Virtual Moon Project: Documentation

# Project Introduction

## Equipment List and Setup – Vacuum chamber and pipework

* 6-way cross vacuum chamber
  + Bottom flange: blank
  + Top flange: viewport
  + Front flange: scroll pump fitting
  + Rear flange: blank
  + Left flange: gas line entry and thermocouple
  + Right flange: pressure reader and air valve (to remove vacuum when needed)
* Scroll pump
* Copper end stage: copper block with a cut-out for curved metal gas pipe to sit, additional copper plate then placed over the top to sandwich the pipe within the block. Sits inside the chamber.
* K-type thermocouple attached to copper end stage, passes out of the vacuum chamber for analysis
* Pressure reader as above
* Air valve as above
* Foam bucket to hold LN2, with copper coil heat exchanger pipe sitting inside
* 6mm PTFE tubing with external foam insulation to carry cooled N2 gas. Runs from

copper coil heat exchanger output to the input of vacuum chamber, and from vacuum chamber output to the outside vent (open window)

* ¼ inch flexible tubing, makes up the remainder of the piping (prior to heat exchanger), handles room temperature N2
* 2x normally closed solenoid valves. ¼ inch tubing runs from N2 gas bottle to a junction of these two solenoids.
  + Solenoid 2 (S2) takes input N2 gas and bypasses the heat exchanger, taking room temp N2 gas to the other side of the heat exchanger and onwards into the vacuum chamber.
  + Solenoid 1 (S1) passes the input N2 gas to a mass flow controller (MFC). ¼ inch tubing then connects the MFC output to the copper coil heat exchanger input
* Alicat Mass Flow Controller (0-1000 SCCM)
* Unions, PTFE tape, nuts, bolts, washers, gaskets etc as required

## Equipment List – Electronics

* Arduino Uno R3
* AD8495 K-Type Thermocouple Amplifier breakout board.
* Arduino 4 relays shield (protects the Arduino, as the solenoids run on 24V)
* Adjustable power supply (to power solenoid valves with 24V)
* 12V power supply for the MFC
* Standard power supply for scroll pump
* “chocolate block” terminal connector strips of different sizes

## Equipment Setup – Electronics

* Arduino connected via USB-A to laptop (5V logic)
* Relay shield fitted to the Uno board using pre-integrated pins
* Terminal block soldered onto breakout board to enable thermocouple to be wired in
* 4x header pins soldered onto breakout board. Jumper wires used to connect this to the Arduino board
  + V+ connected to 5V Arduino pin
  + GND to Arduino GND
  + Output to chosen Arduino analog pin.
  + Additional GND connected to a common ground at the power pack to mitigate inductive interference (using chocolate block connector)
* Solenoids wired to relays 1 and 2 of the Arduino relay shield (V+ and GND) as well as into the adjustable power supply. Chocolate block connector used to achieve this.
* Manufacturer pinout used to wire connections for the MFC.
  + “Analog In” needed to set flow rate using input voltage. Wired back to PWM pin on Arduino
  + GND to complement this (wired to Arduino GND)
  + “Analog out” needed to read back the setpoint flow rate. Wired to analog input pin on Arduino
  + GND to complement this (wired to Arduino GND)
  + Additional GND connected to a common ground at the power pack to mitigate inductive interference (using chocolate block connector)

## Equipment Setup – Software

* Arduino IDE was used for initial tests and holds the majority of the hardware interface code. Required the installation of suitable libraries, detailed in code comments
* MATLAB is currently used to set the desired flow rate over serial connection to Arduino – however, the intention is to eventually run a semi-automated/regulated cycle in which the requirement for the user to set the flow rate is removed. It is likely that this connection will then just be used as a “go” button to start the experiment
* Data is sent from the Arduino to MATLAB over serial for data processing and visualisation. Currently this comprises the measured mass flow rate, temperature and the time at which each datapoint is recorded.
* The desired flow rate on the MFC is set using PWM on the Arduino (5V logic). The TimerOne library was used to enable higher frequency PWM than the Arduino default. This is important to avoid sampling errors, as the MFC samples flow rate at 1 kHz. The chosen PWM frequency is 10kHz.

# Important Notes from Initial Testing

## Thermocouple Breakout Board Implementation

In order for the Arduino to read and interpret the thermocouple temperature data, a specialist breakout board was required. This is essentially a voltage amplifier with a pre-set voltage offset and sensitivity. In this experiment, the AD8495 K-Type Thermocouple Amplifier breakout board was used. This has an accuracy of +/- 2°C within a specified measurement range of -25°C to +400°C. Outside of this range, the accuracy may decrease significantly. Further information can be found at: <https://www.analog.com/en/resources/app-notes/an-1087.html>

The AD8495 is supplied with a manufacturer-specified equation to calculate temperature from the voltage differential supplied to the Arduino:

Within the Arduino processor, the voltage from the breakout board () is read as an analog input and then interpreted into a digital value by the onboard analog-to-digital converter (ADC). The Arduino code therefore includes an equation to recover the analog value of using the reference voltage and the resolution of the ADC (10-bit, representing values from 0 to 1023):

The accuracy of this conversion is therefore affected by the value of , which is the supply voltage given from the Arduino to the breakout board. Although the 5V and GND pins were used, the actual output from these pins may not be exactly 5V. Due to the large amplification factors used in the breakout board architecture, even a small difference in reference voltage can have potentially significant impacts on the accuracy of the calculated temperature readings. It is therefore recommended that a multimeter/voltmeter be used to measure and the code adjusted accordingly.

In initial tests, this measured reference voltage was 5.06V and the code was updated to reflect this. However, the recorded temperatures were still lower than expected when compared to those recorded using the same thermocouple with an external digital reader. Calibration testing should therefore be used to trial further adjustments to in order to more accurately measure the recorded temperature.

This was carried out using an additional K-type thermocouple connected to the digital reader. Both thermocouples were first tested in ambient temperature, and after some trial-and-error, the most suitable value of came out as 5.12V. This represents a +0.06V adjustment from the “true” value.

In order to check the validity of this adjustment over a wider temperature range, the same setup was then tested using ice (to achieve a known reference of 0°C) and boiling water (to achieve a known reference of 100°C). The results of these tests were successful, and suggest that the +0.06V adjustment holds true over the range of 0-100°C.

A further influencing factor was discovered when implementing the full experimental apparatus. Namely, a cable extender was required to connect the Arduino to the laptop due to the spacing and placement of apparatus in the laboratory. The resulting resistive wire losses reduced the reference voltage seen at the Arduino pins down to 5.02V. Further adjustments and testing were therefore required, which resulted in a +0.04V adjustment of to a value of 5.06V. Additional details can be found in the Arduino code comments.

## Electromagnetic Interference

Again, due to the necessary large voltage amplification of the breakout board, the electronics system is especially vulnerable to inductive electromagnetic interference (EMI). Even small voltage oscillations elsewhere in the system can be magnified by the amplifier and interpreted as a temperature reading. This raised significant issues in initial testing of the temperature recording system when other electronics were powered on.

Preliminary testing of the breakout board “on the desktop” used a spare external thermocouple (i.e. not the one located inside the vacuum chamber) and the Arduino Uno board in isolation from any other electronics. Following calibration as described above, these tests proved successful, and no interference was seen.

However, when testing the vacuum chamber thermocouple, significant interference was recorded in a regular oscillating cycle. Temperatures were reported in a range from around 5°C - 40°C rather than reading the approximate ambient temperature of 23°C. Identification of the sources of this interference required extensive testing, where each electrical appliance was powered on in isolation. The effect of each appliance could then be observed in the thermocouple readings.

Following this, the vacuum pump power cable (which ran across the workbench next to the electronics) was found to be the primary source of interference, despite it only being on standby. The MFC was found to be the other source of EMI, but as it runs on a much smaller voltage supply (12V), the effect was significantly smaller than that of the vacuum pump.

To resolve this, several mitigating measures were implemented. Firstly, the pump power cable was rerouted to sit on the floor below the workbench – this proved to be a sufficient distance from the electronics to prevent the inductive interference effects.

In addition, a common ground system was established, where all relevant electronics were grounded to the ground of the adjustable power supply. This included the spare GND on the breakout board, the Arduino GND and a spare GND on the MFC pins.

After this grounding, the oscillatory EMI was no longer present in the temperature readings. However, a small drop in accuracy was noted, with measurements recorded within a 5-6°C range from the mean, rather than the specified 4°C from the manufacturer. Reasons for this are unclear, but for this application this is not critical. In addition, anomalous readings (currently defined as > 3 standard deviations from the mean in either direction) are easily removed in post-experiment data processing and analysis.

It should be noted that any semi-automated/regulated cycle which relies on temperature thresholds to trigger certain processes (e.g. turn off Solenoid 1 when the minimum desired temperature is reached) will be vulnerable to these accuracy deviations. Appropriate measures should be in place to prevent these from affecting the operation of the cycle (e.g. using a moving average, or recording a minimum number of readings over the threshold before the process is triggered).

The pressure monitor on the vacuum chamber has also been known to cause EMI, although this has not been seen consistently. To avoid any unexpected impacts on temperature readings, the monitor should be turned off during experiments (or somehow grounded to the same common ground as the other electronics).

## Important Notes from Initial Testing – Mass Flow Controller

* When setting the flow rate on the MFC, it was noted that low or high flow rates (towards the edge of the range) were often significantly different on the MFC reader than what was set in the code (e.g. a set flow rate of 50 SCCM was interpreted as 39 SCCM). This is likely due to the extremely short “on” (or “off” at the higher end) times associated with these flow rates.
  + This could be approximately corrected by adding a “fudge factor” of around 5 SCCM to the input flow rate. This is currently established in the Arduino code
  + It was also noted that the setpoint stabilised significantly when N2 was actually flowing through the device (compared to initial bench testing with no flow)
* Initial tests with room-temperature N2 at a regulated input pressure of 1.5 bar revealed a significant difference in flow rates between the bypass line and the flow-controlled line. In theory, at maximum set flow rate (1000 SCCM) the two should be similar, so this warranted further investigation
  + The bypass line experienced good flow and pressure throughout, with an audible sound of gas flow at the exhaust
  + However, the flow-controlled line had a significantly lower flow rate even when set to the MFC maximum of 1000 SCCM. Differences could be detected by setting the MFC to 50%, proving that the MFC itself was working, but the flows were so low that these were difficult to ascertain by sound/feel.
  + Possible reasons for this were investigated.
    - It was initially suspected that the copper coil heat exchanger could be throttling the flow due to a narrowed bore diameter at one end (present from manufacturer). This theory was tested by disconnecting the coil from the system and instead connecting a straight section of pipe of constant bore diameter. However, no improvement was noted so this was ruled out
    - A second theory was that the input pressure was too low for the MFC. However when checking on the datasheet, the minimum input pressure specified was 11.5 psia (equivalent to approx. 0.79 bar) so the test input pressure of 1.5 bar was clearly sufficient. In addition the MFC only experiences a maximum pressure drop of 1.5 psid (approx. 0.1 bar) so pressure along the pipe would have been little affected by its presence
    - The current theory is that the MFC was incorrectly chosen based on its flow rate range. When selecting the MFC, a simple online calculator was used to derive expected approximate flow rate from input pressure alone, assuming an operating pressure of 2 bar. This results in around 0.586 litres/min (or 586 CCM) – hence the decision to select the 0-1000 SCCM flow rate controller.
    - However, this was based on Poiseuille’s Law, which has two key assumptions: that the flow is laminar, and viscous. This means it is most applicable to liquids like water at low Reynold’s number. For gaseous N2, very slow velocities would be required for laminar flow, so it is almost definitely not laminar at the chosen input pressure. Gaseous N2 also has a very low viscosity which further invalidates the use of Poiseuille’s Law.
    - This means that the mass flow estimate under-predicted the reality of the gas flow, as the low viscosity results in higher flow speeds and consequently higher flow rates. So it is actually the MFC which is throttling the flow rate, even at its maximum of 1 SLPM. To solve this, a mass flow controller with a larger range should be chosen. However, this was not possible during the summer project duration due to long lead times on delivery. Instead, work progressed with the original MFC with the understanding that cooling times may take longer than expected due to lower flow rates

# Initial Testing with LN2

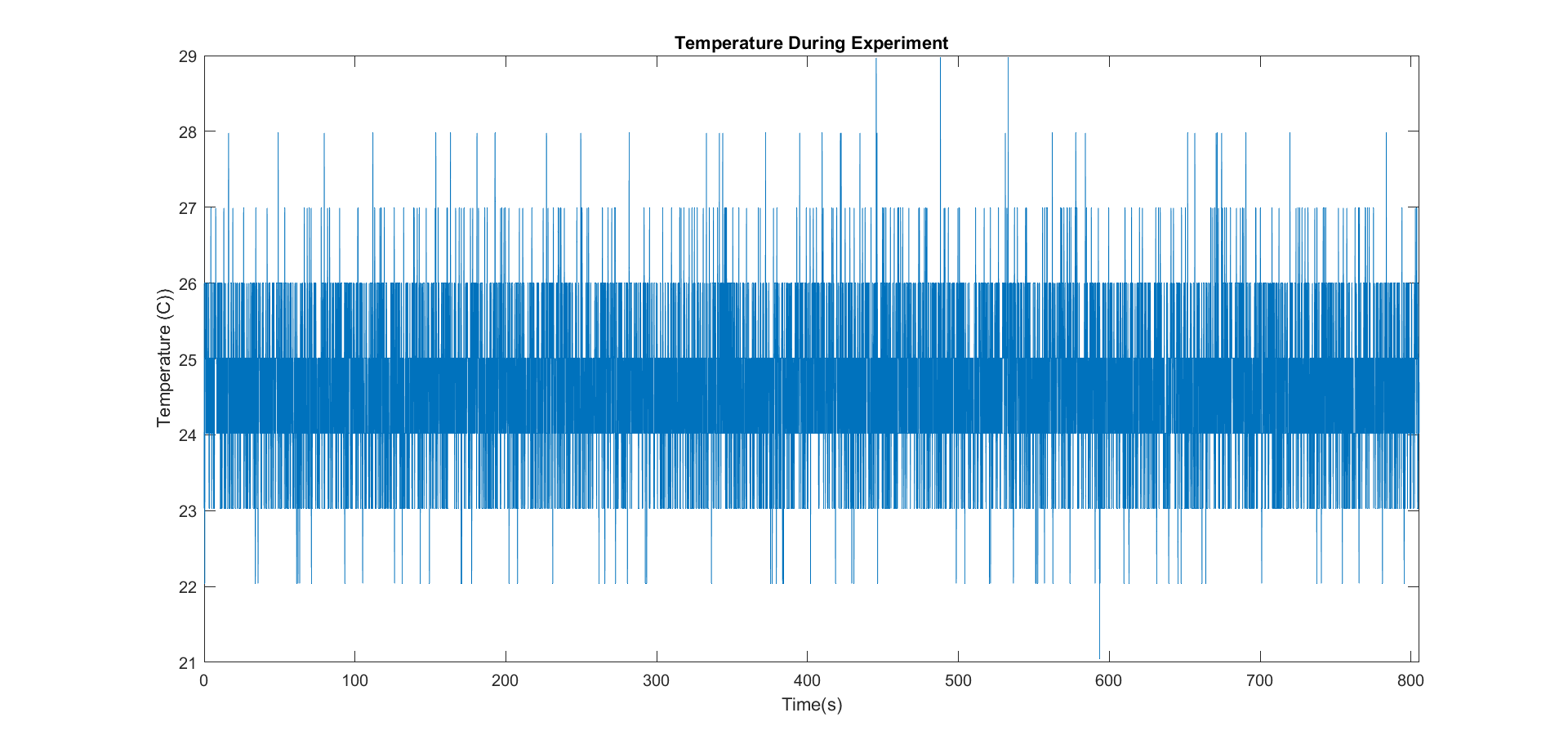
* A testing campaign was carried out to investigate the cooling curves produced by differing mass flow rates. Due to the MFC throttling discussed above, the first test was carried out at the maximum flow rate of 1000 SCCM (or 1 SLPM) with a test duration of 2 hours.
  + This test was set and left to run, however the laptop running the code went to sleep after 5 minutes due to system settings, which paused the computation. When returning after 2 hours, this was rectified, the LN2 topped up, and the code allowed to continue for another 2 hours.
  + However, the code then behaved unexpectedly and did not return the expected number of readings within 2 hours (set a request for 72000 readings which is 10 readings per second for 2 hour duration). Instead, only 7973 readings were returned.
  + The suspicion is that the mass flow controller was causing this, as it showed some unexpected behaviour on the display screen. The measured flow rates fluctuated very rapidly in a range of around 900-1100 SCCM, despite the maximum flow rate being 1000 SCCM. Since data is sent as a package including flow, temperature and time, any issues with the flow rate readings may have affected the speed at which each package was sent. Additionally, in the received data, no readings over 1000 were seen, which doesn’t match with observations seen on the device screen. This reinforces the theory that the flow controller was potentially responsible for the unexpected code behaviour
  + Over the 2-hour run; despite not collecting the expected number of readings, the temperature did not decrease at all. So even at maximum flow rate, the 1000SCCM rate is not enough to cool the end stage (see Fig.1)

Figure 1: Temperature during 2-hour test at 1000SCCM. Time scale does not match 2-hour duration due to code issues

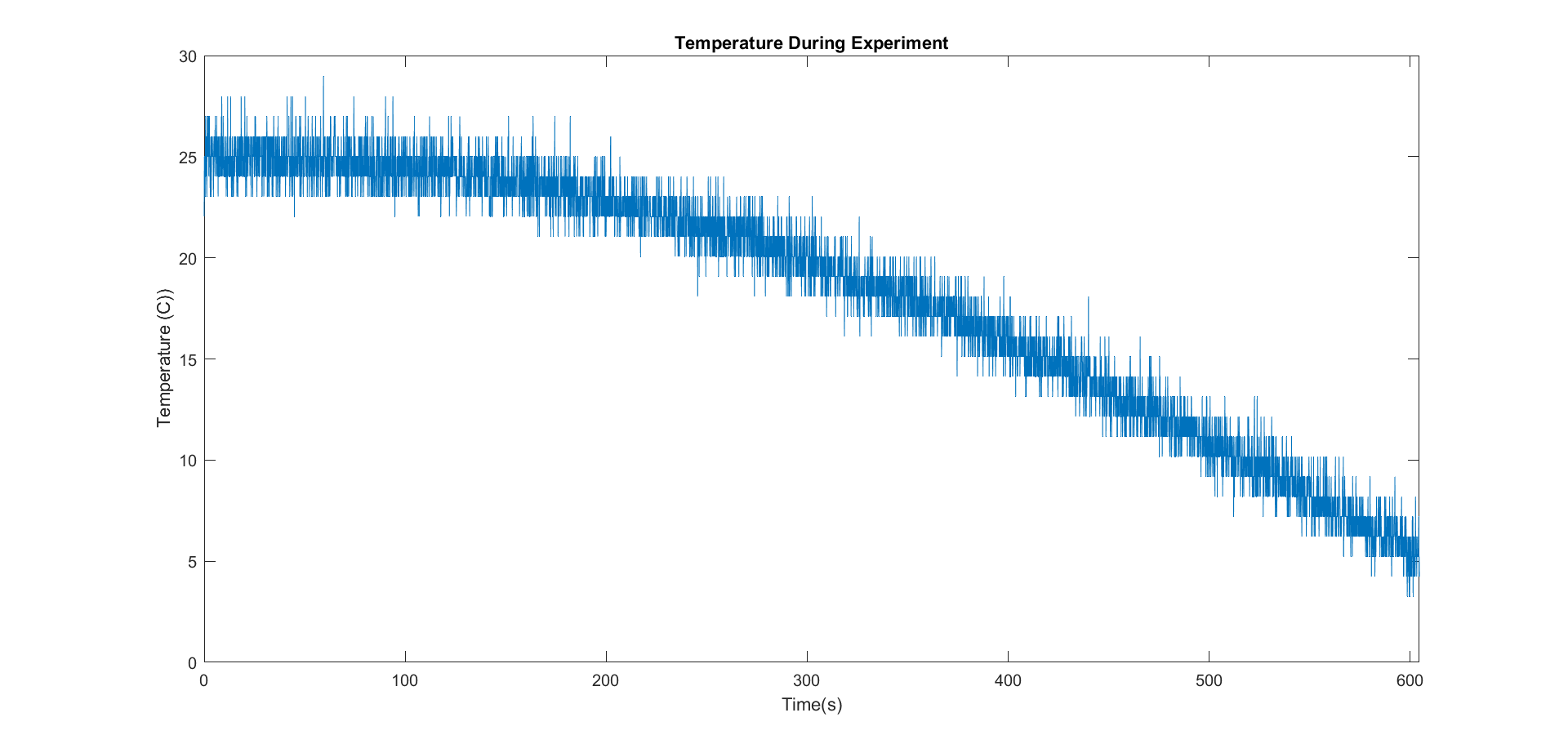
* (insert flow rate graph)
* The MFC was then removed from the system, with the S1 solenoid connected directly to the heat exchanger. The inlet N2 pressure then served as a rough means to control flow rate and was set at 1.2 bar. The experiment was then run for 10 minutes to check whether the code would work as expected and if cooling would now occur:
  + Temperature decreased down to around 3C from 25C start
  + Much more frost formation was observed on the pipework around the heat exchanger

Figure 2: 10-minute test run at 1.2 bar inlet pressure (raw data)

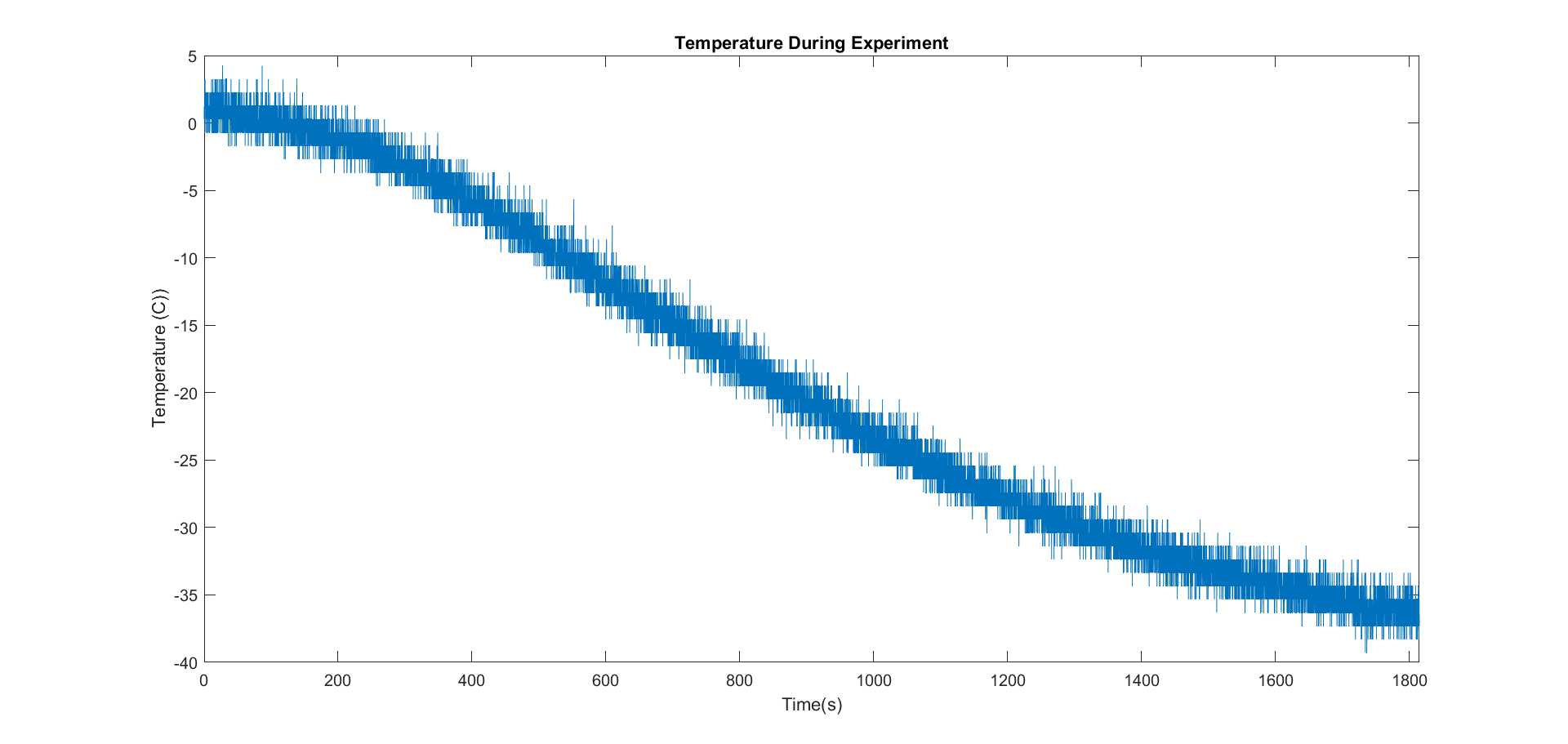
* Immediately after this, the experiment was run for a further 30 minutes to observe cooling:
  + A further decrease to around -38C was observed, however the cooling rate started to flatten out due to low levels of LN2 in the bucket

Figure 3: 30-minute test run at 1.2 bar inlet pressure (raw data)

* + At this pressure the LN2 would need to be refilled every 20-25 minutes to maintain cooling rate, this would essentially require constant work to refill due to the time taken to get LN2 from the tank outside.

# Further Testing with LN2

* Following on from the initial testing, a longer test plan was developed. The mass flow controller was permanently removed from the system as it is not possible to achieve cooling when this is throttling the flow. The bucket was filled with LN2, and the inlet N2 pressure was reduced to 1 bar. This aimed to prolong the lifespan of the LN2 as less boil-off occurs when cooling a smaller amount of gas. The experiment was run for 3 hours, aiming to observe a cooldown cycle to the minimum temperature that can be reached without a LN2 refill, and then observe the warmup cycle as the LN2 is depleted.
  + Initially things seemed positive, with frost formation on pipes at the inlet and outlet of the vacuum chamber as expected
  + After around an hour, a check was performed where it was noted that this frost had melted. After a quick investigation the cause was found to be the N2 gas running out. The gas bottle was changed and then the experiment continued. Temperature was recorded throughout.

Figure 4: 20-point moving mean temperature during 3-hour test run, outliers removed

* + N2 ran out initially at around 900 seconds (15 minutes). The temperature slowly increased until the N2 bottle was changed at 3600 seconds (1 hour). The temperature then increased slightly due to warm N2 gas flowing into the system (it took some time for it to be cooled to the stage temperature)
  + Cooling then resumed until the LN2 was depleted to a critical level. The minimum temperature reached was around -69C (pending confirmation in MATLAB)
  + Warming then occurred as the N2 gas flowing through the system gradually increased in temperature as the LN2 was no longer present
  + The N2 gas ran out again at around 8300 seconds (around 2 hours 20 minutes). This was from a fresh full bottle (2000 litres) at 1 hour into the experiment. So a flow rate of around 25 litres/minute. The warming then slowed as no gas was flowing, and the warming only occurred by heat transfer from the warmer surroundings. This was slow and quite linear in nature, as seen before when the first N2 bottle ran out
  + This could be used to hold the temperature low at the minimum without continually flowing N2 gas through the system (instead using short bursts of flow when the temperature rises above a threshold) – saving N2 and LN2
* A second test was then carried out, with the inlet pressure reduced to 0.5 bar in an attempt to conserve N2 gas and slow the rate of LN2 evaporation. This was run for 2.5 hours due to time constraints. Again the LN2 bucket was filled at the start of the experiment and allowed to deplete naturally (no refills). Checks were carried out more frequently to observe the gas pressure in the supply bottle.
  + At 30 minutes: frost had spread almost all the way down the piping, including the exhaust pipe, further than previously seen. Temperature was down to almost 80C and gas bottle pressure had already dropped to 100 bar (almost half – wasn’t exactly 200 to start with).
  + At 1 hour: reduction in frost seen at the far ends of the exhaust pipe. Temperature was up to approx. -70C. Gas bottle pressure was down to 50 bar. LN2 was still present in the bucket (clouds/vapour seen) but not enough to allow further cooling, so slow warming was occurring instead. LN2 was therefore the limiting factor on this run rather than the N2 supply.
  + At 1.5 hours: frost melted off the entire system – LN2 had completely ran out. N2 bottle ran out completely at 1h 35 minutes. Temperature up to around -30C.
  + At 2 hours (no gas flow): reached around -8C through natural warmup.
  + 2000L (approx.) N2 gas was spent in 95 minutes, giving a rough flow rate of 21 litres/min.



Figure 5: 20-point moving mean for 2.5-hour test run, outliers removed

*Summary of initial issues testing with LN2*

* The flow rates of regulator-set pressure (1 bar or 0.5 bar) are so high that the N2 bottle depletes within a matter of 1-2 hours. With the 5-bar regulator used, finer adjustments are very difficult
* The high flow rates of N2 also cause increased rate of evaporation of LN2 as a large quantity of warm gas is fed into the heat exchanger
* The LN2 reaches a critical level, beyond which no further cooling is possible and slow heating occurs instead. This occurs relatively quickly and creates challenges in terms of resupply
* There is a balance to be made in terms of lower flow rate (prolong the life of N2 gas bottle) with cooling rate. Flows that are too low are already proven to fail at cooling the end stage. Flows that are too high cause rapid loss of N2 and LN2 and may also not cool enough due to the limited time spent in contact with the heat exchanger. Classic optimisation problem?

*Testing a solution: Needle Valve*

* Previous testing identified that the Mass Flow Controller (maximum flow rate of 1000 SCCM) was incapable of providing a high enough flow to cool the copper block. However, relying on regulated input pressure alone results in high flow rates (even at “low” input pressures of e.g. 0.5 bar) which quickly uses up the gaseous N2 and also causes LN2 to boil off quickly. Finer adjustments of input pressure (e.g. to 0.25 bar) may solve this, but are difficult to achieve on a 0-5 bar regulator.
* Consequently a different solution was proposed, using a needle valve in place of the MFC. This aimed to enable finer adjustments of flow rate to achieve cooling using minimal gaseous N2 and LN2
* This was tested on a 2-hour experimental run, including a successful implementation of a new live-plotting function which made it possible to see trends in real-time. The needle valve was opened to full capacity, and input pressure set at 0.5 bar. Unfortunately this did not create a large enough flow rate to create cooling, so at 30 minutes into the experiment, the input pressure was raised to 1.5 bar.
* However this also did not produce meaningful results. Only around 2C of cooling was seen over the whole 2-hour period. Contrasting with previous runs at 0.5 bar without the valve where almost 100C of cooling was seen.
* As a result, the conclusion is that the needle valve (designed for low flow rates) throttles the flow too much, in the same way that the MFC did previously

*Further proposed solutions*

* Recirculation of N2 gas (requires pump/fan) – would reduce N2 usage dramatically as not venting everything to air (idea being an initial cooldown stage which would vent to air, then recirculating cooled N2 back through to reduce N2 usage and LN2 boiloff as lower temperature gradient
* Pre-cooling stage using a Peltier module – could cool incoming N2 gas down from 25C to around -40C which would again reduce the temperature gradient, slowing LN2 boiloff and allowing lower temperatures to be reached
* Nitrogen compressor (long term)? Which would generate LN2 at location – whatever we do, LN2 will not last for more than one cycle and we need to do multiple temperature cycles to accurately replicate multiple lunar nights-days.