




Universitetet
i Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study programme/specialisation: Petroleum Technology Natural Gas Technology	Spring semester, 2020 Open
Author: Kristoffer Sæby	 (signature of author)
Programme coordinator: Supervisor(s): Rabjenja Afamintentiosa, Hans-Joakim Skadsem, Rune Wiggo Time	
Title of master's thesis: Measurement and Control of a Flow Loop System	
Credits: 30	
Keywords: Arduino Process Control Measurement and Control Flow Loop	Number of pages: 56 + supplemental material/other: 12 Stavanger, 13/7/2020 date/year

Measurement and Control of a Flow Loop System

Kristoffer Sæby
228035

13th July 2020

University of Stavanger

This page is intentionally left blank

Abstract

Automation of processes is key to perfecting them. Several systems exist in order to simply this process, although they may be simple to use most are however, expensive to acquire. This thesis attempts to program an Arduino board to record flow, pressure and temperature data to a laptop. This is compared to data that is logged by the Pasco System currently implemented at the University of Stavanger.

This page is intentionally left blank

Acknowledgements

First, I would like to thank Rabenja Afamintentiosa (Bejna), Hans Joakim Skadsem, and Rune Time for giving me the opportunity and guiding me through a project that was both interesting and challenging, while at the same time I had full control over. There are several others I would like to thank here but I should limit this to friends and family. To all those that have endured my presence throughout my bachelors and masters I give you credit for putting up with me and helping throughout these 5 years! Good luck in the future.

This page is intentionally left blank

Contents

List of Figures	x
List of Tables	xi
Nomenclature	xii
1. Introduction	1
2. Theory	3
2.1. Measurement Systems	3
2.1.1. Sensing Elements	4
2.1.2. Signal Conditioning Element	7
2.1.3. Signal Processing Element	9
2.1.4. Signal Sampling	10
2.1.5. Data Presentation Element	10
2.2. Fluid Flow	11
2.2.1. Single-Phase Flow	11
2.3. Fluid Flow in Pipes	12
2.3.1. Two-Phase Flow	12
2.3.2. Laminar and Turbulent Flow	13
2.4. Pressure Loss in Pipes	16
2.4.1. Friction	17
3. Methodology	18
3.1. Flow Loop	18
3.2. Hardware	19
3.2.1. Sensors	20
3.3. Calibration	22
3.4. Experimental Setup and Procedure	23
3.5. Coding	25
3.5.1. Arduino IDE Code	25
3.5.2. Python Code	25
4. Results	27
4.1. Calibration Results	27
4.1.1. Pressure Sensors	27

4.1.2. Temperature Sensors	28
4.1.3. Electromagnetic Flow Sensor	28
4.2. Flow Loop Tests	29
5. Discussion	36
5.1. Flow Loop Tests	36
5.1.1. Run 1-3	36
5.1.2. Runs 4-9	37
5.1.3. Temperature Changes	38
5.2. Limitations and Improvements	38
5.2.1. Arduino Limitations	38
5.2.2. Sensor Limitations	39
6. Conclusions	40
A. Appendix	43
A.1. Arduino Code	43
A.2. Python Code	44
B. Appendix	45
B.1. Calibration Results	45
B.1.1. Temperature and pressure	45
B.1.2. Raw results from individual runs	46

List of Figures

2.1.	Structure of a measurement system, adapted from [Bently, 2005]	4
2.2.	Structure of an electromagnetic flow meter [Morris and Langari, 2011]	7
2.3.	Aliasing, acquired from [Bently, 2005]	10
2.4.	Flow regimes in horizontal pipes [Beggs and Brill, 1973]	14
2.5.	Flow pattern map proposed by [Mandhane et al., 1974]	16
3.1.	Schematic of flow loop setup [Shouib, 2019]	19
3.2.	Rosemount 3051 differential pressure sensor	20
3.3.	Rosemount temperature transmitter and thermocouple installed in the mixing section	21
3.4.	Techfluid XT5 electromagnetic flow sensor	22
3.5.	Sketch of the wiring for the measurement system	24
4.1.	Final wiring of the proposed measurement system	27
4.2.	Arduino run 1-3	32
4.3.	Pasco run 1-3	32
4.4.	Arduino run 4-6	33
4.5.	Pasco run 4-6	33
4.6.	Arduino run 7-9	34
4.7.	Pasco run 7-9	34
4.8.	Temperature from Arduino runs 3, 6, and 9	35
4.9.	Temperature from Pasco runs 3, 6, and 9	35
5.1.	Comparison of results from both measurement systems: run 2	37
A.1.	Arduino code for reading of sensors	43
A.2.	Python code for reading and logging of incoming data from Arduino	44
A.3.	Python code for graphing of results	44
B.1.	Equations for conversion from volt to pressure and temperature	45
B.2.	Arduino run 1 pressure results	46
B.3.	Pasco run 1 pressure results	46
B.4.	Pasco run 2 pressure results	47
B.5.	Pasco run 2 pressure results	47
B.6.	Arduino run 3 pressure results	48
B.7.	Pasco run 4 pressure results	48

B.8. Arduino run 4 pressure results	49
B.9. Pasco run 4 pressure results	49
B.10. Arduino run 5 pressure results	50
B.11. Pasco run 4 pressure results	50
B.12. Arduino run 6 pressure results	51
B.13. Pasco run 6 pressure results	51
B.14. Arduino run 7 pressure results	52
B.15. Pasco run 7 pressure results	52
B.16. Arduino run 8 pressure results	53
B.17. Pasco run 8 pressure results	53
B.18. Arduino run 9 pressure results	54
B.19. Pasco run 9 pressure results	54

List of Tables

4.1. Pressure results from Arduino in run 1-3	30
4.2. Pressure results from Pasco sensor in run 1-3	30
4.3. Pressure results from run 4-6	30
4.4. Pressure results from Pasco sensor in run 4-6	30
4.5. Pressure results from run 7-9	31
4.6. Pressure results from Pasco sensor in run 7-9	31
4.7. Theroetical pressure drop using Colebrook-White and Swamee-Jain equations	31

Nomenclature

μ	Fluid viscosity	Cp
ρ	Density	kg/m^3
ρ_g	Gas density	kg/m^3
ρ_L	Liquid density	kg/m^3
ε	Relative surface roughness	—
A	Area of a surface	m^2
D	Diameter	m
f	Frequency	Hz
f_D	Darcy friction factor	—
g	Gravity	m/s^2
I	Current	A
L	Length	m
m	Mass flow	kg/s
m_g	Mass flow of gas	kg/s
m_l	Mass flow of liquid	kg/s

P	Pressure	Pa/m^2
P_g	Guage pressure	Pa/m^2
P_{abs}	Absolute pressure	Pa/m^2
P_{atm}	Atmospheric pressure	Pa/m^2
Q	Total volumetric flow rate	L/min
Q_g	Volumetric flow rate of gas phase	L/min
Q_l	Volumetric flow rate of liquid phase	L/min
R	Resistivity	Ω
Re	Reynolds number	—
t	Time	s
u	Velocity	m/s
u_{avg}	Average velocity	m/s
U_{gs}	Superficial gas velocity	m/s
U_{ls}	Superficial liquid velocity	m/s
U_{mix}	Superficial mixture velocity	m/s
V	Voltage	V
z	Height	m

1. Introduction

Engineers are reliant on the ability to measure and control various processes, without one of these elements we are essentially lost. Obtaining reliable data from sensors that measure specific processes is used to understand, improve and perfect a process. The sensors must work in unison to provide accurate and simultaneous data for comparison and analysis.

Production engineers are heavily reliant on the acquisition of data from several sensors at various points in the drill string. Process engineers need to understand what fluid types are entering the separators, piping engineers monitor fluid fractions such that stable flow can be maintained. These are only a few examples of where engineers are reliant on sensors to provide accurate information related to fluids in pipes. The data provided by sensors can be used in order to determine the types of fluids present and the flow regime at hand, and adjustments can be made accordingly. With advancements in technology there are now numerous sensors that provide information to develop a clear idea of the situation we have at hand.

Experimental work has long been a major contributor for understanding of flow regimes and patterns that can develop in pipes. From observations and experimental data several models have been developed [Mandhane et al., 1974]. Flow loops built with transparent pipes allow for key observations to be made while adjusting flow speeds. A flow loop can be used in order to simulate a production string or transport pipeline and perform various tests in single- and multiphase flow situations. It is then desirable to have a system that can acquire and record data that is of adequate quality.

The University of Stavanger operates its own small scale flow loop system. A drawback to this system is the lack of communication between the various sensors that are installed. Each sensor outputs data to a separate display with all operation being manual. The loop has been developed and constructed by previous masters students [Shouib,

2019]. While the system functions and experiments can be carried out, the measurement system implemented is not designed optimally. Currently most data is acquired through the Pasco 850 Universal Interface which can collect data on several variables including pressure and temperature. It is not capable of collecting data related to flow measurements and pressure measurements are not reliable. One of the major drawbacks of the Pasco system is that it is relatively expensive. The aim of this study is therefore to design and construct a measurement system that is significantly cheaper in price, that is able to record data from all available devices to one system. This will be done using an Arduino Uno as a data acquisition device in combination with simple python code to log data which can be used in further processing. Several studies have carried out similar setups [Wu et al., 2017] [Ahmed and John, 2018] [Suresh et al., 2014] [Ate, 2018], however, none have attempted to incorporate the Arduino in a large scale flow loop with industrial type sensors installed.

2. Theory

When designing a measurement system it is important to consider the variables we are trying to obtain. We need to understand these variables we want to control and those that are being measured in the system. In this manner we can break down complicated equations into their individual components. By simplifying these equations into their individual components it is possible to introduce these variables into the respective equations. Once introduced into the equations it is possible to make empirical conclusions from the observations and data we have at hand.

2.1. Measurement Systems

Measurement systems are used in all aspects of day to day life that we may sometimes take them for granted. Consider an airconditioning unit in your home, this continuously measures the air temperature and switches on and off in order to maintain a constant comfortable temperature. These systems are composed of several elements that work together to simplify processes. The major elements that are implemented in a measurement system are; a sensing element, a signal conditioning element, a signal processing element and a data presentation element [Bently, 2005]. The general structure of a measurement system is shown in 2.1. During the last few decades major advancements in technology has give us instruments that are highly accurate and responsive. The downside to these technological advancements is that as instruments become more accurate, they also become more expensive. It is always of interest in a process design to have a cost effective system.

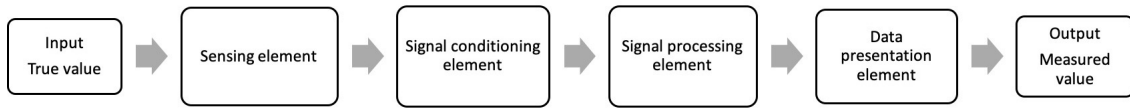


Figure 2.1.: Structure of a measurement system, adapted from [Bently, 2005]

2.1.1. Sensing Elements

A simple way to describe a sensing element is this is the component in the measurement system that is in direct contact with the process that we are attempting to observe. They acquire the true value that is being measure and give an output signal is, mechanical, thermal, or optical [Bently, 2005]. The electrical output from a sensing element can be further subdivided into active and passive type sensors. This classification is based on whether the sensor requires an external power supply (passive) or not (active) to provide an electrical output signal. All sensors operate on various working principles, due to the fact that we are attempting to quantify physical principles and these are not the same for the variables that we are measuring.

Temperature Sensors

The majority of temperature sensors operate on two main principles, either as thermocouples or as resistance temperature sensors. Thermocouples rely on the thermoelectric effect that states when two metals are connected an electromotive force arises. The electromotive force arises at the junction between the two metals and is temperature dependent. In a thermocouple there are two junctions that will essentially have two separate temperatures, one in contact with the process being measured and one that is at standard temperature. The metals used in these thermocouples varies widely and is dependant on the temperature ranges expected in the system, as different metal combinations provide different temperature ranges and tolerances. Thermocouples have to be connected to a transmitter in order to detect these junction potentials [Morris and Langari, 2011].

Pressure Sensors

Pressure measurements are extensively used in almost all industrial processes, and is therefore an important aspect to be considered. Commonly pressure can be described in three separate ways; *absolute*, *gauge*, and *differential* pressure. Where the absolute and gauge pressures are related through the following expression:

$$P_{abs} = P_g + P_{atm} \quad (2.1)$$

The differential pressure is defined as the the difference in absolute pressure at two points. Normally two points along a pipe containing the same fluid, and is commonly used in industrial processes [Morris and Langari, 2011]. For dynamic pressure measurement a capacitive or diaphragm type sensor can be used. Diaphragm type sensors operate by applying pressure to both sides of the diaphragm, a displacement in the diaphragm is detected by a strain gauge which are capable of detecting small variations in movement by the diaphragm plate.

Flow Sensors

In a functioning flow loop system the most important type of sensor to incorporate is a flow sensor. This will constantly provide feedback on either volumetric or mass flow rate. The simplest devices for the measurement of volumetric flow rate are those that obstruct flow in the loop, such as the orifice plate, Venturi tube or electromagnetic flow meter. These devices measure pressure at two points, before the obstruction and after obstruction. As flow enters the constriction in the device the pressure drops, the pressure drop results in an increase in the velocity of the fluid. The most commonly implemented for flow measurements is the orifice plate as it is relatively cheap and simple to install, however, with an accuracy of $\pm 1.5\%$. Although these have been commonly implemented in flow systems, several assumptions are made:

1. No energy loss due to friction
2. No heat transfer to the surroundings

3. Total energy is conserved
4. The fluid is incompressible
5. Horizontal pipe
6. Volumetric flow rate is conserved

For simple systems these assumptions can be made, however these assumptions are not ideal in an industrial system as several of these factors are extremely important. Especially in pipelines there is pressure loss to friction, heat is transferred to the surroundings, fluids are compressible to some degree, and the pipeline is never horizontal.

Mechanical flowmeters are also devices that obstruct the flow through the pipeline. These act as propellers that rotate due the flow, the rotation of the propeller is approximately proportional to the flow rate. The propeller is constructed of a ferromagnetic material this creates a magnetic potential with a permanent magnet, a voltage potential arises between the propeller and the magnet. The voltage potential that arises takes the form of a sine wave that has a frequency proportional to the angular velocity of the rotating propeller [Bently, 2005].

Electromagnetic Flow Sensors

As mentioned, the previously mentioned devices for measuring flow are obtrusive and induce a pressure loss during measurement of flow. Electromagnetic flow meters are mounted on the outside of the pipe and will not have any affect on the flow conditions within the pipe. Since they are not in contact with the fluid being measured they are superior in situations where we deal with fluids that are corrosive or in multiphase flow situations. These devices are applicable in 90% of measurement situations we may encounter.

The devices are constructed with a stainless steel pipe with magnetic coils mounted either side of the pipe which induce a magnetic field in the fluid. The magnetic field induces a voltage which is detected by electrodes mounted perpendicular to the induced field (Figure 2.2). Velocity of the fluid is calculated by the Faraday's law of electromagnetic induction.

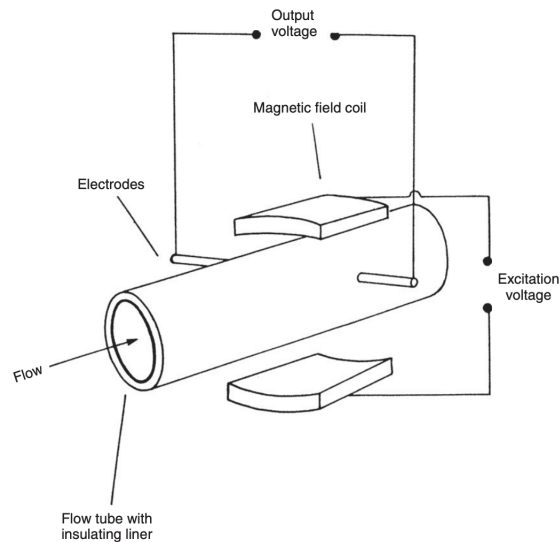


Figure 2.2.: Structure of an electromagnetic flow meter [Morris and Langari, 2011]

2.1.2. Signal Conditioning Element

The goal of a signal conditioning element is to convert the output from the sensing element to a signal that can be used by the processing element. The signal output can be transmitted in various forms; voltage, current or frequency. There are of course both advantages, and disadvantages to the various signal forms.

Analog and Digital signals

Signals in process systems are used to convey information through electrical currents or voltages, though they do exist in many other forms. An analog signal is a continuous signal that can be defined for all values in a time spectrum. The signal output by the device is consequently proportional the measured variable [Yarlagadda, 2010]. In contrast a digital signal is discrete, it is only defined for certain values in time. In

measurement systems analog signals need to be converted into a digital type in order to be processed. This is performed through sampling of the analog signal at defined time intervals and recreating the signal by the use of an analog to digital converter [Baher, 2001].

According to *Bently* 2005, signals are either random or deterministic in nature. A deterministic signal is one where we can predict the outcome of the signal at any point in time. A random signal, as the name suggests, is a signal type where the value of the signal cannot be determined at any point in time. In most measurement systems we deal with signals of this nature. We are able to use two of these five statistical quantities - mean, standard deviation, probability density function, power spectral density and the autocorrelation function - in order to recreate random signals.

Noise in a measurement system can disrupt the incoming signals and as researchers we prefer to keep this to a minimum. The source of a noise signal can be internal or external. The most common of these two is noise arising from external sources, which in a laboratory includes fluorescent lighting and heavy rotating machinery like a pump.

There are several simple methods for reducing the noise that can be experienced in a measurement system. One of the simplest methods is electromagnetic shielding. This is done simply by twisting the wiring that runs in the same circuit together, creating what is known as a twisted pair. this twisted pair cancels out the induced magnetic field that occurs when electricity runs through a conductive material. Filtering is another frequently used method to reduce noise in electrical systems. This can either be analoge filters that use resistors and capacitors to condition a continuous signal. One can also use digital filters that condition the signal after it is received. These are in the form of low pass filters, that filter out noise at a high frequency levels. High pass filters, that filter out low frequency noise. Band pass filters, that filter out both at the high and low levels or band stop filters that filter out noise at both the high and low levels, essentially selecting the desired signal and filtering out the rest of the frequencies that are received.

The 4-20 mA Current Loop

A current loop is a loop that provides an analogue signal from a sensor that can be processed and displayed as an output value. The majority of industrial sensors operate on a 4-20 mA current loop, mostly due to the low voltage required to maintain this current. There are several benefits associated with using a loop in this current range. The most significant of these is the so called false zero. The false zero is an indication that the loop is operating, in other words it is easily identified if the sensor is connected correctly or not. If the sensor is not properly wired, there will be no output from the sensor. With a properly wired sensor it will always output 4 mA. In addition to the false zero, the current loop is also quite robust, it is not so affected by noise from external sources. This resistance to noise is most relevant when the wiring is relatively short. By the simple use of *Ohm's law* and a resistor we can convert out analogue signal (in mA) to a readable voltage level. *Ohm's law* is as follows:

$$V = I \times R \quad (2.2)$$

Where V is the output voltage, I is the current flowing in the circuit, and R is the resistivity in the circuit. In a 4 - 20 mA loop, we know that the current in the loop should be between these two values. By implementing a known resistivity we can determine the output voltage range. In addition to the simple conversion, resistors also act as passive filter in the signal processing stage.

2.1.3. Signal Processing Element

This is the element in the that acquires the signal from the conditioning element and converts the signal into an understandable value, such as the conversion of an analog signal from a pressure sensor into a pressure value. A common element in this step is the use of and ADC. Resistors are implemented as a type of low pass filter which remove high frequency noise present in the incoming signal [Morris and Langari, 2011].

2.1.4. Signal Sampling

Continuous signals are reconstructed in the signal processing stage. This is performed by sampling the signal at regular intervals. The frequency of sampling is given by:

$$f = \frac{1}{\Delta t} \quad (2.3)$$

In signal sampling it is important to sample the signal at an appropriate rate, this is in order to reduce problems that can arise in the processing stage. Sampling at an appropriate rate reduces noise and reduces the possibility for alias signals to arise (see Figure 2.3. An alias is a signal that can match the sample values but is completely different from the original signal.

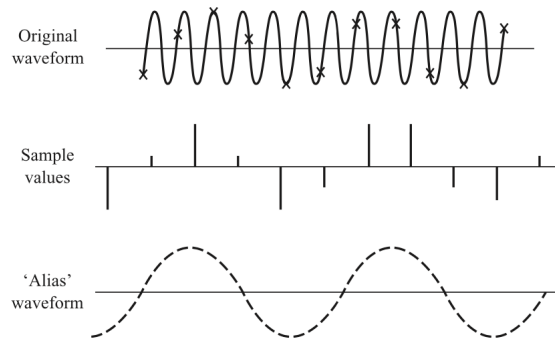


Figure 2.3.: Aliasing, acquired from [Bently, 2005]

For a sampled signal to be adequately represented we must fulfil one criteria, this is that the sampling rate should be at least twice that of the maximum frequency present in the signal. This is more commonly known as the Nyquist sampling theorem.

2.1.5. Data Presentation Element

Data presentation elements can be anything that presents the measured variable as a readable value. The simplest of these are analogue types that display the value on a pointer scale type display. For most projects where several variables are recorded it is common to use a LCD display, in this case a computer screen for presentation of the data.

2.2. Fluid Flow

For the construction of a measurement system it is important to understand what we are attempting to measure. A fundamental understanding of this is paramount to constructing an appropriate system. For the measurement of flow, several variables are of interest. To understand the flow in pipelines, it is important to know what flow speeds we are dealing with, where our pressure losses occur and the density of the fluid we are dealing with.

Fluid flow can be broken down into two main types, single-phase flow or multiphase flow. Single-phase flow involves the flow of either a gas or liquid, whereas multiphase flow is as the name suggests, flow of two or more phases.

2.2.1. Single-Phase Flow

When considering single-phase incompressible flow the approximation established by Bernoulli can be used to derive various parameters related to fluid flow. This equation does have its limitations, by using the Bernoulli equation we make several assumptions including *steady state flow, incompressible and no friction* [Cengel et al., 2012]. Bernoulli's equation is as follows:

$$\underbrace{\frac{P}{\rho}}_{\text{Flow Energy}} + \underbrace{\frac{u^2}{2}}_{\text{Kinetic Energy}} + \underbrace{gz}_{\text{Potential Energy}} = \text{constant} \quad (2.4)$$

The Bernoulli equation is frequently used in situations where we consider steady state incompressible flow. As mentioned earlier, the Bernoulli equation does have its limitations, in that it does not take into consideration that frictional forces, however small, contribute to a pressure loss. The equation does not consider the viscosity of fluids.

2.3. Fluid Flow in Pipes

Transportation of oil and gas occurs for the most part in pipelines over large distances. Over these large distances there is an increasing effect of friction that increases the pressure loss when the fluids are travelling in the pipeline. This is a variable that is not considered in Bernoulli's equations. Friction is one of the most important variables to include when designing a pipeline. Over short distances the pressure loss due to friction is negligible, however, when considering long transport pipelines the pressure loss becomes increasingly important. A pump that is able to overcome this pressure loss must be used otherwise fluids will not be transported.

Several studies have attempted to approximate the pressure loss by friction through experimental observations and correlations. There are several correlations that exist and not all can be considered therefore, only the most widely used have been chosen for consideration.

Fluid flow is an area in engineering that has been extensively studied, with some significant work by [Mandhane et al., 1974], [Beggs and Brill, 1973]. These studies developed the foundation for flow patterns that can be observed in gas-liquid flow situations in horizontal and vertical pipes. Within petroleum engineering this is of particular interest as there are several situations where we have various flow regimes. Commencing with the drilling phase and through to production and transport, there are countless situations where single and multiphase situations arise [Ferdoush and Li, 2014].

2.3.1. Two-Phase Flow

Single phase flow generates the fundamentals for understanding liquid or gas flow, however in several industries it is more common to encounter multiphase flow situations. Two-phase flow has been extensively studied during the last half century, where the major focus has been based on the nuclear and petroleum industries [Beggs and Brill, 1973]. The result of this is a vast understanding of the various flow regimes that occur in horizontal, inclined and vertical pipes.

The most commonly occurring situation in the petroleum industry is the flow of oil and

gas in pipelines, either during production, separation or transport of these fluids.

This section describes the case where we have two phases flowing in a pipe. Most commonly we consider the case where we have a gas-liquid situation flowing in a pipe. However, it is possible to consider other situations such as liquid-solid (cuttings transport), or solid-gas (movement of sand dunes in the desert). [Mandhane et al., 1974]

2.3.2. Laminar and Turbulent Flow

In fluid flow there is agreement that there are two distinct flow types, laminar and turbulent flow. Quantitatively, the flow types can be defined by the Reynolds number (Re). A low value, $Re < 2 \times 10^3$, represents laminar flow, A high value, $Re > 3 \times 10^3$, represents turbulent flow. There is a transitional region between these two flow types. The critical Reynolds number, where flow transitions from laminar to turbulent flow, is accepted to be a value of 2300 in circular pipes [Cengel et al., 2012]. Reynolds number is the ratio internal forces divided by the viscous forces and can be expressed by:

$$Re = \frac{\rho u_{avg} D}{\mu} \quad (2.5)$$

Flow Patterns in Horizontal Pipes

Relatively little has changed since the studies performed by *Mandhane et. al.* 1974, further development has been added to the theory that grounds the flow patterns. It is a general consensus that there are six main flow patterns that can be observed in horizontal pipes are: stratified smooth, stratified wavy, elongated bubble, slug flow, dispersed bubble, and annular wavy (see Figure 2.4). Although these six main flow regimes are quite clearly defined, there is no clear definition as to what flow speeds are required to generate them. This is mostly due to the varying parameters that exist in fluid flow and any small change in one parameter can affect another.

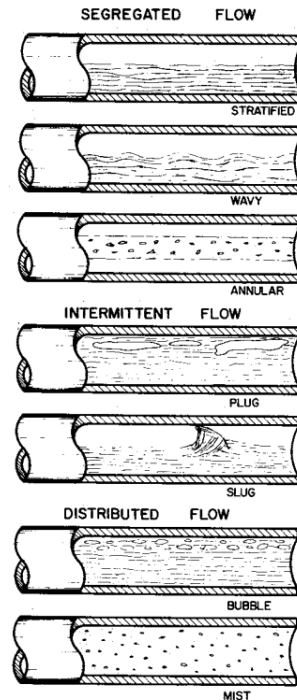


Figure 2.4.: Flow regimes in horizontal pipes [Beggs and Brill, 1973]

Through extensive experimentation approximations can be made on the type of flow present if we are able to obtain specific parameters related to the fluid present, namely the superficial velocity. To arrive at a definition for the superficial velocity, we first need to derive a few other parameters that will lead us to a final definition for this.

Factors That Determine Flow Regimes

If we assume that mass is conserved it is reasonable to assume that mass entering a control volume is the same as mass exiting a control volume. Therefore total mass is the sum of liquid and gas fractions in the control volume, expressed as:

$$m = m_l + m_g \quad (2.6)$$

Where the subscript l represents liquid and the subscript g represents gas. In the same manner, volumetric flow rate can be expressed as:

$$Q = Q_l + Q_g \quad (2.7)$$

While the mass (m) and volumetric (Q) flow rates provide relevant information, it is often more useful to express flow rates as superficial velocities. This is the velocity of the fluid if it were flowing alone in the pipe. It is simply calculated by dividing the volumetric flow rate of the fluid by the cross sectional area of the pipe in which it is flowing:

$$U_{ls} = \frac{Q_l}{A} \quad (2.8)$$

$$U_{gs} = \frac{Q_g}{A} \quad (2.9)$$

With the mixture velocity being equal to the sum of the superficial velocities of the phases present in the pipe, $U_{mix} = U_{ls} + U_{gs}$. The mixture velocity is equal to the actual average velocity of the flow that is present in the pipe [Awad, 2012]. The use of superficial velocities has been used in experimental results in an attempt to make a boundary map of the various flow regimes. There are have been several attempts to quantify a flow transition map from experimental data. One of the most widely accepted maps is the one developed by *Mandhane et. al.* in 1974 (See Figure 2.5).

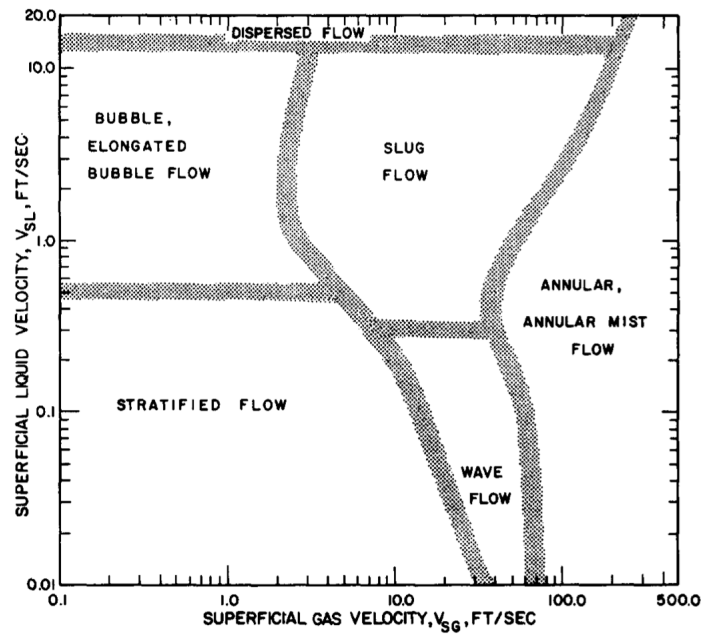


Figure 2.5.: Flow pattern map proposed by [Mandhane et al., 1974]

This flow pattern map is the result of over 5000 experimental results of varying gas and liquid flow rates, and is a log-log plot of the superficial velocities on the x- and y-axes respectively.

2.4. Pressure Loss in Pipes

Pressure loss in pipes is an important consideration when working with fluid flow because it determines the power needed by a pump to overcome the expected pressure loss. The loss of pressure along a pipeline is largely affected by the viscous and frictional forces that are present. It is common to express the pressure loss in both laminar and turbulent situations as:

$$\Delta P = f_D \frac{L}{D} \frac{\rho V_{avg}^2}{2} \quad (2.10)$$

2.4.1. Friction

Frictional pressure losses are possibly the factor that has the largest effect on loss of pressure along a pipeline. It is rarely relevant to exclude frictional pressure losses due to the lengths of pipeline that fluids flowing in gas and oil pipelines are transported. Several attempts have been made to approximate the friction factor with both explicit and implicit derivations. The most commonly used implicit equation is the Colebrook-White equation (2.11) [Colebrook and White, 1937] and the most commonly used explicit equation is the Swamee-Jain equation (2.12) [Swamee and Jain, 1976] these equations are:

$$\frac{1}{\sqrt{f_D}} = -2 \log \left[\frac{\varepsilon/D}{3.71} + \frac{2.51}{Re\sqrt{f_D}} \right] \quad (2.11)$$

$$f_D = \frac{0.25}{\left[\log \left(\frac{\varepsilon}{3.7D} \right) + \left(\frac{5.74}{Re^{0.9}} \right) \right]^2} \quad (2.12)$$

The Colebrook-White equation requires a number of iterations in order to find a solution but is considered the most accurate. The explicit solution method using the Swamee-Jain equation is far easier to solve as no iterations are required.

3. Methodology

3.1. Flow Loop

The flow loop (see schematic in Figure 3.1) was originally designed as a two pump system for oil/water/gas test system, but is currently being used as water gas system without the second pump. It is a 20m loop system that transports water from a tank (4 in Figure 3.1) using a 50 Hz helix pump (7 in in Figure 3.1) delivered by Froster. This pump is the KL30S type capable of a maximum pump rate of 14 m³/h. This pump is controlled by a Yasakawa V1000 control drive, which varies the pump frequency from 0-50 Hz. In the loop the water initially travels through flexible hoses that run through an electromagnetic flow meter (10 in Figure 3.1). Flow is then directed upwards to a horizontal test section (13, 14 in Figure 3.1) composed of transparent plexiglass pipes that are 5cm in diameter. At the start of the test section there is a mixing plate (12 in Figure 3.1) where gas can be introduced into the system for two-phase experiments. The horizontal section is where fluid flow is visible and observations of flow regimes can be made. Along the horizontal section several taps are installed where differential pressure sensors can be connected for the measurement of pressure drop. Prior to entering the separator the fluid flows through a Coriolis flow meter from Endress Hauser.

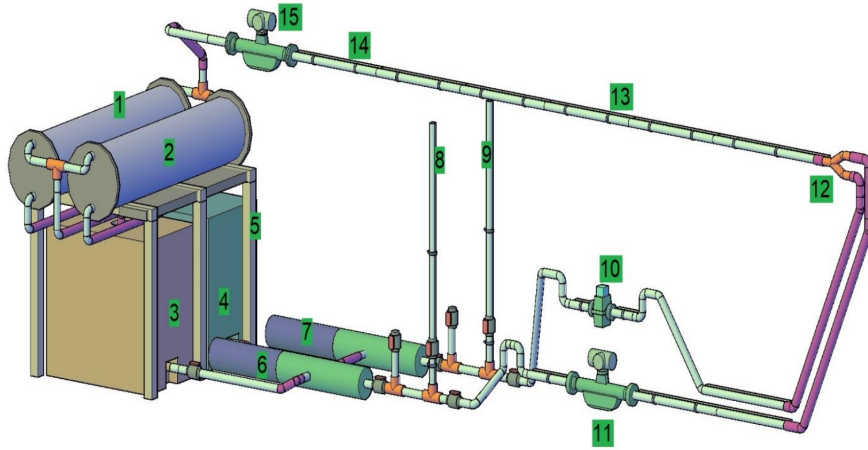


Figure 3.1.: Schematic of flow loop setup [Shouib, 2019]

Where:

- 1, 2: Separators
- 3, 4: Storage tanks
- 5: Separator support table
- 6, 7: Pumps
- 8, 9: Safety lines
- 10: Electromagnetic flow sensor
- 11: 15: Coriolis flow meters
- 12: Water/gas/oil mixing unit
- 13, 14: Test section

3.2. Hardware

All sensors are supplied with electricity from a wall socket, 230 V at 50 Hz, which enters a switching power source that delivers the sensors with 24V at 1 A. This is more than sufficient to power all sensors currently installed and can power additional sensors if it is desired to add more variables into the system.

3.2.1. Sensors

Pressure Sensors

Rosemount 3051 differential pressure sensors (Figure 3.2) are installed in the flow loop, there are four in total each measuring pressure between two separate points. The sensors are calibrated between 0 and 62.3 mBar providing an analog signal between 4 and 20 mA. These were attached to the horizontal section of the pipe through taps drilled on the bottom of the pipe with one meter intervals between taps. Measurements were obtained from two points that were 2.01 m apart.

For comparison, more sensitive sensors from Validyne Engineering were introduced into the system. These sensors are calibrated to a smaller pressure drop, values between ± 860 Pa. Specifically, the P55 pressure transducer from Validyne engineering was used. These sensors require 10-55 Vdc to function while also providing an analog output of between 4 - 20 mA. For comparison the currently used Pasco dual pressure sensors were also run simultaneously with the two other sensors installed.

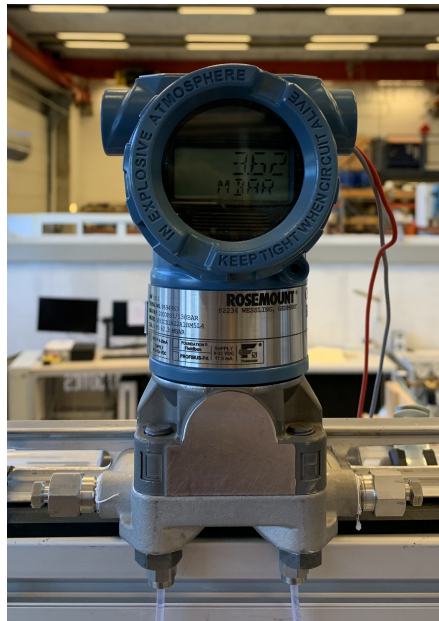


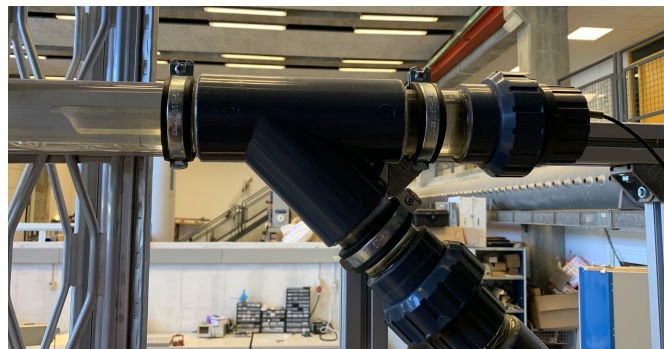
Figure 3.2.: Rosemount 3051 differential pressure sensor

Temperature Sensors

Temperature data was acquired through the Rosemount 3144P temperature transmitter (Figure 3.3) and a 3-wire thermocouple installed. Temperature measurement was obtained from the mixing section as not to obstruct flow and such that the thermocouple was in full contact with the fluid. This sensor having an analog output, also provides a 4-20 mA signal between 0 - 100°C.



(a) Rosemount 3144P



(b) Thermocouple

Figure 3.3.: Rosemount temperature transmitter and thermocouple installed in the mixing section

Electromagnetic Flow Meter

The loop has a XT5 electromagnetic flow meter installed (Figure 3.4), this was factory calibrated prior to tests, and previous work has shown this calibration to be accurate. The frequency regulator is directly related to the volumetric flow through the equation:

$$Q = 0.1436 \times f + 0.0203 \quad (3.1)$$

The XT5 can be setup to provide several output signals in different circuits. For simplicity the 4 - 20 mA analogue output signal type was chosen. As this has been previously

shown to be accurate with regards to the calibration.



Figure 3.4.: Techfluid XT5 electromagnetic flow sensor

3.3. Calibration

All sensors are factory calibrated, however, all were put through a simple calibration to ensure that the signal output was adequate.

Pressure Sensors

The pressure sensors were calibrated using a simple setup using a liquid column for a straight line calibration. The sensors were connected to a liquid column with a known height, using a multimeter the sensors were checked to determine if they were providing an adequate signal at full scale pressure (20 mA). For the Validyne sensors it was possible to adjust the zero and span of the sensors. Once sensors were determined to be providing the correct signal the liquid column was drained in steps of 10 cm starting at 50 cm down to 10 cm. For each drainage step both signal output and liquid column height was recorded. For confirmation of results, a Pasco absolute pressure sensor was connected

to the same liquid column. Using the simple equation $\rho gh = P$ it is then possible to determine the pressure at the sensor. Voltage output was then plotted against recorded pressure it is possible to acquire a straight line plot, which is used for the conversion of voltage to pressure. Each sensor was run through the calibration stage three times in order to have an acceptable average and to confirm that the output was repeatable.

The Pasco dual pressure sensor was run through a similar calibration however the Pasco Capstone software has a calibration function that allowed for a simple two point calibration. This was obtained at 70 cm and 20 cm and saved as the calibration function in the software.

Temperature Sensors

Both the Pasco and Rosemount temperature sensors were calibrated using the Fluke 9102S Dry-Well Calibrator. This device allows for the calibration of thermocouples and temperature probes with a range of -10°C and 122°C . When calibrating the Rosemount temperature sensor, a six point calibration was chosen with points at 10°C to 60°C . The voltage output of the sensor was recorded at each stage, and an average of 100 points was used to create a straight line calibration. This straight line relates the voltage output to temperature. this a straight line that relates voltage output to temperature was created.

The Pasco Capstone program has an inbuilt calibration function, using this a two point standard calibration was used. With temperatures of 10°C and 70°C being used for calibration points.

3.4. Experimental Setup and Procedure

Prior to implementation of the measurement system, extensive small scale testing of the sensors was carried out on a breadboard. This was performed to ensure that the circuitry was functioning before implementing the system in a larger scale.

The two pressure sensors and the temperature transmitter are powered by a switching

power supply that draws power directly from the wall socket and converts it to a useable 24 V at 1 A. The XT5 flow meter requires a direct power source of 220V and was therefore not included in the main circuitry. The output was wired to the Arduino so that the analog output could be recorded. The sensors were wired in series, and each was wired to its own 220Ω resistor. This was to bring the voltage down to 4.8V, due to the max voltage input on the Arduino board is limited to 5 V. A sense wire was connected from one side of the resistor to the analog input and a second wire was connected ground of the Arduino. The serial port of the Arduino was used for communication between a laptop and Aduino. The final wiring schematic was as follows:

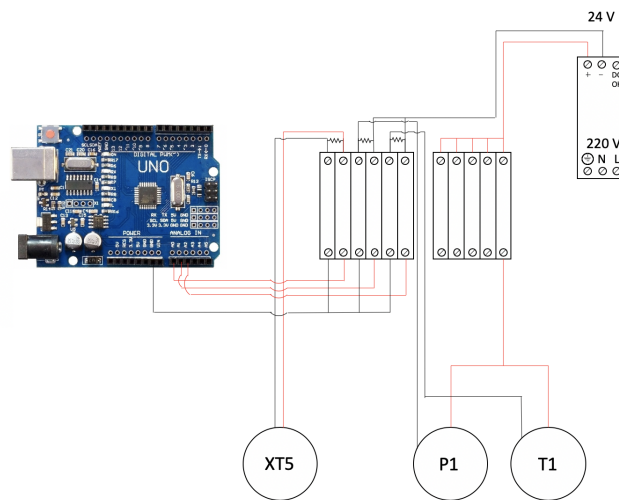


Figure 3.5.: Sketch of the wiring for the measurement system

It has been previously shown [Shouib, 2019] that the separator affects pressure measurements in the flow loop, especially when this is open or closed to atmospheric pressure. It was therefore chose to keep this open to the atmosphere while running tests.

The measurement system was put through several different test situations to test how the sensors react to changing flow situations. The loop was run before recoding such that all of the piping was water filled before the commencement of tests. All tests were approximately 13 minutes long, where a stabilisation period of 1 minute was recored before the pump was started. Each increase or decrease in flow was recored for 3 minutes to allow for stabilisation of flow before changing the pump frequency.

The first test was a low to high test where the pump was started at 20 Hz and increased to 50 Hz in increments of 10 Hz. The second test was a low-high-low test where the pump

was started at 30 Hz, increased to 50 Hz (with 10 Hz increments) before being decreased to 40 Hz. The third test was runs as a low-high-low-high test where the pump rate was started at 30 Hz, increased to 50 Hz, decreased to 30 Hz, and finally increased to 50 Hz again. Both systems were run at a sampling rate of 10 Hz (10 samples per second). All tests were run three times to ensure repeatability of the measurement system. In addition all tests were run as single phase flow, with only water being introduced into the flow loop.

3.5. Coding

The code that controls the system was written in two parts. Firstly, a code that was uploaded to the Arduino was written in the Arduino IDE software. Subsequently, a Python code was written to communicate and extract information from the Arduino. Finally a third code was written to perform calculations and create visual representations of the incoming data. A full summary of the various code that has been used can be seen in Appendix A.

3.5.1. Arduino IDE Code

The code written in the IDE contains three elements, of which all are sent to separate analog inputs on the Arduino board ($A_0 - A_2$). Incoming data from each sensor is sent to the serial monitor of the Arduino IDE. The full code can be seen in A.1.

3.5.2. Python Code

Python code had to be written to access the serial monitor application of the Arduino IDE. This code reads the incoming data and stores it in a comma separated values (CSV) formatted file. The CSV file format is extremely useful as it is simple to export to excel or other data processing applications for further use. The CSV file can opened in a second code that uses the data to make plots of the data that has been stored. This

was used as a quick reference to ensure that adequate data had been obtained. See A.2 and A.3 for a full summary of the codes.

4. Results

The final system was capable of logging the pressure, temperature and flow simultaneously. The final wiring of the system can be seen in figure 4.1

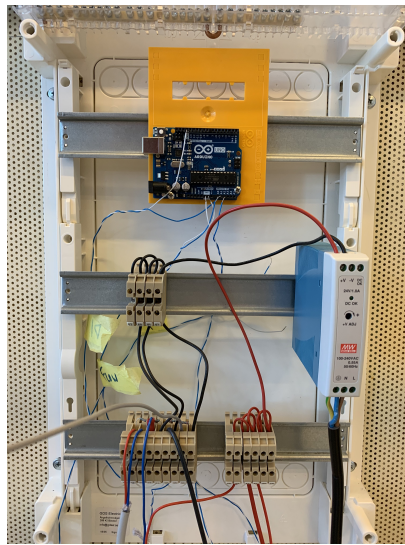


Figure 4.1.: Final wiring of the proposed measurement system

4.1. Calibration Results

4.1.1. Pressure Sensors

Following several tests, the Validyne P55 sensors were found to be unreliable when measuring pressure when compared to the currently implemented Pasco sensors. It was decided to use only the Rosemount sensors, despite the range being much larger on these sensors, 0-63.2 mBar compared to ± 860 Pa, the output from the Rosemount sensors was

found to be much more reliable. Calibration of the Rosemount sensors resulted in the following function:

$$y = 1694.8x - 1553.6 \quad (4.1)$$

Where y is the pressure output in Pa and x is the input from the sensor in volts.

4.1.2. Temperature Sensors

Results from the calibration of the Rosemount temperature sensor can be seen in figure B.1a. It is clear that the sensor follows the straight line principle that relates the voltage output from the sensor to temperature. The function obtained is seen in equation 4.2 and was used when running tests in the flow loop to convert the voltage output to temperature.

$$y = 27.969x - 24.297 \quad (4.2)$$

The conversion is as follows; y is the temperature output in °C and x is the input in volts from the sensor.

4.1.3. Electromagnetic Flow Sensor

Using the active output from the sensor the XT5 was programmed to provide an output of 4 mA at 0 l/min flow and 20 mA at 7.2 l/min. A two point calibration resulted in a function to convert the voltage output to flow in l/min which is:

$$y = 33.858x - 29.424 \quad (4.3)$$

Where y is the flow in l/min and x is the voltage output from the device.

4.2. Flow Loop Tests

The variable that is constantly changing in a flow loop is the pressure drop along the 2.01m section, this is especially varying with turbulent flow. Temperature is constant throughout all tests, however there is a slight increase from run 1 to run 9.

Several problems were encountered while performing tests with the flow loop, issues including erroneous results from pressure and flow sensors had to be worked out before reliable results were obtained. The implemented system seems to provide accurate and repeatable results for all changes in flow velocity. All results are represented in Figure: 4.2, 4.3, 4.4, 4.5, 4.6, and 4.7. For plotting of data a moving average of 50 points from pressure data was selected to smooth out data. Raw results are presented in Appendix B.1.2. Table: 4.1, 4.2, 4.3, 4.4, 4.5, and 4.6 present averaged results from 20 seconds at each flow rate. Table 4.7 are the theoretically expected results.

Q (L/min)	Run 1		Run 2		Run 3	
	Average	Standard	Average	Standard	Average	Standard
	Pressure (Pa)	Deviation (Pa)	Pressure (Pa)	Deviation (Pa)	Pressure (Pa)	Deviation (Pa)
0.00	-68.341	4.072	-65.738	4.141	-63.672	3.000
48.40	5.635	3.810	10.002	4.583	9.218	4.316
72.06	75.543	4.922	75.689	4.667	75.192	4.906
95.86	158.731	5.087	157.325	5.295	159.145	4.952
119.27	260.555	6.695	260.162	5.817	261.114	5.616

Table 4.1.: Pressure results from Arduino in run 1-3

Q (L/min)	Run 1		Run 2		Run 3	
	Average	Standard	Average	Standard	Average	Standard
	Pressure (Pa)	Deviation (Pa)	Pressure (Pa)	Deviation (Pa)	Pressure (Pa)	Deviation (Pa)
0.00	20.207	2.869	66.461	3.142	62.280	2.846
48.40	114.042	38.779	123.804	38.270	126.706	36.315
72.06	182.702	14.821	187.958	20.182	185.958	23.253
95.86	258.719	12.126	262.045	12.652	263.465	13.530
119.27	393.811	14.286	396.577	11.695	397.171	12.708

Table 4.2.: Pressure results from Pasco sensor in run 1-3

Q (L/min)	Run 4		Run 5		Run 6	
	Average	Standard	Average	Standard	Average	Standard
	Pressure (Pa)	Deviation (Pa)	Pressure (Pa)	Deviation (Pa)	Pressure (Pa)	Deviation (Pa)
0.00	-37.399	2.244	-36.531	3.081	-36.284	3.298
72.06	105.925	5.058	110.315	4.285	108.617	4.714
95.86	196.007	5.234	193.729	5.527	201.494	5.188
119.27	296.815	6.288	295.778	6.081	295.097	5.901
95.86	197.871	5.610	197.209	4.771	194.724	4.801

Table 4.3.: Pressure results from run 4-6

Q (L/min)	Run 4		Run 5		Run 6	
	Average	Standard	Average	Standard	Average	Standard
	Pressure (Pa)	Deviation (Pa)	Pressure (Pa)	Deviation (Pa)	Pressure (Pa)	Deviation (Pa)
0.00	60.602	3.069	66.616	3.552	69.539	2.477
72.06	190.198	42.542	192.369	30.507	188.758	14.221
95.86	270.583	16.136	264.789	14.553	271.740	12.326
119.27	400.862	15.168	402.823	13.550	406.019	13.392
95.86	323.683	11.311	325.163	11.581	324.363	10.522

Table 4.4.: Pressure results from Pasco sensor in run 4-6

Q (L/min)	Run 7		Run 8		Run 9	
	Average	Standard	Average	Standard	Average	Standard
	Pressure (Pa)	Deviation (Pa)	Pressure (Pa)	Deviation (Pa)	Pressure (Pa)	Deviation (Pa)
0.00	-36.573	3.041	-35.047	17.658	-33.864	17.139
72.06	106.133	6.002	37.176	4.881	106.547	5.008
119.27	294.641	5.555	293.297	6.431	292.239	5.952
72.06	115.244	5.376	115.513	5.190	111.620	4.061
119.27	291.683	6.559	291.166	5.953	292.263	6.567

Table 4.5.: Pressure results from run 7-9

Q (L/min)	Run 7		Run 8		Run 9	
	Average	Standard	Average	Standard	Average	Standard
	Pressure (Pa)	Deviation (Pa)	Pressure (Pa)	Deviation (Pa)	Pressure (Pa)	Deviation (Pa)
0.00	59.919	2.954	73.465	8.724	75.653	19.058
72.06	196.360	32.231	196.493	29.686	197.665	17.401
119.27	409.426	12.691	409.982	13.805	412.332	12.913
72.06	263.005	10.792	264.613	8.845	264.807	9.852
119.27	410.175	13.016	403.707	12.340	410.592	13.236

Table 4.6.: Pressure results from Pasco sensor in run 7-9

Q (L/min)	Reynolds Number (Re)	f_D Colebrook- White	f_D Swamee- Jain	Theoretical C-W (Pa)	dP S-J (Pa)
0.00	0.000	0.0000	0.0000	0.000	0.000
48.40	21372.406	0.0253	0.0254	85.788	88.634
72.06	31819.421	0.0231	0.0230	173.519	177.995
95.86	42327.766	0.0216	0.0216	287.181	293.975
119.27	52665.613	0.0209	0.0205	429.065	432.146

Table 4.7.: Theoretical pressure drop using Colebrook-White and Swamee-Jain equations

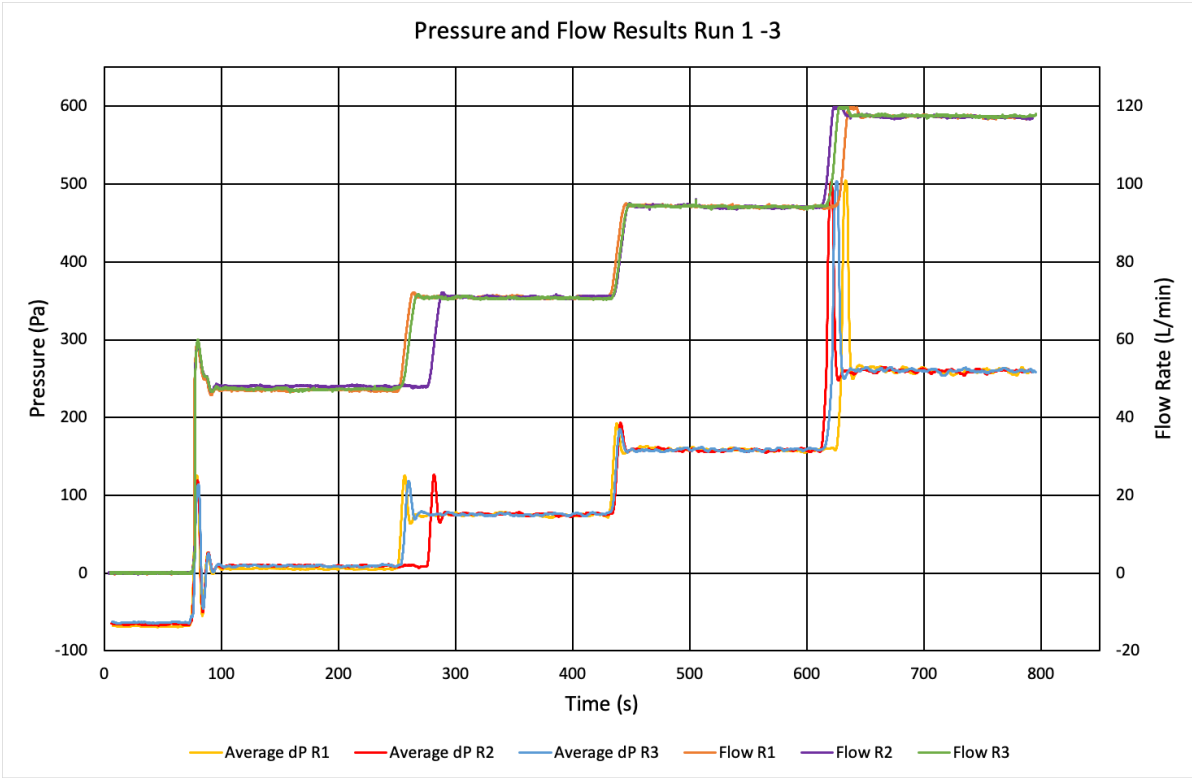


Figure 4.2.: Arduino run 1-3

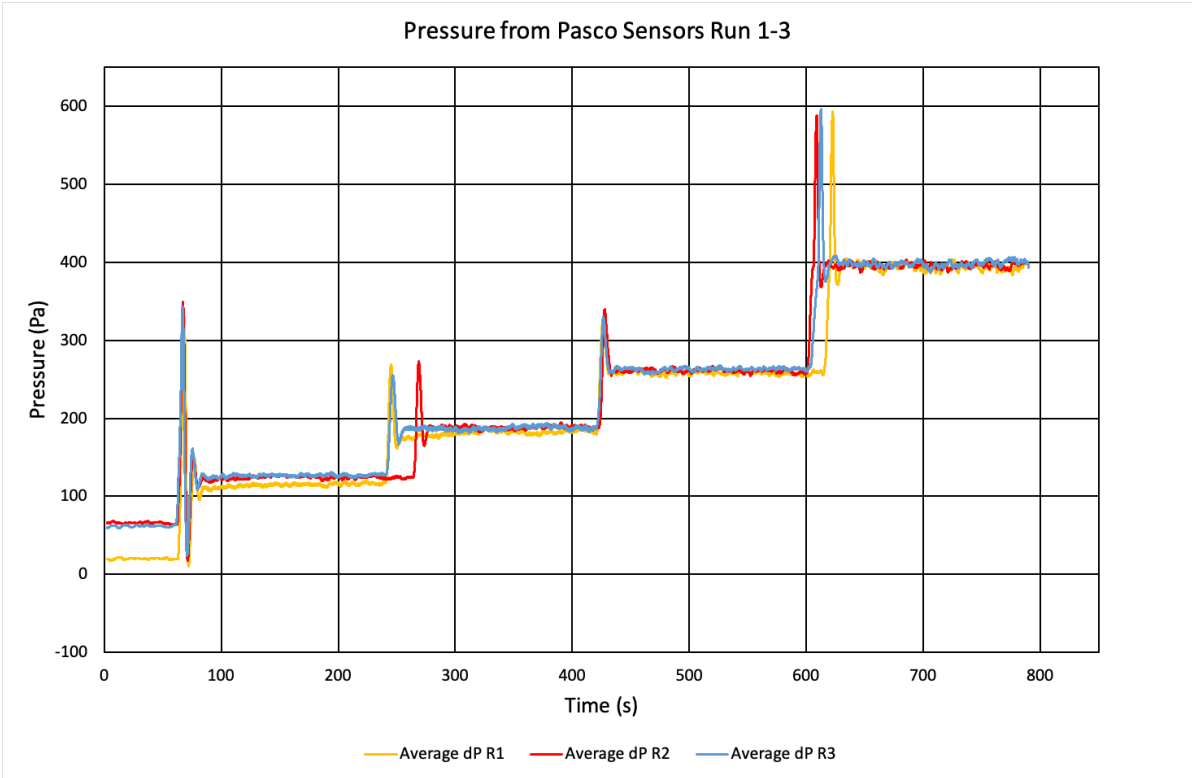


Figure 4.3.: Pasco run 1-3

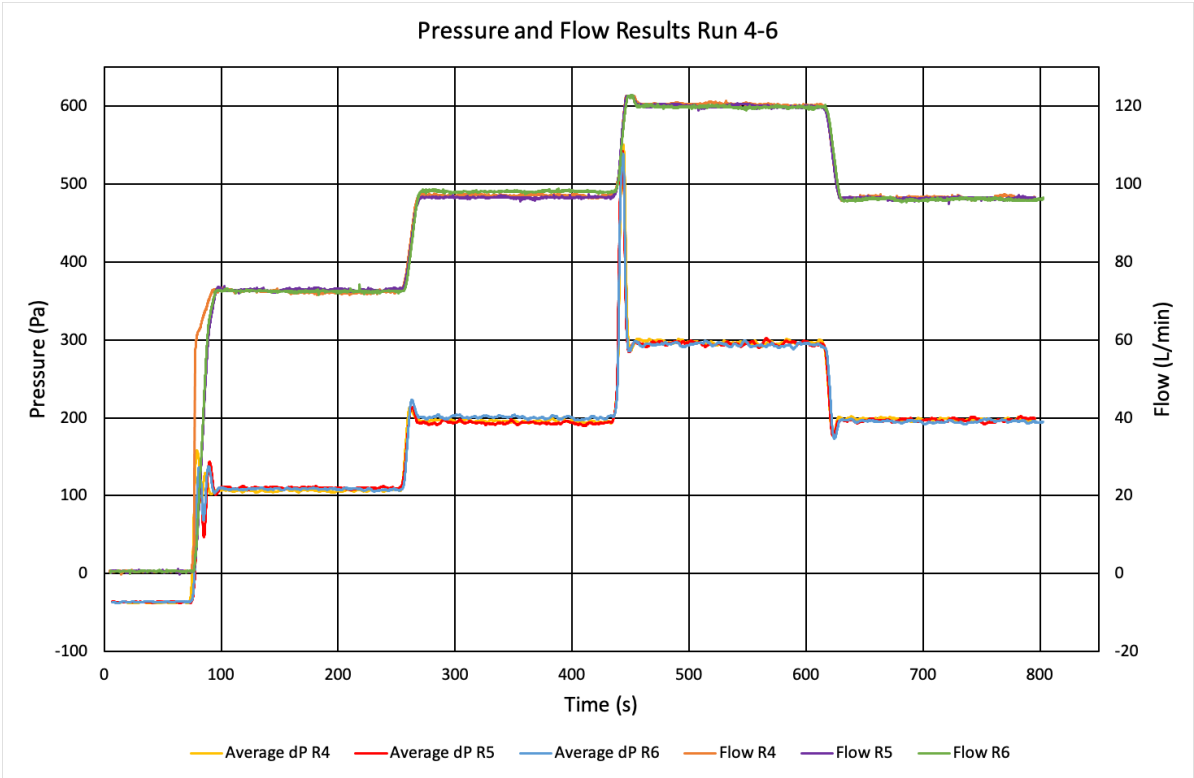


Figure 4.4.: Arduino run 4-6

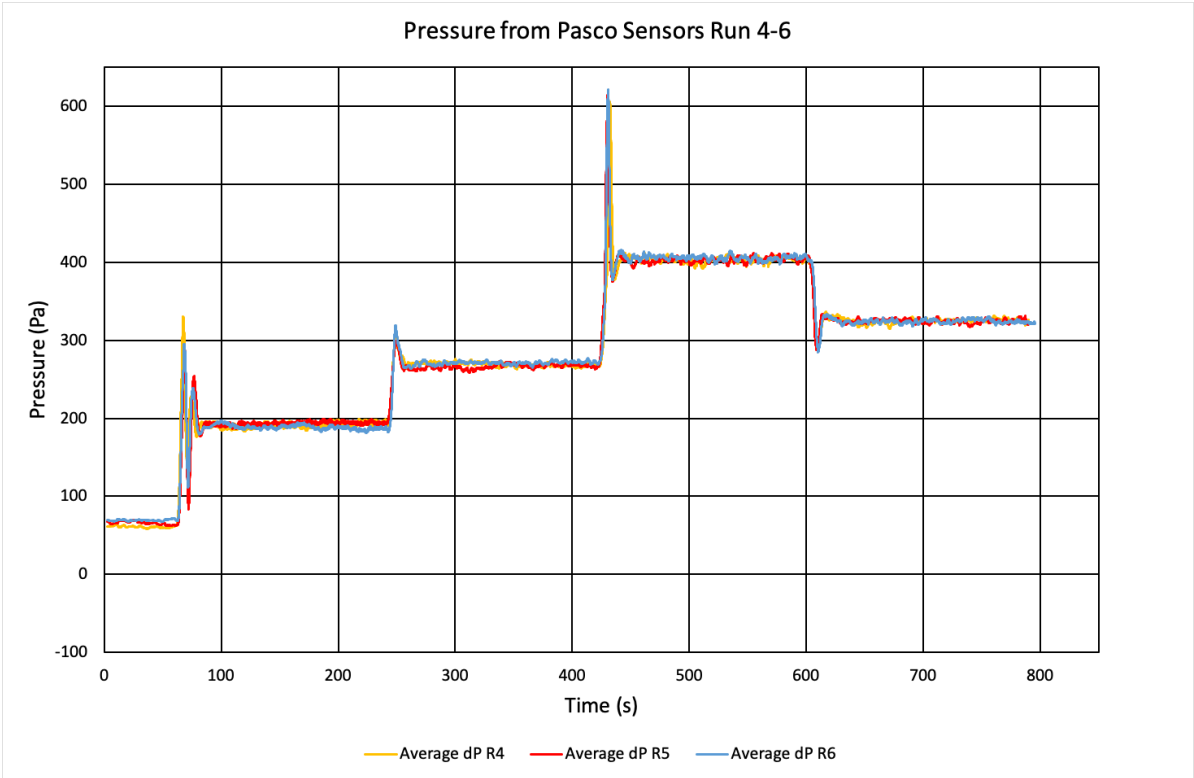


Figure 4.5.: Pasco run 4-6

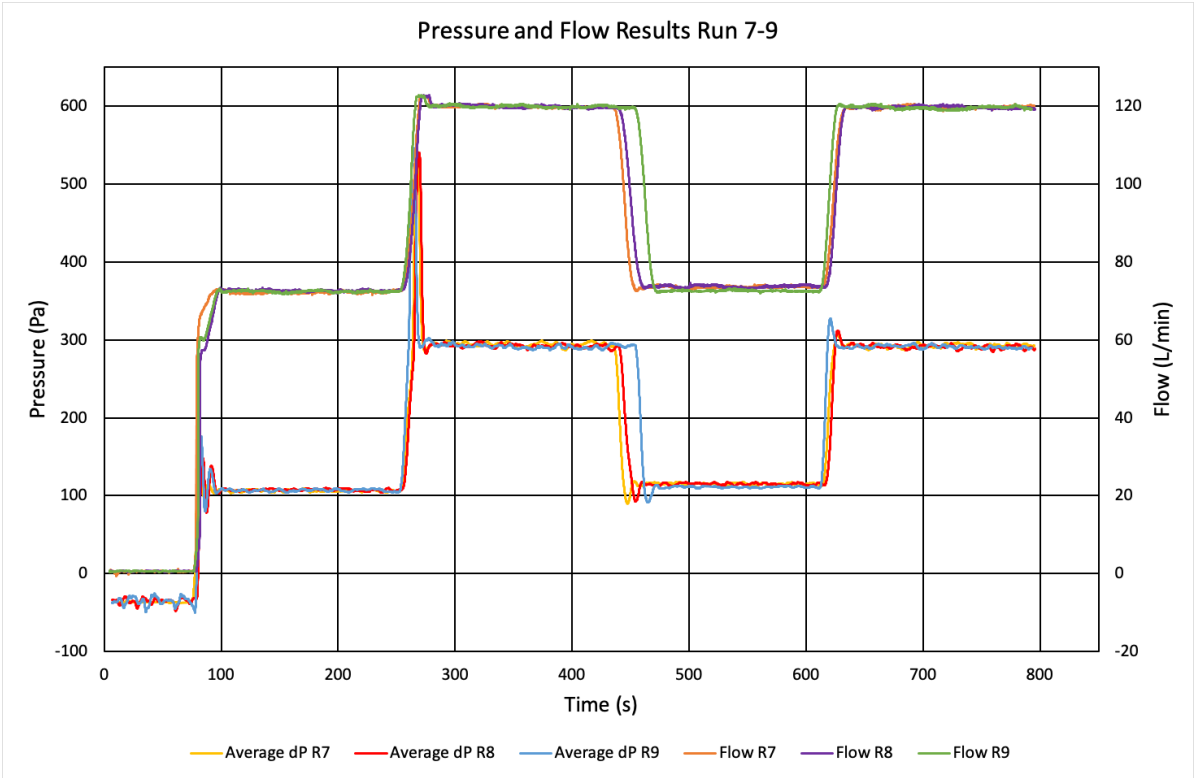


Figure 4.6.: Arduino run 7-9

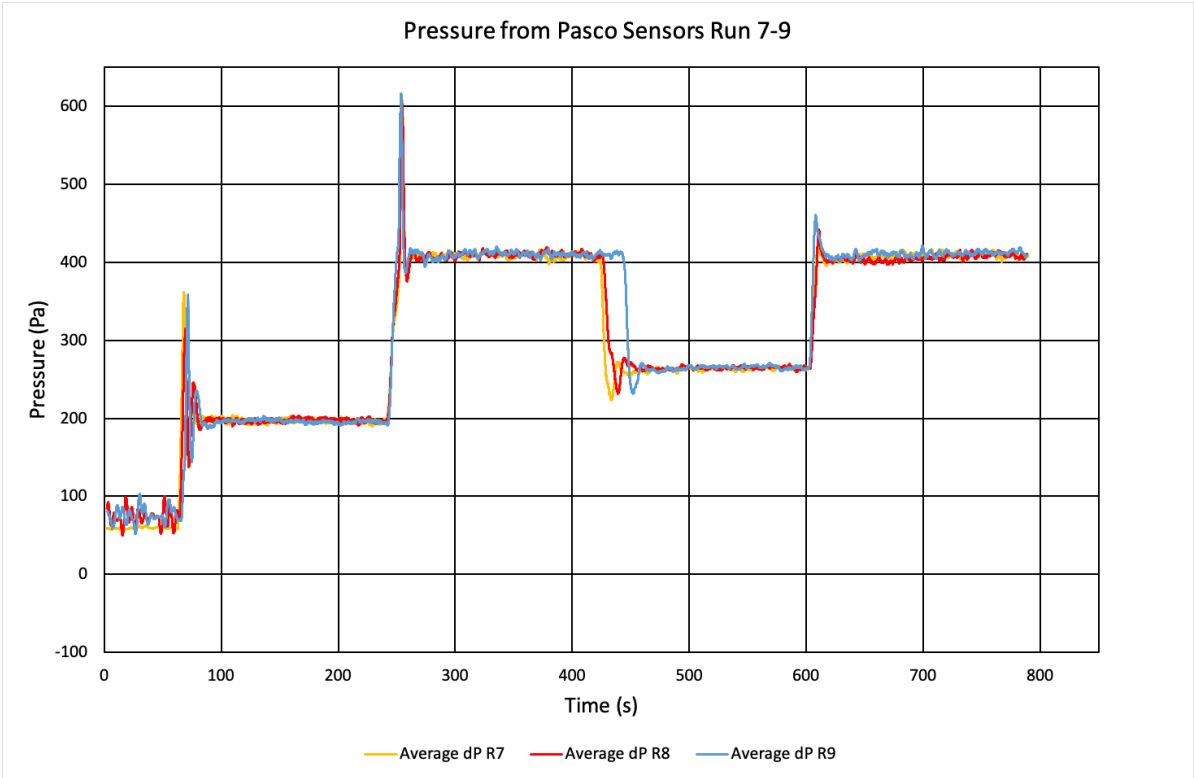


Figure 4.7.: Pasco run 7-9

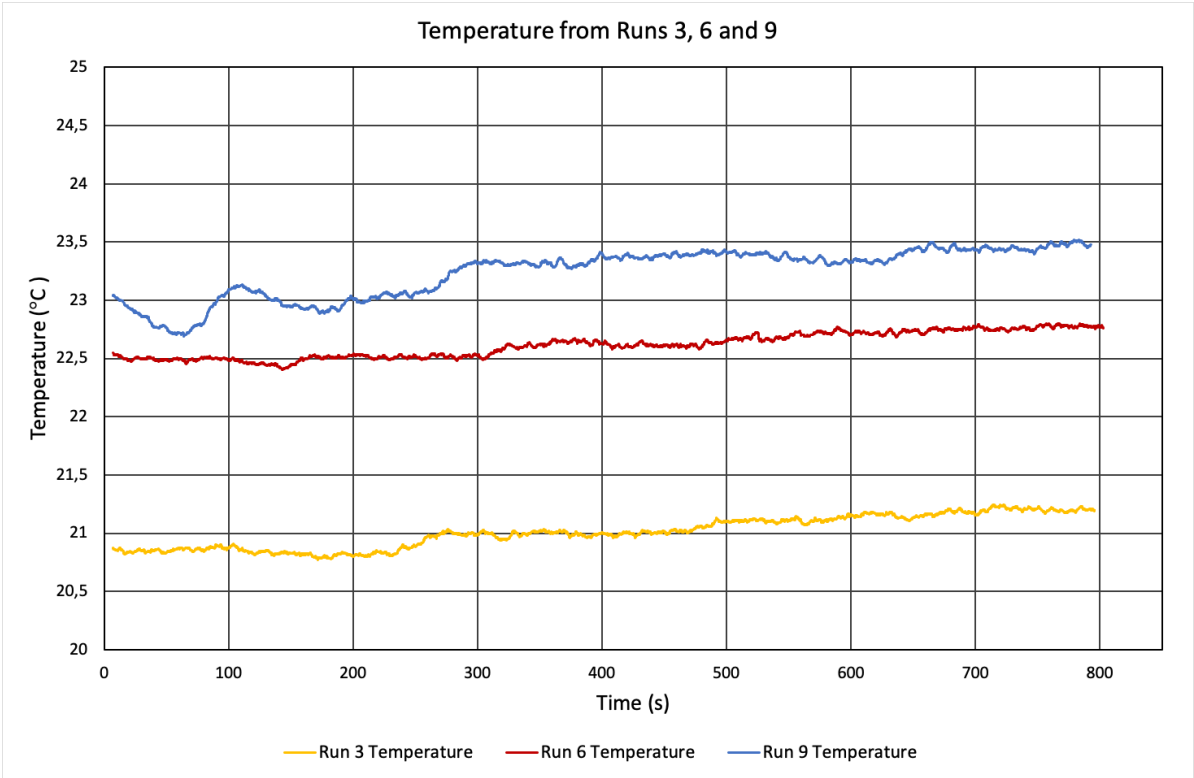


Figure 4.8.: Temperature from Arduino runs 3, 6, and 9

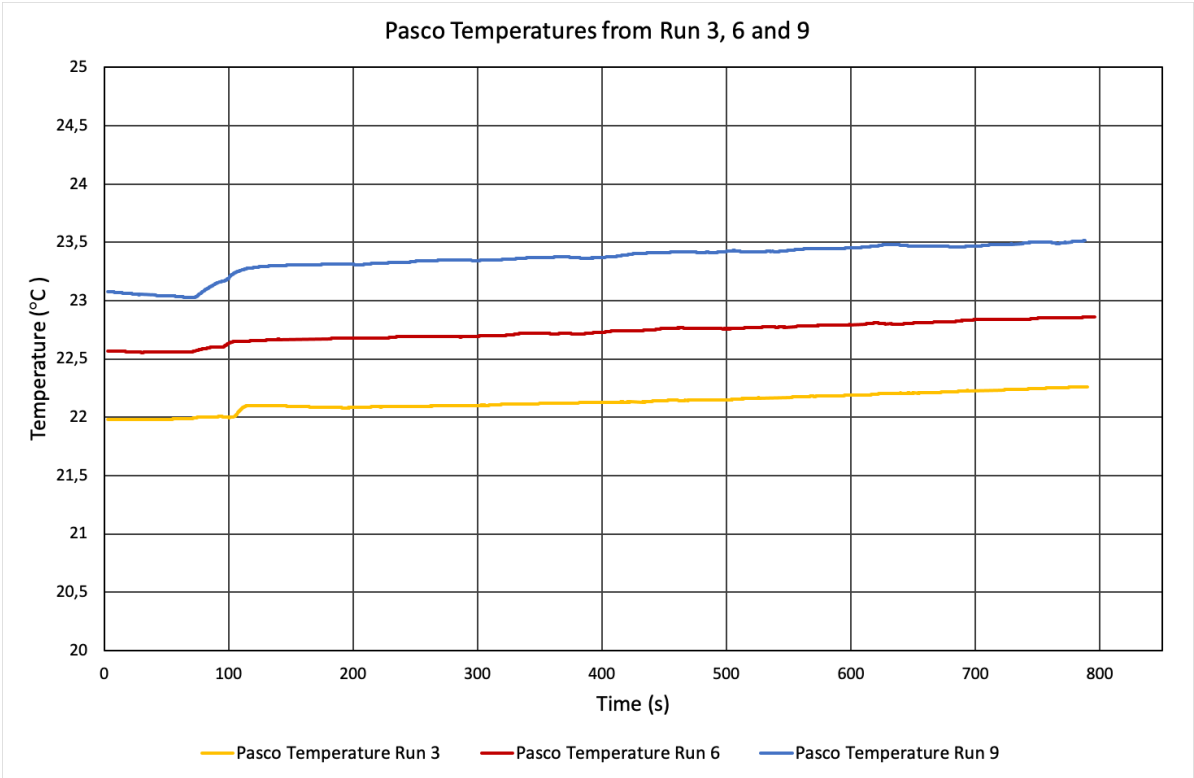


Figure 4.9.: Temperature from Pasco runs 3, 6, and 9

5. Discussion

Observing the results from the various runs, it is clear that a functioning measurement system has been presented. The system functions in the desired manner, capable of acquiring data from various sensors, with capacity to install other sensors. The results present in Figures 4.2, 4.4, and 4.6 also show that the obtained results are repeatable. However, there is some lack of accuracy as the pressure sensors seem to underestimate the pressure loss along the pipe section.

5.1. Flow Loop Tests

The experimental method was run in three different flow cases, therefore results from each will be discussed separately. Pressure results from both the Arduino and Pasco systems will be compared for accuracy and repeatability. Results from the temperature sensors are similar, as this should remain constant throughout tests. However, a slight increase in temperature was observed during consecutive tests.

5.1.1. Run 1-3

From both the Arduino and Pasco systems the results are repeatable with all three runs from both systems providing roughly the same averaged results each time the flow loop was run. Looking at the raw data, we see a much smaller spread of data using the Arduino compared to the Pasco system. For most flow rates the Pasco system has a standard deviation that is almost twice that of the Arduino for the same flow rate (See tables 4.1 and 4.2). Especially at lower flow rates the Pasco dual pressure sensors struggles to accurately determine the pressure drop. At $Q = 48.40$ L/min the Pasco

system has a standard deviation of 38.8 Pa compared to just 3.8 Pa from the Arduino measurement system. This higher variation in pressure output from the Pasco system could be due to the fact that the sensor cannot be in contact with liquids, and therefore a cushion of air had to be left at the sensor input, this is relevant for all tests. Figure 5.1 is an example of the difference in results from one run for the two systems.

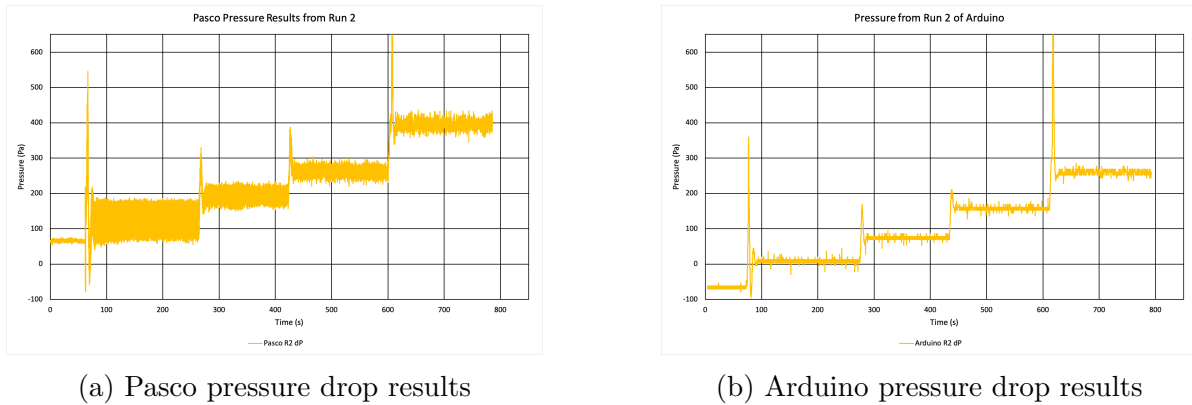


Figure 5.1.: Comparison of results from both measurement systems: run 2

As seen in figure 5.1 the Arduino system seems to underestimate the pressure drop, at 0 L/min there is a negative pressure drop of 68.34 Pa, in theory this should be 0 Pa. In comparison the Pasco system overestimates this pressure drop (20.21 Pa). Despite this, the Pasco pressure sensors seem to more accurately determine the pressure drop compared to the theoretically expected results.

5.1.2. Runs 4-9

These six runs show similar trends with regards to the Pasco sensors and will therefore be compared in the same section. There is a similar trend with these sensors in regards that they overestimate pressure loss at $Q = 0$ L/min providing values between 60 and 75 Pa for all six runs. Comparing this to the Rosemount sensors installed on the Arduino board, these sensors underestimate the pressure at $Q = 0$ L/min by between -33 and -37 Pa. The Rosemount sensors are obviously more consistent with less variation in the pressure drop at this stage. This is also clear in the situations with flow, where the standard deviation is much lower, between 3 and 6 Pa.

Setting aside these differences the Arduino system represents the pressures drop more consistently, returning to the same pressure drop when stepping down to a lower pressure. The Pasco sensors consistently overestimate this pressure drop. In runs 4-9 (see 4.5 and 4.5) the pressure drop recoded by the Pasco proves a higher value when returning to a lower pressure. This is consistently 60 Pa higher than the previous value at the same flow rate.

5.1.3. Temperature Changes

Temperature results from both measurement systems showed a similar trend. As these are minimally affected by the fluid dynamic properties they remain almost constant. There is a slight increase in temperature over the course of the runs that were carried out. This increase in temperature can be explained by the pump that delivers a large amount of heat to the system and will over the course of time lead to a temperature increase in a small loop system. The temperature changes also show different trends due to the placement of the sensors. The pasco sensor was placed in the flow stream where it would acquire the most accurate temperature readings. For the thermocouple there were no tappings in the flow stream that would allow for temperature readings here. This can explain the temperature jump we see at approximately 100 seconds in the Pasco sensor (Figure 4.9), while for the thermocouple there is a more gradual increase in temperature (Figure 4.8).

5.2. Limitations and Improvements

5.2.1. Arduino Limitations

One major drawback to using the Arduino Uno is that the board is not capable of handling input signals that are more than 5V. Most of the industrial sensors used operate on a high input voltage with the output between 4-20mA. It was possible to draw the output signal down to a tolerable voltage for the Arduino, however this reduced the resolution of the data. Using a 220 Ω resistor the voltage was brought to between 0.88V and 4.4V, giving a range of 3.52V.

Another drawback to the Arduino Uno board is the number of analog inputs that are available on the circuit board. There are a maximum of 6 inputs available for analog variables, limiting it to 6 sensors. This can be overcome by using an external ADC with more analog inputs. As well as adding more analog inputs a higher resolution ADC can be used. Standard on the board is a 8 bit microcontroller capable of representing 256 values. Using an external microcontroller with a higher bit rate (12 or 16 bit) it is possible to increase the resolution of incoming data that can be comparable to that of the current Pasco system.

5.2.2. Sensor Limitations

Pressure measurements that occur over short distances in small pipes are extremely small, in this case the theoretical maximum pressure drop at the highest flow rate is 432.146 Pa. The sensor used in this system is calibrated between 0 and 63.2 mBar (0 and 6320 Pa), although the sensors are calibrated to function in this range the pressure drops experienced are in the lower end of the calibration with very small changes in pressure occurring with changing flow rates.

6. Conclusions

- The designed system functions and consistently reads pressure, temperature and flow results.
- The Pasco system is not consistent enough in reading of pressure drop.
- More should be done to understand the pressure loss that occurs in the flow loop.
- Improvements should be made to python code so that conversion of volt to flow, pressure and temperature does not need to be performed in excel.
- The system should be further developed to include control of the drive system that controls the flow rate in the loop.
- Data should be represented live as recording is being performed.
- The Coriolis flow meter installed in the flow loop should be incorporated into the the recording system.
- Further work should be performed with multiphase flow situations.
 - In order to to monitor multiphase situations a gas flow meter should be installed to measure control the volume of gas entering the system.

Bibliography

- [Ahmed and John, 2018] Ahmed, S. A. and John, B. (2018). Liquid – liquid horizontal pipe flow – a review. *Journal of Petroleum Science and Engineering*, 168:426 – 447.
- [Ate, 2018] Ate, A. (2018). *Controlling the temperature reactor based on Raspberry Pi system control*. PhD thesis.
- [Awad, 2012] Awad, M. (2012). *An Overview of Heat Transfer Phenomena*. Intech Open, first edition.
- [Baher, 2001] Baher, H. (2001). *Analog and Digital Signal Processing*. John Wiley and Sons Ltd, second edition.
- [Beggs and Brill, 1973] Beggs, D. H. and Brill, J. P. (1973). A study of two-phase flow in inclined pipes. *Journal of Petroleum technology*, 25(05):607–617.
- [Bently, 2005] Bently, J. P. (2005). *Principles of Measurement Systems*. Pearson Education Limited, fourth edition.
- [Cengel et al., 2012] Cengel, Y. A., Cimbala, J. M., and Turner, R. H. (2012). *Fundamentals of Thermal-Fluid Sciences*. McGraw Hill, fourth edition.
- [Colebrook and White, 1937] Colebrook, C. and White, C. (1937). Experiments with fluid friction in roughened pipes. *Proceedings of the Royal Society of London*, 161(906):367–381.
- [Ferdoush and Li, 2014] Ferdoush, S. and Li, X. (2014). Wireless sensor network system design using raspberry pi and arduino for environmental monitoring applications. *Procedia Computer Science*, 34:103 – 110.
- [Mandhane et al., 1974] Mandhane, J., Gregory, G., and Aziz, K. (1974). A flow pattern map for gas-liquid flow in horizontal pipes. *International Journal of Multiphase Flow*, 1(4):537 – 553.
- [Morris and Langari, 2011] Morris, A. S. and Langari, R. (2011). *Measurement and Instrumentation: Theory and Application*. Elsevier Science and Technology.

- [Shouib, 2019] Shouib, A. K. (2019). Multiphase air-water flow loop, construction and experiments. Master's thesis, University of Stavanger.
- [Suresh et al., 2014] Suresh, N., Balaji, E., Anto, K., and Jenith, J. (2014). Raspberry pi based liquid flow monitoring and control. *International Journal of Research in Engineering and Technology*, 03(07):122–125.
- [Swamee and Jain, 1976] Swamee, P. and Jain, A. (1976). Explicit equation for pipe flow problems. *Journal of the Hydraulics Division*, 102(5):657–664.
- [Wu et al., 2017] Wu, B., Firouzi, M., Mitchell, T., Rufford, T. E., Leonardi, C., and Towler, B. (2017). A critical review of flow maps for gas-liquid flows in vertical pipes and annuli. *Chemical Engineering Journal*, 326:350 – 377.
- [Yarlagadda, 2010] Yarlagadda, R. (2010). *Analog and Digital Signals and Systems*. Springer, first edition.

A. Appendix

A.1. Arduino Code

```
Pressure_test §
const int sensorPin0=A0; // Initiates analog input A0
const int sensorPin1=A1; // Initiates analog input A1
const int sensorPin2=A2; // Initiates analog input A2

void setup() {
  Serial.begin (9600);} // Determines baudrate

void loop() {

  float PSensor1 = analogRead(sensorPin1);
  float P1;
  float TSensor2 = analogRead(sensorPin2);
  float T2;
  float flow = analogRead(sensorPin0);
  float flows;

  flows = flow*(5.0/1023);
  Serial.print(flows, 4); // Prints flow data as a voltage
  Serial.print(",");

  P1 = PSensor1*(5.0/1023.0);
  if(P1 > 0)
  {Serial.print(P1, 4); //Prints pressure data as a voltage
  Serial.print(",");}
  else
  Serial.print("Sensor Error!"); // Sends an error message if a negative value is taken

  T2 = TSensor2*(5.0/1023.0);
  if(T2 > 0)
  {Serial.print(T2, 4); // Prints temperature data as a voltage
  Serial.println(",");}
  else
  Serial.println("Sensor Error"); // Sends an error message if a negative value is taken

  delay (100);} // Adjust frequency of readings are taken
```

Figure A.1.: Arduino code for reading of sensors

A.2. Python Code

```

1 import serial
2 import time
3 import csv
4
5 f=open('Run1.csv', 'wt') # Creates the datafile where data is stored for graphing (wise to change filename between
6 # tests as not to overwrite previous data
7 writer = csv.writer(f, delimiter=',') # Specifies the delimiter (',') used when writing to csv
8 writer.writerow(['Time (s)', 'Flow Speed (m3/s)', 'Pressure 1 (Pa)', 'Temperature (°C)']) # Variables collected
9
10 ser = serial.Serial('/dev/tty.usbmodem141201', baudrate=9600, timeout=2) # specifies the serial port to read as well as
11 # the baudrate and timeout
12
13 while True:
14     User_Input = input('Start data collection (y or n)?') # User controls when to start recording
15
16     if User_Input == 'y':
17         try:
18             time.sleep(2)
19             for i in range(10000): # User defines number of data points to collect
20                 ser_bytes = ser.readline().decode('ascii').strip('\r\n') # Decodes incoming data
21                 data_col = ser_bytes.split(',')
22                 print(time.perf_counter(), ser_bytes) # Prints incoming data
23                 writer.writerow([time.perf_counter(), data_col[0], data_col[1], data_col[2]]) # Writes data in four
24 # columns
25
26         except KeyboardInterrupt:
27             print("Keyboard Interrupt")
28             break
29
30     if User_Input == 'n':
31         break

```

Figure A.2.: Python code for reading and logging of incoming data from Arduino

```

1 import csv
2 from matplotlib import pyplot as plt
3 filename = 'Run1.csv'
4
5 with open(filename) as f: # Opens specified filename
6     reader = csv.reader(f)
7     header_row = next(reader)
8     time = [] # Creates a list of time variables
9     flow = [] # Creates a list of flow variables
10    P1 = [] # Creates a list of pressure variables
11    Temp = [] # Creates a list of temperature variables
12    for row in reader:
13        if row == '':
14            continue
15        times = float(row[0])
16        flows = float(row[1])
17        P1s = float(row[2])
18        Temps = float(row[3])
19        time.append(times)
20        flow.append(flows)
21        P1.append(P1s)
22        Temp.append(Temps)
23
24    fig, ax = plt.subplots(2, 2) # Plots variables in four subfigures
25    ax[0,0].plot(time, P1, label = P1)
26    ax[0,1].plot(time, Temp, label = Temp)
27    ax[1,0].plot(time, flow, label = flow)
28    ax[1,1].plot(time, time)
29    plt.xlabel('Time (s)')
30    plt.ylabel('Pressure (Pa)')
31    plt.show()

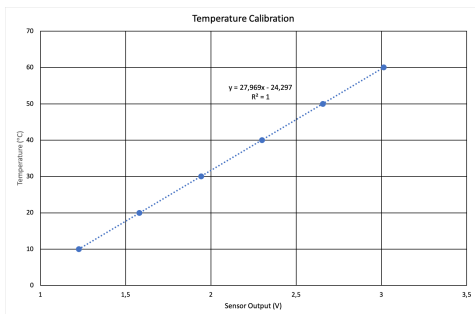
```

Figure A.3.: Python code for graphing of results

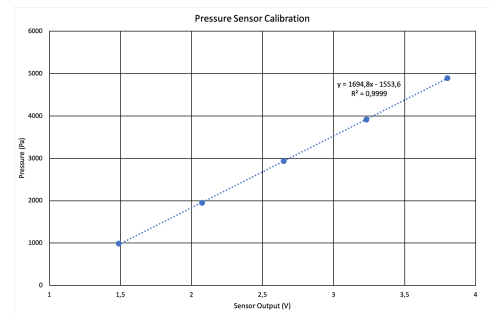
B. Appendix

B.1. Calibration Results

B.1.1. Temperature and pressure



(a) Temperature calibration



(b) Pressure calibration

Figure B.1.: Equations for conversion from volt to pressure and temperature

B.1.2. Raw results from individual runs

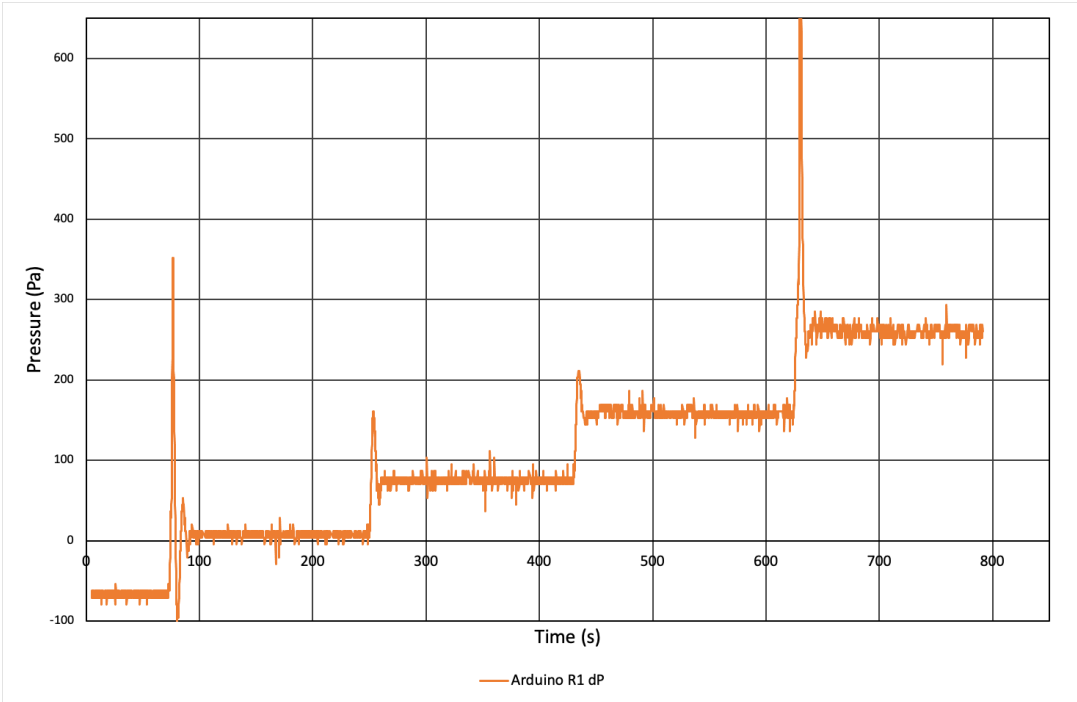


Figure B.2.: Arduino run 1 pressure results

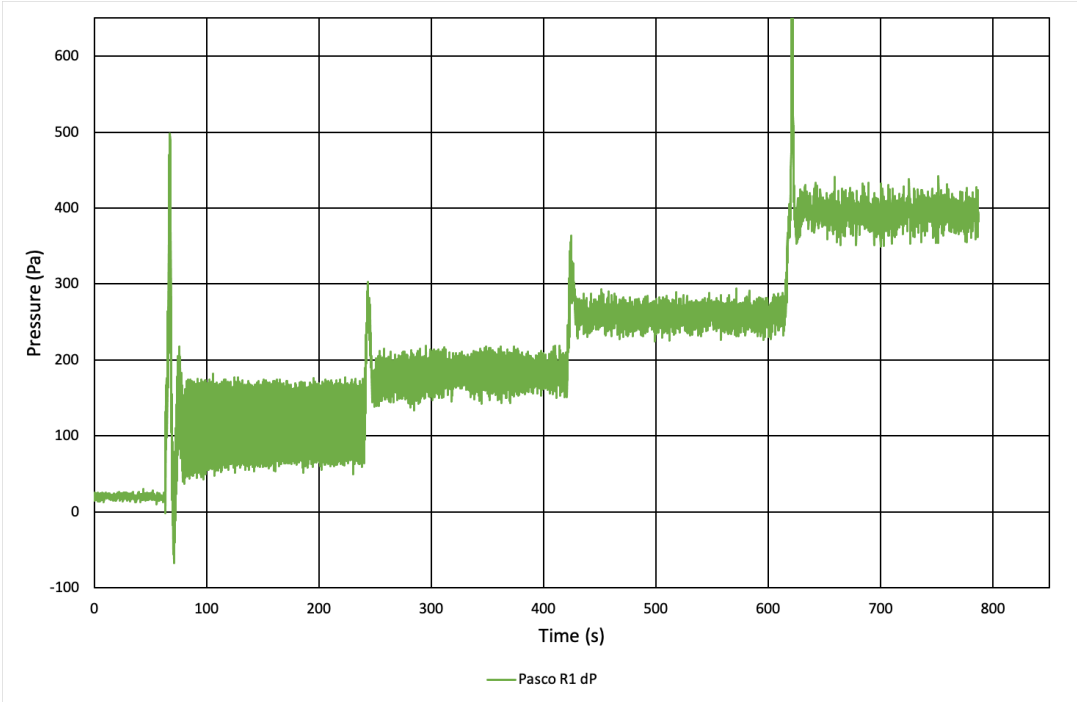


Figure B.3.: Pasco run 1 pressure results

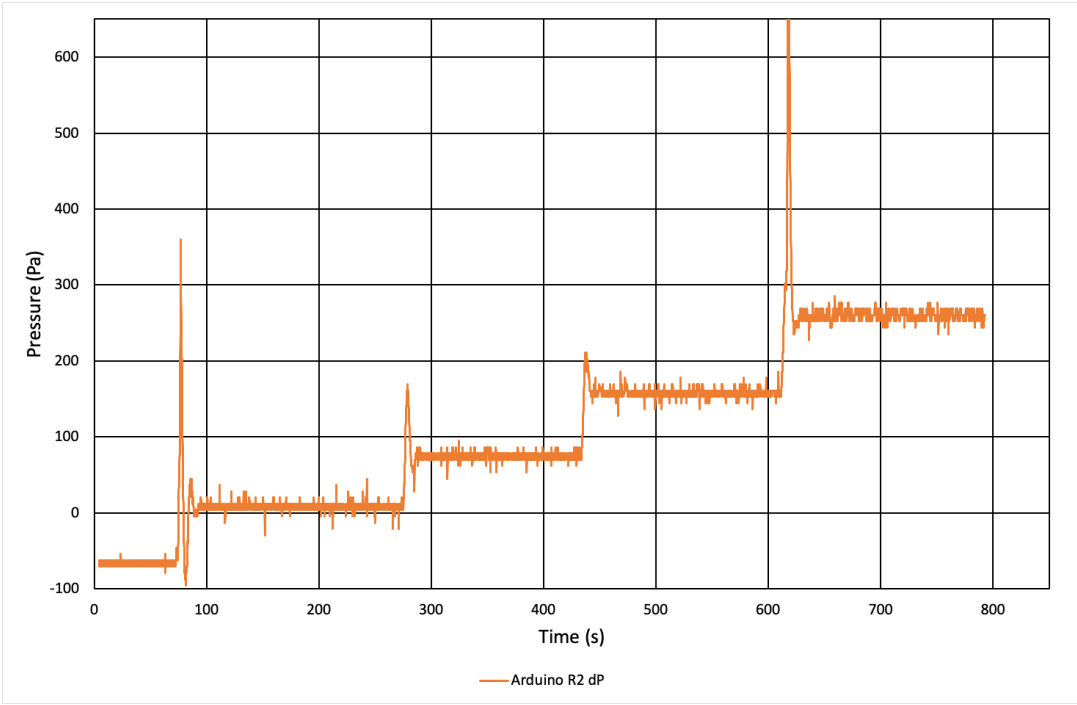


Figure B.4.: Pasco run 2 pressure results

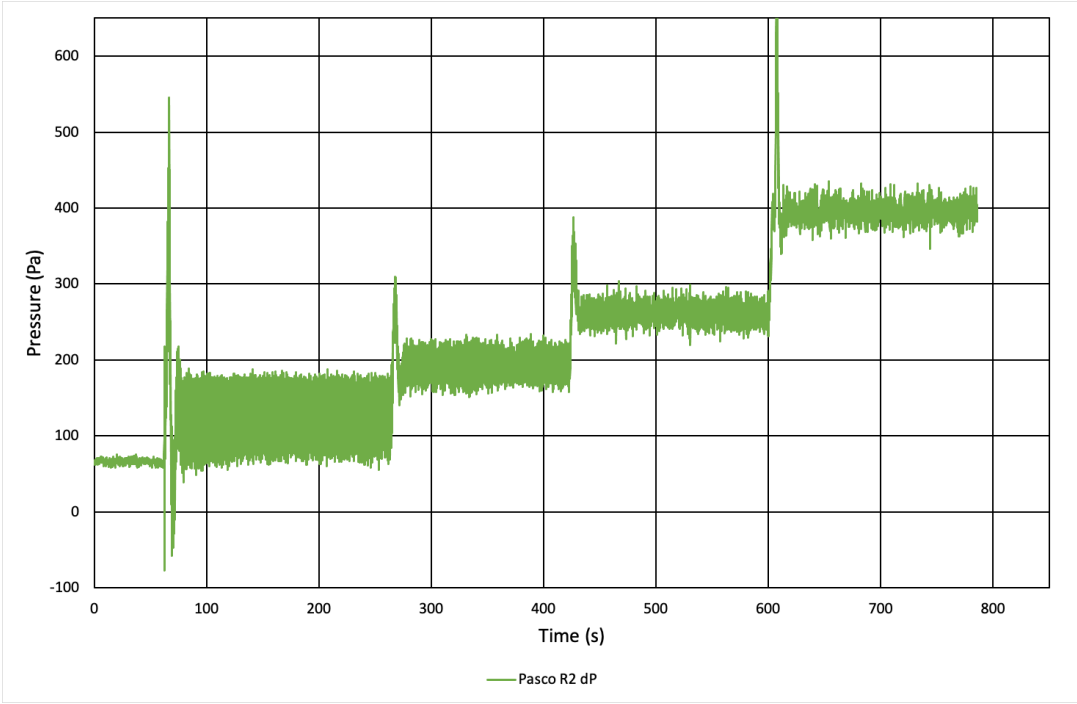


Figure B.5.: Pasco run 2 pressure results

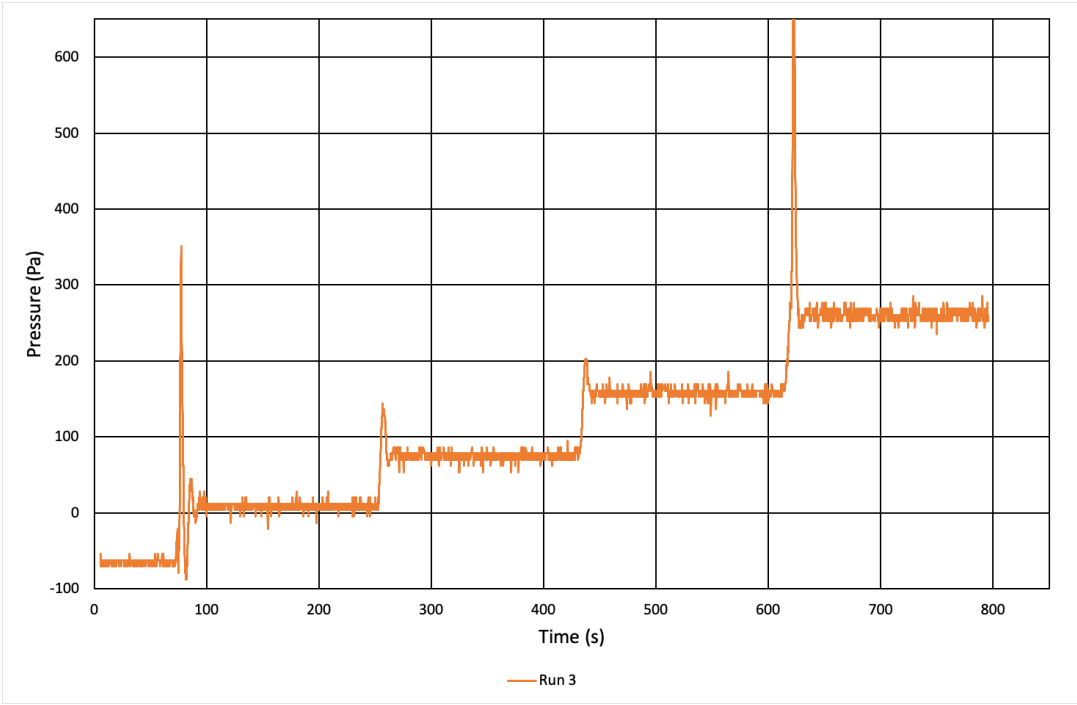


Figure B.6.: Arduino run 3 pressure results

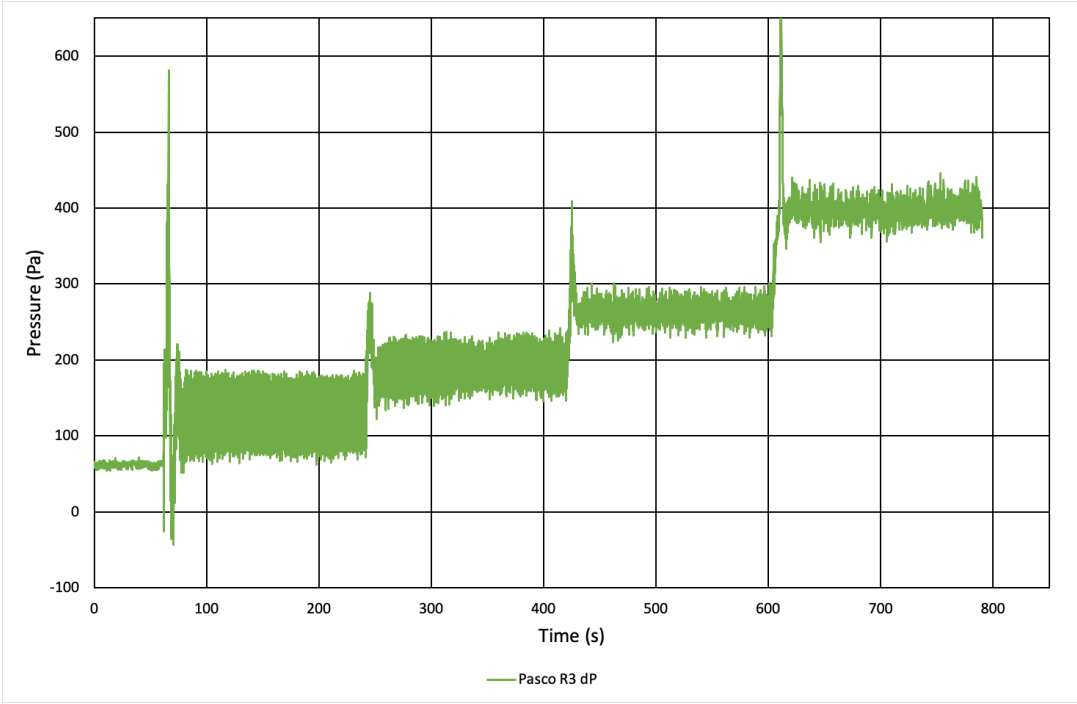


Figure B.7.: Pasco run 4 pressure results

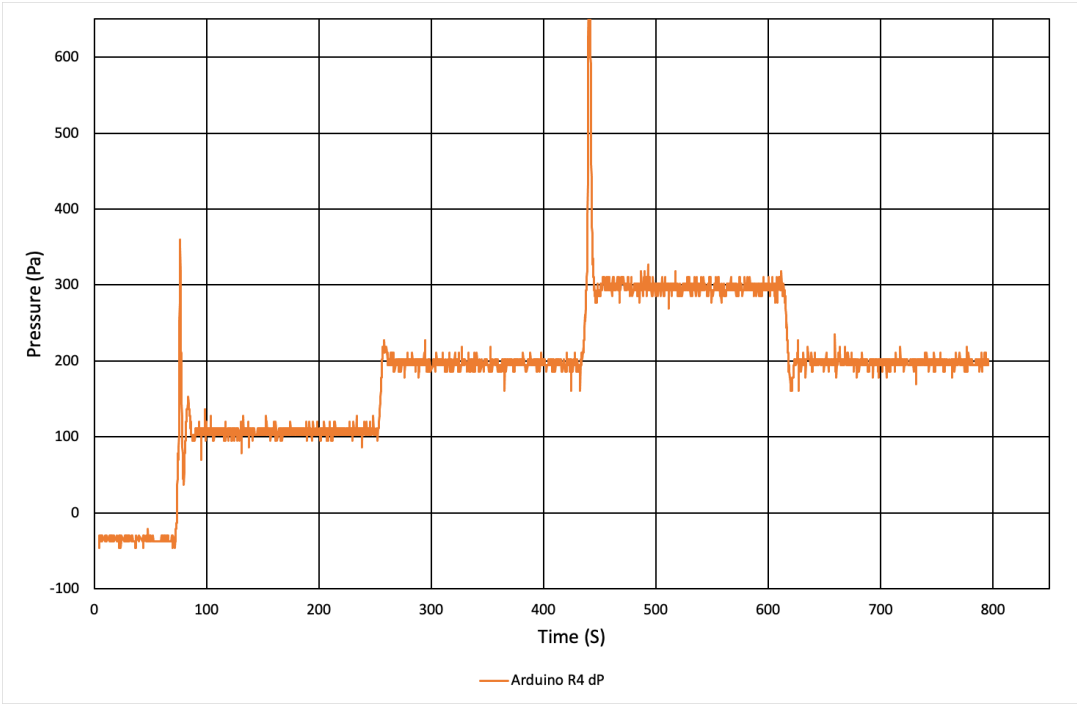


Figure B.8.: Arduino run 4 pressure results

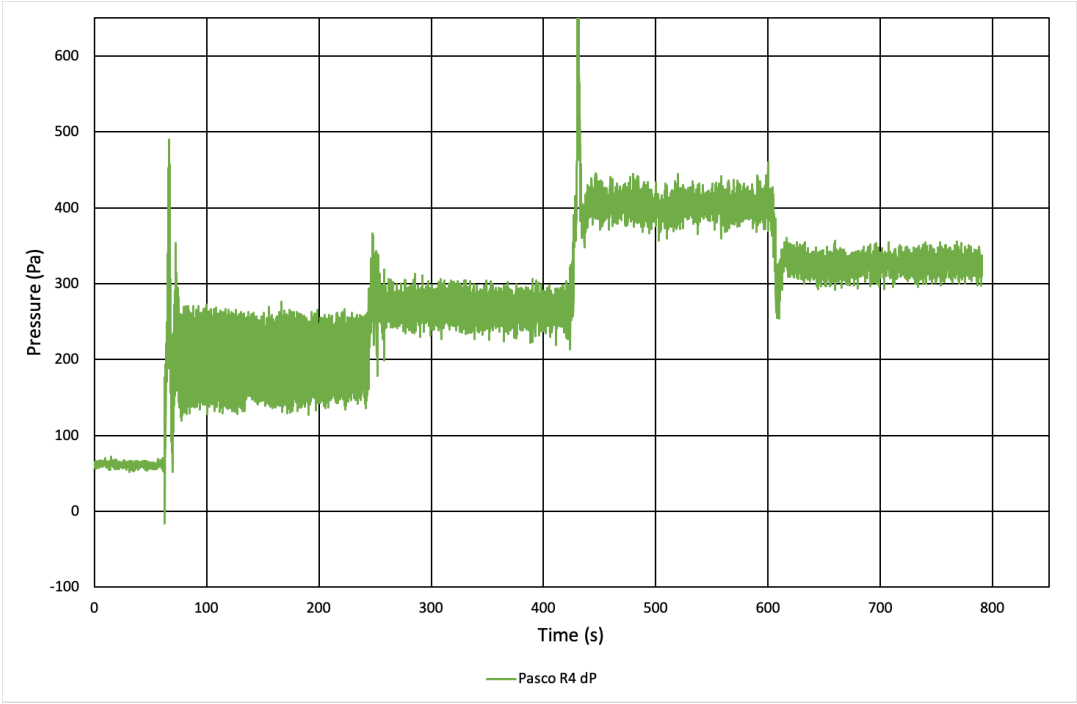


Figure B.9.: Pasco run 4 pressure results

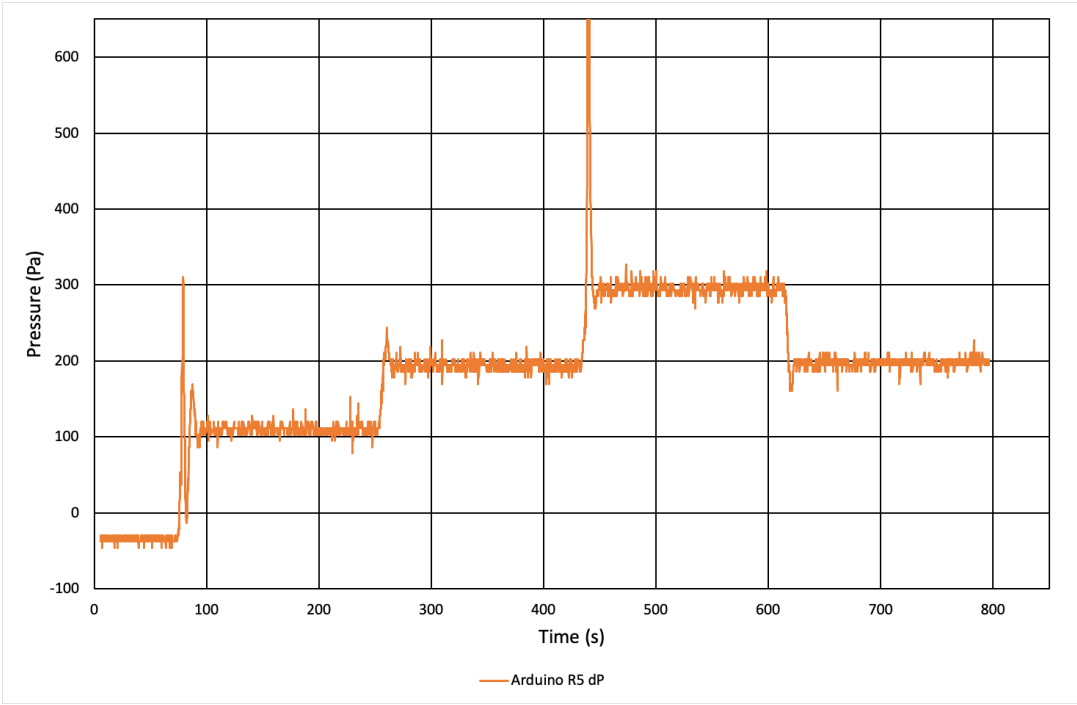


Figure B.10.: Arduino run 5 pressure results

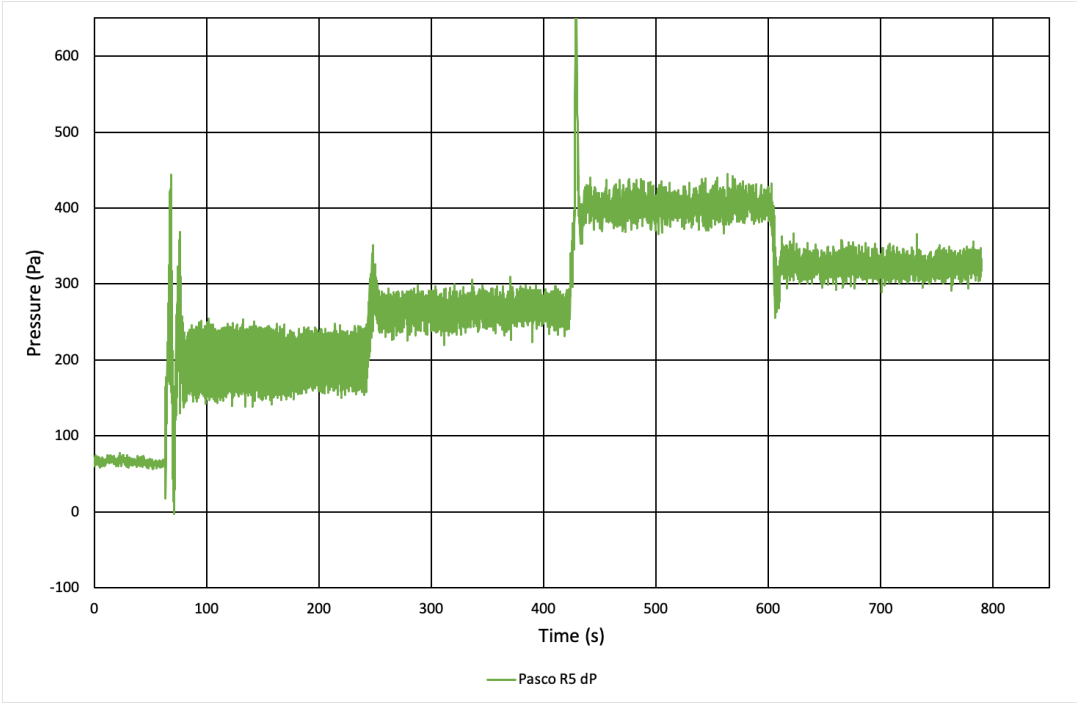


Figure B.11.: Pasco run 4 pressure results

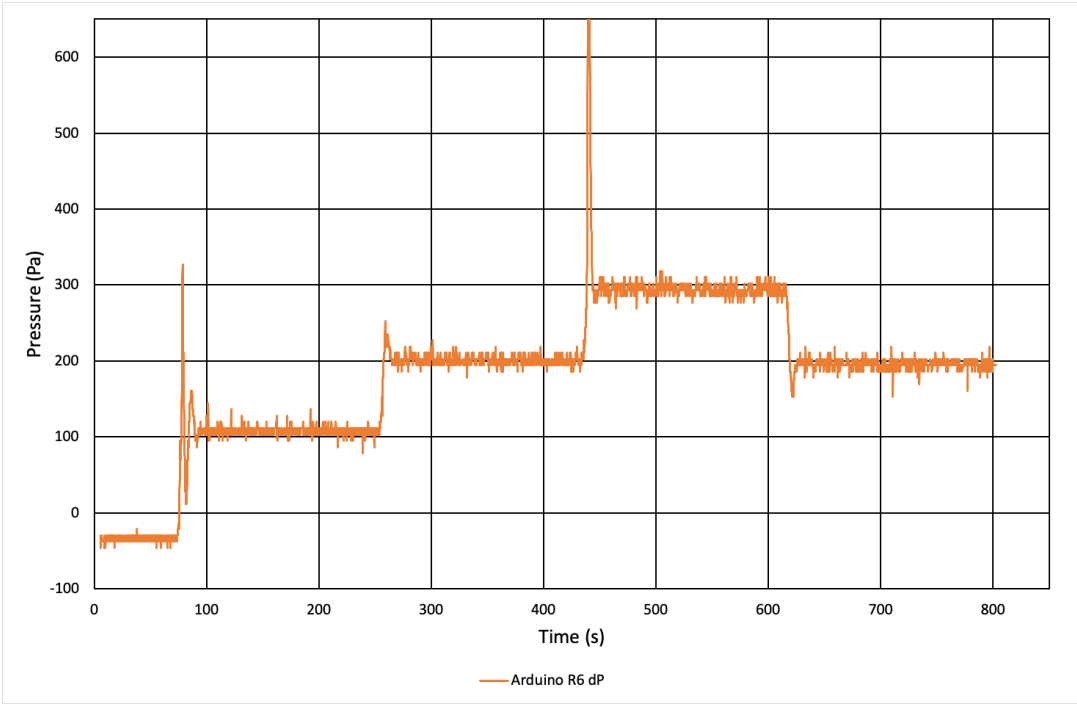


Figure B.12.: Arduino run 6 pressure results

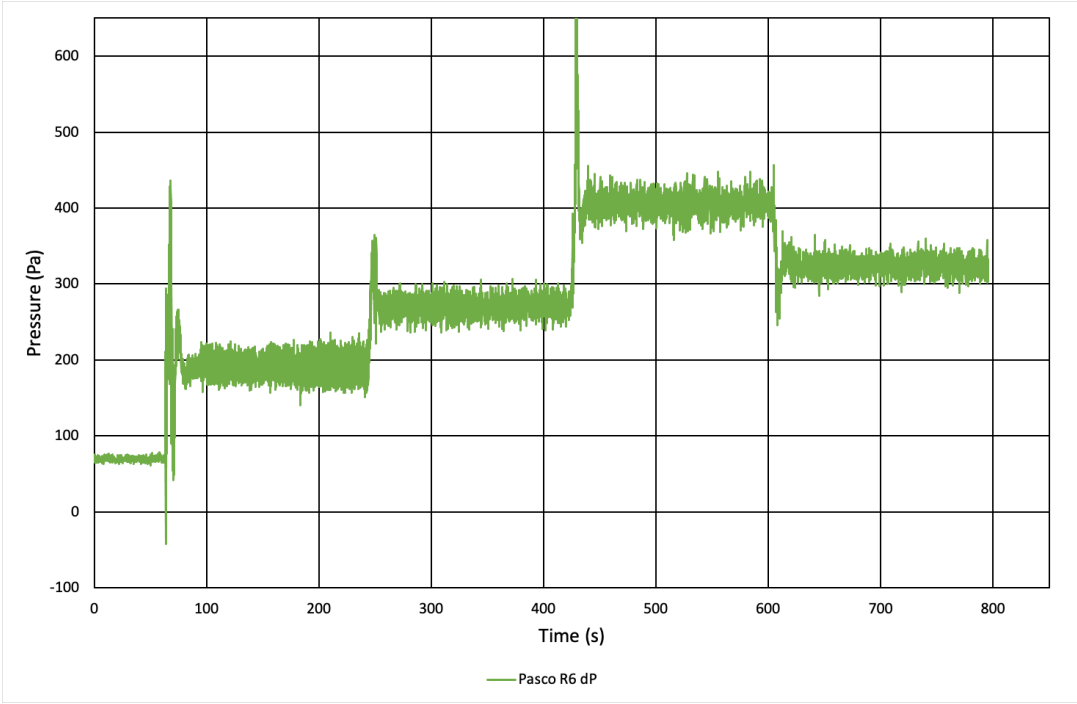


Figure B.13.: Pasco run 6 pressure results

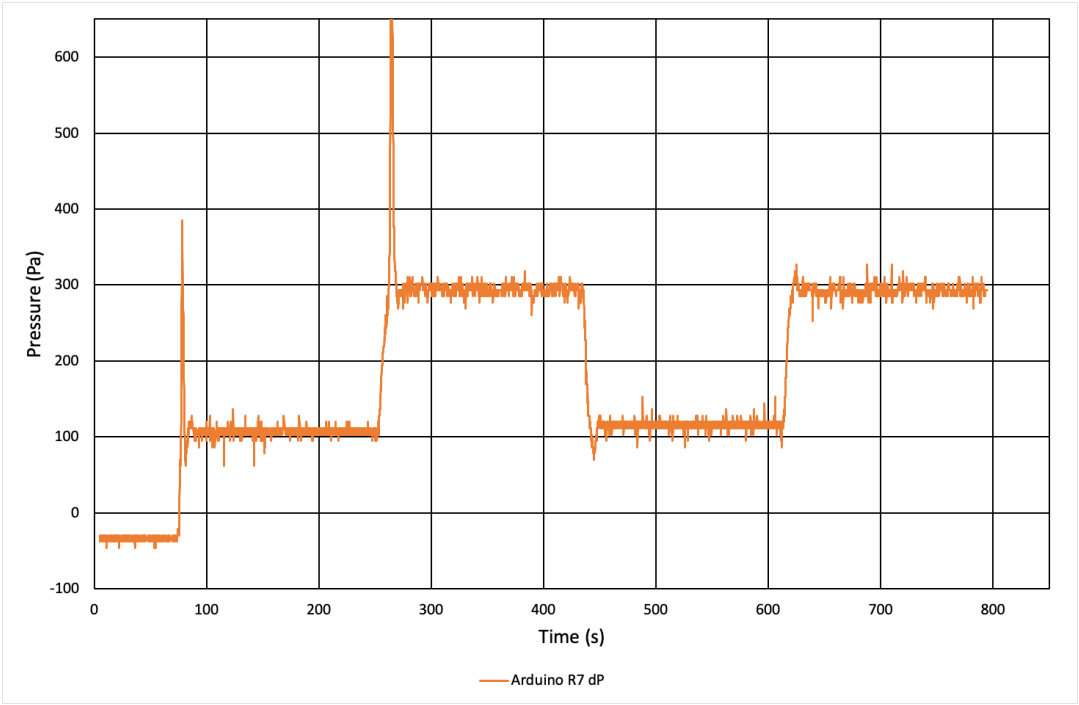


Figure B.14.: Arduino run 7 pressure results

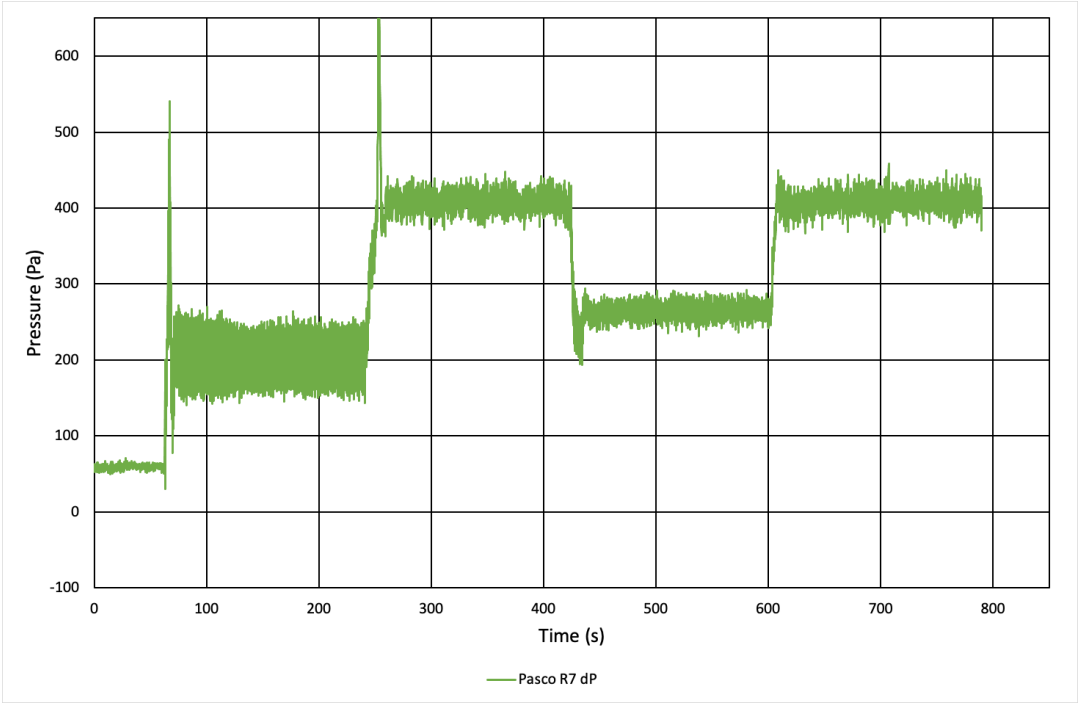


Figure B.15.: Pasco run 7 pressure results

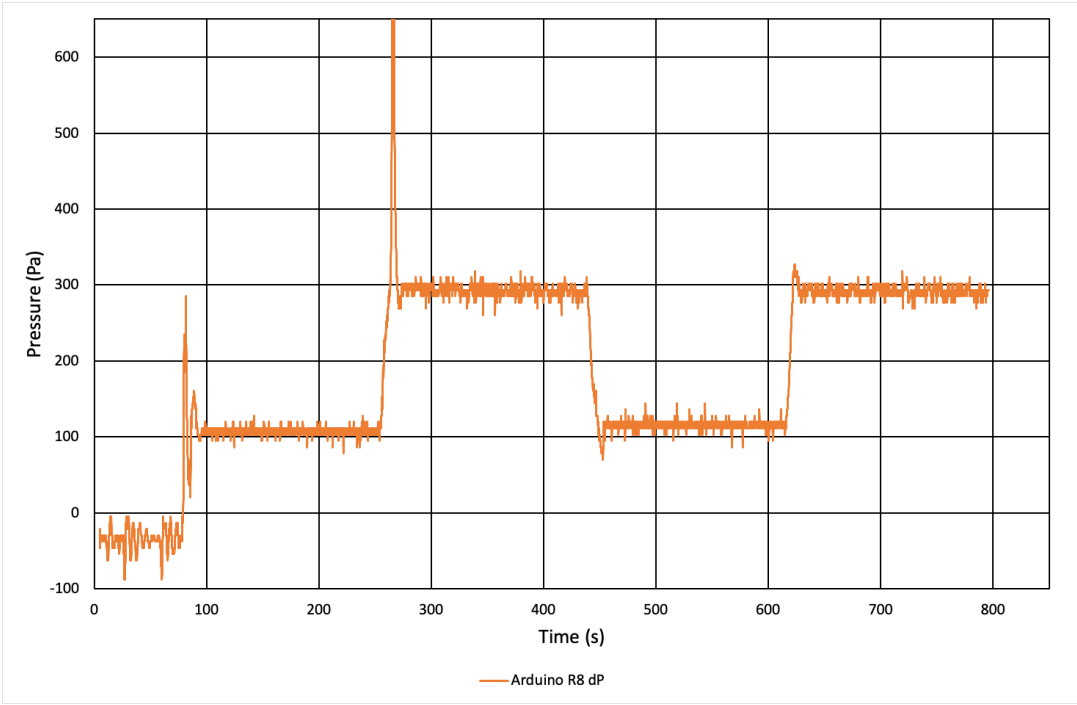


Figure B.16.: Arduino run 8 pressure results

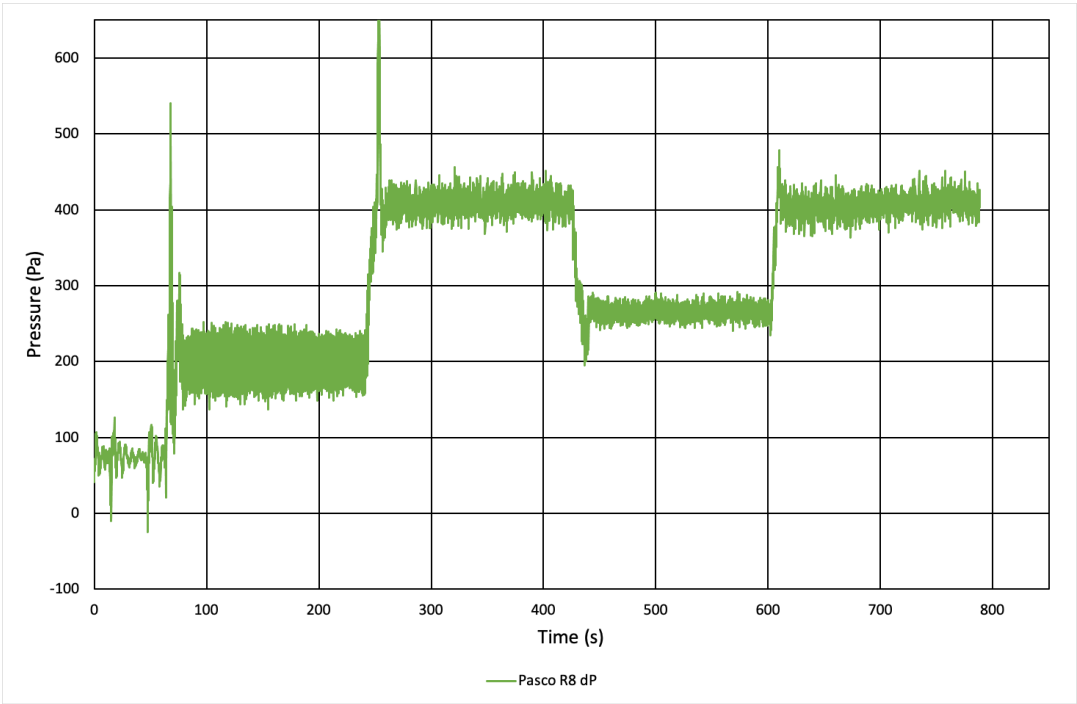


Figure B.17.: Pasco run 8 pressure results

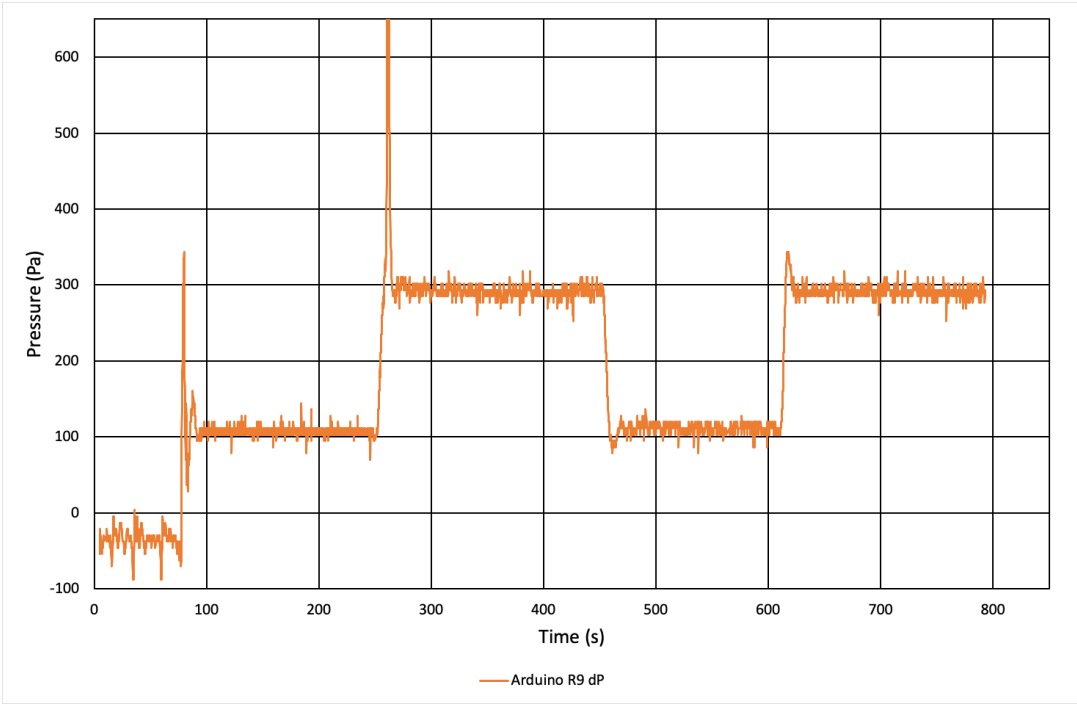


Figure B.18.: Arduino run 9 pressure results

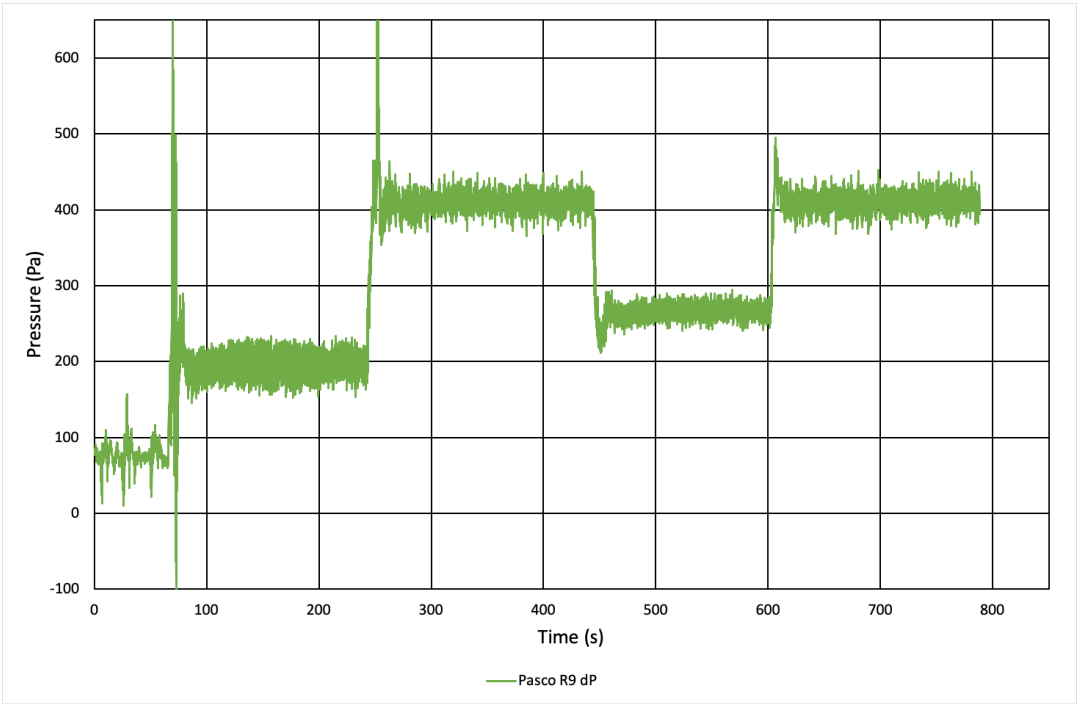


Figure B.19.: Pasco run 9 pressure results