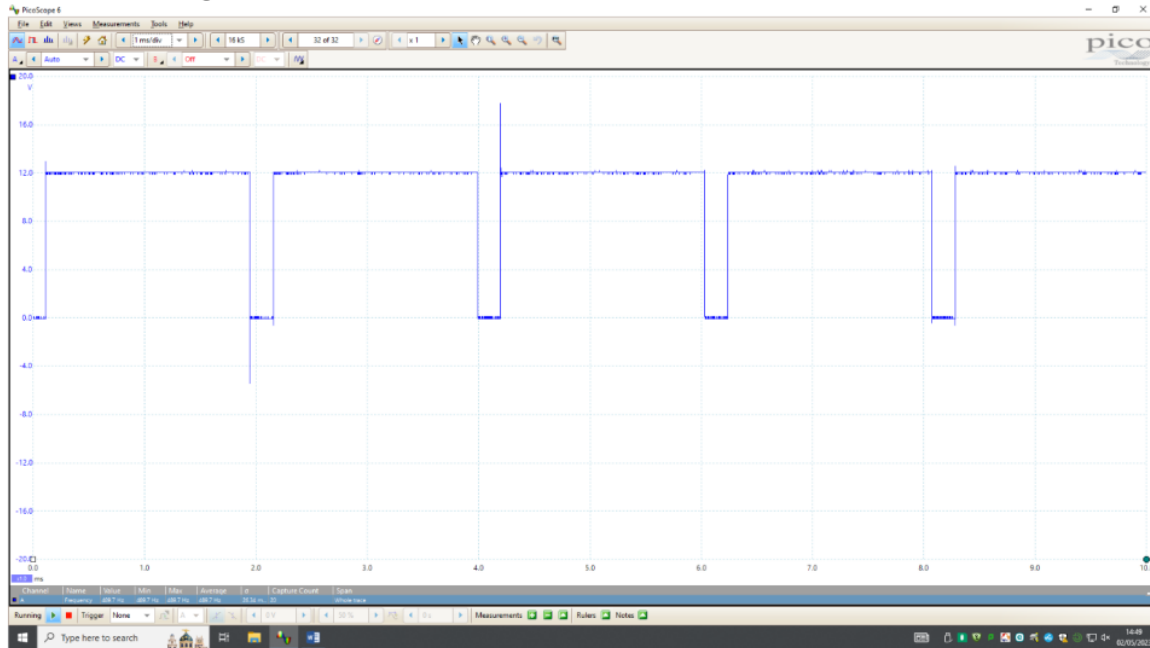


Figure 51 90%

70% using the Potentiometer:

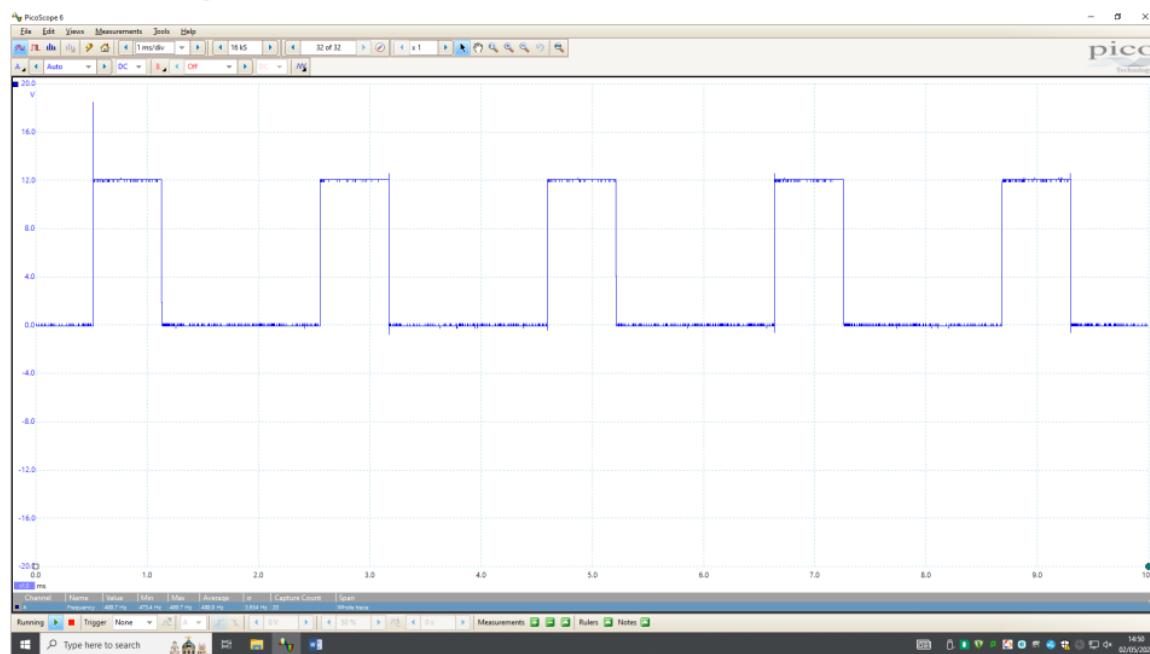


Figure 52 70%

PWM output control and frequency is demonstrated

100% using the Potentiometer LCD Display:

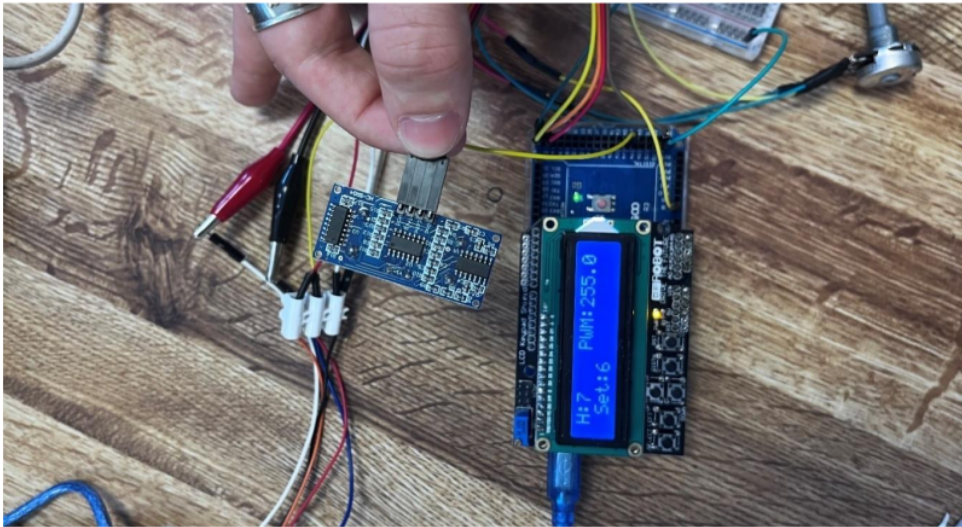


Figure 53 100% LCD Display

50% using the Potentiometer LCD Display:

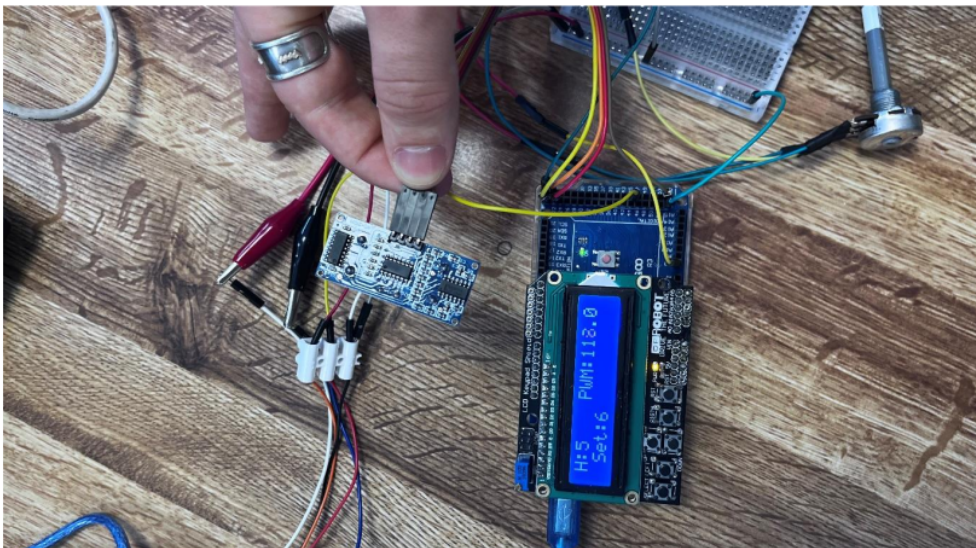


Figure 54 50% LCD Display.

70% using the Potentiometer LCD Display:

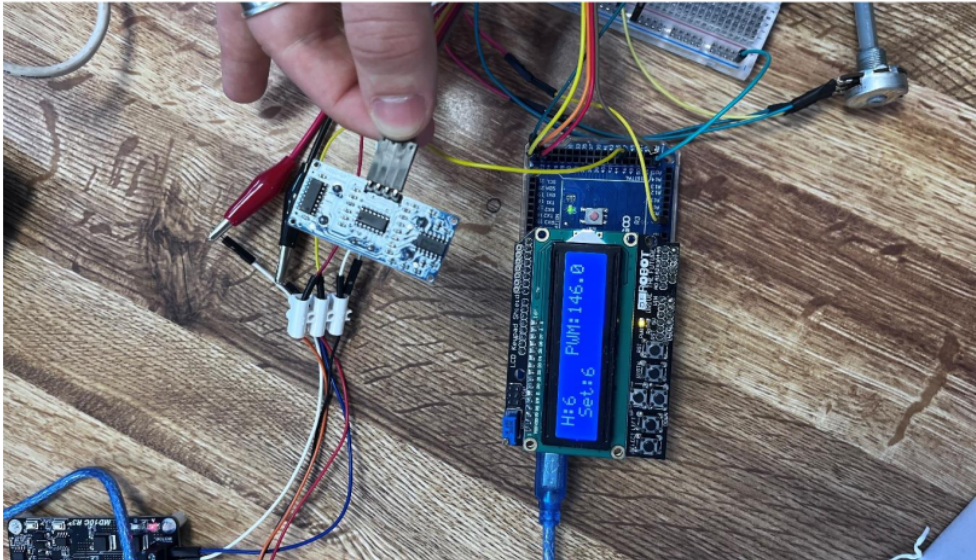


Figure 55 70% LCD Display.

Results are displayed on LCD

All the Arduino code can be found in figure 80, 81 and 82 in the appendix.

Prototype 8 (Final Prototype)

The Prototype 8 should work exactly how I wanted the initial project plan to work as it's the final prototype.

The LabVIEW vi user screen can be seen below:

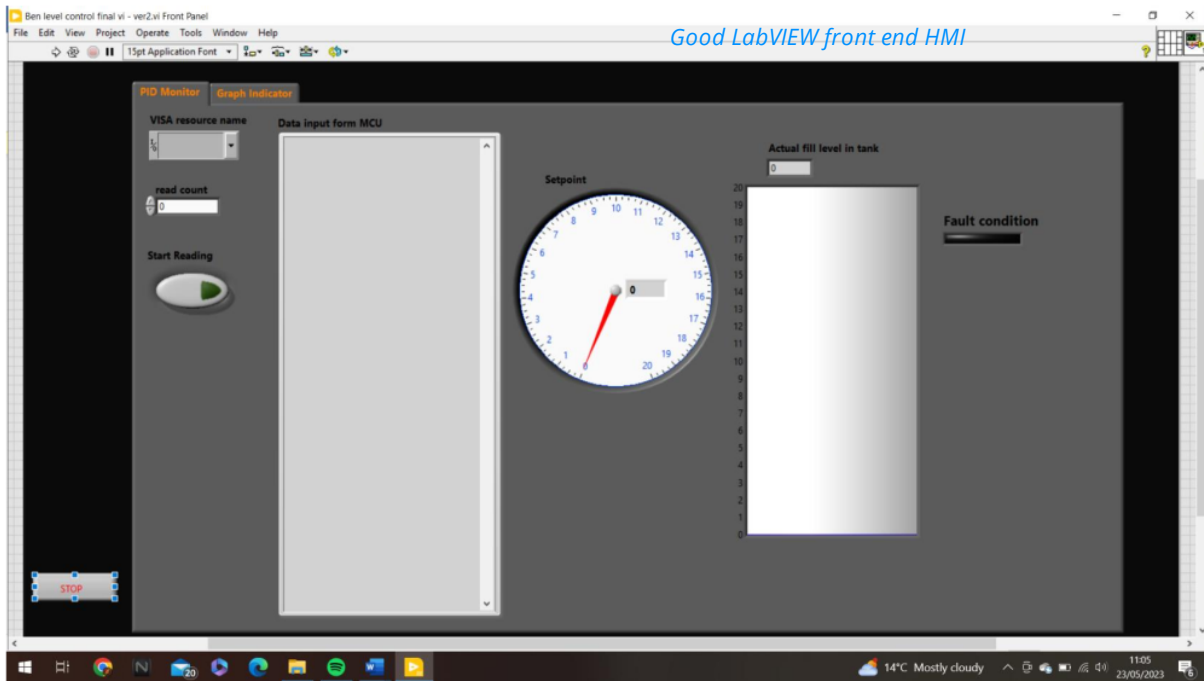


Figure 56 Main vi user screen.

The block diagram for the screen can be seen below:

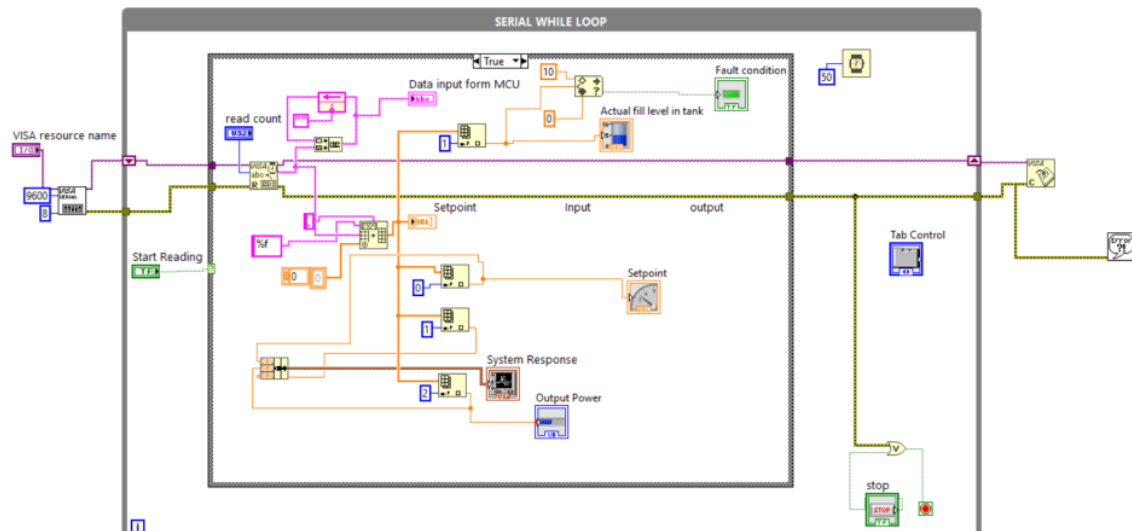


Figure 57 Final prototype screen.

Communication link to Arduino



Serial data extracted by elements from array for graphical display, setpoint dial and output power slider indicators.

Conditions set for fault from less than 10cm from sensor.

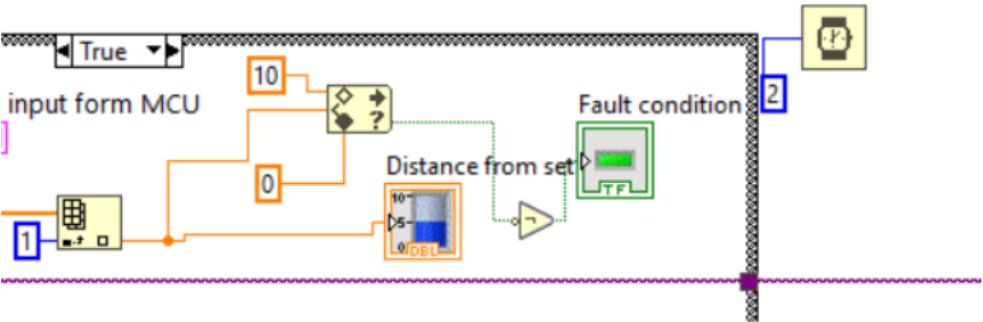


Figure 60 conditions

Fault conditions set for max 10 and min 0 and indicator via LED on from panel.

Significant blocks used in program:



Figure 61 Significant blocks.

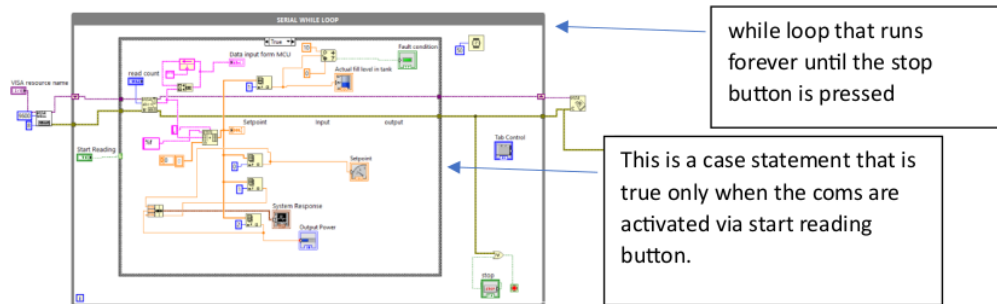


Figure 62 screen explanation 1

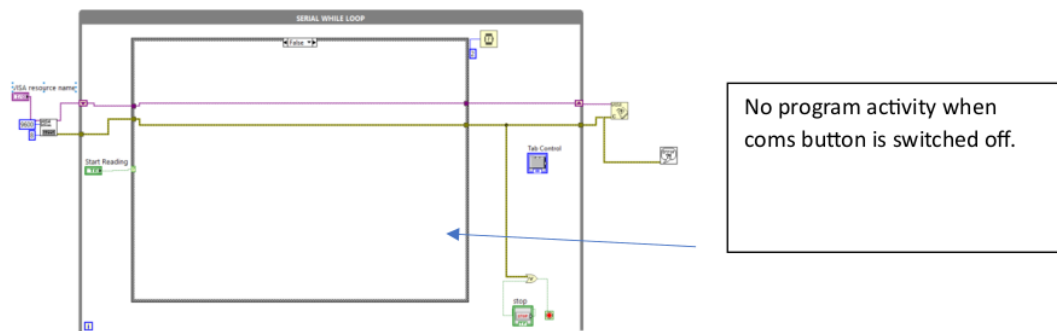


Figure 63 screen explanation 2

Good explanation of LabVIEW program

The Arduino code:

I have defined my variables:

```
9 // defines variables
10 const int max_value = 20; //max height of tank
11 long duration; // variable for the duration of sound wave travel
12 int distance; // variable for the distance measurement
13 int analogInPin2 = A8; //set point value
14 //const int analogInPin = A7; // Analog input pin that the potentiometer is attached to
```

Figure 64 Prototype 8 code 1

I have setup my PID:

```
21 //Setup PID
22 double Setpoint, Input, Output; //I/O for PID
23 double Kp=30, Ki=10, Kd=2; //original Kp=1, Ki=0.05, Kd=0.25
24 PID myPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, DIRECT); //Initialize PID
25
```

Figure 65 Prototype 8 code 2

Adjusted so I get a PWM frequency of 3921.16Hz:

```
43 //PID Setup
44 myPID.SetMode(AUTOMATIC);
45 //TCCR2B = TCCR2B & 0b1111000 | 0x01; //adjust the PWM Frequency, note: this changes timing like delay()
46 TCCR5B = TCCR5B & 0b1111000 | 0b0000010; // for PWM frequency of 3921.16 Hz
47
```

Figure 66 Prototype 8 code 3

The void Loop starts with clearing the trig pin and setting it from LOW to HIGH. On line 69 it calculated the distance:

```
59 void loop()
60 {
61     // Clears the trigPin condition
62     digitalWrite(trigPin, LOW);
63     delayMicroseconds(2);
64     // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
65     digitalWrite(trigPin, HIGH);
66     delayMicroseconds(10);
67     digitalWrite(trigPin, LOW);
68     // Reads the echoPin, returns the sound wave travel time in microseconds
69     duration = pulseIn(echoPin, HIGH);
70     // Calculating the distance
71     distance = duration * 0.034 / 2; // Speed of sound wave divided by 2 (go and back
72     distance = max_value - distance;
73 }
```

Figure 67 Prototype 8 code 4

Good commented Arduino C++ code

The code below reads the analog input :

```
74 //sensorValue = analogRead(analogInPin);
75 sensorValue2 = analogRead(analogInPin2);
76 //sensorValue = map(sensorValue, 0, 1023, 255, 0);
77 sensorValue2 = map(sensorValue2, 0, 1023, 255, 0);
78 Setpoint=sensorValue2;
79 Setpoint = Setpoint -2;
80 Input = distance;
81 if (Input >= 20){
82     Input = 20;
83 }
```

Figure 68 Prototype 8 code 5

The serial print can be seen below:

```
107 if (Output <= 80)
108 {
109     Output = 0;
110 }
111
112 Serial.print(Setpoint);
113 Serial.print(",");
114 Serial.print(Input);
115 Serial.print(",");
116 Serial.println(Output);
117 delay (500);
118 }//end loop
```

Figure 69 Prototype 8 code 6

Most of the code of Prototype 8 is the same as prototype 7 but it has been adjusted. The full Arduino code for prototype 8 can be seen in figure 83, 84 and 85 in the appendix.

The testing of prototype 8 is in task 6 as it's the final prototype.

6 - Testing and Results

The testing plan from task 3 was revisited in this task with the inclusion of test results and has been supported by screenshotted graphs of the outputs and analysed with data tables.

PWM frequency set for 3921HZ in C++ program in Arduino

PWM duty cycle at 58%

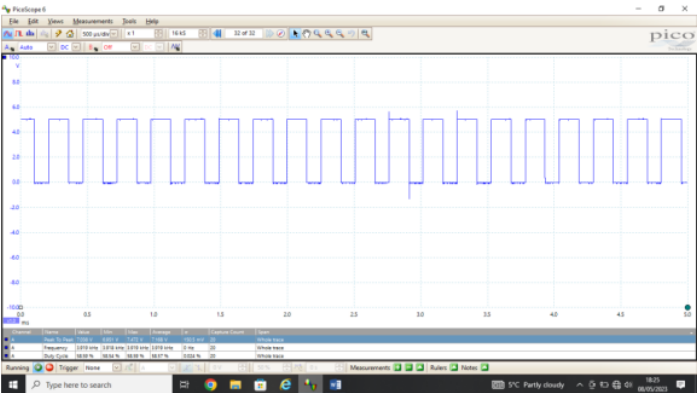


Figure 70 58%

Channel	Name	Value	Min	Max	Average	σ	Capture Count	Span
A	Peak To Peak	7.038 V	6.951 V	7.472 V	7.168 V	150.5 mV	20	Whole trace
A	Frequency	3.919 kHz	3.918 kHz	3.919 kHz	3.919 kHz	0 Hz	20	Whole trace
A	Duty Cycle	58.59 %	58.54 %	58.59 %	58.57 %	0.024 %	20	Whole trace

Figure 71 58% results

PWM duty cycle at 81%

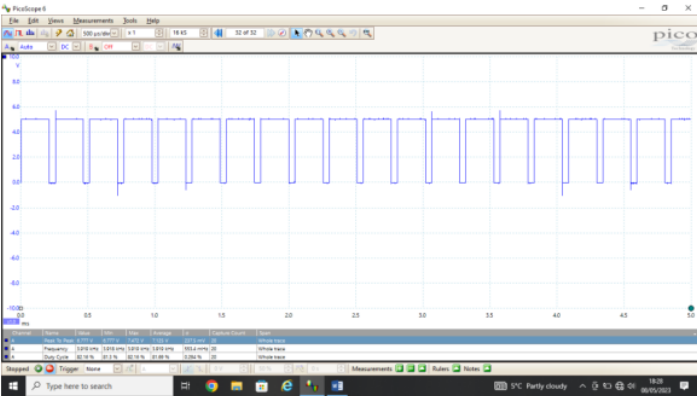


Figure 72 81%

Channel	Name	Value	Min	Max	Average	σ	Capture Count	Span
A	Peak To Peak	6.777 V	6.777 V	7.472 V	7.125 V	237.5 mV	20	Whole trace
A	Frequency	3.919 kHz	3.918 kHz	3.919 kHz	3.919 kHz	553.4 mHz	20	Whole trace
A	Duty Cycle	82.16 %	81.3 %	82.16 %	81.69 %	0.294 %	20	Whole trace

Figure 73 81% results

PWM duty cycle at 16%

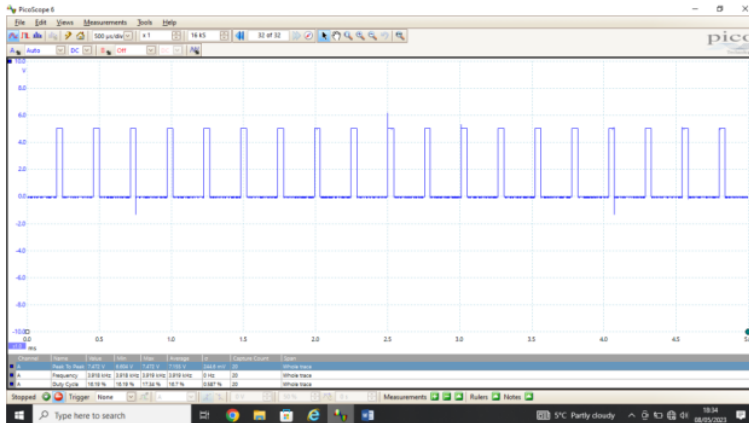


Figure 74 16%

Channel	Name	Value	Min	Max	Average	σ	Capture Count
A	Peak To Peak	7.472 V	6.604 V	7.472 V	7.155 V	244.6 mV	20
A	Frequency	3.918 kHz	3.918 kHz	3.919 kHz	3.919 kHz	0 Hz	20
A	Duty Cycle	16.19 %	16.19 %	17.34 %	16.7 %	0.587 %	20

Figure 75 16% results

The Arduino Serial Plotter Testing can be seen below for the Input from sensor exceeds the setpoint, Input from sensor fall below setpoint and out reacts, Input equals setpoint and output, Input fall 2cm below setpoint and output:

Input from sensor just exceeds the setpoint

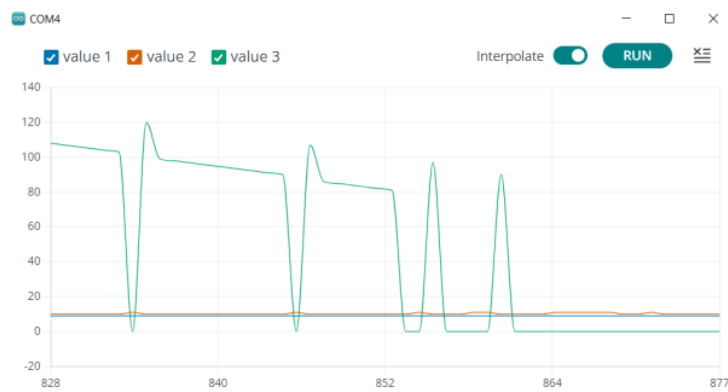
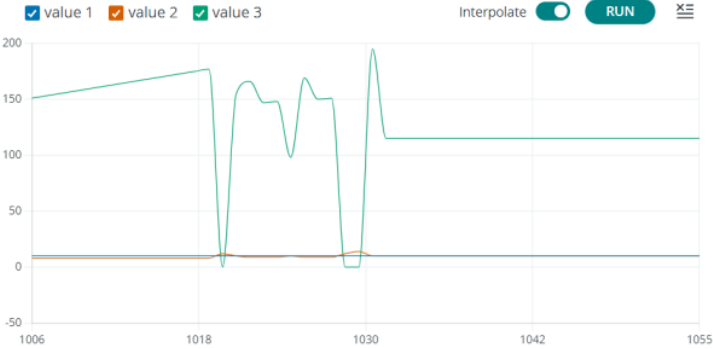


Figure 76 Exceeds.

Input fall below setpoint and out reacts



Good evidence of PID process from Arduino graph

Figure 77 out reacts

Input equals setpoint and output hold the position



Good graph

Figure 78 equals

Input fall 2cm below setpoint and output reacts by increasing toward its max value of 255

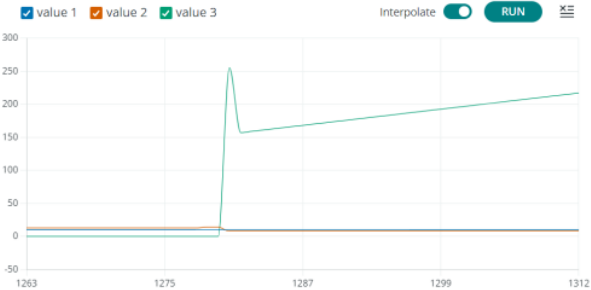


Figure 79 2cm below

LabVIEW Test

LabVIEW front end display

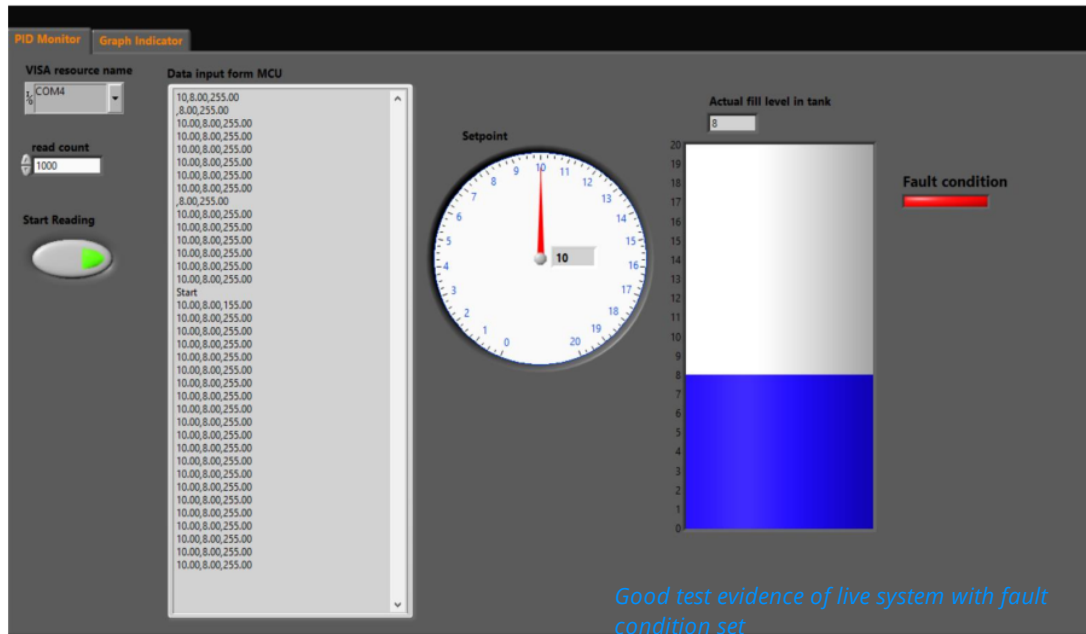


Figure 80 front display

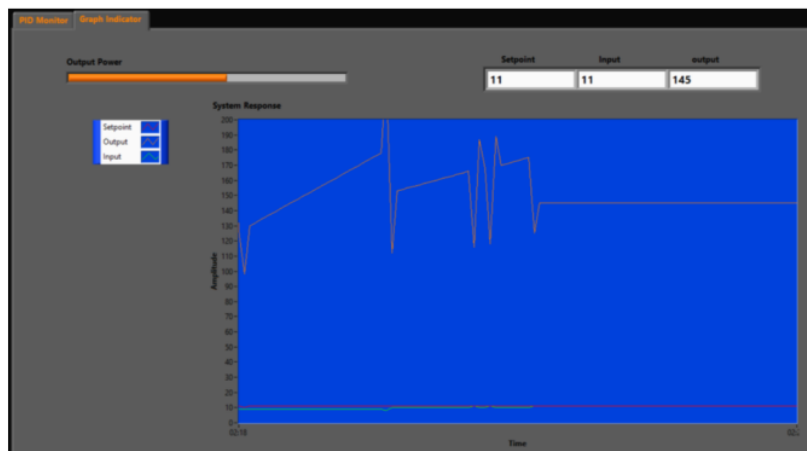
The image demonstrates the values of setpoint and input from sensor, with the serial values from Arduino in the scrolling pane. The data from the Arduino are the actual values from the PID control system; the setpoint, input and output values respectfully.

The dial displays the actual setpoint from the Arduino, and the tank level is the value from the sensor via Arduino.

The fault indicator is activated because the setpoint for the fault is $< 10\text{cm}$ in the tank, as indicated in the LabVIEW display.

Good explanation

LabVIEW response graph



Good evidence of LabVIEW graph

Figure 81 response graph

The graph demonstrates the response of the system; the output is steady and holding the tank level equal to the setpoint.

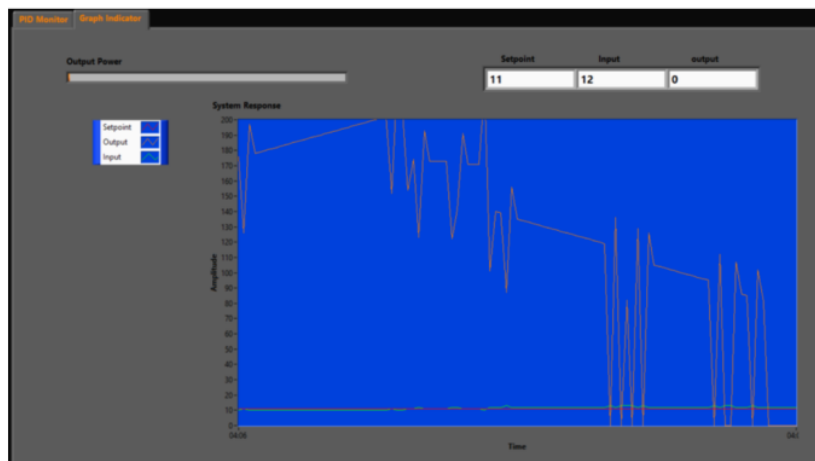


Figure 82 steady output response

The graph demonstrates the system response when the tank level decreases, and the output increases to raise the tank level. When the tank level reaches the setpoint the output can be seen to decrease and when the level is close to the setpoint, the output goes to zero.

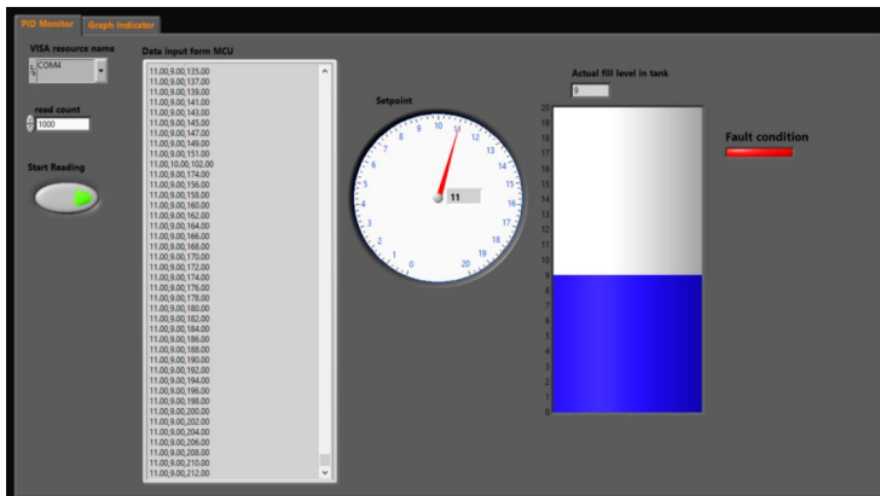
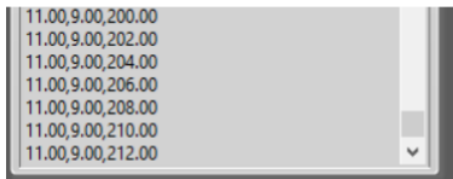


Figure 83 below 10cm

The display indicates the setpoint on the dial and the tank level in the level indicator that shows the level is below 10cm. The fault alarm indicator is activated due to the tank level being below 10cm.

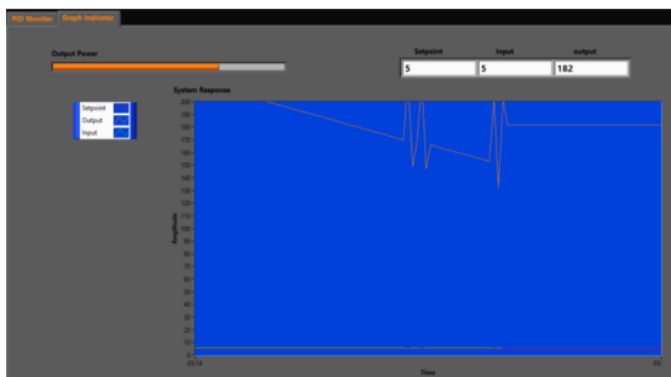
The actual data from the Arduino is corroborated in the data scrolling window as demonstrated. The setpoint is 11cm, the input from the level sensor is 9cm, and the output is driving the level toward the setpoint at a value near the maximum (255) duty cycle.



Evidence indicating live data stream

Figure 84 11cm

Setpoint at 5cm



Evidence of live system function

Figure 85 5cm

Setpoint	Input	output
5	5	182

Figure 86 setpoint, input and output

5.00,5.00,182.00
5.00,5.00,182.00
5.00,5.00,182.00

Figure 87 results

Setpoint at 10cm

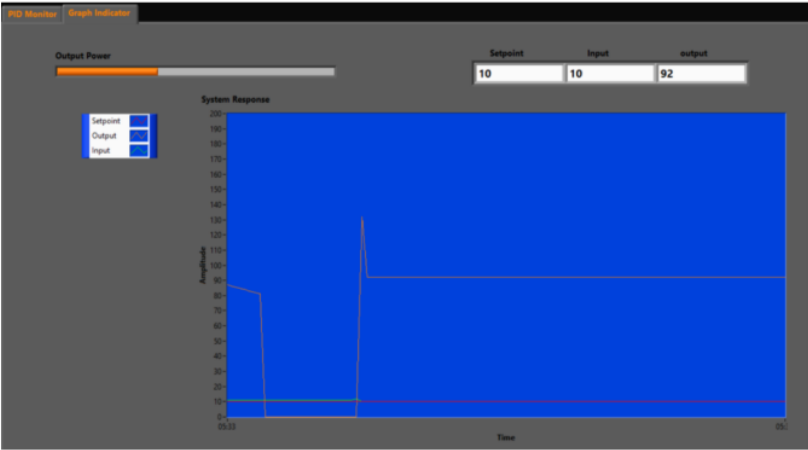


Figure 88 setpoint 10cm

Setpoint	Input	output
10	10	92

Figure 89 setpoint, input and output

10.00,10.00,92.00
10.00,10.00,92.00
10.00,10.00,92.00
10.00,10.00,92.00

Figure 90 results

Setpoint at 12cm

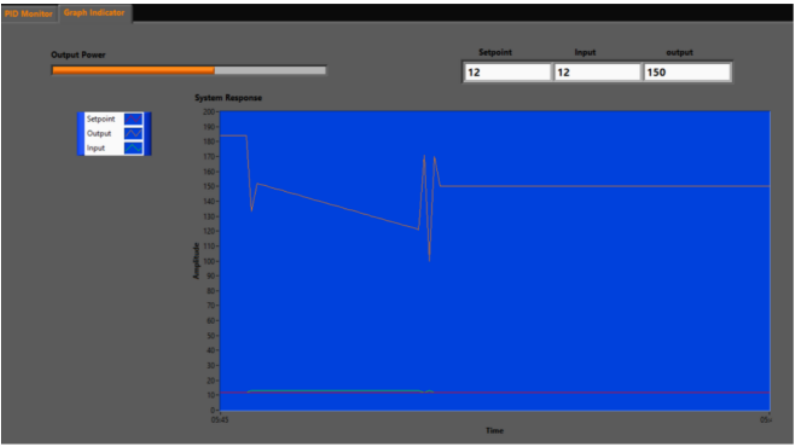


Figure 91 setpoint 12cm



Figure 92 setpoint, input and output

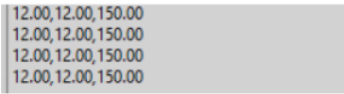


Figure 93 output

Setpoint at 15cm

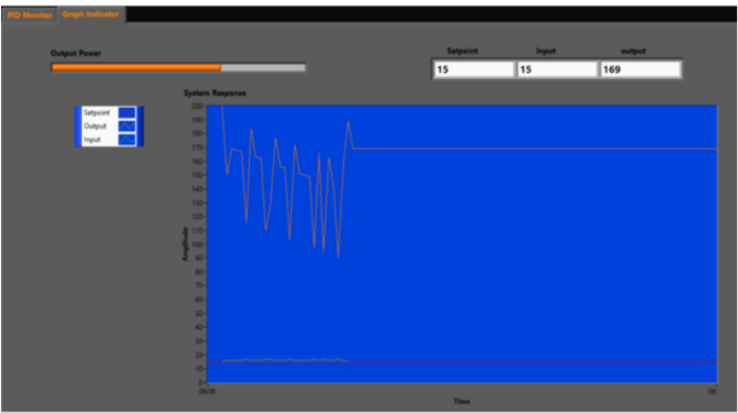


Figure 94 setpoint 15cm

Setpoint	Input	output
15	15	169

Figure 95 setpoint, input and output

```
15.00,15.00,169.00
15.00,15.00,169.00
15.00,15.00,169.00
15.00,15.00,169.00
```

Figure 96 output

Very good evidence of functional tests

Table of results

Good test table

Test No	Set value CM	Expected Value CM	Actual Value CM	Error CM	PWM Duty Cycle Value	LabVIEW Fault Detection	Pass / Fail						
1	5	5	4	1	<table><tr><th>Setpoint</th><th>Input</th><th>output</th></tr><tr><td>5</td><td>5</td><td>182</td></tr></table>	Setpoint	Input	output	5	5	182	Fault condition	pass
Setpoint	Input	output											
5	5	182											
2	10	10	9	1	<table><tr><th>Setpoint</th><th>Input</th><th>output</th></tr><tr><td>10</td><td>10</td><td>92</td></tr></table>	Setpoint	Input	output	10	10	92	Fault condition	pass
Setpoint	Input	output											
10	10	92											
3	12	12	11	1	<table><tr><th>Setpoint</th><th>Input</th><th>output</th></tr><tr><td>12</td><td>12</td><td>150</td></tr></table>	Setpoint	Input	output	12	12	150	Fault condition	pass
Setpoint	Input	output											
12	12	150											
4	15	15	14	1	<table><tr><th>Setpoint</th><th>Input</th><th>output</th></tr><tr><td>15</td><td>15</td><td>169</td></tr></table>	Setpoint	Input	output	15	15	169	Fault condition	pass
Setpoint	Input	output											
15	15	169											

Results demonstrate that the level control system is working according to the initial specifications; however, there is an error of 1cm which can be either eliminated through the program or recalibrating the tank level.

The fault indicator activated with level values from the sensor of less than 10cm; equal to 10cm is enough to cancel the fault indicator.

The evidence indicates the functions of the controller, with random setpoints and disturbances are correct as the tuning allowed. For the purpose of this level system the accuracy meets requirements.

7 - Discussion, Conclusions and Recommendations

First person

Overall, the project was a success as it did the specifications, I laid out for it before starting the project. Each prototype worked as intended and where all tested appropriately. The final Prototype did measure the distance from the ultrasonic sensor and gave that input into the MCU where the program and the Labview worked together to fill the tank appropriately to the correct amount by using PWM to get it to the setpoint I set for the tank level to aim for and the Labview gave a easy to read vi screen that was appropriate for its application with a visually aid of the water tank. The final prototype was fully tested by opening the valve to see if the tank can maintain a setpoint value which it did and also when the valve was shut then stopped filling the tank so that it would remain on its setpoint. The system was very accurate with a slight error from around 1cm at most and as it's a small tank the error that I had as a result was an acceptable amount.

To make the system better more fine tuning to eradicate as much error as possible would be good and also a protector for the sensor to stop it from getting any water damage. From doing this assignment my skills using C++ coding has dramatically improved and so has my ability to use the Labview interface. Adding the protector for the sensor would also increase the maintenance cost of the system as the sensor will be less likely to get damaged. The solution that I chose was definitely the more ethical and environmentally friendly version of the solution as it was much cheaper to buy and also to create the parts needed. Through the project I made sure to work safely at all times and worked with ethical and professional conduct testing the prototypes constantly to eradicate risk of failure. The project was finished in time, and I was within my time restraints of the project meaning that the Gant chart that I created was accurate and trustworthy.

Good informative conclusion with valid recommendations for improvement

8 – Reference list and Bibliography

Cytron Technologies, 2022. *Cytron Technologies*. [Online]

Available at: <https://www.cytron.io/p-10amp-5v-30v-dc-motor-driver>

[Accessed 23 05 23].

eTechnophiles, 2023. *eTechnophiles*. [Online]

Available at: <https://www.etechnophiles.com/how-to-change-pwm-frequency-of-arduino-mega/>

[Accessed 23 05 23].

Arduino Store, 2023. *Arduino Store*. [Online]

Available at: https://store.arduino.cc/products/arduino-mega-2560-rev3?srltid=AR57-fDuC7FCzl0So8puslIRxTnHktW8OwA_uG7xQFF7mFcCGOObrLI8_aw

[Accessed 16 05 23].

NATIONAL INSTRUMENTS CORP, 2023. *NATIONAL INSTRUMENTS CORP*. [Online]

Available at: <https://www.ni.com/en-gb/shop/product/labview.html>

[Accessed 23 05 23].

PLC City, 2023. *PLC City*. [Online]

Available at: <https://www.plc-city.com/shop/en/siemens-software-tia-portal-v17-simatic-step-7-professional.html>

[Accessed 16 05 23].

RS Hydro, 2023. *RS Hydro*. [Online]

Available at: <https://www.rshydro.co.uk/water-level-sensors/siemens-ultrasonic-level/siemens-ultrasonic-level-transducers/st-h-transducer/>

[Accessed 16 05 23].

RS, 2023. *RS Components*. [Online]

Available at: <https://uk.rs-online.com/web/p/plcs-programmable-logic-controllers/8624468>

[Accessed 16 05 23].

The Pi Hut, 2023. *The Pi Hut*. [Online]

Available at: https://thepihut.com/products/ultrasonic-distance-sensor-hcsr04?variant=1054704288¤cy=GBP&utm_medium=product_sync&utm_source=google&utm_content=sag_organic&utm_campaign=sag_organic&srltid=AR57-fD5t4TcdySwEHa5BlBfB1Ztypi4EYh_o2mCYihWVeP9me7PiI7mX-U

[Accessed 23 05 23].

A good number of ref sources

9 – Appendix



```
read_data.ino
1  int diginputpin = 8;
2  int labview = 0; //initialise to 0
3
4  void setup() {
5      Serial.begin(9600); //baud rate for serial comms
6      pinMode(diginputpin, INPUT); //dig input pin declared
7
8  }
9
10 void loop() {
11     labview = digitalRead(diginputpin);
12     Serial.println(labview);
13     delay(100);
14 }
15
16
```

Figure 97 Code for Prototype 1



```
Analogue_read.ino
1  int sensorValue = 0; //declare analogue sensor variable with designated pin no
2  int diginputpin = 8; //declare dig input variable with designated pin no
3  int readinputpin8 = 0; //initialise to 0
4
5  void setup() {
6      Serial.begin(9600);
7      pinMode(diginputpin, INPUT); //dig pin 8 declared as INPUT
8  }
9
10 void loop() {
11     readinputpin8 = digitalRead(diginputpin);
12     sensorValue = analogRead(A0);
13     Serial.print(sensorValue);
14     Serial.print(",");
15     Serial.println(readinputpin8);
16 }
17
```

Figure 98 Prototype 2 Arduino code.

```

1  #include <LiquidCrystal.h> //Please replace the single quote characters (') with the parenthesis character (<>)
2
3  LiquidCrystal lcd(8, 9, 4, 5, 6, 7); // Creates an LCD object. Parameters: (rs, enable, d4, d5, d6, d7)
4
5  const int trigPin = 23; //10
6  const int echoPin = 25; //9
7  long duration;
8  int distanceCm, distanceInch;
9
10 void setup() {
11
12  lcd.begin(16,2); // Initializes the interface to the LCD screen, and specifies the dimensions (width and height) of the display
13  pinMode(trigPin, OUTPUT);
14  pinMode(echoPin, INPUT);
15
16  }
17
18 void loop() {
19
20  digitalWrite(trigPin, LOW);
21  delayMicroseconds(2);
22  digitalWrite(trigPin, HIGH);
23  delayMicroseconds(10);
24  digitalWrite(trigPin, LOW);
25
26  duration = pulseIn(echoPin, HIGH);
27  distanceCm= duration*0.034/2;
28  distanceInch = duration*0.0133/2;
29

```

Figure 99 Prototype 4 code 1.

```

30  lcd.setCursor(0,0); // Sets the location at which subsequent text written to the LCD will be displayed
31  lcd.print("Distance: "); // Prints string "Distance" on the LCD
32  lcd.print(distanceCm); // Prints the distance value from the sensor
33  lcd.print("  cm");
34  delay(10);
35  lcd.setCursor(0,1);
36  lcd.print("Distance: ");
37  lcd.print(distanceInch);
38  lcd.print("inch");
39  delay(10);
40
41  }

```

Figure 100 Prototype 4 code 2.

```

1  #include <PID_v1.h>
2  #include <LiquidCrystal.h>
3
4  //Definitions
5  #define PUMP 45 // Output pin for pump
6
7  int analogInPin2 = A8;
8  const int analogInPin = A7; // Analog input pin that the potentiometer is attached to
9
10 int sensorValue = 0; // value read from the pot
11 int sensorValue2 = 0;
12 LiquidCrystal lcd(8, 9, 4, 5, 6, 7); //set up LCD
13
14
15 //Setup PID
16 double Setpoint, Input, Output; //1/O for PID
17 double Kp=10, Ki=5, Kd=2; //original consKp=1, consKi=0.05, consKd=0.25, Conservative Turning,20,10,10
18 PID myPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, DIRECT); //Initialize PID
19
20 //Interface
21 //int timeCounter;
22 void setup()
23 {
24 // start serial port for distance readings
25 Serial.begin(9600);
26 Serial.println("Start");
27
28
29 // read the analog in value:
30 sensorValue = analogRead(analogInPin); //reads value from potentiometer
31 sensorValue2 = analogRead(analogInPin2);
32 sensorValue = map(sensorValue, 0, 1023, 100, 0);
33 sensorValue2 = map(sensorValue2, 0, 1023, 100, 0);
34 Output = map(Output, 0, 1023, 255, 0);
35
36 //PID Setup

```

Figure 101 Prototype 5 code 1

```

36 //PID Setup
37 myPID.SetMode(AUTOMATIC);
38 //TCCR2B = TCCR2B & 0b11111000 | 0x01; //adjust the PWM Frequency, note: this changes timing like delay()
39
40 //Setup Pins
41 pinMode(PUMP, OUTPUT); // Output for pump speed, 0 to 255
42
43 //Setup LCD 16x2 and display startup message
44 lcd.begin(16, 2);
45 lcd.setCursor(0,1);
46 lcd.print(" Starting Up");
47 delay(3000);
48 lcd.clear();
49 }
50
51 void loop()
52 {
53 delay(200);
54 //Get distance and give it to the PID input
55 sensorValue = analogRead(analogInPin);
56 sensorValue2 = analogRead(analogInPin2);
57 sensorValue = map(sensorValue, 0, 1023, 0, 50);
58 sensorValue2 = map(sensorValue2, 0, 1023, 0, 50);
59 Setpoint=(sensorValue2);
60 Input = (sensorValue);
61
62 //print out info to LCD
63 lcd.setCursor(0,0);
64 lcd.print("P:");
65
66 lcd.print((int)Input);
67 lcd.print(" ");
68 lcd.setCursor(7,0);
69 lcd.print("Pwm:");
70

```

Figure 102 prototype 5 code 2

```

71 lcd.print(output);
72 lcd.print(" ");
73 lcd.setCursor(1,1);
74 lcd.print("Set:");
75 lcd.print((int)Setpoint);
76 lcd.print(" ");
77
78 //Compute PID value
79 myPID.compute();
80 //write PWM to output
81 analogWrite(PUMP,output);
82
83 Serial.print(Setpoint);
84 Serial.print(",");
85 Serial.print(input);
86 Serial.print(",");
87 Serial.println(output);
88
89 } //end loop
90

```

Figure 103 Prototype 5 code 3

```

1  #include <PID_v1.h>
2  #include <LiquidCrystal.h>
3
4  //Definitions
5  #define PUMP 45 // Output pin for pump
6  #define echoPin 29 // attach pin D2 Arduino to pin Echo of HC-SR04
7  #define trigPin 27 //attach pin D3 Arduino to pin Trig of HC-SR04
8
9  int analogInPin2 = A8;
10
11
12  int sensorValue = 0; // value read from the pot
13  int sensorValue2 = 0;
14  LiquidCrystal lcd(8, 0, 4, 5, 6, 7); //set up LCD
15  // defines variables
16  long duration; // variable for the duration of sound wave travel
17  int distance; // variable for the distance measurement
18
19  //Setup PID
20  double Setpoint, Input, Output; //I/O for PID
21  double Kp=10, Ki=5, Kd=2; //original consKp=1, consKi=0.05, consKd=0.25, Conservative Turning,20,10,10
22  PID myPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, DIRECT); //Initialize PID
23
24  //interface
25  //int timeCounter;
26  void setup()
27  {
28  pinMode(trigPin, OUTPUT); // Sets the trigPin as an OUTPUT
29  pinMode(echoPin, INPUT); // Sets the echoPin as an INPUT
30  Serial.begin(9600); // // Serial communication is starting with 9600 of baudrate speed
31  Serial.println("Ultrasonic Sensor HC-SR04 Test"); // print some text in Serial Monitor
32  Serial.println("with Arduino UNO R3");
33
34  // start serial port for distance readings
35  Serial.begin(9600);
36  Serial.println("Start");

```

Figure 104 Prototype 6 code 1.

```

39 // read the analog in value:
40 sensorValue2 = analogRead(analogInPin2);
41 sensorValue = map(sensorValue, 0, 1023, 100, 0);
42 sensorValue2 = map(sensorValue2, 0, 1023, 100, 0);
43 Output = map(Output, 0, 1023, 255, 0);
44
45 //PID Setup
46 myPID.SetMode(AUTOMATIC);
47 //TCCR2B = TCCR2B & 0b11111000 | 0x01; //adjust the PWM Frequency, note: this changes timing like delay()
48
49 //Setup Pins
50 pinMode(PUMP, OUTPUT); // Output for pump speed, 0 to 255
51
52 //Setup LCD 16x2 and display startup message
53 lcd.begin(16, 2);
54 lcd.setCursor(0,1);
55 lcd.print(" Starting Up");
56 delay(3000);
57 lcd.clear();
58 }
59
60 void loop()
61 {
62 // Clears the trigPin condition
63 digitalWrite(trigPin, LOW);
64 delayMicroseconds(2);
65 // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
66 digitalWrite(trigPin, HIGH);
67 delayMicroseconds(10);
68 digitalWrite(trigPin, LOW);
69 // Reads the echoPin, returns the sound wave travel time in microseconds
70 duration = pulseIn(echoPin, HIGH);
71 // Calculating the distance
72 distance = duration * 0.034 / 2; // Speed of sound wave divided by 2 (go and back)
73 // Displays the distance on the Serial Monitor

```

Figure 105 Prototype 6 code 2.

```

76 delay(200);
77 //Get distance and give it to the PID input
78 sensorValue2 = analogRead(analogInPin2);
79
80 sensorValue2 = map(sensorValue2, 0, 1023, 0, 20);
81 Setpoint=(sensorValue2);
82 Input = (distance);
83
84 //print out info to LCD
85 lcd.setCursor(0,0);
86 lcd.print("P:");
87
88 lcd.print((int)Input);
89 lcd.print(" ");
90 lcd.setCursor(7,0);
91 lcd.print("PWM:");
92
93 lcd.print(Output);
94 lcd.print(" ");
95 lcd.setCursor(1,1);
96 lcd.print("Set:");
97 lcd.print((int)Setpoint);
98 lcd.print(" ");
99
100 //Compute PID value
101 myPID.Compute();
102 //write PWM to output
103 analogWrite(PUMP,Output);
104
105 Serial.print(Setpoint);
106 Serial.print(",");
107 Serial.print(Input);
108 Serial.print(",");
109 Serial.println(Output);
110
111 }//end loop

```

Figure 106 Prototype 6 code 3.

```

1  #include <PID_v1.h>
2  #include <LiquidCrystal.h>
3  ///////////////////////////////////////////////////////////////////
4  //Definitions
5  #define echoPin 22 // attach pin D2 Arduino to pin Echo of HC-SR04
6  #define trigPin 23 //attach pin D3 Arduino to pin Trig of HC-SR04
7  #define PUMP 45      // Output pin for pump to Cytron board
8
9  // defines variables
10 long duration; // variable for the duration of sound wave travel
11 int distance; // variable for the distance measurement
12 int analogInPin2 = A8; //set point value
13 //const int analogInPin = A7; // Analog input pin that the potentiometer is attached to
14
15 int sensorValue = 0; // value read from the pot
16 int sensorValue2 = 0;
17 LiquidCrystal lcd(8, 9, 4, 5, 6, 7); //set up LCD
18 ///////////////////////////////////////////////////////////////////
19
20 //Setup PID
21 double Setpoint, Input, Output; //I/O for PID
22 double Kp=30, Ki=10, Kd=2; //original consKp=1, consKi=0.05, consKd=0.25, Conservative Turning,20,10,10
23 PID myPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, REVERSE); //Initialize PID
24 ///////////////////////////////////////////////////////////////////
25 //interface
26 //int timeCounter;
27 void setup()
28 {
29   pinMode(trigPin, OUTPUT); // Sets the trigPin as an OUTPUT
30   pinMode(echoPin, INPUT); // Sets the echoPin as an INPUT
31   // start serial port for distance readings
32   Serial.begin(9600);
33   Serial.println("Start");
34 }
35
36 // read the analog in value:

```

Figure 107 Prototype 7 code 1.

```

37 //sensorValue = analogRead(analogInPin);
38 sensorValue2 = analogRead(analogInPin2);
39 //sensorValue = map(sensorValue, 0, 1023, 100, 0);
40 sensorValue2 = map(sensorValue2, 0, 1023, 100, 0);
41 Output = map(Output, 0, 1023, 255, 0);
42
43 //PID Setup
44 myPID.SetMode(AUTOMATIC);
45 //TCCR2B = TCCR2B & 0b11111000 | 0x01; //adjust the PWM Frequency, note: this changes timing like delay()
46
47 //Setup Pins
48 pinMode(PUMP, OUTPUT); // Output for motor speed, 0 to 255
49
50 //Setup LCD 16x2 and display startup message
51 lcd.begin(16, 2);
52 lcd.setCursor(0,1);
53 lcd.print(" Starting Up");
54 delay(3000);
55 lcd.clear();
56 }
57 //////////////////////////////////////////////////////////////////end setup..runs onlu once////////////////////////////////////
58 void loop()
59 {
60   // Clears the trigPin condition
61   digitalWrite(trigPin, LOW);
62   delayMicroseconds(2);
63   // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
64   digitalWrite(trigPin, HIGH);
65   delayMicroseconds(10);
66   digitalWrite(trigPin, LOW);
67   // Reads the echoPin, returns the sound wave travel time in microseconds
68   duration = pulseIn(echoPin, HIGH);
69   // Calculating the distance
70   distance = duration * 0.034 / 2; // Speed of sound wave divided by 2 (go and back)
71
72   //sensorValue = analogRead(analogInPin);

```

Figure 108 Prototype 7 code 2.

```

73  sensorValue2 = analogRead(analogInPin2);
74  // sensorValue = map(sensorValue, 0, 1023, 255, 0); //map 10 bit to 8 bit
75  sensorValue2 = map(sensorValue2, 0, 1023, 255, 0);
76  Setpoint=sensorValue2;
77  Input = (distance);
78
79  if (Input >= 10){
80      Input = 10;
81  }
82
83  //print out info to LCD
84  lcd.setCursor(0,0);
85  lcd.print("H:");
86  lcd.print((int)Input);
87
88  lcd.print(" ");
89  lcd.setCursor(7,0);
90
91  lcd.print("PWM:");
92  lcd.print(Output);
93  lcd.print(" ");
94  lcd.setCursor(1,1);
95
96  lcd.print("Set:");
97  lcd.print((int)Setpoint);
98  lcd.print(" ");
99
100 //Compute PID value
101 myPID.Compute();
102 //write PWM to output
103 analogWrite(PUMP,Output);
104
105 if (Output <= 50)
106 {
107     Output = 0;
108 }
109
110 Serial.print(Setpoint);
111 Serial.print(",");
112 Serial.print(Input);
113 Serial.print(",");
114 Serial.println(Output);
115
116 } //end loop

```

Figure 109 Prototype 7 code 3.

```

1  #include <PID_v1.h>
2  #include <LiquidCrystal.h>
3
4  //Definitions
5  #define echoPin 22 // attach pin D2 Arduino to pin Echo of HC-SR04
6  #define trigPin 23 //attach pin D3 Arduino to pin Trig of HC-SR04
7  #define PUMP 45      // Output pin for pump
8
9  // defines variables
10 const int max_value = 20; //max height of tank
11 long duration; // variable for the duration of sound wave travel
12 int distance; // variable for the distance measurement
13 int analogInPin2 = A8; //set point value
14 //const int analogInPin = A7; // Analog input pin that the potentiometer is attached to
15
16 int sensorValue = 0; // value read from the pot
17 int sensorValue2 = 0;
18 LiquidCrystal lcd(8, 9, 4, 5, 6, 7); //set up LCD
19
20
21 //Setup PID
22 double Setpoint, Input, Output; //I/O for PID
23 double Kp=30, Ki=10, Kd=2; //original Kp=1, Ki=0.05, Kd=0.25
24 PID myPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, DIRECT); //Initialize PID
25
26 //interface
27 //int timeCounter;
28 void setup()
29 {
30   pinMode(trigPin, OUTPUT); // Sets the trigPin as an OUTPUT
31   pinMode(echoPin, INPUT); // Sets the echoPin as an INPUT
32   // start serial port for temperature readings
33   Serial.begin(9600);
34   Serial.println("Start");
35
36   // read the analog in value:
37   //sensorValue = analogRead(analogInPin);
38   sensorValue2 = analogRead(analogInPin2);
39   // sensorValue = map(sensorValue, 0, 1023, 100, 0);
40   sensorValue2 = map(sensorValue2, 0, 1023, 100, 0);
41   Output = map(sensorValue2, 0, 1023, 255, 0);
42
43   //PID Setup
44   myPID.SetMode(AUTOMATIC);

```

Figure 110 Prototype 8 code 1

```

45 //TCCR2B = TCCR2B & 0b11111000 | 0x01; //adjust the PWM Frequency, note: this changes timing like delay()
46 TCCR5B = TCCR5B & 0b11111000 | 0b00000010; // for PWM frequency of 3921.16 Hz
47
48 //Setup Pins
49 pinMode(PUMP, OUTPUT); // Output for pump speed, 0 to 255
50
51 //Setup LCD 16x2 and display startup message
52 lcd.begin(16, 2);
53 lcd.setCursor(0,1);
54 lcd.print(" Starting Up");
55 delay(3000);
56 lcd.clear();
57 }
58
59 void loop()
60 {
61 // Clears the trigPin condition
62 digitalWrite(trigPin, LOW);
63 delayMicroseconds(2);
64 // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
65 digitalWrite(trigPin, HIGH);
66 delayMicroseconds(10);
67 digitalWrite(trigPin, LOW);
68 // Reads the echoPin, returns the sound wave travel time in microseconds
69 duration = pulseIn(echoPin, HIGH);
70 // Calculating the distance
71 distance = duration * 0.034 / 2; // Speed of sound wave divided by 2 (go and back
72 distance = max_value - distance;
73
74 //sensorValue = analogRead(analogInPin);
75 sensorValue2 = analogRead(analogInPin2);
76 //sensorValue = map(sensorValue, 0, 1023, 255, 0);
77 sensorValue2 = map(sensorValue2, 0, 1023, 255, 0);
78 Setpoint=sensorValue2;
79 Setpoint = Setpoint -2;
80 Input = distance;
81 if (Input >= 20){
82 Input = 20;
83 }
84
85 //print out info to LCD
86 lcd.setCursor(0,0);
87 lcd.print("H:");
88 lcd.print((int)Input);|

```

Figure 111 Prototype 8 code 2

```

89
90     lcd.print(" ");
91     lcd.setCursor(7,0);
92
93     lcd.print("PWM:");
94     lcd.print(Output);
95     lcd.print(" ");
96     lcd.setCursor(1,1);
97
98     lcd.print("Set:");
99     lcd.print((int)Setpoint);
100    lcd.print(" ");
101
102    //Compute PID value
103    myPID.Compute();
104    //write PWM to output
105    analogWrite(PUMP,Output);
106
107    if (Output <= 80)
108    {
109        Output = 0;
110    }
111
112    Serial.print(Setpoint);
113    Serial.print(",");
114    Serial.print(Input);
115    Serial.print(",");
116    Serial.println(Output);
117    delay (500);
118 } //end loop
119

```

Figure 112 Prototype 8 code 3.

FINAL GRADE

GENERAL COMMENTS

Instructor

63/100

PAGE 1

PAGE 2

Text Comment. The contents table should be more detailed

PAGE 3

PAGE 4

PAGE 5

Text Comment. Good extensive list of figures

PAGE 6

Text Comment. Avoid first person terms

Text Comment. Introduction is informative. There are grammatical errors

PAGE 7

Text Comment. Grammatical erros

Text Comment. Good image to support requirements

PAGE 8

Text Comment. Good supporting image

Text Comment. Good image

PAGE 9

Text Comment. First person used

Text Comment. Good Gantt chart that includes comparison timelines

Text Comment. Good key

Text Comment. Good test plan

PAGE 10

Text Comment. Good evaluation

Text Comment. Good supporting ref source and image

PAGE 11

Text Comment. Good evaluation

Text Comment. This proves your point

Text Comment. Good argument for chosen solution

PAGE 12

Text Comment. Good effort here. Components are identified and explained with ref sources

Text Comment. Good point made supported with ref source

PAGE 13

Text Comment. This should be deliverables

Text Comment. Grammatical errors

Text Comment. Good informative block diagram

PAGE 14

Text Comment. Good evidence of prototype test outcome

Text Comment. Proof of concept here with serial link operational

Text Comment. First person

Text Comment. Good evidence

Text Comment. A very good prototype progression that is significant for your final outcome.

Text Comment. Good evidence

Text Comment. Good evidence and explanation

Text Comment. Prototype 3 demonstrates good progress with complexity

Text Comment. Good evidence

Text Comment. Good explanation and evidence

Text Comment. Main serial function blocks are identified

Text Comment. Good evidence of fault condition blocks

Text Comment. Good evidence and explanation

Text Comment. Good explanation of program sample

PAGE 23

PAGE 24

Text Comment. First person

Text Comment. Good explanation and supporting image

PAGE 25

Text Comment. Good explanation of sample code

PAGE 26

Text Comment. Good evidence

PAGE 27

PAGE 28

Text Comment. Good progression demonstrated in this prototype

PAGE 29

Text Comment. Good evidence

PAGE 30

PAGE 31

PAGE 32

Text Comment. Good supporting image of test setup

PAGE 33

PAGE 34

Text Comment. PWM output control and frequency is demonstrated

PAGE 35

PAGE 36

Text Comment. Results are displayed on LCD

PAGE 37

Text Comment. Good LabVIEW front end HMI

Text Comment. Good block program

PAGE 38

PAGE 39

PAGE 40

Text Comment. Good explanation of LabVIEW program

PAGE 41

Text Comment. Good commented Arduino C++ code

PAGE 42

PAGE 43

PAGE 44

PAGE 45

Text Comment. Good evidence of PID process from Arduino graph

Text Comment. Good graph

PAGE 46

Text Comment. Good test evidence of live system with fault condition set

Text Comment. Good explanation

PAGE 47

Text Comment. Good evidence of LabVIEW graph

PAGE 48

Text Comment. Evidence indicating live data stream

Text Comment. Evidence of live system function

PAGE 49

PAGE 50

PAGE 51

Text Comment. Vvery good evidence of functional tests

Text Comment. Good test table

PAGE 52

Text Comment. First person

Text Comment. Good informative conclusion with valid recommendations for improvement

PAGE 53

Text Comment. A good number of ref sources

PAGE 54

PAGE 55

PAGE 56

PAGE 57

PAGE 58

PAGE 59

PAGE 60

PAGE 61

PAGE 62

PAGE 63

GRADING FORM: CSG-RUBIC



RESEARCH-INFORMED LITERATURE



Background research includes the process requirements. This could have been a wider research into similar process control systems. The research for the solutions includes the references.

10/20

KNOWLEDGE AND UNDERSTANDING



Spec includes all the significant points, and goes further with identifying the spec for components for the final solution requirements. 6/10
Solutions considered are valid, and are explained with good arguments. The chosen solution is justified by performance and cost evaluation. 6/10
Explanations include significant relevant points, however, the quality of the writing could be improved.

12/20

ANALYSIS



Very good structured test method used with supporting evidence. The testing is extensive and this is evident in the report. 8/10
Conclusion and recommendations include good relevant points. 6/10

14/20

PRACTICAL APPLICATION AND DEP.



Good planning methods demonstrated using Gantt chart for timeline plan and evaluation, and test plan using tables with ref to evidence in appendix. 3.5/5

Solutions considered are good arguments for alternative solutions. 6/10

Implementation methods are demonstrated using prototypes, with supporting evidence and explanations. 7.5/10

17/25

SKILLS FOR PROFESSIONAL PRAC.



The writing quality could be better

The report has a good structure and sequence

10/15