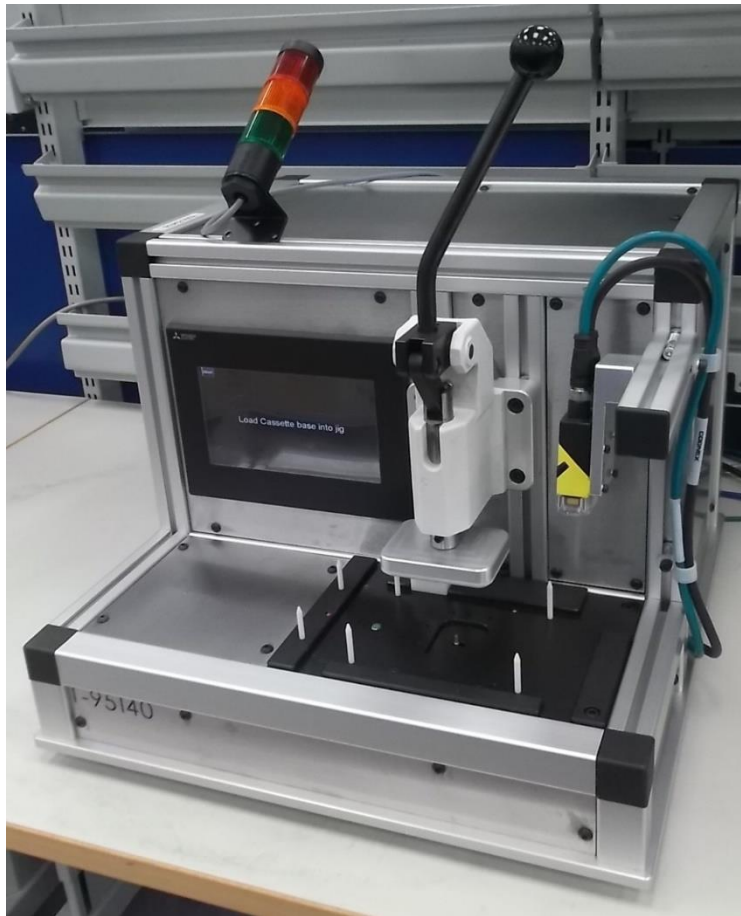




CONTROL SYSTEM MODELLING

CONTROL TECHNOLOGY

ACMA6001

Ryan Harris

Student: 2115244

Contents

Table of Figures	2
Introduction	3
Task 1	4
Product/Jig Overview	4
ATOM Burn-in Chamber	5
System Breakdown.....	6
Sensor-Controller-Actuator	6
Sampling Time.....	8
Software used to analyse the system.....	8
Brief Description of System Communication	9
Improvement to the system.....	9
Task 2	10
A Car Cruise Control System.....	10
Encoding & Decoding for the Sensor, Demand & Actuator	10
Feedback Controller & Selecting Suitable Gains	11
Suitable Gain Selection	12
Trade-off between velocity response and control effort	14
Number of Bits and the Affect on Closed Loop Performance	15
Sampling Time.....	16
List of References	17
Appendices	18
Appendix 1 – Process flow for the ATOM Chamber Burn-in process.....	18

Table of Figures

Figure 1 - PID Controlled Furnace (Inst Tools, 2023)	3
Figure 2 - ATOM Burn-in Chamber CAD (front)	4
Figure 3 - Renishaw ATOM Encoder	4
Figure 4 - PCB board & Cassette.....	4
Figure 5 - ATOM Burn-in Chamber Key Features.....	5
Figure 6 - OMRON Temperature Controller	6
Figure 7 - Top Level System	6
Figure 8 - Block Diagram	7
Figure 9 – LabView Visualisation Software Tool.....	8
Figure 10 - System Response Mapped Per Slot Position	8
Figure 11 - System Communication	9
Figure 12 - Encoding & Decoding Script.....	10
Figure 13 - Feedback Controller	11
Figure 14 - Modeling Response with Low Proportional Gain.....	12
Figure 15 - Modeling Response with High Proportional Gain.....	13
Figure 16 - Increasing P Gain and Introducing I and D	13
Figure 17 - Finding a Balance for the System	14
Figure 18 - Decreasing Number of Bits	15
Figure 19 - Increasing Number of Bits.....	15

Introduction

Control systems are used frequently in modern industry, finding ways to manufacture more efficiently, control variables and execute operations at a pace that meets demand has become critical. A control system is defined as a variable quantity or set of variable quantities that are made to conform to a set of conditions. Either holding the values of the controlled quantities constant or cause them to vary or react in a set way (Brittanica, 2023).

There are two main types of control systems, open loop, and closed loop. The open loop control system is a type of control system in which the output has no effect on the control action. Therefore, an open loop system is said to have no feedback loop to allow in sequence monitoring to adjust the output (Ragno, 2023).

In contrast to an open loop system, a closed loop system utilizes feedback to monitor and adjust its output according to the error feedback. The system is described as self-adjusting and is a far more accurate system than open loop. An example of a closed loop system is a cars cruise control system in which the speed of the vehicle is maintained by the sensing of speed and the resulting adjustment in throttle.

One of the most common types of control system used in industry is the PLC (Programmable Logic Controller) that is programmed to manipulate inputs and outputs to control a system. A PID (Proportional-Integral-Derivative) controller is again another control device that can be used to regulate a systems temperature, flow, pressure, or speed. A PID controller is described by OMEGA as the most accurate and stable controller, utilizing control loop feedback as a mechanism to control set variables (OMEGA, 2023).

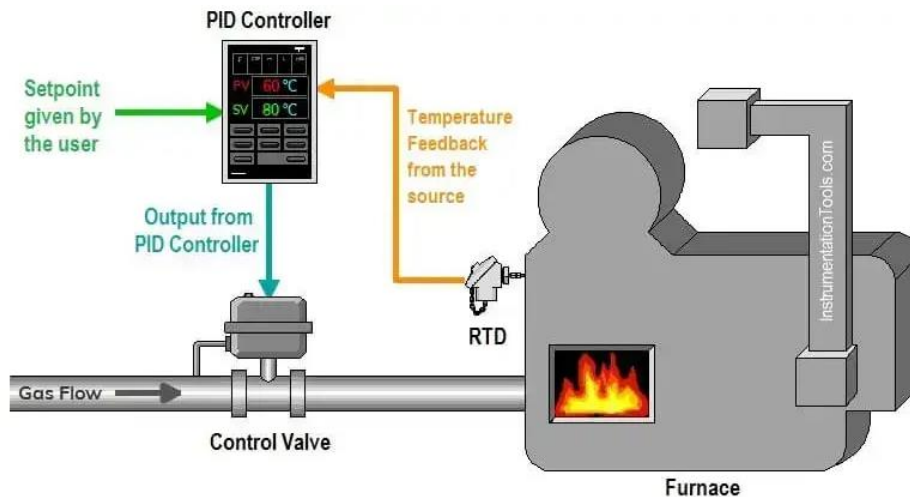


Figure 1 - PID Controlled Furnace (Inst Tools, 2023)

Task 1

Product/Jig Overview

The ATOM DX burn-in chamber is a system that is used by Renishaw during the manufacturing process of the ATOM encoder (Figure 2) that is sold on a worldwide scale by the company. The chamber pictured in Figure 3 is used to run the Burn-in process for the PCB's used in the ATOM encoder. This process is a temperature/signal testing procedure that allows the 1-week continuous life testing to confirm the stability of the product. The chamber has 24 slot positions within, this is where a PCB board with 24 individual PCBs (Figure 4) are tested. These boards are placed within another tool which is called a cassette as pictured in Figure 4, The cassettes are then placed in the 24 positions within the chamber. This means that 576 boards can complete the Burn-in process per week per chamber.



Figure 3 - Renishaw ATOM Encoder

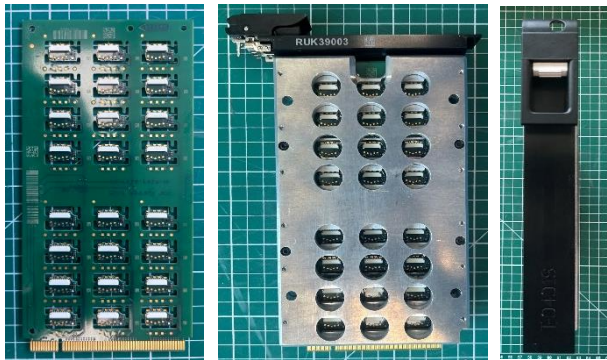


Figure 4 - PCB board & Cassette

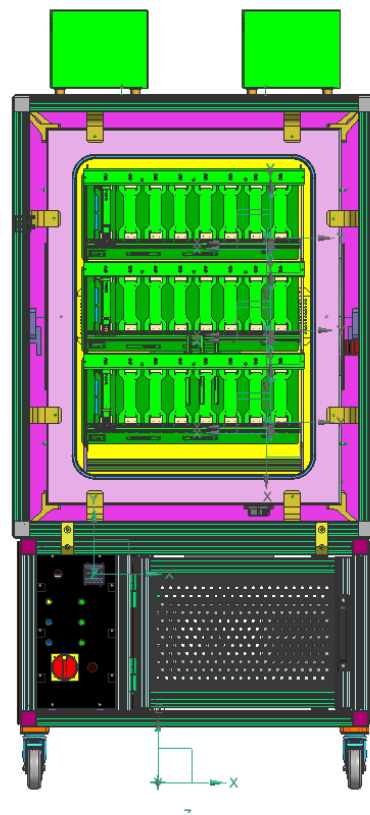


Figure 2 - ATOM Burn-in Chamber CAD
(front)

ATOM Burn-in Chamber

Some of the key features of the Atom Burn-in chamber include the exhaust fans that control the cooling process of the chamber, the side heater assembly which controls the heating of the chamber, the Oven Frame Assembly which houses the PCB's during testing and the OMRON PID controller that ensures a constant temperature of 50°C. With reference to Appendix 1 the process of Burn-in is as follows:

1. Empty chamber door opens via interlock release.
2. Operator loads 24 x Cassettes into the Oven Frame Assembly within the Chamber.
3. Operator closes door interlock confirms.
4. Side heaters switch on until the temperature sensor confirms 40°C.
5. PCBs are powered up.
6. Temperature sensor confirms 45°C.
7. Exhaust fan begins to run to maintain internal temperature at 50°C for one week.
8. After a week, PCBs powered off and exhaust fans run until 25°C.
9. Temperature sensor confirms, door opens operator unloads and Burn-in process is complete.

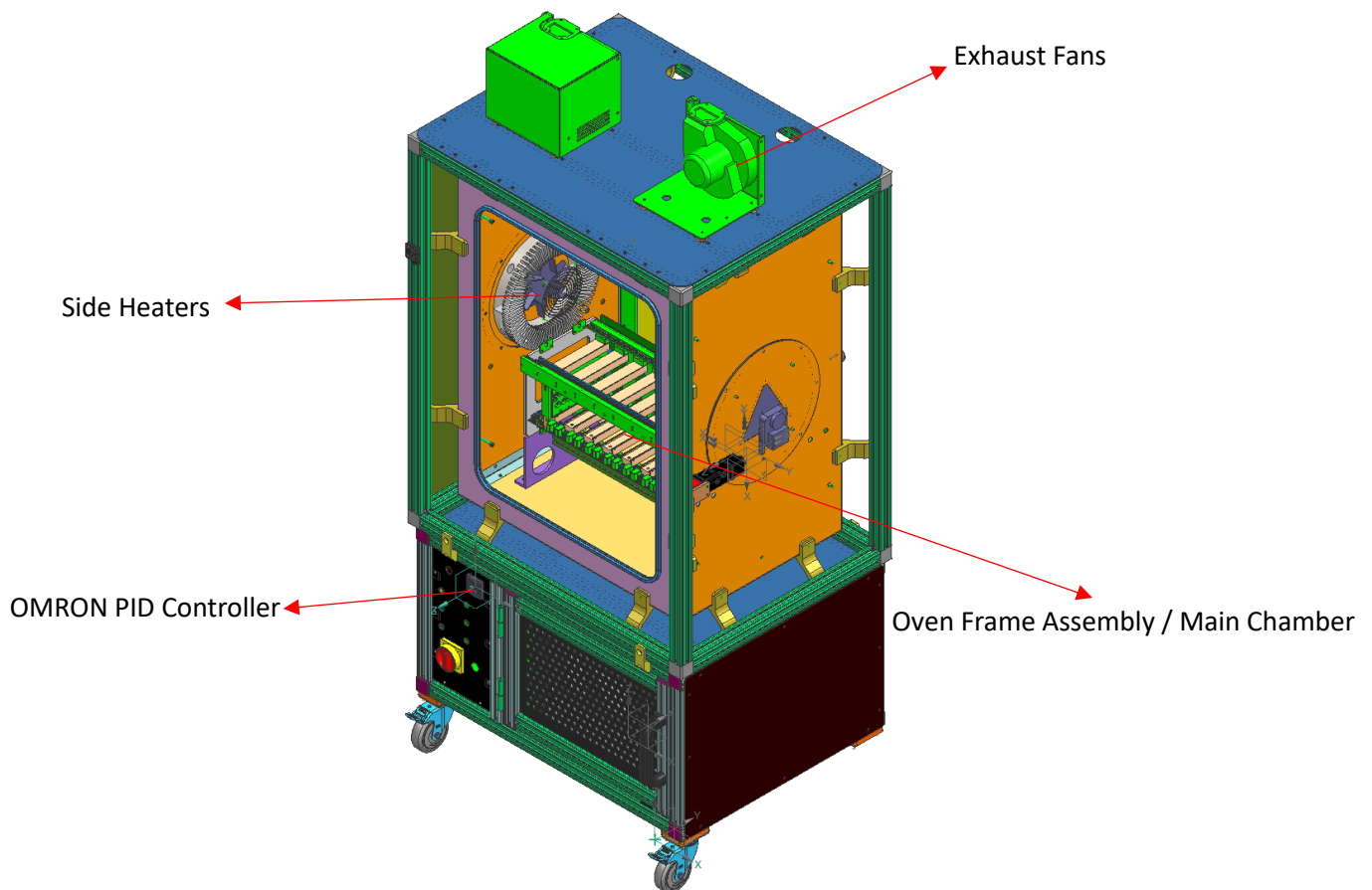


Figure 5 - ATOM Burn-in Chamber Key Features

System Breakdown

The controller that's used in the Atom Burn-in system is an OMRON temperature controller E5CN-HV2MD-500 as shown in Figure 6. Some of the key features of this controller include indication accuracy for thermocouple input of $\pm 0.3\%$ of PV (process variable), Pt (Platinum Thermometer) input of $\pm 0.2\%$ of the PV and analogue input $\pm 0.2\%$ FS. It also has the capability of sampling at 250ms. There are many temperature controller manufacturers which all tend to have different functionalities and ideas of accuracy required. The reason Omron was used is due to its precision, user friendliness and overall performance. The auto tune feature has been used frequently to find the optimal values for the proportional, integral, and derivative parameters.



Figure 6 - OMRON Temperature Controller

Sensor-Controller-Actuator

There are two main mechanisms of control, heat generated by the side heaters and the exhaust fans which help to control and maintain internal temperature. The PID is required to control this temperature and ensure that the temperature is stable throughout the Burn-in process. A top level of the sensor-controller-actuator system is shown below:

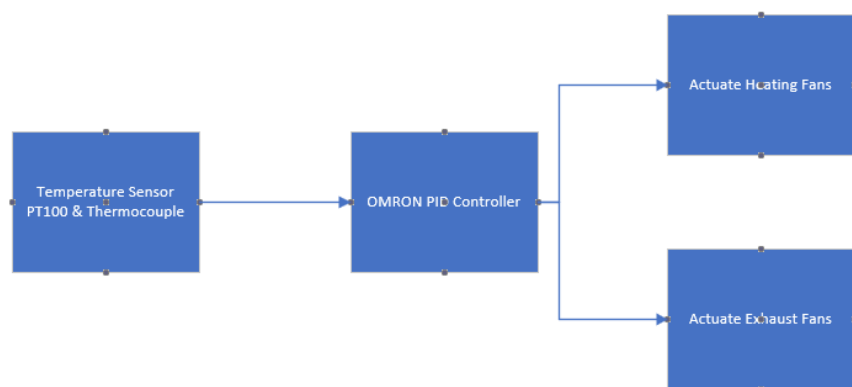


Figure 7 - Top Level System

The temperature sensor (PT100) senses the chambers internal temperature which is fed back to the PID controller. The PID controller then decides whether to actuate side heating fans to build up temperature or actuate the exhaust fans to cool the system. This is to keep a constant 50°C throughout the process.

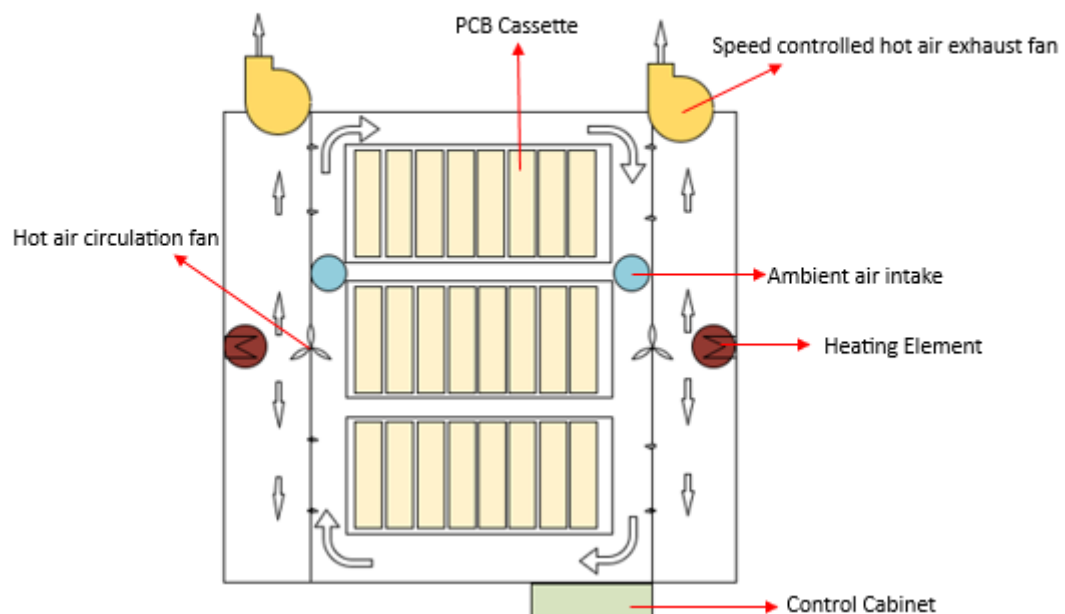


Figure 8 - Block Diagram

A factor in the chamber that was initially overlooked was the extra temperature that would be generated by the power up the PCBs within the chamber. The PCB's house LEDs amongst other components that create an extra heat variable. This is why good circulation was required by the circulating fans to monitor the ambient air as closely as possible. If there was poor air circulation, then the temperature sensor may not be feeding accurate data back to the PID controller.

There's a slight difference between the heating and cooling phase, during the cooling phase the fan speed can range from 20% to 100% effort. Whereas the heating is either on or off. The cooling fan is controlled by the LabVIEW programme and voltage is in turn converted to fan speed, the more voltage across the exhaust fans the faster the exhaust hence the cooling.

Pulse width modulation is used to control the heating element on and off – if a small amount of heat is required then it's a long time base PWM, Increasing the frequency on and off increases number of seconds heating. Will be on longer if more heat is required. 50-60Hz is normal pulse width modulation in systems however the chamber uses relays that will become damaged if constant switching was required. The PID controller controls the frequency of the PWM and will dictate how long heating element is required to be on. As stated by Salus Controls (2023) In PWM, the difference between the set temperature and the current temperature determines how long the heating device will run.

Sampling Time

The system samples every 30s checking how the temperature has changed. This is to enable the PID to adjust required temperature, the reason the sampling time is infrequent is because temperature is not a quick changing variable and sampling time is less critical. Sampling time becomes more important in systems that require continuous monitoring such as medical ventilators, robotics, and automations as well as process control in areas such as the chemical industry. MATHWORKS (2023) explain that controlling the rate of block execution through appropriate sampling time is critical to a control systems performance.

Software used to analyse the system

The ATOM chamber utilises LabView as the main software to run the programme. Data is logged in LabView via the visualisation tool. With reference to the figure 9, there are three different gain settings that are called by the software. For example, in PID set 3 there is no integral or derivative gain, and the proportional gain is maximised. When this set is cooled the exhaust fans run at 100% capacity rapidly cooling the chamber, allowing operators to remove the PCB's. Figure 10 shows the temperature and response for each of the 24 slot positions, this again is used to analyse system performance.

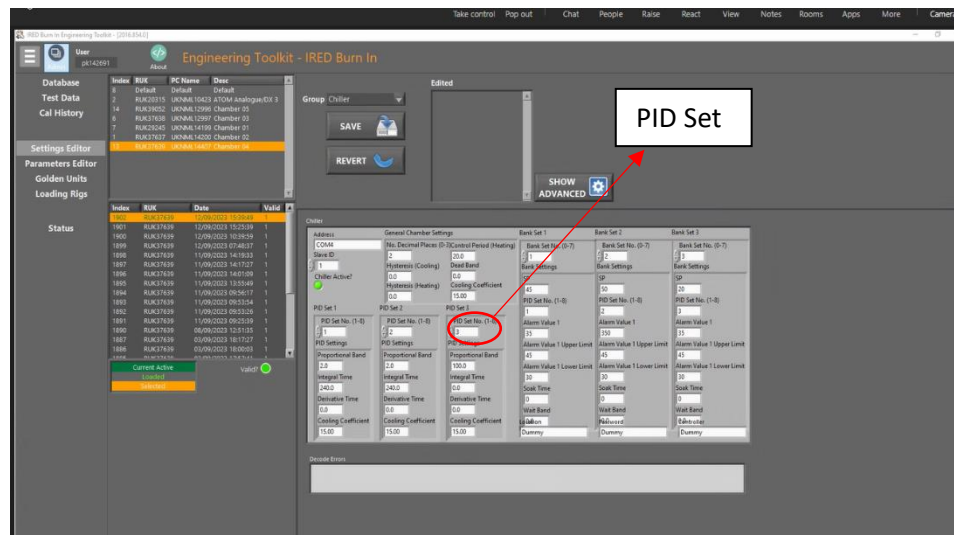


Figure 9 – LabView Visualisation Software Tool

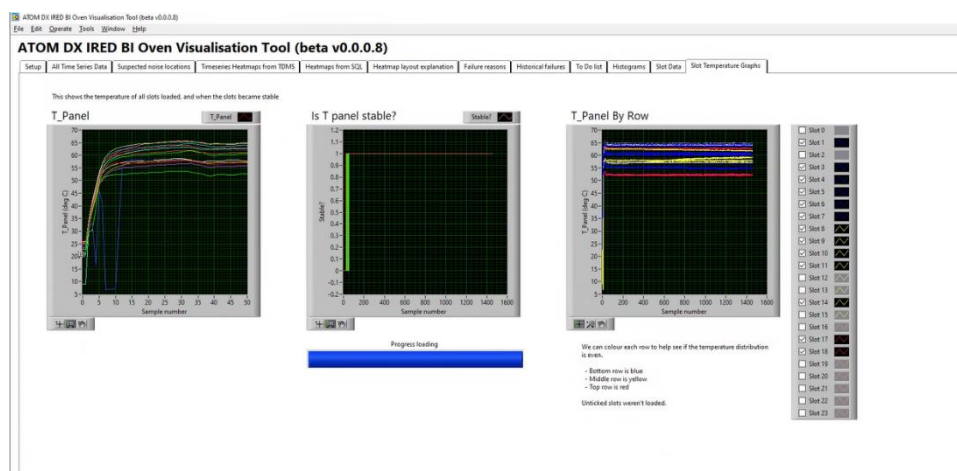


Figure 10 - System Response Mapped Per Slot Position

Brief Description of System Communication

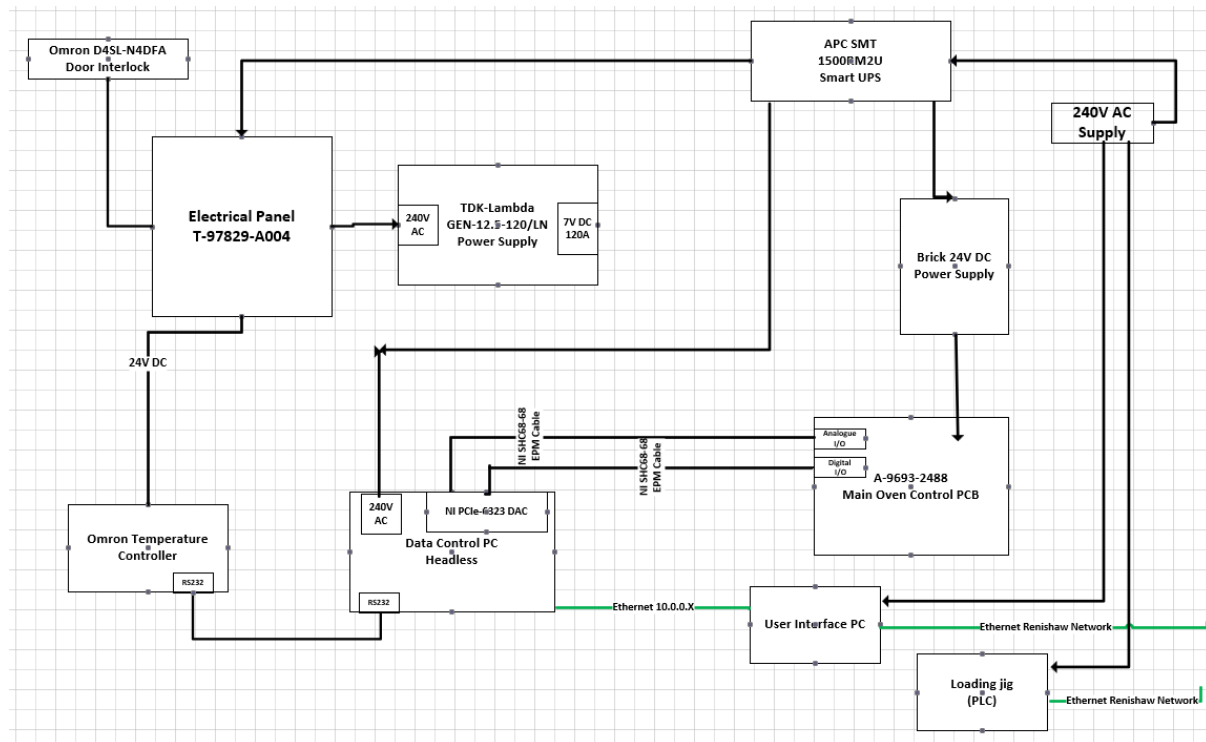


Figure 11 - System Communication

Figure 11 is a top-level system diagram; this shows that the Omron temperature control is powered via 24v from the electrical panel. This is where the relays are stored for the switching of both the heating and cooling phases. There is RS232 port comes between the Data control PC and the PID controller. Some other notable communication includes sensor PC160 analogue input, LabVIEW coms from the PC is serial RS232 this is not continuous and not part of the PID loop. With cooling fan 0-10v converted to PWM signal that controls the speed. Heating element is a digital output either on or off solid-state relay with high switch frequency.

Improvement to the system

As a brief mention, the only area of improvement would be the circulation of the air. This is difficult to control because of the oven rack assembly. It is not the most aerodynamic this can be a problem circulating ambient air efficiently cooling the chamber.

Task 2

A Car Cruise Control System

A car cruise control system is common in many vehicles that are being produced today. The system controls the speed of a car by adjusting the throttle position. The system sends a signal to the vacuum actuator which in turn increases and decreases the speed of the car via the throttle. An ideal cruise control system according to Nice (2021) accelerates aggressively to speed without overshooting, maintaining this speed with little deviation no matter the variables such as road condition and weight.

A Proportional-Integral-Derivative (PID) controller is widely used in regulating a systems output based on feedback. The Proportional part of the system determines the strength of the response of the car, ideally reaching the desired speed without overshoot or oscillation. The integral gain eliminates ESS (steady state error), this is the error persisting after the proportional term has acted. The derivative gain further dampens the overshoot, allowing the system to settle quickly.

The remainder of this report sets to evaluate a cruise control system modelled in MATLAB which features a virtual driver with a mapped response of the cruise control system.

Encoding & Decoding for the Sensor, Demand & Actuator

```
%% (4) simulation loop %%
velocity_real = 0; %Change this to 0 - this defines starting speed of the vehicle

for j = 1:length(T) % for every simulation step

    loops_remaining = length(T)-j % display progression

    % get vehicle raw velocity sensor measurement from the communication bus
    [velocity_raw, velocity_real, throttle_setting] = getRawVelocity(n_bits_sensor_signal, velocity_real, P, throttle_setting);

    % decode raw bus signal (from x-bit binary to mph)
    velocity = ((bin2dec(velocity_raw).*(M_velocityS-m_velocityS))/R)+m_velocityS;

    if mod((j-1)*sim_step, Ts) == 0 % every sampling time

        % get driver's velocity demand raw signal from the communication bus
        demand_raw = getRawVelocityDemand(n_bits_sensor_signal);

        % decode raw signal (from x-bit binary to mph)
        demand = ((bin2dec(demand_raw).*(M_demand-m_demand))/R)+m_demand; % determining demand signal

        % calculate control signal (range [-100,100])
        [control, error] = calcController(velocity, control, demand, P, I, D)

        % encode control signal into x-bit binary number
        control_raw = dec2bin(((control-m_ControlSig)/(M_ControlSig-m_ControlSig)).*R) ;

        % send control command to throttle/brake actuator via communication bus
        sendControlCommand(control_raw, n_bits_actuator_signal, actuator_signal_delay);

    end
```

Figure 12 - Encoding & Decoding Script

Figure 9 illustrates a part of the MATLAB script which sets to encode and decode raw velocity data from binary to miles per hour. Some of the notable parts of the script are as follows:

READING VELOCITY DATA: These include variables stored which directly relate to the velocity of the car, information regarding 'velocity_raw', 'velocity_real'. There is also a function stored within this code that sets about retrieving the raw velocity via a communication bus, this is known as 'getRawVelocity'.

DECODING: The decoding of the velocity is an important part of the code; the system decodes binary format to real velocity. The binary code is required for the cruise control system to understand demands. The part of the code 'velocity_raw' is converted to miles per hour (MPH). This is then stored as a new variable 'velocity' ready for further utilization by the code.

READING VELOCITY DEMAND (DRIVER): A raw signal from the simulated driver 'demand_raw' is received which indicates the velocity required to cruise. The code then retrieves the raw demand signal 'getRawVelocityDemand'.

DECODING DEMAND: 'demand_raw' is another binary format converted to real world MPH. this decoded demand is the stored as a variable 'demand'.

CONTROL CALCULATION: This part calculates the control signal based on the actual velocity, desired demand, and the gains (PID).

ENCODING: The control signal that is calculated is encoded back into binary a format that is suitable for the controller.

SENDING CONTROL COMMAND: the control signal is finally transmitted from the controller to the actuator in this case throttle for velocity increase and brake for velocity decrease.

In essence the encoding and decoding code forms part of the control loop, reading and translating velocity and demand information, processing, calculating control signal and sending it to the controller in the correct binary format to limit or increase the vehicles velocity.

Feedback Controller & Selecting Suitable Gains

The feedback control is a fundamental part of any control system. In this case, it allows the vehicle to automatically adjust to the driver's demand creating a smooth drive. The feedback control script used in the model of a cruise control system is a function called 'calcController'. A snippet of this code is illustrated in Figure 10.

```
function [contout, error] = calcController(velocity,control,demand,P,I,D)

global sim_step u_vec

error = demand - velocity; % calculates the error value

dist_int_diff = 6;

if dist_int_diff >= sum(size(u_vec)) % uses 'control' when u_vec is less than 6 (dist_int_diff)
    prev = control
else
    prev = u_vec(end-dist_int_diff-1) % when u_vec is greater than dist_int_diff find 6th from last value
end

P_part = error * P; % Proportional multiplied by the error

Int_part = (((control+prev)/2)*(sim_step*dist_int_diff-1))* I; % Integrate and multiply by 'I' gain

diff_part = (((control-prev)/(sim_step*dist_int_diff-1)))* D; % Differentiate and multiply by 'D' gain

contout = P_part+Int_part+diff_part;

m_ControlSig = -100;
M_ControlSig = 100;

if contout > M_ControlSig
    contout = M_ControlSig
end

if contout < m_ControlSig
    contout = m_ControlSig
end
```

Figure 13 - Feedback Controller

The first step of the code is to **calculate the error** this is calculated via the difference between the desired velocity and the vehicles actual velocity. The code then moves on to **determine past control value** 'dist_int_diff' is used to understand the previous control values. If there are fewer values in 'u_vec' storage, then the current 'control' value is taken as 'prev' or previous.

The next part of the feedback control script is to calculate the gains, The **proportional** term is calculated first by multiplying the error by the **P** gain. This represents the immediate response to current error. The **integral** term is the error accumulated over time and used to minimise the ESS (Error). This is calculated via the average of the current and previous control values and multiplied by the **I** gain and time step 'sim_step'. The **derivative** term is the final calculation which helps to dampen the response. This is calculated via the difference between current and previous control values, divided by the time step 'sim_step' and multiplied by the **D** gain. These gains are then combined to obtain the final control output identified by the script as 'contout'.

The control signal is constrained within the range 'm_ControlSig' and 'M_ControlSig' this prevents the system from exceeding safe values or maximum speed. The function then returns the control output and overall error.

Suitable Gain Selection

Having modelled the cruise control system, it was important to understand how gain changes affect the controller's performance. Tuning the PID gains is a critical part to understand how to get the best out of the controller. The proportional gain provides response to the error signal which improves response, selecting too high a proportional gain will lead to oscillations. The integral gain is used to eliminate steady state error and the derivative gain helps to further dampen a systems response.

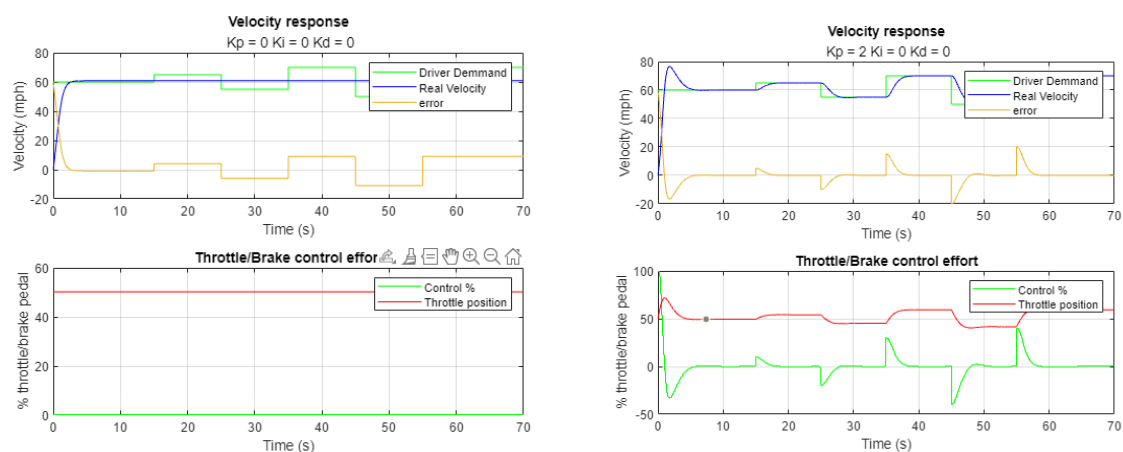


Figure 14 - Modeling Response with Low Proportional Gain

At zero gain the system is not responsive. Error matches the drivers demand as the controller is not matching the speed with cruise control.

At lower proportional gain values, the system has sluggish response to changing velocity. The controller effort is low.

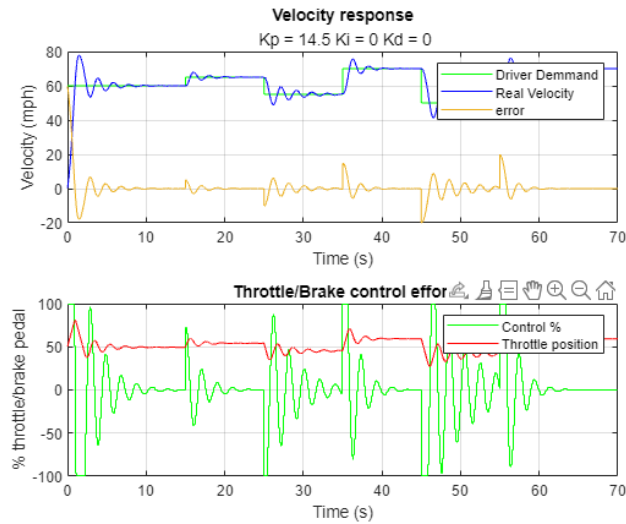


Figure 15 - Modeling Response with High Proportional Gain

With a large proportional gain with no integral or derivative gain, the controller is very unstable and has increased oscillation. The throttle and break response are unstable, and this would lead to a very uncomfortable drive.

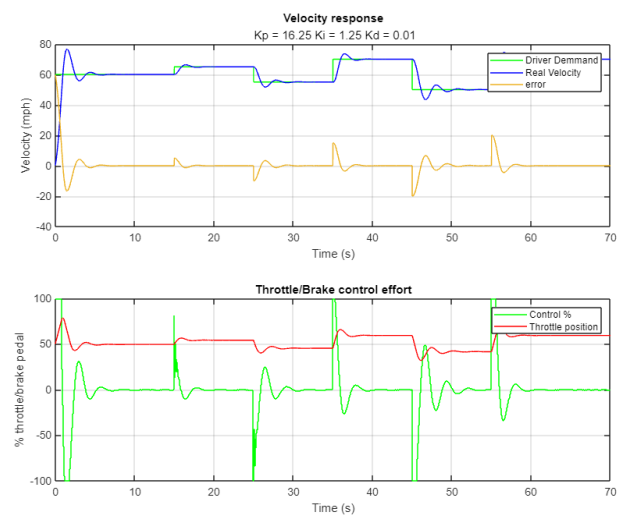


Figure 16 - Increasing P Gain and Introducing I and D

At this point the velocity response is mapping well, however there are some slight oscillations as the cruise controller begins to match the speed, at this point the other began to increase the integral and derivative gain to dampen the response.

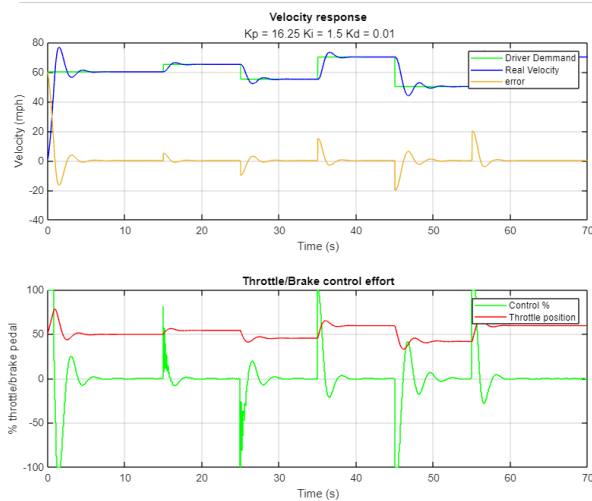


Figure 17 - Finding a Balance for the System

Final Gain figures, $P=16.25$, $I=1.5$ and $D=0.01$. This controller's response is a good indication of progression yet still not perfect. The controller is quickly responding to the demanded change in velocity with minimal oscillation. The throttle response is uniform meaning that the drive is a lot smoother. There is further improvement and as the author has mentioned, it's important that time is spent tuning a PID controller not only in simulation but in the real world.

Trade-off between velocity response and control effort

Velocity response is how quickly and accurately the cruise control system reaches and maintains the demanded speed. Increasing the P gain will lead to a faster response to reach desired speed. A higher P gain leads to faster velocity response whereas a lower P gain leads to a lower response.

Control effort refers to the amount of throttle or braking required in maintaining the demanded velocity. A higher integral gain reduces steady state error but increases the control effort required. In terms of trade off, there needs to be a good balance of both adequate velocity response mixed with a controller effort that is not too demanding.

It's critical that PID tuning is simulated and applied in a real-world scenario to ensure a fast response without sacrificing drivers' comfortability and safety.

Number of Bits and the Affect on Closed Loop Performance

Changing the systems bit rate will affect the accuracy of the signals represented within the system. Changing the bit rate affects the encoding and decoding of signals related to Measured velocity, demanded velocity and overall system control.

There are many trade-offs to consider when changing the bit rate of a control system. Having more bits in a system provides a higher signal accuracy, however, require a higher level of computing resource. Finer adjustments are possible with higher bit rate, but this will reduce data that is represented.

The author experimented with changing bit rates and found that with 8 bits in the system the control effort signal Became unstable as shown in Figure 18. As Fiore (2021) explains, lower resolution leads to nosier signals, With 8 bits there are fewer discrete values that are represented. Increasing the number of bits from 16 to 32 had little affect some explanations may be that 16 bits is already accurate enough. Further testing would be required.

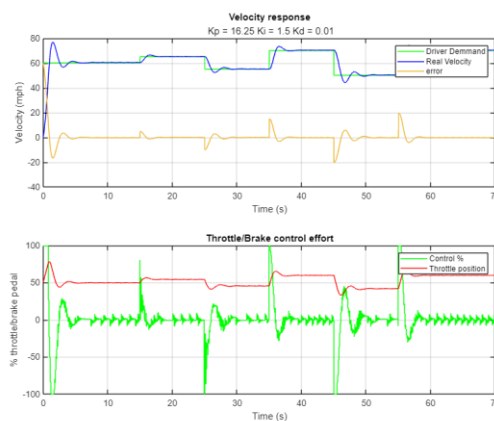


Figure 18 - Decreasing Number of Bits

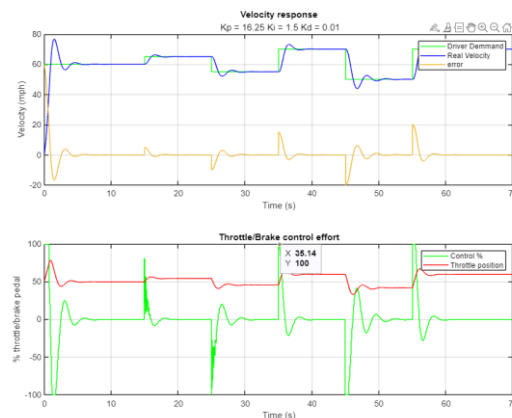


Figure 19 - Increasing Number of Bits

Sampling Time

Sampling time is a critical parameter that determines how often the system is monitored. According to MathWorks (2021), A faster sample time generally improves the systems performance, the bandwidth and robustness as well as the gain and phase margins.

At a higher sample rate, the control system responds faster to changes or driver's demand. At a lower sample rate, the system reduced demand of the computer, but this leads to a slower response which means a poorer performance. A high sample rate can also stabilise the system, this is particular to systems that are constantly changing.

Again, with all the variables it is critical to select a sample time that does not place too much stress on the controller but still allows adequate performance.

List of References

Britannica. (2023) *Instrumentation Technology*. Available at: [Tableware | Britannica](#) (Accessed: 28 August 2023).

Fiore, J. (2023) *Resolution and Sampling Rate*. Available at: [12.3: Resolution and Sampling Rate - Engineering LibreTexts](#) (Accessed: 18 September 2023).

InstTools. (2023) *What is PID Control*. Available at: [What is PID Controller ? - Instrumentation Tools](#) (Accessed: 01 September 2023).

MATHWORKS. (2023) *Three Ways to Speed Up Model Predictive Controllers*. Available at: [Three Ways to Speed Up Model Predictive Controllers White Paper \(mathworks.com\)](#) (Accessed: 27 September 2023).

MATHWORKS. (2023) *What is Sample Time?* Available at: [What Is Sample Time? - MATLAB & Simulink - MathWorks United Kingdom](#) (Accessed: 06 September 2023).

Nice, K. (2021) *How Cruise Control Systems Work*. Available at: [How Cruise Control Systems Work | HowStuffWorks](#) (Accessed: 08 September 2023).

OMEGA. (2023) *What is a PID Controller?* Available at: [PID Controller: Types, What It Is & How It Works | Omega](#) (Accessed: 01 September 2023).

Ragno, M. (2023) *Understanding Open-Loop and Closed-Loop Control Systems*. Available at: [Open-Loop vs Closed-Loop Control Systems: Features, Examples, and Applications — RT Engineering](#) (Accessed: 01 September 2023).

SALUS CONTROLS. (2023) *Temperature Thermostats*. Available at: [Learn about heating control algorithms - SALUS Controls \(salus-controls.eu\)](#) (Accessed: 03 September 2023).

Appendices

Appendix 1 – Process flow for the ATOM Chamber Burn-in process

