
# Development and implementation of an opensource DIC acquisition system, focusing on temporal synchronization with the Gleeble 3800.

Presented by: Maxwell Vos
Supervisor: Dr Sarah George

*Submitted to the Department of Mechanical Engineering at the University of Cape Town in partial fulfilment of the requirements for a Master of Science degree in Mechanical Engineering.*

*2024*

# Student Plagiarism Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this Capstone Report which was taken from the work(s) of other people has been attributed and has been cited and referenced. Any section taken from an internet source has been referenced to that source.
3. This Capstone Report is my own work and is in my own words (except where I have attributed it to others).
4. I have not paid a third party to complete my work on my behalf. My use of artificial intelligence software has been limited to grammar editing.
5. I have not allowed and will not allow anyone to copy my work with the intention of passing it off as his or her own work.
6. I acknowledge that copying someone else's assignment or essay, or part of it, is wrong, and declare that this is my own work.
7. Dishonesty, including plagiarism or the submission by students of other people's work as their own, or the writing of work on behalf of others, in an examination or any other form of assessment will be dealt with in terms of the disciplinary rules and as per Examination and Assessment Committee (E&AC).

Signed by: *Maxwell Vos*

| | |
|---|---|
| Student name and surname: | Maxwell Vos |
| Student number: | VSXMAX001 |
| Date: | 25 May 2024 |
| Signature: | Signed by candidate |

## Abstract

The Gleeble 3800 is a thermomechanical processing simulator with the ability to perform high temperature, high strain rate deformation experiments. The standard Gleeble 3800 uses electrical resistance heating for temperature control and a closed loop hydraulic actuator for displacement control. In order to extract greater accuracy and control of simulated processing parameters, the incorporation of a synchronised Digital Image Correlation (DIC) capability with equipment such as the Gleeble 3800 has become essential.

Current commercial DIC systems offer limited versatility, particularly with regards to integrating DIC and thermal imaging on the Gleeble 3800 in addition to being financially prohibitive for many researchers. Current open-source acquisition solutions lack the specifications required for most DIC applications and custom solutions are overly complex for many DIC users. This research therefore aimed to develop an open-source DIC acquisition system for materials research applications that fulfil these shortcomings. The DIC Capture system addresses these limitations by using high quality hardware that interfaces with user orientated software, thus facilitating complex material testing protocols at a reduced cost.

The DIC Capture system comprises of a hardware data acquisition (DAQ) board, full-field camera system and control computer. The DAQ is controlled by an Arduino Nano ESP32 that sends camera trigger times, voltage values from the ±10V ADS8584S analogue to digital converter (ADC) and sync times with the Gleeble 3800. The camera trigger frequency follows a predetermined sequence and is iterated through a command in the Gleeble Script Language (GSL) program, which controls the Gleeble 3800, thus allowing for event-based frame-rate changes. The Python based software displays and saves images from the DIC cameras, synchronizing all data received from the DAQ board and any other image recording sources.

The DIC Capture system can produce high quality DIC data sets and integrates with the Gleeble 3800 for high temperature DIC applications. The DIC capture system has been successfully tested in three different case studies across a range of testing conditions.

# Acknowledgements

Dr Sarah George for the guidance and support that was provided throughout this project.

A/Prof Thorsten Becker for technical guidance at key stages of the project.

David Dallas for technical assistance with electronic and coding aspects of this project.

James Dicks for the assistance in writing and project structuring.

Penny Louw for the running of the CME labs and keeping a tight ship.

The UCT Mech Eng Workshop staff; Pierre Smith, Grant Springle, Thulani Leike, Wayne Brandt, Gavin Doolings and Thando Mobo. For always going above and beyond, helping in numerous aspects of this project.

Daniel Slater for help with setting up the GUI.

James Hepworth for technical assistance and guidance with electronics.

# Table of Contents

# List of figures

# List of abbreviations

| | |
|---|---|
| ADC | Analogue to Digital Converter |
| API | Application Programming Interface |
| BNC | Universal Asynchronous receiver / transmitter |
| CME | Centre for Materials Engineering |
| COM | Communication |
| CPU | Central Processing Unit |
| DAQ | Data Acquisition |
| DIC | Digital Image Correlation |
| FEA | Finite Element Analysis |
| FPGA | Field-Programmable Gate Array |
| FPS | Frames Per Second |
| GPIO | General-Purpose Input/Output |
| HDD | Hard Disk Drive |
| IDE | Integrated Development Environment |
| IR | Infrared Radiation |
| LED | Light Emitting Diode |
| LPBF | Laser Powder Bed Fusion |
| LQFP | Low-profile Quad Flat Package |
| LVDT | Linear Variable Differential Transformer |
| NAS | Network-Attached Storage |
| NI | National Instruments |
| NTP | Network Time Protocol |
| NUC | Non-Uniform Correction |
| PCB | Printed Circuit Board |
| PSC | Plain Strain Compression |
| RAM | Random Access Memory |
| RGB | Red Green Blue |
| ROI | Region Of Interest |
| RTC | Real Time Clock |
| SSD | Solid State Drive |
| TC | Thermocouple |
| UART | Universal Asynchronous Receiver / Transmitter |
| UCT | University of Cape Town |
| USB | Universal Serial Bus |
| VFM | Virtual Fields Method |
| VHT | Very High Temperature |

# 1 Introduction

## 1.1 Background and context

Digital Image Correlation is a full field, non-contact, displacement measurement technique that is widely used in a variety of materials testing applications. Commercially available DIC systems are often prohibitively expensive for many researchers and are challenging to modify to suit researchers needs due to the proprietary software and hardware required for these DIC systems. By creating an opensource DIC acquisition system, researchers would be able to capture DIC data for a fraction of the current costs, whilst also being able to easily modify the DIC acquisition system to best suit their needs.

At the Centre for Materials Engineering (CME), a research centre within the University of Cape Town (UCT), Digital Image Correlation (DIC) is frequently utilized during the mechanical testing of materials. The three DIC systems currently in use at the CME are Instra 4D, LaVision and most recently a MatchID system. These systems all have unique benefits and drawbacks, some more than others; however, they are all expensive systems.

Integrating DIC with the Gleeble 3800 has always been a challenge at the CME. The versatility of the Gleeble 3800 often leads to it being used for complex material testing protocols; which, in turn, makes recording DIC data for the experiments with existing systems challenging. In the past, researchers[1] at the CME used a National Instruments (NI) Data Acquisition (DAQ) device in conjunction with Dantec's Instra4D software to hardware trigger cameras and send Analogue to Digital Converter (ADC) values from the Gleeble force output to Instra4D. A distinct force feature could then be used to temporally match the DIC images with the Gleeble output data with a custom script. This worked well at the time, but it was limited in versatility due to the reliance on the Instra4D software and required an advanced level of electronics and software competence to be used effectively. One of the major limitations is that there was no live communication with the Gleeble control unit, so acquisition parameters cannot be event dependent. For tests where a combination of high and low frame rates are necessary, being able to control the frame rate via the Gleeble Script Language (GSL) is hugely beneficial.

An increased demand for the integration of thermal imaging with DIC has presented a further technical challenge to testing. Acquiring thermal images that are temporally synced to the Gleeble and DIC data is a challenge, even for basic experiments, and there are currently no elegant solutions. Given the high cost of DIC and thermal cameras, users

[1] Dr Richard Curry and Richard Wittemore

will often be limited to the hardware that they already have or can afford, therefore solutions that allow for integration of additional cameras should follow a generalised approach that can be applicable for all current and future camera configurations.

## 1.2 Aims and objectives

The aim of this work was to create a custom DIC acquisition system that is not reliant on commercial DIC software or NI hardware and software. The objectives of the project were to develop a system that would be:

1. Accessible to as many users as possible by keeping the system simple, and affordable.
2. Achieve a suitable level of accuracy for the majority of DIC tests.
3. Be focused on integrating with the Gleeble 3800, the Telops M350 thermal imaging camera and MatchID, whilst being versatile enough to be integrated with various systems in future.
4. Reliable during testing to ensure data is not lost unnecessarily.
5. Opensource so that users can modify the system to suit their needs and ensure the system is freely available to anybody who wishes to build their own system.

The challenge of this project lies in the integration of many incompatible systems in an elegant, robust and reproducible fashion. This DIC acquisition system is referred to as "DIC Capture" within this report.

Figure 1 shows a simplistic diagram of the major systems that need to be integrated with the DIC Capture system; these systems are the Gleeble 3800, DIC analysis software (MatchID for this project), Thermal Camera, at least two DIC cameras, Analogue to Digital Converter (ADC) and a control computer.

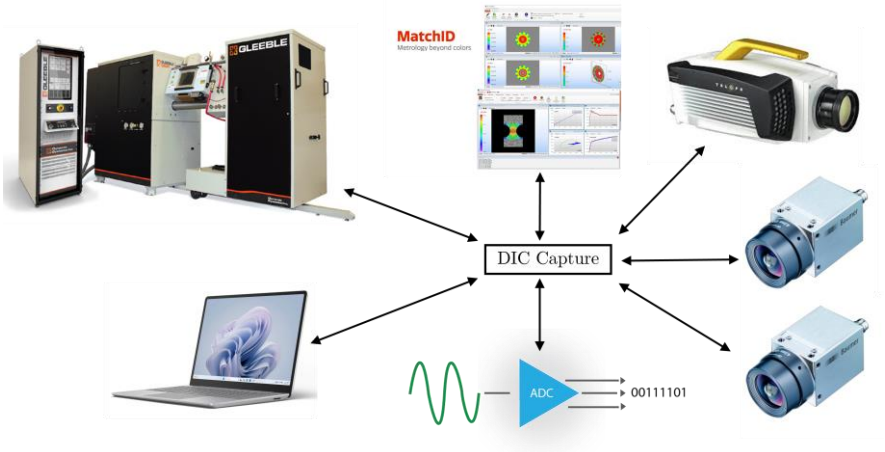



Figure 1: Simplified overview of various systems that are required to interface with DIC Capture

## 1.3 Scope

The project was broken up into multiple stages to ensure steady progress and avoid omitting unforeseen useful features. These stages are shown in Figure 2. The hardware and software that comprise the trigger system are developed to meet the needs of the case studies, which increase in complexity with each iteration. The case studies were chosen to validate each version of the DIC Capture system whilst also producing valuable research data for research projects which focused on materials testing within the CME.

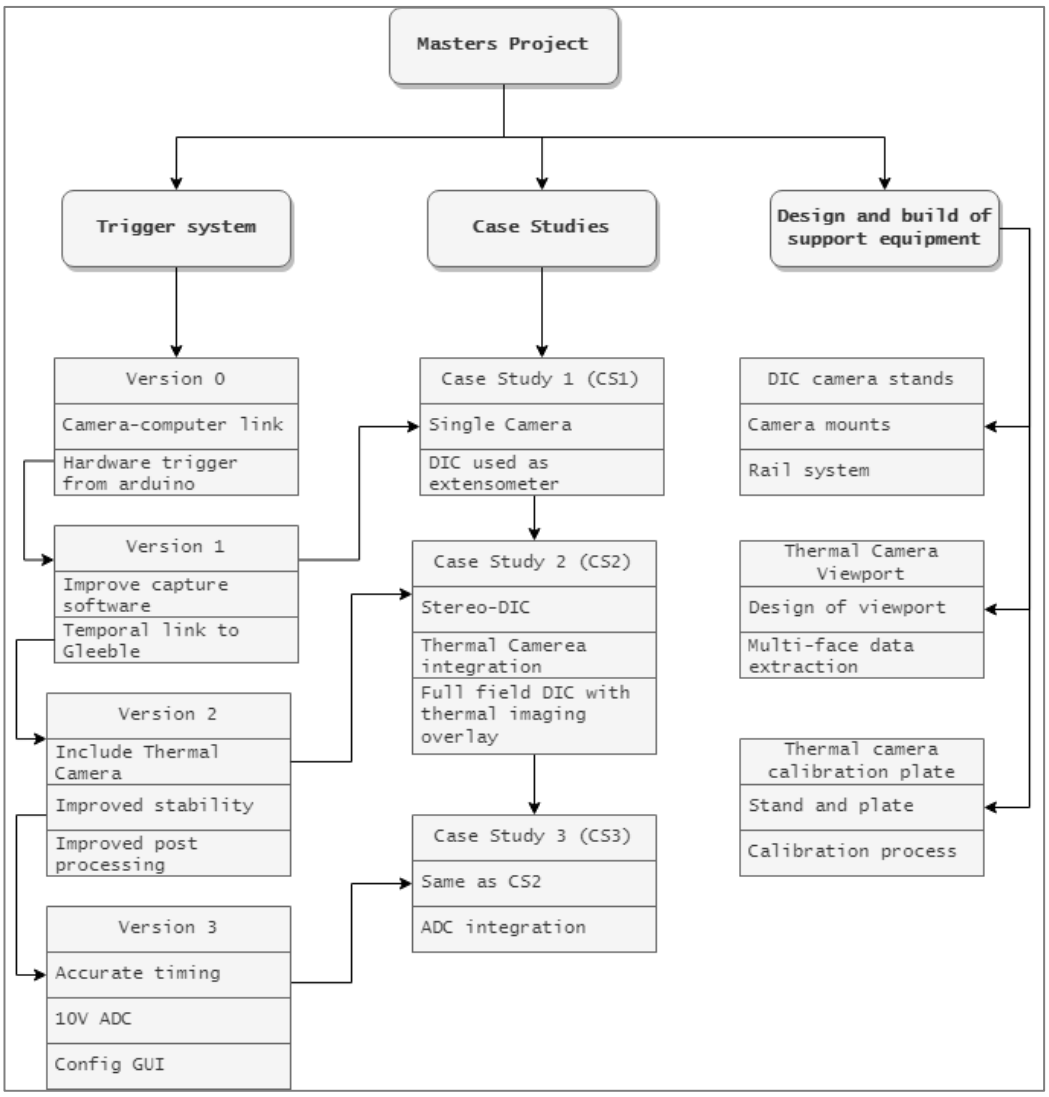



Figure 2: Work breakdown structure for design and testing phase of project.

# 2 Literature review

## 2.1 Overview of DIC

Digital Image Correlation (DIC) is a widely accepted and commonly used tool for quantifying the displacement and deformation of multiple points on a samples surface [1]. For most DIC applications, digital cameras are used to capture a sequence of images of a surface with a randomly speckled pattern applied. A grouping of these speckles act as unique subsets; and the relative movement of each subset, within an image sequence, can be calculated.

DIC is broken up into four main steps: calibration, correlation, displacement transformation and strain calculation. These four steps are described in detail in a paper written by Atkinson and Becker [2]. The quality of the DIC results are highly reliant on the data used in the calibration and correlation steps. The International Digital Image Correlation Society (iDICs) was formed to facilitate the development of DIC standards and best practices for users to follow when generating DIC data, thus increasing the quality of data generated by users who adopt their guidelines [3].

### 2.1.1 Commercial DIC systems

Several companies offer a wide variety of software and/or software packages that provide a streamlined DIC system for the customer and are generally expensive to purchase [4]. A non-comprehensive list of these companies include:

1. MatchID
2. Dantec Dynamics
3. LaVision
4. GOM
5. Imetrum
6. CorreliSTC

MatchID was founded in 2008 by leading experts in DIC and have focused on developing an open and holistic DIC-platform where DIC, FEA and VFM tools are combined to create a powerful software platform for users [5]. MatchID does not use proprietary file formats when using their grabber system; enabling researchers the freedom to use existing or experimental hardware to capture images and DAQ data that can be imported into the MatchID platform.

Dantec Dynamics focuses on turnkey advanced measurement systems for fluid dynamics and solid mechanics applications. Dantec has the Instra4D software package for 2D and 3D DIC that can be configured with various application specific upgrade modules [6].

LaVision offer various measurement solutions, with a focus on fluid mechanics, combustion, and particle diagnostics. The StrainMaster systems are turnkey imaging solutions for materials testing applications which integrate with the DaVis image analysis software. The LaVision system makes use of a double level calibration plate that is capable of spatially calibrating a stereo camera system with a single image from each camera [7]. Being able to calibrate with a single image greatly simplifies the calibration process when working in confined spaces. High temperature testing in the Gleeble 3800 is one such use case. In a paper by van Rooyen and Becker, the double level calibration plate is effectively used for calibration of the LaVision DIC system where the space constraints of the Gleeble vacuum chamber make multi-image calibration challenging and tedious [8].

GOM specialises in material independent measurement systems with high performance integrated hardware and a focus on industrial environments. The ARAMIS 3D camera system can be used with the Zeiss Inspect Correlate, formerly known as GOM Correlate Pro, for analysis of movement data [9].

Imetrum offers 3D measurement systems that have fixed camera systems and integrated hardware. The Mobius is one of their products used for DIC [10]. The Mobius system is orientated towards industrial applications and offers minimal configuration options.

CorreliSTC is a DIC analysis software package that can import images in various formats. There is a large focus on the aeronautics sector with one of their partners being the Airbus Group [11].

### 2.1.2 Opensource DIC systems and tools

There are a wide range of opensource DIC software tools that have been published and continue to be developed. However, there is a lack of opensource acquisition systems that can be used to capture high quality data for DIC. The reason for this is unclear. Some papers mention that the task of acquiring images is of less research relevance [4], which may be the case for the most basic of DIC experiments but becomes more complex when using DIC as a research tool. A Raspberry Pi based imaging system developed by Miikki *et al.* [12] is a good candidate for a system with a constrained budget; however, the lack of a monochrome sensor, hardware triggering, synchronisation, and an ADC limit the system from being useful in most DIC applications.

iCorrVision-2D and iCorrVision-3D [13] offers a GUI software grabber, calibration and correlation module. The grabber module connects to Basler cameras through the pypylon library with a software trigger. The calibration and correlation module offer functionality that would meet the requirements of most DIC users. Overall, the iCorrVision is an impressive tool that can be modified by experienced users as the code is openly available on the creators GitHub repository [14]. Due to the iCorrVision acquisition system being software based; there is no integrated DAQ device to read additional signals such as force data from a load cell and no hardware triggering or hardware I/O connections, which is inconvenient in many situations. These inconveniences can be resolved on a case-by-case situation depending on the users needs, but this would rely heavily on the user's technical competency.

Open-source DIC analysis software tools have been created by various independent developers. Whilst these software tools may not be as polished as the commercial systems, they are powerful tools that can be used in many applications. Some of these software tools are listed below:

1. Py2DIC
2. Ncorr
3. iCorrVision-2D and iCorrVision-3D
4. DICe
5. MultiDIC
6. DuoDIC
7. µDIC

Py2DIC is an open-source Python-based toolkit with a basic input GUI, Py2DIC has been validated against the commercial VIC-2D and opensource Ncorr and DICe software and was found to produce consistent results [15].

Ncorr offers a user friendly GUI interface for the input of data and visualisation of results; the GUI runs in the MATLAB program with many of the analysis algorithms being written in C++ [16].

iCorrVision-2D and iCorrVision-3D, previously mentioned in the context of their grabber software, both offer a user-friendly GUI for the analysis and visualization of DIC data. At the time of writing iCorrVision-3D and DICe [17] were the only non-MATLAB based open-source DIC software capable of stereo DIC through a GUI interface. MATLAB stereo DIC tools such as MultiDIC [18] and DuoDIC [19] are useful tools for users with access to a MATLAB licence.

µDIC is a Python based toolkit with no GUI designed to offer simplified DIC analysis tools for users with limited C++ programming experience [20].

### 2.1.3 Integration of Thermal imaging into DIC

DIC has proved to be an effective strain measurement technique for high temperature testing, primarily due to its robust and contactless attributes. Alternatives strain measurement techniques such as strain gauges, optical fibre sensors and clip gauges offer inferior performance in high temperature applications as compared to DIC [21]. When performing high temperature DIC without thermal imaging, a uniform or assumed temperature profile is used to calculate temperature dependant properties. The integration of thermal imaging into DIC systems offer many advantages to high temperature DIC applications and have therefore recently been added as features in leading commercial DIC systems. MatchID [5] and Correlated Solutions [22] have synchronized DIC and thermal imaging systems with spatial calibration techniques in their calibration software packages.

## 2.2 Overview of the Gleeble 3800

The Gleeble 3800 is a versatile thermal-mechanical physical simulation system that is widely used in academic and industrial research applications [23]. High temperature mechanical testing is a common application for the Gleeble 3800, with oxidization mitigation by testing within the vacuum chamber. High heating rates are possible through the ohmic heating system and medium heating rates with the optional induction heating system. Both tensile and compression type tests are possible with various test specific modules.

The Gleeble 3800 has large viewing windows on the front and rear of the vacuum chamber providing a clear view of the sample when using DIC at elevated temperatures. DSI, Gleeble's parent company, offer a customised camera system and licenced software [24]. The Gleeble DIC system is essentially a stand-alone commercial DIC system that receives start/end triggers and a single configurable ±10V analogue signal from the Gleeble control consol.

## 2.3 Relevant electronics theory

### 2.3.1 ADC

An Analogue to Digital Converter (ADC) is used to measure an input voltage and produce a digital output that is proportional to the input voltage and is an integral

component in a data acquisition system. Sensors such as load cells, displacement transducers, temperature probes etc., send signals to purpose-specific signal conditioning electronics that amplify and filter the signal into a voltage which is converted to a digital signal by an ADC [25].

Some critical properties of an ADC are its range, resolution, and speed. The range of an ADC refers to the difference between the maximum and minimum input voltages; for most lab and industrial devices, a ±10V range is standard. The resolution refers to the precision with which an ADC can convert a voltage into a unique binary value. A shorthand of "N-bit resolution" is commonly used with ADC's and can be converted using equation 1.

$$Resolution = \frac{V_{input\ range}}{2^n} \tag{1}$$

For example; a 16-bit, ±10V ADC will have a resolution of :

$$Resolution = \frac{10V - (-10V)}{2^{16}} = \frac{20}{65536} = 0.3052mV \tag{2}$$

In practice, the resolution of the ADC is seldom the limiting factor for the accuracy of the ADC as the noise floor of the input voltage is usually greater than the resolution of the ADC.

### 2.3.2 Relevant connections and transfer protocols

Universal Serial Port (USB) is a common camera interface for machine vision applications due to it ubiquity and high transfer rates [26]. USB 3.1 Gen 1, formerly USB 3.0, was the fastest commonly available USB standard used on machine vision cameras at the time of writing. Power delivery over USB means that most USB cameras do not require an additional power supply. If higher transfer speeds are required for specialized applications: 10 GigE (PoE) interface is capable of 10Gb/s over longer distances. High speed cameras with large buffers are usually a more practical option for DIC applications if the USB 3.0 transfer speed is a bottleneck, as DIC tests generally happen over a limited time frame [27].

Universal Asynchronous Receiver-Transmitter (UART) is a serial communication protocol commonly used for communication between microcontrollers and computers [28]. Both devices must be initialized with the same transfer speed, referred to as baud rate, which is defined in bits per second. A common baud rate for Arduino's is 115200

bit/s which is significantly slower than most other data transfer rates and is generally only used for transferring text data.

Analogue voltage via Bayonet Neill-Concelman (BNC) connection is a general purpose connection type that has been adopted in various industries and is ideal for when a high quality analogue connection with a robust connector is required [29]. The BNC connector is widely used in the materials testing environment where a sensors full scale reading is typically converted to a ±10V signal that is outputted via an BNC connector and measured by the ADC on a DAQ. BNC connections are also commonly used to transfer synchronisation and trigger pulses in applications where timing is important.

# 3 Project Development

As shown previously in Figure 2, this project used an incremental development process which was broadly categorized into Version 0, 1, 2 and 3. Version 0 was used to test and develop the individual components of the DIC capture system separately before attempting to make one cohesive system. Version 1 combined the components from Version 0 into a prototype system that was applied in the data capturing for Case Study 1. Version 2 improved on some reliability issues in Version 1, used two DIC cameras, a thermal camera and was applied in Case Study 2. Version 3 had significant improvements made to the control board, a basic GUI was added, thermal camera integration improvements, ADC implementation and was validated through its application in Case Study 3. Version 3 is considered to be the final version of the DIC Capture system for this work. The versions are described in detail in this section.

## 3.1 Version 0

Version 0 topics include:

1. Camera-Computer link.
2. Timing and hardware triggering.
3. Saving Camera and Arduino data.
4. Gleeble related timing.
5. Frame rate changes.
6. Live display of camera images.

The image saving algorithm, hardware triggering and timing were improved for Version 3 after being used in Version 1 and 2.

### 3.1.1 Linking cameras to a computer

For most DIC applications, industrial machine vision cameras are used. A pair of Baumer VCXU-50M cameras were used in this work. Baumer offers an Application Programming Interface (API) called ‘neoAPI’ that can be used to control the cameras via a Python script interface. Most industrial camera manufacturers offer similar API’s with their cameras. The VCXU-50M is primarily limited in frame rate by the USB 3.0 data transfer speed of 5Gb/s, thus ensuring that the system will be capable of receiving data from at least two cameras at USB 3.0 speeds. Faster connections, such as 10GigE, are possible at the time of writing but add unnecessary cost and complexity. For the purposes of DIC, it is preferable to achieve higher frame rates by having a larger image buffer on the camera rather than a faster connection as real time analysis of the images is not a focus of this project. USB 3.0 strikes a balance of affordability, ubiquity and

expandability, which made it a suitable choice for the connection standard for this project.

The VCXU-50M have two interface ports. The USB 3.0 connection for transferring image and camera data to a computer whilst also providing power to the camera. The general-purpose input/output (GPIO) ports are generally used for hardware triggering of the camera.

### 3.1.2 Image Triggering

Most industrial cameras can either be software or hardware triggered. A software trigger is susceptible to interruptions and delays from the control computer. Hardware triggers offer accurate trigger timings when using appropriate hardware and near instantaneous synchronization between multiple cameras. Version 1 and 2 used a timing register on the Arduino Uno to provide hardware interrupts that switched the state of the camera trigger pin every x milliseconds. Using the timer interrupts on the Arduino, instead of any built in delay() function isolates any time drift to the drift of the quartz crystal oscillator and is independent of code that may be running on the Arduino. The drift of this oscillator can be compensated for partially; however, there is a temperature dependent factor which contributes to timing errors over longer periods. For Version 1 and 2, the time between trigger signals from the Gleeble is used to correct the Arduino time which can then be used to sync trigger times with the Gleeble output data, explained further in the Temporal synchronisation section of this document.

As a precautionary measure, instead of applying a voltage directly to the frame trigger pins on the camera (pin 1/8 shown in Figure 3), an optocoupler was used to short pin 1 with pin 7. Shorting pin 1 and 7 took the Field-Programmable Gate Array (FPGA) input voltage from the onboard 3.3V to ground which registered as a falling edge. This worked well to isolate the cameras from the experimental control board and potential grounding issues. However, this approach resulted in a rising edge from the control board being interpreted as a falling edge from the camera. In future iterations of the design, this should be amended. Worth noting is that the trigger input for the Telops M350 thermal camera requires a 5V step signal and does not work with the opto-coupler shorting technique; so, there is a voltage trigger source on all versions of the board that could be split to multiple cameras should the user requires this.

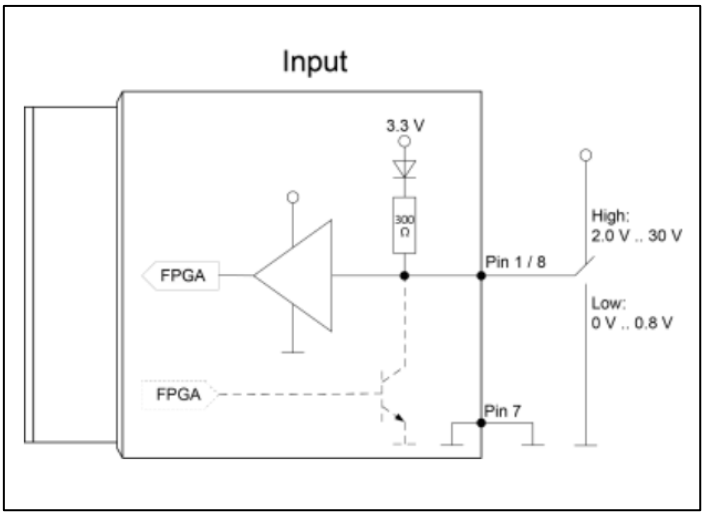


Figure 3: Wiring diagram of triggering circuit for the Baumer VCXU-50M [30]

### 3.1.3 Start, end and frame rate changes.

It is common for there to be too few controllable signals from materials testing equipment that can be modified for custom applications or external system integration. On the Gleeble 3800, it was found that the Quench 3 and 4 outputs could be used as a common signal. Quench 3 and 4 activates the quench module without turning the vacuum pumps off and is seldomly instead of Quench 1 and 2, which disable the vacuum pumps during a quench. By setting Quench 4 to 'on' in the GSL program, 24V is supplied to the quench module solenoid and there is a binary change in the Gleeble output file for the Quench 4 data stream. By stepping the 24V signal down to 5V, a change in the Quench 4 state can be used to iterate through a pre-determined sequence of frame rates on the DIC control board. This frame rate sequence is sent to the Arduino via a serial link to the control computer at the start of the Gleeble test and compiled into an array. An example of a frame rate array would be [0,50,2,0] for a test where a high frame rate is needed for a fast event and a lower frame rate captures slower effects after the initial event. The leading and trailing zero's put the system into a standby state so that the first frame is synchronized with the first Quench 4 signal from the Gleeble, and the last Quench 4 stops the frame acquisition. A delay of 30ms (see section 4.2.3 for details on this delay) is included after each frame rate change to eliminate the chance of two triggers being sent to the cameras at a higher frequency than the cameras are capable of capturing, which would cause a frame to be dropped.

### 3.1.4 Saving image data

The Baumer VCXU-50M has an internal image buffer of 445MB which stores approximately 20 images, if the capture rate exceeds the transfer rate to the computer, the internal buffer will start to fill up and eventually frames will be dropped if the buffer is exceeded. When the camera receives a hardware trigger, the resulting image is stored in the internal buffer queue, the computer then sends a pull request to the camera, the oldest image in the internal buffer is sent to the computer via the USB connection and removed from the camera buffer once the computer has received the image. The timestamp of the image is therefore determined by the time of the hardware trigger, not the transfer or save time of the image.

The next step is saving the transferred image to the computer without bottlenecking the transfer speed of the camera. With data transfer rates of up to 4Gb/s for each camera, at high frame rates it is possible that a frame can be dropped if the computer's storage device is momentarily overloaded. Writing the images directly to a RAM buffer can help prevent this but one must be careful not to exceed the RAM capacity of the computer whilst doing so. For Version 1 and 2; the saving algorithm used a batch cycling RAM buffer, which is set by the user at the start of the test. By setting the RAM buffer to five images for example, the control computer would receive the first five images from a camera and store them to RAM whilst displaying only the first image of that batch, when the sixth image is received, a second RAM buffer is created to which the next five images are transferred to whilst the images from the first RAM buffer are saved to the control computers storage. The cycle would then repeat, switching between the two RAM buffers for the duration of the recording. By displaying only the first image of the RAM buffer, processing and displaying the images does not bottleneck the system during high frame rate sections.

This batch cycling RAM buffer technique minimized RAM usage; however, required fast data storage speeds to be reliable. The saving method was therefore changed for Version 3, see *Redesign of the image saving* algorithm section, but worked well for Versions 1 and 2.

The images from each camera are saved in their respective folders with a naming convention which best suites the projects requirements. All image names must contain a trigger count in their name to keep them in order and match frames across different systems. With Case Study 2 for example, *testID_FrameID_ CameraNumber* naming convention was used.

### 3.1.5 Saving timestamp and ADC values

A serial communication is established between the control computer and the Arduino, when the Arduino triggers the cameras, it records frame ID and the local time for the trigger, which is then sent via serial to the control computer. The control computer then appends this data to a tab delimited text file which was set up at the start of the test. In Version 3, the voltage values of the ADC are included in this communication with the ability to record ADC values at a higher sampling rate than the camera frame rate. This is further described in *Chapter 4.3.1.4 .*

Using the metadata of the incoming DIC images, an additional text file is created for each camera which records the FrameID and the Baumer cameras internal timestamp. These camera text files are compared with the Arduino text file to verify that all frames captured correspond to the correct trigger sent from the Arduino. If a frame is dropped, the exact dropped FrameID can be identified from the discrepancies between the camera and Arduino timing files. The user can compensate for the dropped frame during the analysis of the data. This error correction ability is extremely useful and many commercial DIC acquisition systems, such as MatchID, do not have systems in place to correct a test with dropped frames which means that the test results have to be thrown out. Ideally, a frame should never be dropped; but this is easier said than done, even for commercial DIC acquisition systems.

### 3.1.6 Temporal synchronisation

With the quench 4 output already being used to change camera frame rates, this offered a convenient means to temporally match the timing data from the Arduino and the Gleeble output data. The Gleeble output data contains the state of quench 4 as either a 1 or 0 (on or off) and the Arduino records its internal time each time the voltage from quench 4 changes state, which is recorded in the fifth column (“Last_QTime”) of the Arduino serial output file as time in milliseconds since the first quench 4 state change.

A Python script is then used to match each systems local time with a common number. Either system can then be chosen as the reference system. The Arduino frame trigger times are translated and scaled to synchronise the image capture times with the Gleeble output data.

Given that there should be minimal temperature fluctuations of the crystal oscillator on the Arduino during these short tests, this should have worked well for the first prototype, but it was observed that there was relative timing difference of 0.1%. This error is far larger than the expected error of 0.001% from a crystal oscillator but is within the 0.5%

error of a typical lead zirconium titanate ceramic resonator [31]. This can be attributed to instances where Arduino Uno use a resonator instead of the usual crystal oscillator as a clock, which explains the large clock drift. Whilst it is possible to correct for this temporal drift by using the Gleeble clock as the master clock, having all timing components in the system as accurate as possible minimises any error that may occur when syncing the Arduino board time with the Gleeble Time.

For Version 3, the Arduino Nano ESP32 is equipped with a crystal oscillator that provides accurate time data over the expected durations of most Gleeble tests. The temporal synchronization algorithm modification for Version 3 is described in *Chapter 4.3.1.6*

### 3.1.7 Live display of camera feed

Good DIC results require a well set up camera system as errors such as overexposure, motion blur or poor focus can invalidate an experiment. To reduce the probability of these errors occurring, tools can be used to aid the user when setting up the camera system.

Converting the monochrome image from the cameras to a heatmap colour image provides a more intuitive view for gauging the exposure levels of the image. It is often best practice to slightly underexpose your image as it reduced the likelihood of overexposure causing a loss of data, these underexposed images are often interpreted as too dark by new DIC users, and they compensate by increasing the lighting or exposure time to create a visually pleasing image which results in overexposed images. The heatmap colour image more clearly shows the details within darker images and overexposed pixels are easier to identify. Figure 4 shows the heatmap images for context.

The incoming images are usually of higher resolution than the computer monitor that would display them, so the images are scaled down to fit. As the images are scaled down, the average value of multiple pixels is used to create the single display pixel. In the case where one of these original pixels is overexposed whilst the others are not; the new averaged pixel will not be overexposed which could be misleading to the user. By adding a zoom function into the live display, the user can select their Region Of Interest (ROI) by clicking and dragging, the ROI is displayed at a 1:1 resolution with a histogram showing the distribution of pixel values within the ROI; see Figure 4 where the full field, zoom and histogram view is shown for each camera. A thick red bar on the right-hand side of the histogram is displayed if there are any overexposed pixels in the ROI. Over exposed pixels have a much higher weighting in the histogram count, which makes it

easier for the user to identify when only a few pixels are overexposed (see ‘Hist. 1’ on Figure 4).

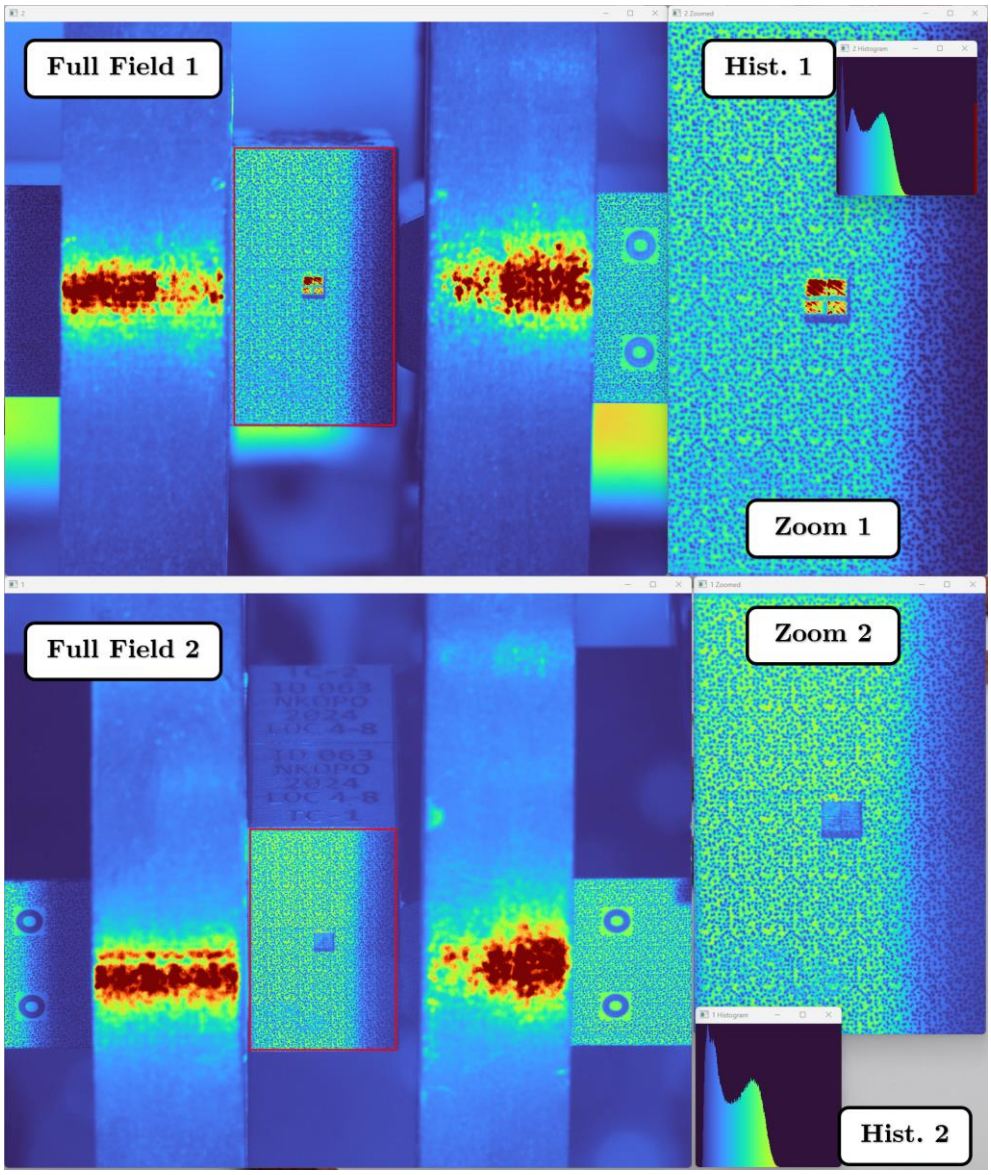



Figure 4: Display of CAM 1 and 2 with Zoom and Histogram windows for each.

Once the camera setup is deemed acceptable, the live display is closed and the ‘record mode’ is set to true. In Version 3, the run record mode button is clicked on the GUI. When the cameras are triggered in this mode, all incoming images and data is recorded to the test_ID folder entered by the user at the beginning of the test. There are currently no means to adjust the camera parameters during the test, as this risks unexpected

software crashes. One exception to this is the lighting, the user can adjust the lighting on longer duration tests if the ROI has an undesirable exposure level.

# 4 Detailed discussion of Versions 1-3

## 4.1 Version 1

### 4.1.1 Version 1 Design and build

For stereo DIC, it is common for two cameras to be mounted to a single bar to keep the relative position of the two cameras constant throughout the duration of the test. This bar is attached to a tripod for easy positioning and alignment of the camera system. Dantec, LaVision and MatchID all have their own systems which can be costly and often have limitations. For example, LaVisions system mounts the cameras, lights and control board all to one bar. Having everything on one bar increases the size of the system making it challenging to fit into small spaces or work around.

The custom camera stand was designed to be made from standard components and access to an angle grinder and some basic hand tools. The first design made use of components that were readily available. Therse were: a length of HGR20 linear rail and three Manfrotto 496 ball head mounts. Inexpensive ungraded HGR20 linear rails, at approximately $30/metre at the time of writing, have hardened surfaces and can be arranged to create a dovetail feature. By using two short sections of HGR20 rail bolted to a plate, the plate can slide along a longer section of rail and conveniently be locked in place by tightening a single bolt. Such a configuration is shown in Figure 5, where the base plate that is bolted to the short section of rail is laser cut from 6mm steel and holds up to two Manfrotto 496 ball head mounts. There is a smooth sliding action when adjusting the position of the cameras on the bar, which is unlikely to degrade with time due to the mating surfaces being hardened steel.

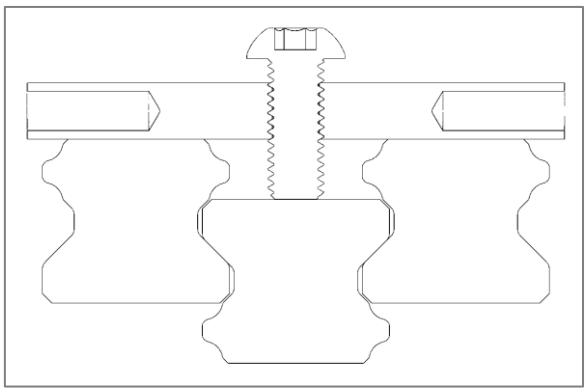


Figure 5: Illustration of dovetail feature on HGR20 linear rail

The HGR20 rail comes with 6mm diameter holes spaced 60mm apart along the length of the rails. The Manfrotto 496 mounts have a ¼”-20 standard camera stud which can be unscrewed after removing the top rubber layer. The ¼”-20 stud has a M6 thread on the side that screws into the ball joint. A single M6 button head bolt is then used to secure the longer length of HGR20 linear rail to the ball joint. The rubber layer provides enough friction to prevent unscrewing of the bolt and the whole system can be easily moved along the tripod in 60mm increments. Having the whole rail being supported by one ball joint makes it possible to place the two cameras in nearly any orientation, most commonly: vertical or horizontal. A dual camera setup in the horizontal configuration with Version 1 of the DIC Capture board on the right is shown in Figure 6. For many Gleeble tests, the ability to position the cameras on the vertical plane rather than the horizontal plane, is extremely advantageous.

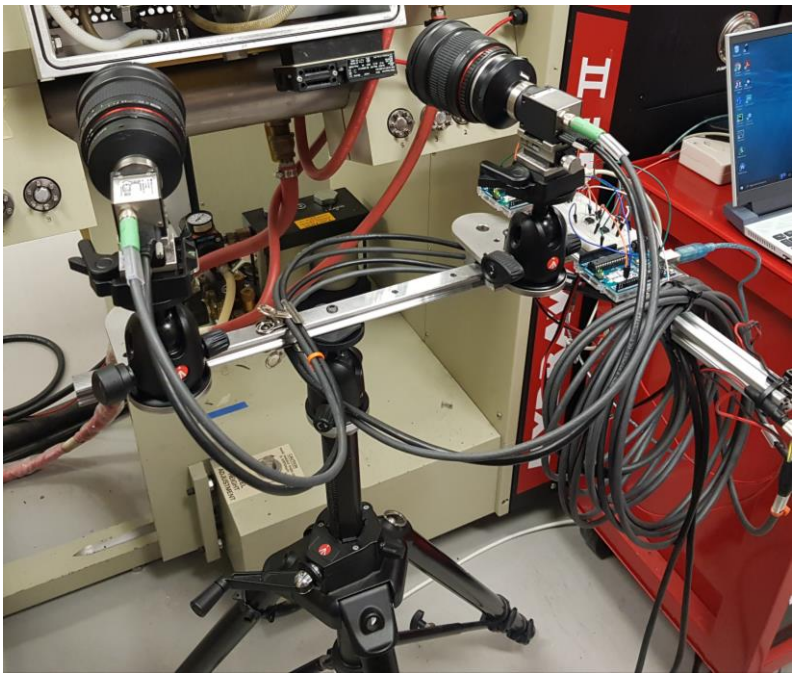


Figure 6: Horizontal configuration of camera rail

By mounting the light to the Gleeble door (see Figure 27) instead of the camera rail, powerful lighting that required a fan for cooling could be used without worry that the fan vibrations could affect image quality. Magnetic indicator arms worked well for positioning lights for most experiments. The Arduino system is placed next to the control computer for easy user access with 3m long USB and trigger cables being used to link the cameras to the system. This camera mounting system worked as intended so it was left as is with no improvements to later versions.

### 4.1.2 Case study 1

As part of the experimental work performed for the MSc project by Borrangeiro [32], plain strain compression (PSC) on the Gleeble 3800 was required. The objective of the testing was to monitor creep strain after compression at temperatures during an isothermal hold under constant pressure. The test protocol for the Gleeble consisted of a heat-up and soak, then a deformation where the strain rate was varied across tests, the sample was then held under constant load for 15 minutes. Obtaining accurate strain values during the constant load portion of the test is challenging. The Jaw reading from the Gleeble uses an Linearly Variable Differential Transformer (LVDT) to accurately measure the displacement between its two mounting locations as seen in Figure 7. The Jaw measurement was the most accurate deformation measurement technique available on the Gleeble 3800 for this test type.

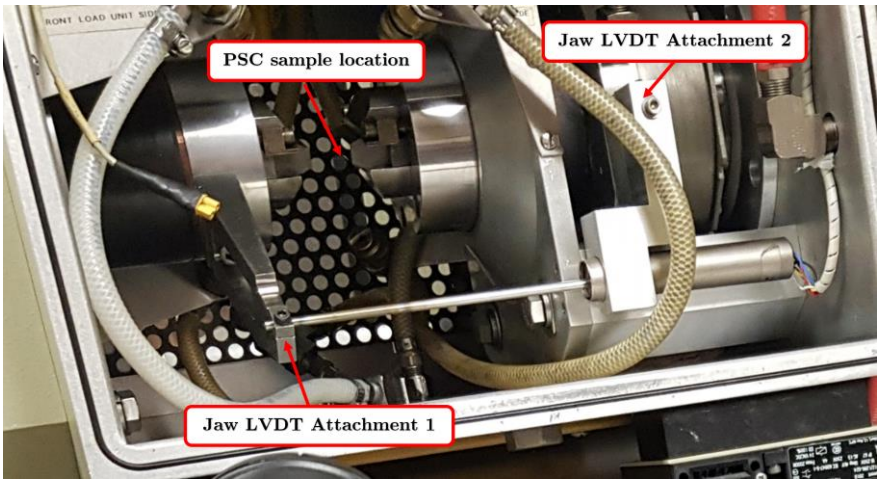


Figure 7: LVDT mounting position relative to PSC sample

The Jaw measurement is affected significantly by thermal expansion of the components between the two LVDT attachment points, which causes complications when trying to accurately measure the deformation experienced by just the PSC sample.

To achieve a more accurate measurement of the relative anvil displacement during the creep phase of the tests, the PSC anvils were speckled for DIC. A single camera was positioned perpendicular to the anvil face plane, centred on the final centre of the deformed sample. The camera was approximately 450 mm away from the sample. A single camera was used for this test as the DIC Capture system was only barely functional at this point in the development phase and having a single camera setup simplified calibration, troubleshooting and postprocessing significantly. Only one camera was necessary as images captured for this test, as negligible out of plane motion was expected due to the constrained motion of the anvils. The field of view of the camera in this setup can be seen in Figure 8 with the approximate locations of the two facets used to calculate the DIC extension shown in red.

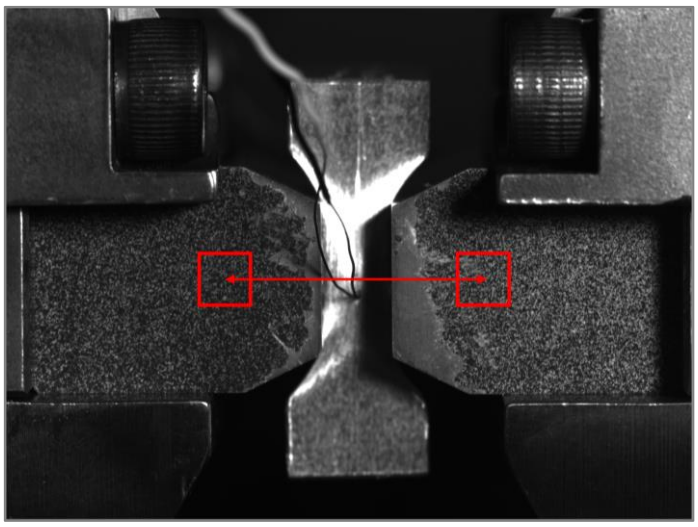


Figure 8: Camera view during creep phase of case study 1 testing.

The DIC extensometer provided a much more reliable and accurate measurement of the sample deformation for Borrageiro's results. The DIC Capture system Version 1 successfully captured data for 25 of these tests. Figure 9 shows how that Jaw and DIC extensometer differ during the heat and soak stage of the test and demonstrate the benefit of the DIC extensometer, note the frame rate change after the initial soak. Experiments revealed the necessity of an initial embedding of the PSC sample onto the anvils. This was achieved by compressing the PSC sample slightly during the initial heating, resulting in a uniform heat distribution during the first 170 seconds of the test, after which the sample is held at a constant temperature before the deformation.

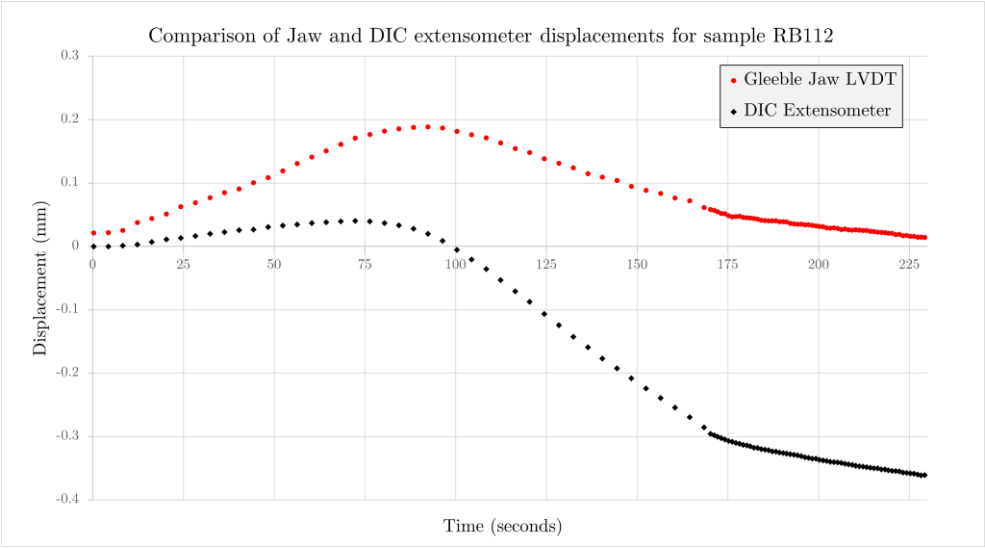



Figure 9: Comparison of LVDT and DIC extensometer measurements.

Case study 1 was considered a success and Version 2 was started shortly after testing was completed.

### 4.1.3 Problems identified during Version 1

The FPS sequence for the test was as follows: [0,0.25,1,32,1,0.2,0]. A low FPS was used during the heat up and hold phase of the test. 32 FPS was used for the fast deformation section of the test and a slower FPS of 1 and eventually 0.2 FPS was used for the creep section of the test. The images captured during the heat up and hold phase were primarily used for trouble shooting and checking for any pre-deformation creep.

The higher FPS during the deformation was used to determine the accuracy of the quench trigger mechanism. The data was first synchronised using the quench sync pulse. Fitting a line to the jaw and extensometer data during a rapid deformation, the temporal offset between the lines was calculated. There appeared to be a delay of approximately 200 ms when syncing using the quench signal. It was unclear at the time of testing what the primary cause of this delay was. For the long duration of the creep section of these tests, a 200 ms offset did not affect the results. Given that the use of the quench signal for timing purposes is outside the intended use of the Gleeble, there was no useful information on the topic.

An initial theory for the trigger delay was that the quench 4 output ramped between states. To measure this, an oscilloscope was connected to the quench 4 output of the Gleeble 3800. The oscilloscope showed that the Quench 4 voltage switching could be considered instantaneous on the timescales used for this system (see Figure 10) with ramp times in the low micro-seconds.

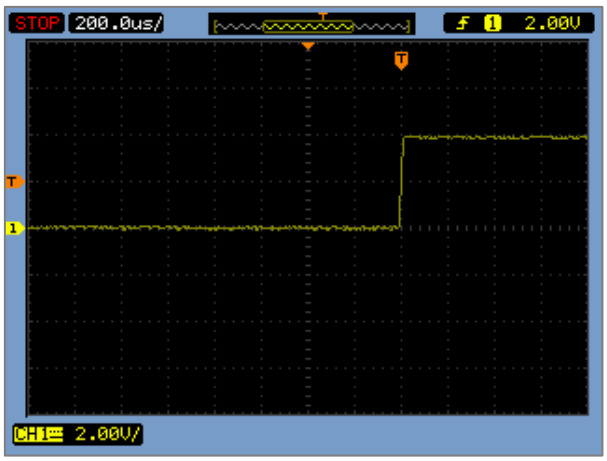



Figure 10: Measurement of Quench4 signal using an oscilloscope.

When viewing the quench data output from the Gleeble, the ramp time between state changes was consistently 100 ms, irrespective of sampling frequency, as seen in Figure 11. The hypothesis for this is that a 100 ms running average filter is applied to the quench data as it is saved. Ideally gaining access to the code used to filter the incoming data would reveal a more concrete solution. Assuming the use of a running average, one can calculate the exact trigger time in the Gleeble data by calculating a line of best fit for the quench values between 0 and 1 that is used to determine the intercept with 0 for a rising edge or the intercept with 1 for a falling edge. The time value at the intercept represents the moment the quench signal was observed in the Gleeble data and should correspond to the voltage change used to iterate the frame rates on the DIC acquisition system.

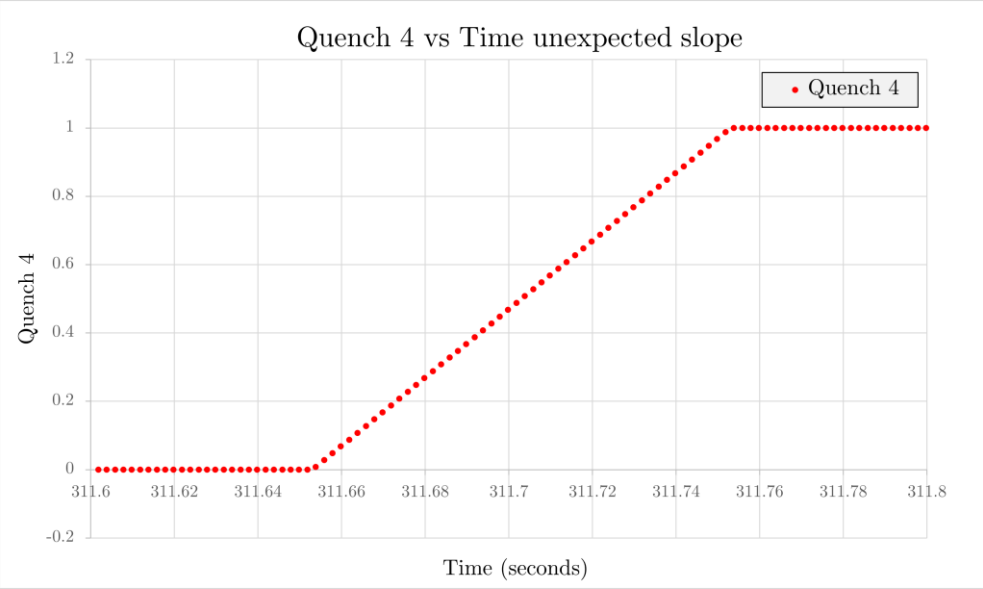



Figure 11: Gleeble output data of a Quench 4 state change showing a 100ms ramp time

An intermittent error was identified whereby a test would fail due to an extra trigger from the quench 4 signal being received. During Version 1 of this work, the triggers sent during cooling activation and safety mechanisms were not known but it was found that the noise from the quench 4 signal was significant. To mitigate this, an interim solution was to ground the Arduino to the Gleeble controller which reduced the noise amplitude by an order of magnitude. Another fix implemented was to change the quench 4 trigger condition from being monitored by a digital input pin to an analogue pin with the tighter thresholding for a change in state to be registered. These two approaches improved the reliability of the system, with the other sources of false quench 4 signals being fixed in Version 2 of the system.

## 4.2 Version 2

### 4.2.1 Version 2 Design and build

During a stereo calibration for Blackwells testing [33], the chamber door must be opened and closed for each position of the calibration plate to account for the effects of the glass between the cameras and the sample. The makeshift solution is a large piece of Prestik to hold the calibration plate. The repeatability and accuracy of this manual step was poor. Further to this, this manual approach was made more difficult because it was necessary to position the plate such that the reverse side of the plate was required to be visible in order to spatially calibrate the thermal camera with the DIC cameras, due to the placement of the thermal camera on the rear side of the sample. Figure 12 shows the relative position of the cameras.

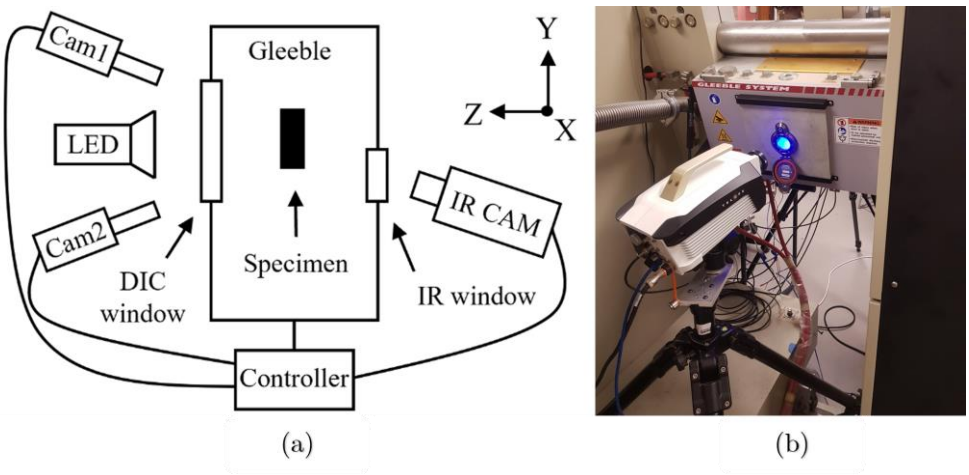



Figure 12: (a) Diagram of camera positions relative to sample [33]. (b) Thermal camera positioned to view rear of sample through Fluke viewport

Figure 13 shows how the problem of positioning of the calibration plate was addressed. A mirror image of the MatchID 12x9x2.5mm calibration plate was machined from 3mm thick aluminium. The dots were holes drilled with a 1mm drill bit and the three fiducial markers were made by adhering a thin wire in the centre of the hole with cyanoacrylate glue. The surface was then sanded to a consistent 600 grit finish on a flat backed surface with sandpaper. The thermal calibration plate had four M3 threaded holes which were used to align and attach the MatchID calibration plate to the thermal calibration plate.

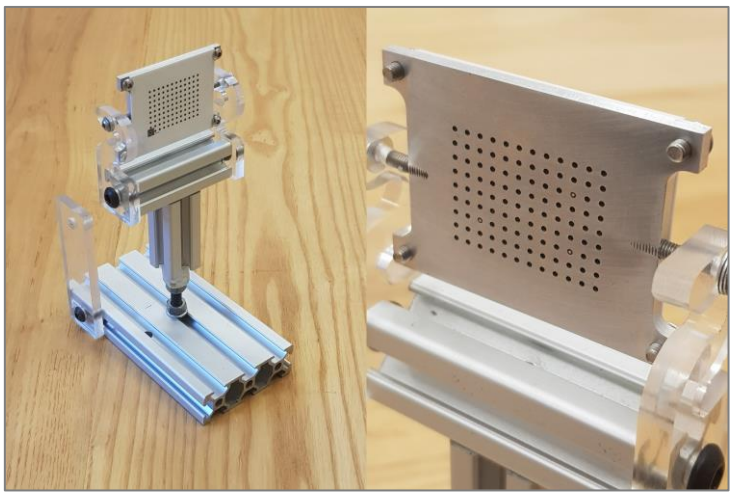


Figure 13: Calibration plate stand with thermal calibration plate.

Experimenting with various options to create adequate contrast when using the thermal camera to image the calibration plate showed that having hole features on a reflective metal surface provided the best contrast. Figure 14 shows the thermal image of the thermal calibration plate. The target style fiducial markers are clearly visible, and the contrast was sufficient for the special calibration we needed for the test. Further details regarding the thermal-DIC calibration process are described by Blackwell [33]. A more general thermal-DIC calibration process used for the DIC Capture system is described in *Chapter 4.3.1.7*

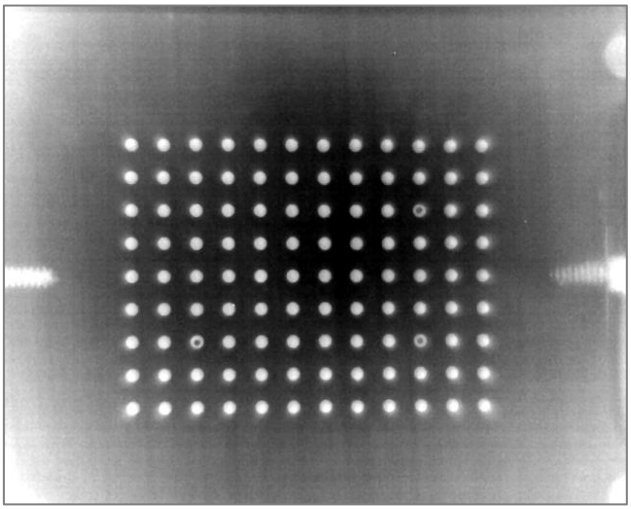


Figure 14: Non-Uniformity Correction (NUC) thermal image of the aluminium calibration plate.

### 4.2.2 Case Study 2

Case Study 2 focused on the work by Blackwell [33] where a novel testing approach was used in the Gleeble 3800 Pocket Jaw to investigate high temperature mechanical properties of Inconel 718 samples produced using Laser Powder Bed Fusion (LPBF). Owing to the high cost of the samples and wide temperature range required for this research, a novel approach using stereo DIC with thermal imaging integration was used to reduce the number of samples necessary for statistically meaningful data. By measuring a strain field and corresponding temperature field, the temperature dependant mechanical properties of a single sample can be calculated for a range of temperatures. The technique is described in the document by Blackwell [33]. Factors from that work that are relevant to this report are:

1. Stereo DIC with synchronised thermal imaging.
2. Synchronisation with the Gleeble 3800 with Gleeble output data being linked to each DIC image.

The initial approach by Blackwell attempted to use the commercial DIC system LaVision for this testing technique, but there was no reliable means to synchronise the Gleeble data and thermal images with the LaVision system. LaVision's dual level calibration plate [34] could not be used for this test due to low contrast on the thermal camera and technical difficulties in modifying the calibration data natively in LaVision to match the camera system.

The DIC Capture Version 2 system comprised of an ad hoc disorganised arrangement of hardware and software initially. Figure 15 shows the hardware used for camera triggering and synchronisation with the Gleeble. Here, a breadboard was used to allow for easy modifications as needed during the development stage. Details of the Version 2 circuit and software were not systematically documented, but those for Version 3 are discussed in *Chapter* 4.3.

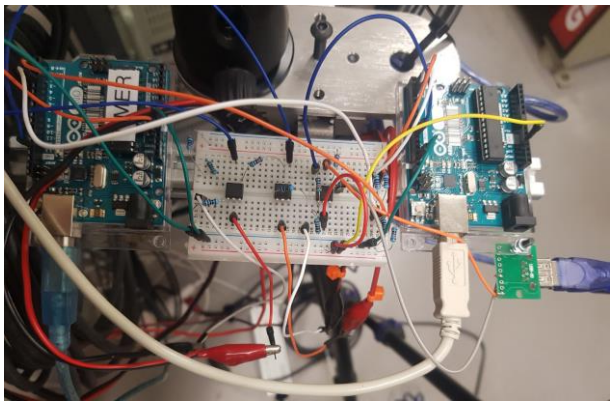


Figure 15: Electronic hardware that was used during case study 2

While functional, the components used in V2 as part of the DIC capture system, Gleeble 3800 and Telops thermal camera were not neatly and optimally arranged. Figure 16 shows the setup during testing where multiple independent systems can be identified. The improvement of the arrangement of the DIC capture system was identified as a crucial requirement for Version 3.

Figure 16: Gleeble room during case study 2 testing. Blackwell (left) and Vos (right).

Version 2 of the system met the requirements for the successful implementation for the work by Blackwell [33], which was to achieve the overlay of thermal data on the DIC image, such as the example shown in Figure 17 [33].

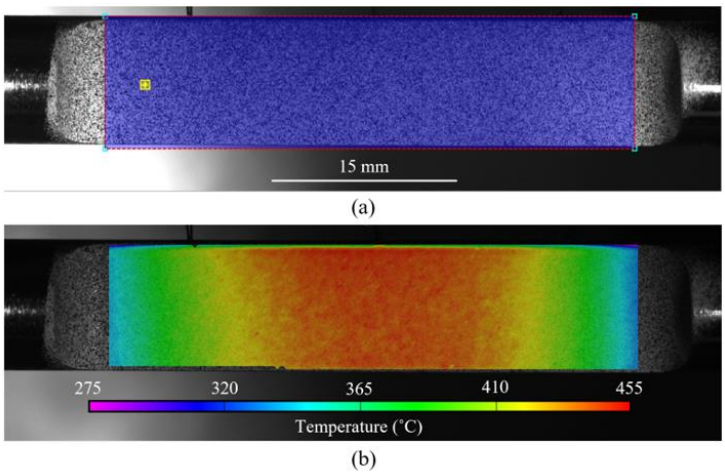



Figure 17: (a) DIC region of interest. (b) Thermal data overlay. Image from Blackwell's thesis [33].

Another successful outcome is shown in Figure 18 [33], where the temperature dependant mechanical properties were determined. These results confirm that the DIC Capture system Version 2 could generate useful research data and provided strong motivation for a Version 3.

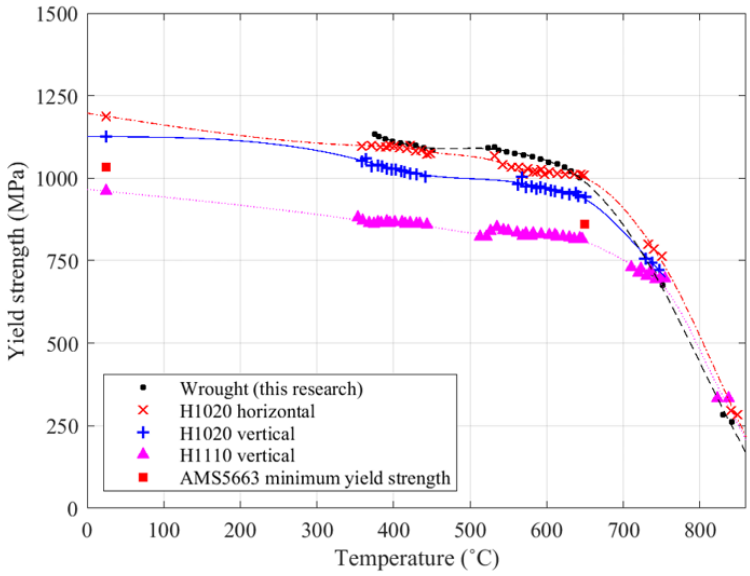



Figure 18: Results of temperature dependant yield strength. Image from M. Blackwell's thesis [33].

### 4.2.3 Problems identified during Version 2

It was found that using features on the Gleeble that integrate with the built-in safety system, such as opening the door resulted in the activation of a short pulse output from

the Quench 4 output. This short pulse output was also observed when the chiller system would be activated mid-way through a test. These instances were an intermittent problem and took some time to isolate. An example of this short pulse from the chiller activation can be found in Figure 19.

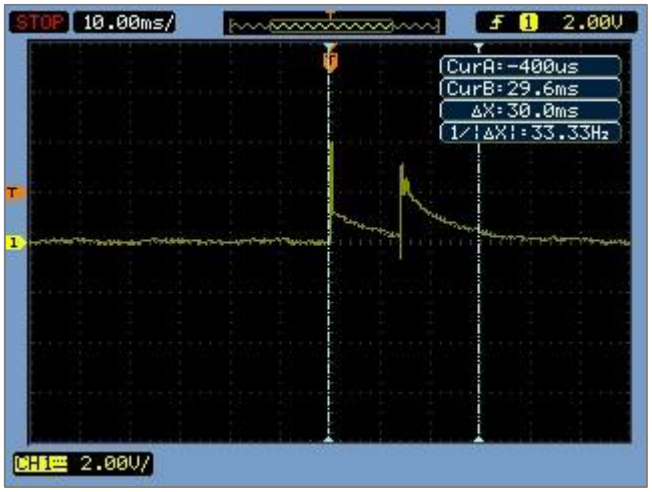



Figure 19: Quench 4 signal on chiller activation.

Whilst using the Version 2 system for Case Study 2, the extended duration of the tests would occasionally experience a false trigger. A temporary solution to this was the change of the frame rate array to continue […,0.125,0,0.125], resulting in image acquisition to continue until manually stopped if there was a false trigger. This temporary safety mechanism helped prevent a number of tests from losing DIC data, but this was not an elegant solution. To permanently fix this problem in Version 3; the 30 ms delay was used to prevent possible dropped frames and was used to verify that the state change of the FPS change pin was consistent for at least 30 ms. Only if this consistency requirement was met, could the FPS be iterated.

## 4.3 Version 3

Version 3 is the final version of the DIC Capture system for this investigation. Version 3 was required to be an effective tool for capturing raw DIC data by offering a simple user experience that most researchers would be able to use for standard DIC tests; whilst offering a robust framework from which the system can be modified for advanced DIC tests should the user be so inclined.

It is worth noting that Version 3 was developed during the upgrade of the Gleeble 3800 controller, which included the Gleeble DIC package and induction heating module. This allowed the DIC Capture system to be verified for compatibility with both the original,

and newest, version of the Gleeble 3800 GTC control module. The new GTC controller has a DIC system incorporated with a DIC.trigger signal, which could now be used instead of the Quench 4 signal for Version 3. However, there is no functional difference between Quench 4 and DIC.trigger after addressing the false trigger issue experienced in Version 2.

The documentation of Version 3 is divided into two sections: *Python Software* and *Arduino triggering system*.

### 4.3.1 Python Software

The core capturing software remained the same from Version 2, with the main new features being:

1. GUI interface for capturing system.
2. Migration to Git version control.
3. Redesign of image saving algorithm to be less CPU intensive.
4. Integration with new communication protocols for the latest triggering system.
5. Live camera exposure adjustment and during Test Mode.
6. Redesign of synchronization algorithm.
7. New algorithm for reformatting thermal images to be compatible with MatchID.

These new features are described in greater detail in the following section. From a user perspective, these upgrades have transformed the system from an ad hoc system into a coherent and useful research tool for others to use.

#### *4.3.1.1. GUI interface for capturing system.*

The GUI's (Graphical User Interface) primary purpose is to expand the usability of the software to those who are less familiar with programming. Daniel Slater assisted with the setup of the GUI and provided a robust framework upon which the final version of this research could be made accessible to users through the GUI.

After running the GUI module, the window showing Figure 20 is generated. The user selects a main working folder for the project they are working on. Various configurations can be saved as config.json files and loaded from this folder for future tests. This ensures that all test variables are consistent between tests and shortens the setup time.

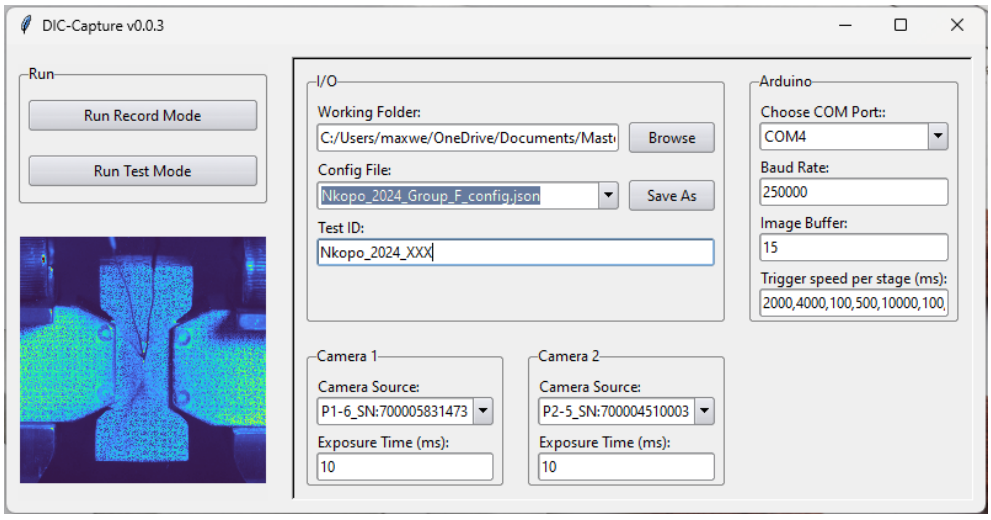



Figure 20: DIC Capture GUI

There are two run modes. “Run Test Mode” is used when setting up an experiment; the cameras are configured to run as fast as possible and functions such as *zoom*, *show histogram, adjust exposure* and *display live ADC values* are all enabled. Only Camera 1 and 2 parameters need to be set before clicking the “Run Test Mode” button, all other relevant fields are automatically populated.

“Run Record Mode” saves all incoming data and is used when running an experiment for results, all additional features are disabled to prioritize compute resources and maximize stability. A unique “Test ID” must be set before entering record mode, a test folder is created with the Test ID name, where all data relevant to the current test are stored. By creating a new folder for every test, the possibility of overwriting previous data is eliminated and if the file name already exists, the program does not proceed. A default version of the test ID is also loaded with the selected config file. An example is shown in Figure 20 where the default in this case was “Nkopo_2024_XXX” where the “XXX” would be replaced by the appropriate test ID for the experiment.

In the Arduino section of the GUI, a list of available COM port devices is automatically populated. The user is responsible for selecting the COM port associated with the triggering device; in practice though, there is seldom more than one COM port option. The baud rate should not be changed from 250 000 unless a different microcontroller is used for the triggering device. This field is automatically populated. Running at a higher baud rate will increase serial data transfer rates; but 250000 bits/second was found to be the upper limit for reliable operation.

The “Image buffer” indicates that every n’th image received from the cameras will be displayed on the computer. This was included to reduce the hardware requirements for the DIC capture system. When receiving high bandwidth data from the cameras, the computer may not be able to process these images into a displayable format faster than the images are coming in. This could result in frames being dropped. In practice, using an image buffer of 3 is more than safe enough for most computers.

The ”Trigger speed per stage” is one of the main reasons for embarking on this project. For tests where events happen at different time scales, selecting one frame rate to adequately capture DIC data is often impractical. Most commercial DIC systems had not implemented an effective solution to this at the time of development. By inputting a string of comma delimited integer values, which represent the triggering period in milliseconds, the DIC capture system will iterate through these values each time a FPS change signal is received. This FPS change signal is a 5 V rising edge signal sent from the Gleeble, either by calling the DIC.trigger or Quench 4 function in the GSL file being used for the current Gleeble test. An example of this code is seen in Figure 21, if the user does not have the DIC module from Gleeble, they can simply connect it to the Quench 4 output from the Gleeble and replace the DIC.trigger with a Quench 4 on/off pulse. Note that Quench 4 has a minimum cycle time of approximately 500 ms. When a FPS change pulse is received by the DIC Capture system, a 30 ms delay is included to ensure that the cameras are not accidentally double triggered during a FPS change. This delay is not present for the first FPS pulse that it received in order to maintain synchronous starting times. It is important to sample at a high frequency when calling the DIC.trigger as the recorded DIC.trigger data is used to synchronize the Gleeble output data with the DIC capture data. It was found that a 100 ms delay before, and a 130 ms delay after the DIC.trigger is required to ensure that the rising edge is captured at 1000 Hz. Shorter delays may be possible; however, this is challenging to optimise due to the limitations of the GSL system of the Gleeble 3800.

```
//=============================================(/1)
// Start of soak
// Frame Period: 4000ms
repeat 1
    sample at 1000Hz
    sync
    delay 100msec
    sync
    set DIC.trigger to on
    sync
    delay 130msec
    sync
    sample at 20Hz
    sync
end
//=============================================(1/)
```

Figure 21: FPS change code in GSL

The camera 1 and 2 input fields are used to select the USB port ID for each camera. The drop-down box selection is automatically populated by using the Baumer neoAPI to find the port ID and associated serial number for each camera. It is generally recommended to make camera 1 the reference camera as this makes the calibration process easier during post processing. Users can either match the serial numbers on the camera to those shown in the GUI or unplug one camera to identify which camera is associated with its respective port.

At the time of writing, only Bamer cameras are natively supported in the DIC Capture system as time constraints did not allow for making multiple camera brands compatible. Further compatibility with different camera manufactures' is currently being worked on as part of a parallel project.

#### *4.3.1.2. Migration to GitHub*

GitHub is a widely used software development platform used by developers for collaboration, version control and much more. The primary reasons for migrating to GitHub for this project were to make the process of using the latest version of the code on multiple computers easier and to release the code as an opensource project so that others can modify it. The documentation of each commit was another feature that proved useful when developing the newer versions of the DIC capture software.

Currently the user would have to have some very basic programming knowledge to run DIC Capture in an IDE as an executable version of DIC Capture has not yet been made available. Releasing a program as an executable version limits its adaptability to the end user and often becomes incompatible with future updates of Windows. In future, it would

make sense to have a basic form of DIC Capture available as an executable, with the full version available through GitHub.

#### *4.3.1.3. Redesign of the image saving algorithm*

In Version 2 of the DIC capture software, two arrays were used to read and save the data coming from the camera as described in *Saving image data* section. This worked well on a relatively powerful computer but there was room for improvement.

Whilst working on a less powerful laptop (ASUS Vivobook 16x), the storage write speeds were bottlenecking the maximum frame rate of the system to below that which was possible with the theoretical transfer rates of USB 3.0. It was decided that making DIC Capture more dependent on RAM instead of CPU performance would be beneficial for many users as upgrading a systems RAM capacity is significantly less expensive than upgrading its CPU. The new saving algorithm creates a large *super_img_arr* for each camera during initialization. Incoming images are sequentially appended to the array whilst old images are simultaneously being removed and saved from the tail of the array. The *super_img_arr* length is set to 300 (images) by default as this allocates a maximum RAM usage of approximately 3GB per camera. Due to the images being removed from the array after they are saved, if the save speed of the images exceeds read speed from the cameras, there should only be a maximum of one image in the *super_img_arr.* During higher frame rates, the save speed may be less than the read speed from the cameras and an increasing number of images will be stored by RAM within the *super_img_arr.* If this continues for an extended period, the RAM will fill up and the DIC capture software will start dropping frames by overwriting existing images in the *super_img_arr.* This is extremely unlikely to occur during a standard DIC test as most DIC test acquire less than 300 images and even mid-range laptops with 16GB of RAM and an SSD should be able to manage at least 900 images being read at the maximum transfer speed of two USB 3.0 ports. In unique cases where large data sets are expected, upgrading the computers RAM to 32GB or 64GB would be a reasonably cheap solution.

#### *4.3.1.4. Integration with new communication protocols*

The formatting of the serial data from the Arduino has changed from the previous version. Data column now have the following names:

“trigger_frame_count,
adc_frame_count,
test_run_time,

fps_change_count,
fps_change_time,
adc.ch1_volts,
adc.ch2_volts,
adc.ch3_volts,
adc.ch4_volts,
data_delta".

A description of each category is listed below:

1. *trigger_frame_count*: a count value starting at 0 which increments each time a trigger pulse is sent to the cameras. *trigger_frame_count* is equal to zero for the first image.
2. *adc_frame_count:* similar to *trigger_frame_count*. A count value starting at 0 which increments each time the ADC records an input voltage. Currently the system is set to record 10 ADC records for each image which is acquired. So if the triggering frequency of the cameras is 10 Hz, the ADC will record and save data at 100 Hz.
3. *test_run_time:* the time in milliseconds since the first image was taken.
4. *fps_change_count:* A count value which is incremented each time a FPS change signal is received. Mainly used for data management purposes.
5. *fps_change_time:* The exact *test_run_time* when an external FPS change signal is received. This is used for syncing data with external systems. For the Gleeble, the DIC.Trigger or Quench 4 data can be acquired to match Gleeble data with the DIC data. In general; any system where the time of a rising edge signal is known, can be temporally synchronized to the DIC Capture system. It is worth noting that only a start and stop signal are required if no FPS changes are required during the test.
6. *adc.chX_volts:* Output from each ±10V ADC channel. There are 4 channels which can be use simultaneously. It is left to the user to convert the voltage into the equivalent unit for their sensor output. It is the user's responsibility to calibrate their system. This can be done by inputting known voltages and reading what is recorded in the *adc.chX_volts,* a calibration curve can then be generated from the differences in these values.
7. *data_delta:* Used to track if the serial communication speed is keeping up with the data generation. This should always be zero. If it increases, one should decrease the amount of ADC records taken between frames. In the Current configuration, 33 FPS is the maximum frame rate with a 10:1 ratio of ADC to frame triggering. 33 FPS is the maximum continuous speed for the cameras we generally use (Baumer VCXU50M) at 12-bit pixel depth, hence the choice of

10:1 ratio. On board storage solutions could be implemented in future to increase maximum data rate; however, this was omitted due to time constraints and limited applicability to current research projects at the CME.

#### *4.3.1.5. Live camera exposure adjustment in Test Mode.*

In Version 2 of the DIC capture software, the camera exposure time would have to be set before viewing the images in test mode. This made adjusting the exposure time tedious as the program would have to be restarted every time the user wanted to change the exposure time. A quick solution of using the arrow keys on the keyboard to increment the exposure time by 1 ms and print the current exposure time was used. The up-down arrow keys are used for camera 1 and the left-right arrow keys are used for camera 2. The current exposure time for the relevant camera is printed in the Python console. The user can then use this value for when they next run the DIC test. This feature is not available during Record Mode to improve stability, but the software could easily be modified if the user needed to.

#### *4.3.1.6. Redesign of synchronization algorithm.*

Version 3 of the synchronization algorithm incorporated the new communication protocol with the Arduino, making use of rising-edge instead of any-edge as *fps_change* triggers and utilized the better timing accuracy of the ESP32 based Arduino. The algorithm used in the *'post_processor_syncronization.py'* program adaptively reads the Gleeble data from the Gleeble output file for the test. If the Gleeble output file is not automatically detected, the user will be prompted to select the Gleeble output file. The Gleeble timestamps for the rising edges of the DIC.Trigger dataset are compared to the *fps_change_time* in the Arduino output file. The average difference between the two is used to shift the *test_run_time* of the DIC data. The DIC data is then automatically interpolated into the Gleeble data and saved as a separate file. The user has the option to generate a Matplotlib graph of the *Gleeble Force* and *DIC ADC Force* vs *time* to ensure that the synchronization was successful. Figure 22 shows this graph. The DIC Capture ADC values are shown in red when an image is captured and blue for the ADC subsampling, the black plot is the Force output from the Gleeble data. Only a rough conversion from ADC volts to Force was used for demonstration purposes, the user should verify that the conversion ratio is correct for their test setup. The Gleeble data appears to be averaged between samples when sampling at lower frequencies, which can create discrepancies between the DIC Capture ADC values and Gleeble data if there is a significant difference in sampling frequency between the two systems.

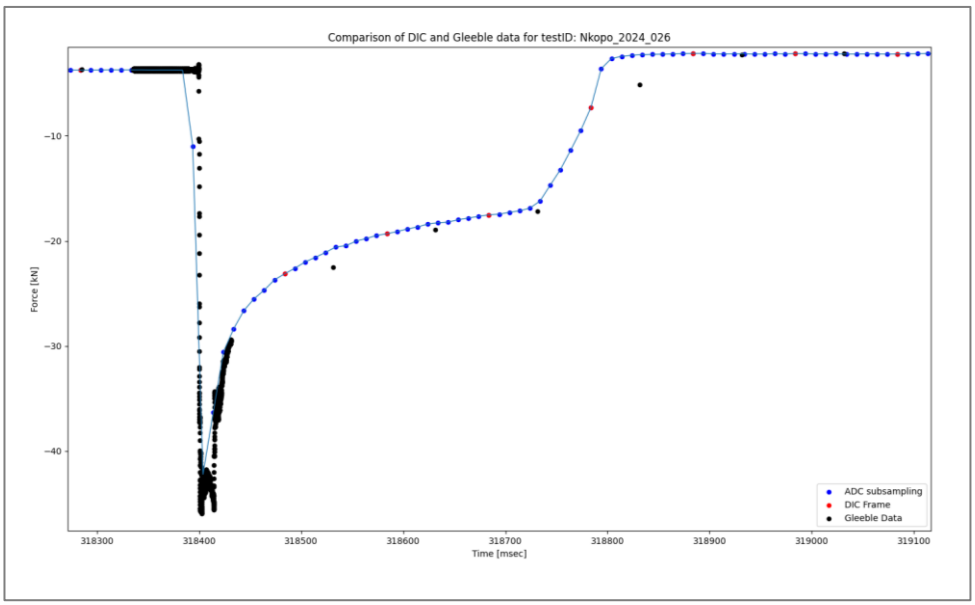



Figure 22: Graph generated during synchronization showing PSC Hit

It is worth noting that in addition to the average offset determined using the DIC.Trigger times, there seems to be a constant offset of approximately 113 ms on the Force data from the DIC.Trigger data. The reason for this is unclear as deeper access to the Gleeble signal processing systems would be required; however, it appears to be constant across tests. An *offset_constant* variable was therefore included at the start of the code for users to make fine adjustments to timing offsets for their setup should they require it.

No scaling of data was implemented between each DIC.Trigger events which was necessary in Version 2, as this was seen to introduce error due to the unreliable nature of the DIC.Trigger and Quench 4 recording from the Gleeble. The 2 ppm timing drift accuracy of the ESP32 should be adequate for most tests assuming a comparative timing drift accuracy from the Gleeble system. A quick confirmation of this assumption is found by comparing the DIC capture systems data with the Gleeble data at multiple times where there are sharp force changes. Figure 22 and Figure 23 occur more than 5 minutes apart from each other and the temporal synchronisation is adequate for almost all DIC applications at the intended frame rates of the DIC Capture system. If the user wished to gain further timing accuracy, matching the data using the force data would provide better results but require a more complex test set up and offer less flexibility.

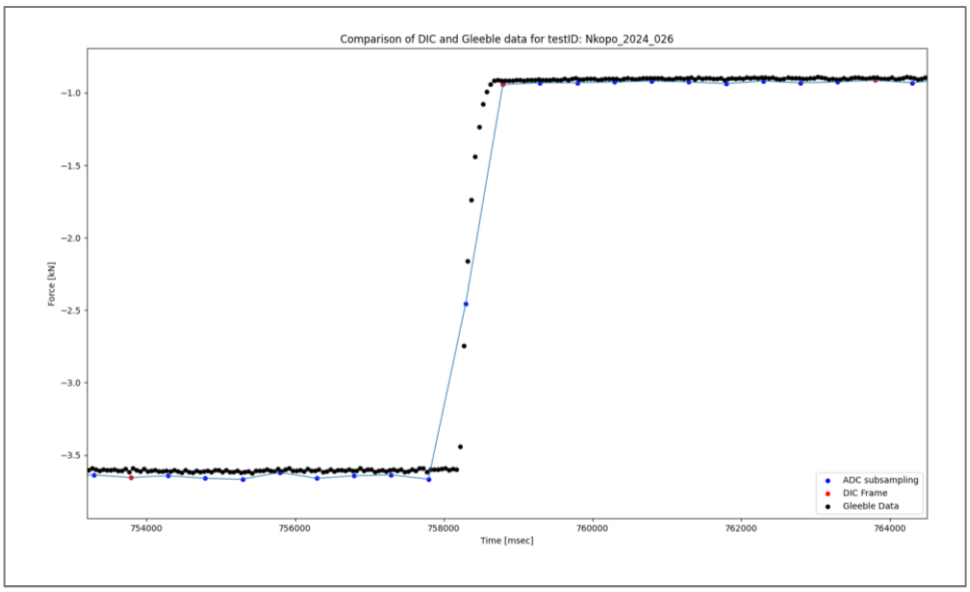



Figure 23: DIC capture data and Gleeble data when turning air ram off

#### *4.3.1.7. Reformatting thermal images for MatchID.*

Thermal images are captured using the Telops FAST M350 thermal camera with the Reveal IR V1.10.1 software package. Each sequence is saved in a proprietary .hcc project format. Images can be exported as raw .tiff files; note that the older versions of the Reveal IR software cannot export in this format so one should update the software if using an old computer. The raw .tiff files have floating point pixel values which represent the temperature in Celsius for each pixel. This is different from the usual convention of representing pixels on an integer scale between 0 and $2^n$-1. This unique format is convenient for temperature data as no further conversions are required; however, this format is not widely used and cannot be displayed on most software.

When spatially calibrating the thermal camera with the primary DIC camera, the thermal image must have the exact same format at the DIC camera. This is done in the *'telops_to_matchid_reformat.py'* program by performing the following operations to the thermal image:

1. Cycle through each image in the user selected directory, loading each image into a NumPy array at the start of the reformatting loop.
2. Normalise the pixel values of the image to a scale between 0 and 255. 8-bit format was used as accurate bit depth is not required for calibration.
3. The image is resized to the same resolution as the DIC image. If the thermal and DIC images have a different aspect ratio; a black bar is added to either the right

or bottom side of the image, depending on the aspect ratio. A linear interpolation is used during the upscaling of the thermal image to the DIC image resolution.

4. The pixel values are rounded to the nearest integer value and converted to the UINT-8 format.
5. The array is saved as a .tiff image in a separate '*Camera_3_Thermal_Resized'* folder with an image naming convention which matches the DIC image convention, where the camera ID is 3.

Once calibration is completed in MatchID using the reformatted images, the thermal images used for the *Thermal Analysis* module in MatchID need to be formatted specifically to work with the module. These images are saved as pseudo text files with the extension of .thermal.tiff. There was little available documentation for this during the initial development of this work. However, there is now sufficient documentation of file format on the MatchID wiki. This .thermal.tiff format requires the structure seen in Figure 24 and is further outlined in the comments of the *'telops_to_matchid_reformat.py'* program.

```
***MatchID Thermal
<Width>=<2448>
<Heigth>=<2048>
<Unit>=<1>
<Matrix>
24.07;24.07;24.07;24.07;24.07
24.07;24.07;24.07;24.07;24.07
24.07;24.07;24.07;24.07;24.07
24.06;24.06;24.06;24.06;24.06
24.05;24.05;24.05;24.05;24.05
24.04;24.04;24.04;24.04;24.04
```

Figure 24: MatchID .thermal.tiff formatting

The process for converting the exported .tiff images into the MatchID compatible thermal.tiff format using the '*telops_to_matchid_reformat.py'* program is summarized as:

1. Build a header string consisting of the first 5 lines shown in Figure 24. Note the spelling of 'Heigth', at the time of writing, the typo must be included. The image width and height are defined at the start of the '*telops_to_matchid_reformat.py'* program and should match the resolution of the DIC cameras used for the experiment.
2. Cycle through each image in the user selected directory, loading each image into a NumPy array at the start of the reformatting loop.
3. The image is resized to the same resolution as the DIC image. If the thermal and DIC images have different aspect ratios; a black bar is added to either the right

or bottom side of the image, depending on the aspect ratio. A linear interpolation is used during the upscaling of the image.

4. Use the NumPy '*savetxt()*' function to save this image with the format shown in Figure 25 where the *output_match_ID_path* is the file name of the image ending in '.thermal.tiff'. These images are saved to a '*Camera_3_Thermal_MatchID'* folder withing the main folder for the test. It is worth noting that in the 'fmt' section the number of decimals can be changed; in Figure 25, the temperatures are rounded to whole numbers, but this can be increased if needed. Adding more decimal places increases the file size of each image, and the file sizes are already sizable due to the unique format, so this should be considered if there are many images per test.

```
np.savetxt(output_match_ID_path,
           boarder_img,
           fmt='%1.0f',
           delimiter=';',
           newline='\n',
           header=heading_string,
           comments='',
           encoding=None)
```

Figure 25: NumPy savetxt format for MatchID .thermal.tiff files

### 4.3.1 Arduino triggering system

Version 3 of the Arduino triggering system needed a full overhaul from Version 2. Version 2 used an Arduino Uno R3, which was released 12 years before the time of writing, and newer versions offered superior performance and features at a similar price point. Using one of the newer versions would also assist in future proofing the project somewhat. The Arduino Nano ESP32 was chosen as a replacement for the following reasons:

1. Recently released at the time of development with a widely used processor (ESP32) that offered sufficient performance for this application.
2. Easy to integrate into circuit boards due to its standard pin layout.
3. Cheap and readily available.
4. Wi-Fi and Bluetooth built in which may be useful in future.

Further useful features which were identified in Version 2 that were added to Version 3 are:

1. ±10 V ADC
2. Multiple connection options for the ADC and camera triggers.

3. State and trigger indication light.
4. RTC option for longer duration tests.

The Arduino coding and circuit board design for Version 3 was performed in collaboration with David Dallas.

Version 3 of the circuit board is show below in Figure 26

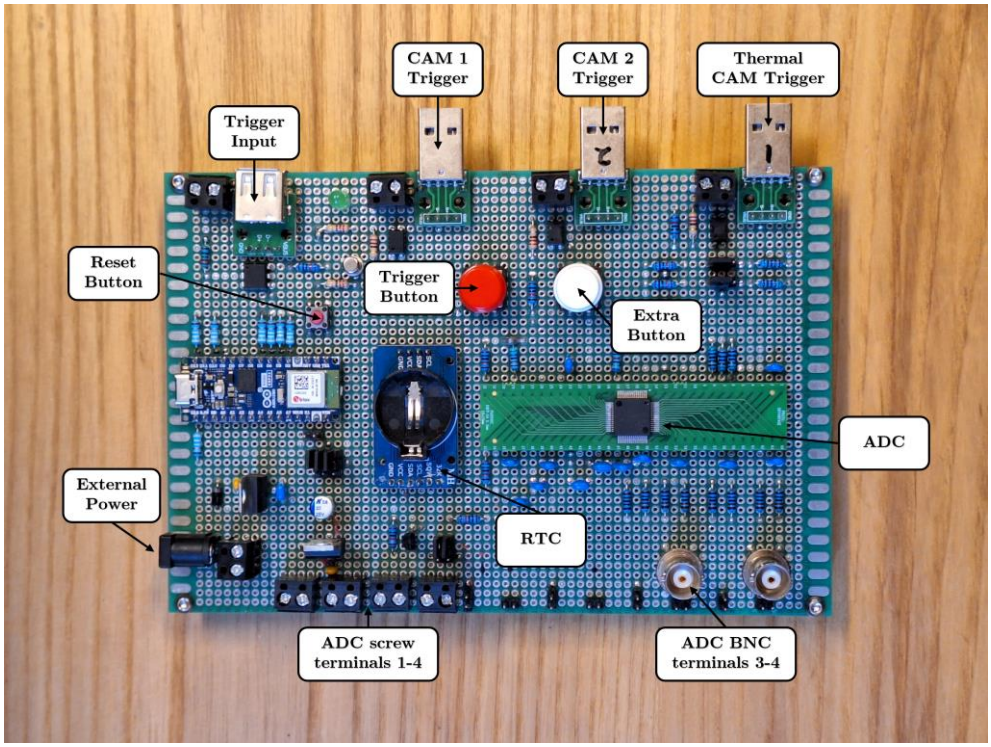



Figure 26: Diagram of Version 3 control board.

#### *4.3.1.1. ADC selection*

There were no ±10 V ADC modules designed to be easily integrated with the Arduino that could be purchased at the time of this project. After experimenting with converting a ±10 V signal to be compatible with commonly available 0 to 5 V ADC modules by using op-amps, it was determined that building a custom ADC module would be the best long-term solution as a high-performance ADC could be used for a very low cost.

The ADS8584S from Texas Instruments was chosen for this project. The ADS8584S met the following criteria:

1. ±10V range with simultaneous sampling on 4 channels. There are seldom scenarios where more than two ADC inputs are needed; however, having the 4

inputs would be useful for more complex experiments and the additional cost to include the extra 2 channels was negligible.

2. 16-Bit resolution should ensure that the resolution of the ADC is not the limiting factor of the system accuracy.
3. More than enough speed, with a maximum of 330 thousand samples per second per channel, multiple samples can be averaged to produce a more stable reading.
4. Cost effective and readily available from www.digikey.com (part number 296-50900-1-ND) who ship worldwide.

One challenge with using ADS8584S is that it comes in a 64-Pin LQFP package that is designed to be soldered directly to a circuit board. Given that the development of this board was still in the prototype stage; having a circuit board made was impractical as the turn-around time on iterations would be too long. Since only the ADS8584S needed to be a surface mount component, a 64-Pin breakout board (part number 315-PA0096C-ND from www.digikey.com) was used to make the chip compatible with standard Vero board. This offered a good compromise between prototyping with an optimal ADC whilst reducing the turnaround time between iterations. At the time of writing, the Trigger board V3 has proved to be reliable enough to create a circuit board design for in future. As soon as this PCB design is complete, it will be released in a format that can be made by PCB manufactures such as JLCPCB, making it easy for future users to build their own Trigger boards or to purchase them as pre-assembled kits if someone offers this service.

#### *4.3.1.2. Multiple connection for the ADC and camera connections.*

The GPIO connections for industrial cameras are not standardized and therefore often must be stripped and soldered to connections that are suitable for the triggering system. In Version 2, USB-A connections were used as they were convenient board mountable solutions that could also be connected to the GPIO cables for the camera. This worked well and was also used in Version 3, with the addition of screw terminals for each camera, ADC, and trigger connection. This will allow for easy connection of any cable directly to the board for future users if they do not wish to use USB connections. BNC ports were added to the ADC channels as this is a standard connection for analogue voltages. BNC ports will be added to the camera trigger circuits in future as well. Adding a BNC port for a opto-coupled open-close trigger and a 0-5 V trigger signal isolated from the normal camera triggers would further improve the boards versatility. This was outside the scope of work for the testing in Case Study 3 and was therefore left for future iterations of the board.

#### *4.3.1.3. State and trigger indication light.*

When using Version 2 of the trigger board, an oscilloscope was used to track the frame triggers to monitor for any problems. The inability to easily monitor the current state of the Arduino created opportunity for confusion or mistakes when testing. Version 3 has a green LED wired to the camera trigger signal that shows when each camera is triggered. There is a RGB led built into the Arduino Nano ESP32 that is utilized to indicate that the board is in one of the following states:

1. Solid blue: Arduino setup code is running.
2. Flashing blue: ready to receive frame rates from the control computer.
3. Solid cyan: paused and waiting for first trigger signal.
4. Solid green: Cameras are being triggered.
5. Solid yellow: Camera triggering has been paused (occurs when a zero millisecond period is sent), but the test is not over.
6. Solid red: Test is over, indicating that the any capturing software attached to the board should be stopped, saved or reset. The Arduino can then be reset for the next test.

#### *4.3.1.4. RTC option for longer duration tests.*

Timing accuracy was of particular concern as a development from Version 2. There was uncertainty as to whether the quarts oscillator built into the ESP32 would have a low enough drift to be useful. A DS3231 real time clock (RTC) was included in the circuit design as a back-up for if there were problems with the ESP32. It was determined that the timing accuracy from the ESP32 was sufficient for the purposes of this work and the DS3231 was never integrated into the code despite being soldered to the trigger board. In future it may be useful for long duration tests, over multiple days, and with the Wi-Fi connectivity of the Arduino Nano ESP32, the RTC can be calibrated using Network Time Protocol (NTP) which could be used for other systems which are synced to NTP such as satellites or remote sensors. The DS3231 has an included battery that would maintain time even if power is lost to the system. Given the promising potential that the RTC offers, it was left as a future project, to be implemented as a software upgrade when the need arises.

### 4.3.2 Case Study 3

The intention of Case Study 3 was to verify the performance of DIC Capture Version 3 by testing all the new features discussed in the previous sections. The reliability of the system will be assessed through real world DIC tests on the Gleeble 3800. It is worth noting that the direct use of the generated data is outside the scope of this project. The objective is to generate data that proves the DIC Capture system integrates with the

Gleeble 3800 and MatchID software and produces useful raw DIC data for other researchers.

Case Study 3 was setup to assist with multiple projects currently under way at the CME, these projects are listed below:

1. Aid in the testing of Nkopo Chaole's PSC samples which would be used for a master's thesis project.
2. Verify uniform heating of PSC samples during the installation of the new Gleeble induction heating module.
3. Provide data for an upcoming paper comparing resistive and inductive heating of PSC on the Gleeble 3800.
4. Be a test run for using DIC with PSC and provide the framework for a future paper on the subject.

#### *4.3.2.1. Noteworthy additions to the experimental setup*

A few new additions to the Gleeble and camera setup were required to accommodate the new system.

##### *4.3.2.1.1. Thermal viewport for Gleeble 3800 Hydrawedge*

For Case Study 3, both thermal and DIC images are temporally and spatially calibrated. However, for PSC tests, the same assumptions made in Case Study 2 regarding uniform temperature through the thickness of the sample cannot be made. The thermal camera therefore needs to image the same surface seen by the DIC cameras. The CME owns a large IR window (Fluke CV400) which is transparent to both visible and thermal wavelengths, but it is not possible to achieve line of sight to the sample with both DIC cameras and the thermal camera through the CV400 window.

Instead of looking through one window, a custom mounting for the CV400 was designed and manufactured as part of this project and was bolted to the bottom viewport of the Gleeble chamber, as seen in Figure 27. The DIC cameras can then easily be positioned to look through the standard glass viewport on the chamber door. Due to the geometry of the Gleeble chamber, the viewing angle of the front face of the sample will be approximately 45 degrees, which is ideal as both the front and bottom face can be seen, providing useful information on the temperature distribution across the PSC sample. Additionally, this camera poisoning makes it possible to view a calibration plate with all three cameras at once. The standard DIC and Thermal Camera calibration process can therefore be used for Case Study 3.

#### *4.3.2.1.2. Thermal camera stand.*

The limited working space between the DIC tripod and the Gleeble chamber made positioning the thermal camera a challenge. The Telops M350 weighs approximately 6 kilograms, and therefore requires a sturdy and compact tripod to ensure that the very expensive and delicate camera is protected. A solution was the design and installation of a custom steel plate with a large diameter threaded bar to be attached to the base. The base had a M6 threaded hole placed in each corner where bolts and locking nuts were used to stabilize the system if the floor was not flat. A locking flexure clamp was used for vertical movement and a Manfrotto 410 Tripod Head was used for fine 3 axis tilt adjustment to align the field of view of the thermal camera with the sample.

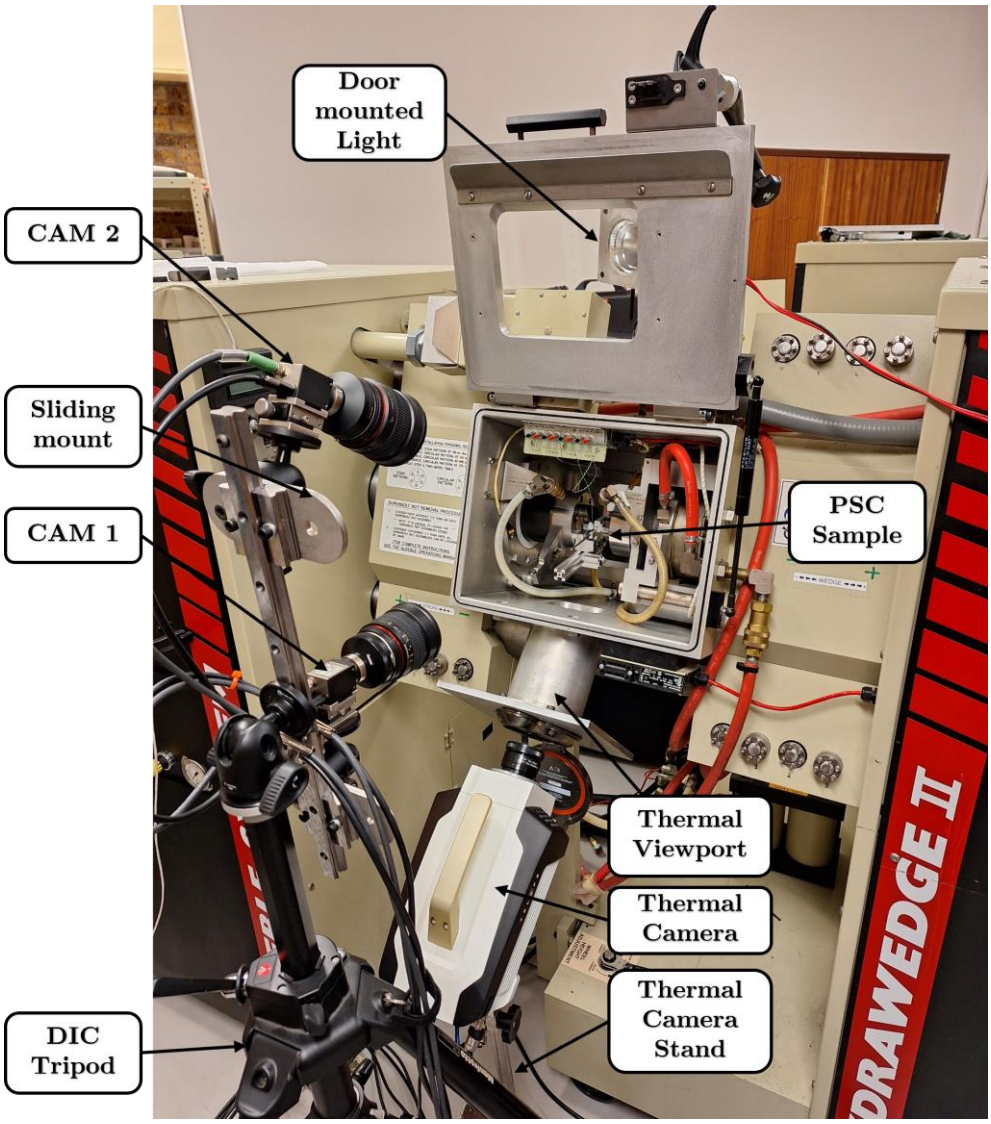



Figure 27: Camera setup for case study 3.

This stand functioned well, despite its rough design. If this system is used frequently, a more robust solution may be warranted to minimize the chance of the camera moving due to accidental bumps to the stand, and possibly make the alignment process easier.

#### *4.3.2.1.3. New speckling technique*

Creating a uniform, high quality DIC speckle pattern is a challenging exercise which requires both practice and skill if using standard spraying techniques. For Case Study 3, a speckling technique which uses laser marking of paint was developed. A galvo-scanning 1064 nm wavelength 30 W pulsed fibre laser was used to mark Very High Temperature (VHT) white paint (Dupli Color Inc., Cleveland, Ohio, United States) with a low power setting being used to turn the white paint black, whilst not ablating the paint. This was remarkably effective and produced high contrast dots of approximately 100 microns in diameter which can be seen in Figure 28. The laser file used for these samples is included in the supplemental material. A dot size of 100 microns was limited by the laser beam diameter when focused. The speckles were intentionally placed more sparsely than the suggested density for DIC as it was anticipated that during compression of the sample, the features may become too crowded to correlate.

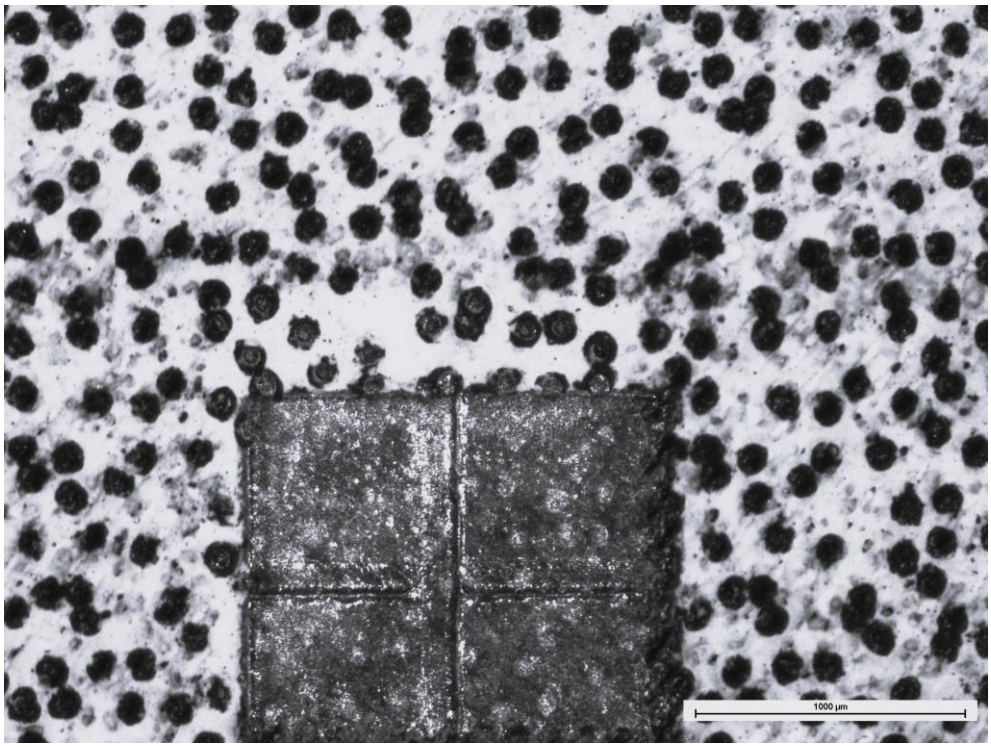



Figure 28: Laser speckle pattern close up.

An additional benefit to using the Fibre laser for marking is that a small square area of paint can be removed for the welding of the thermocouple. This enables the thermocouple welding to be done after applying speckles which greatly speeds up the testing process. The lines in the centre of the square are scribed before painting and are used to position the thermocouple when welding. If the thermocouple is not accurately placed; it can move far from the centre of the sample due to the material flow during PSC, which negatively effects the temperature measurement accuracy. Figure 29 shows the top half of the PSC sample front face which will be seen by the DIC cameras. A 2.5 x 2.5 mm area of random speckles was generated and then repeated to fill the painted surface of the sample. Ideally, repeating grids should be avoided; however, the current laser software configuration makes it difficult to externally generate files that produce high quality dots. The dot features therefore must be manually positioned which would have taken too long for the whole sample. In practice, the 2.5 mm grid size is significantly larger than the speckle and corresponding facet sizes used during analysis; and there are no correlation issues, provided that excessively large facets are not used.

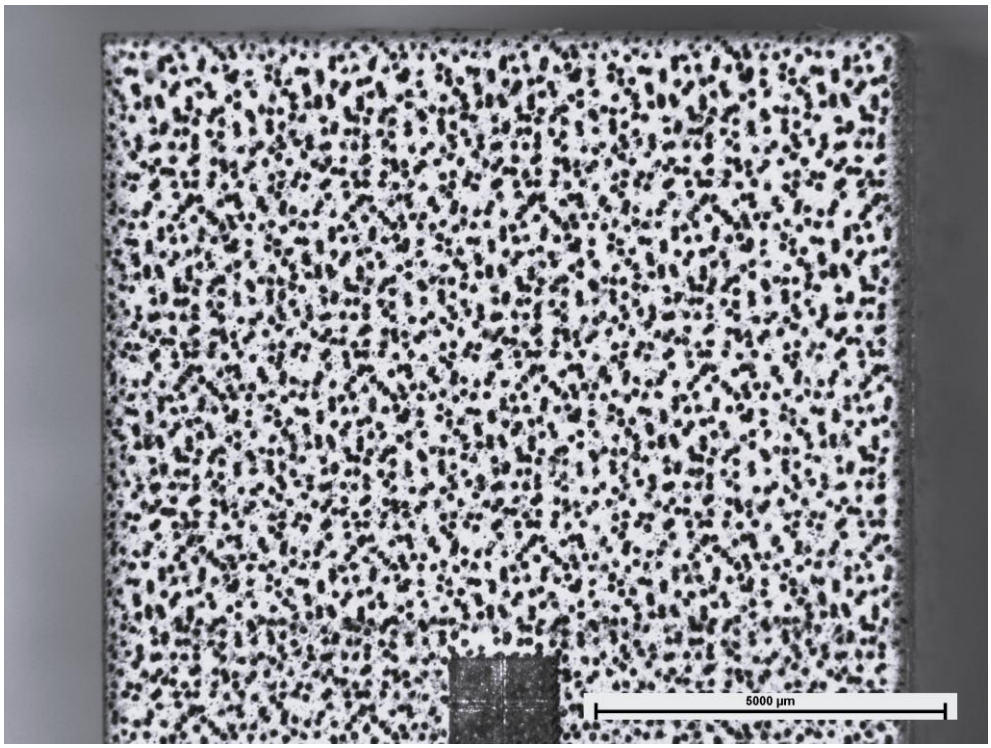



Figure 29: Half of PSC face visible to DIC cameras with TC weld pad.

#### *4.3.2.2. Testing procedure*

The testing for Chaole included multi-deformation PSC tests on AA3104 with deformation temperatures of 360 °C, 330 °C and 300 °C; at strain rates of 30 /s, 50 /s

and 100 /s for true strains of 0.3, 0.32 and 0.4 respectively. The interpass times were varied to observe the effects of differences in interpass times experienced by the leading and tailing edge of aluminium stock in a rolling mill. The results of this Chaole's experiments have not yet been published at the time of writing.

The DIC cameras available were not fast enough to capture useful DIC data during the deformations. Instead, the DIC capture system was used to inspect for non-uniform heat distribution over the sample during the test. Other useful information that can be obtained through DIC is the quantification of slow deformation during inter-passes due to the holding pressure; similar to the focus of Case Study 1, this would not be possible using the standard LVDT set-up due to compliance and thermal expansion uncertainties.

The primary reason for using the DIC capture system during these tests was to quantify its reliability and act as a validation for future PSC testing with DIC. A total of 44 Gleeble tests were performed for Chaole's work, 32 of those test incorporated DIC with thermal imaging integration. Of those 32 DIC tests, five tests failed. All five of these failed tests presented as the computer restarting in the middle of a test and each happened sequentially after an unscheduled Windows update whilst conduction test number 032. Unfortunately, there is little one can do to prevent such failures, and it is highly probable that this is an isolated incident related to a faulty Windows update. Due to the isolated nature of this incident, no changes to the gui.py or run.py programs were made after the testing and the reliability of the system is considered adequate.

The general testing process with DIC for each sample is outlined in the following steps:

1. The sample is temporarily held in position using the custom PSC loader. The 'TC1' marking on the sample is always on the top side of the sample and closest to the operator, this ensures a consistent sample orientation. Figure 30 shows a typical un-deformed sample.

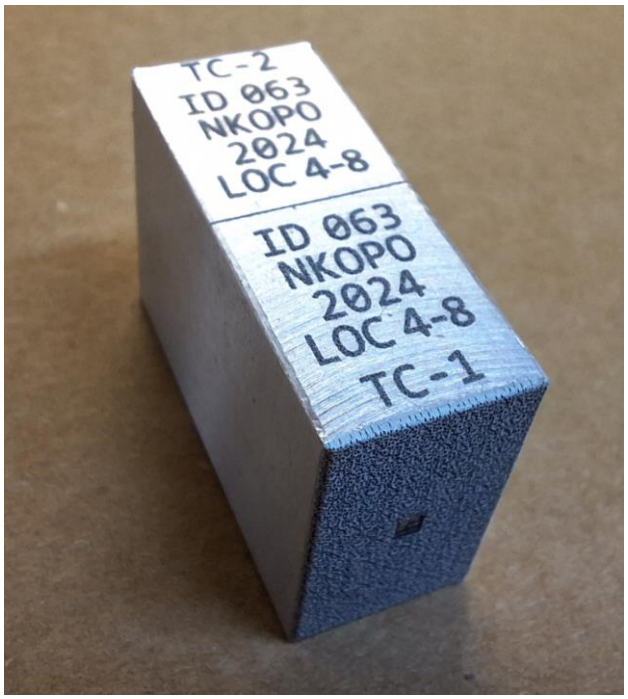



Figure 30: Example of markings and speckles on typical sample in case study 3

2. Air ram is turned on in compression, holding the sample between the anvils with AirRam force. The custom PSC loader is then removed and the Gleeble is ready, (see Figure 31) to run the GSL script for the relevant test (see Appendix for example code).

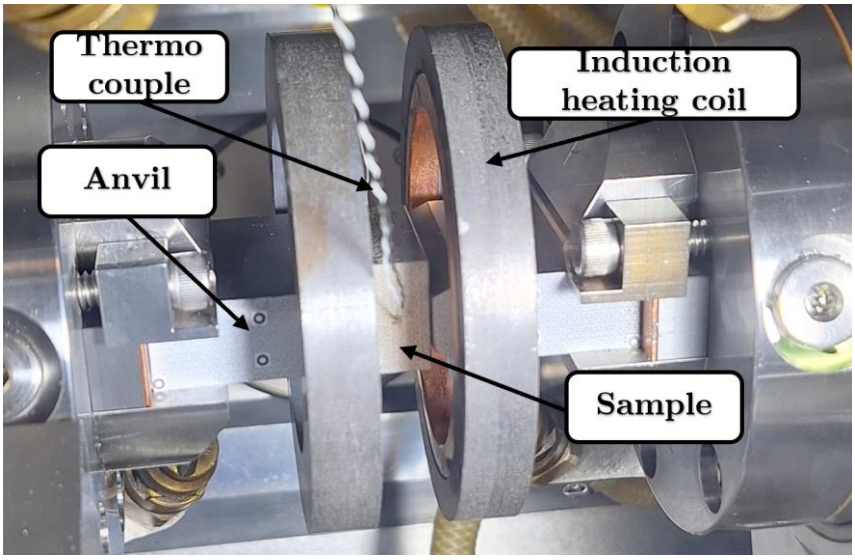



Figure 31: View of PSC being held by silicon nitride anvils surrounded by induction heating coil.

3. The *gui.py* program is run. The experiment's working folder is selected and the config file for the testing group is loaded. The test ID is entered, and the rest of the fields are confirmed to be correct.

4. Within the Reveal IR software, the thermal camera is configured to Falling Edge hardware triggering and the output directory is set to a file with the same name as the test ID. Check that the record mode is in radiometric temperature and that the frame rate is set to a value which exceeds the maximum frame rate that will be outputted from the DIC Capture system. The 'HDD' button is then clicked, indicating that all images will be recorded from this point onwards.

5. The 'Run Record Mode' button can then be pressed on the DIC Capture GUI. This will initialise the Arduino, DIC Cameras and notify the user when it is ready to run, as seen in Figure 32.

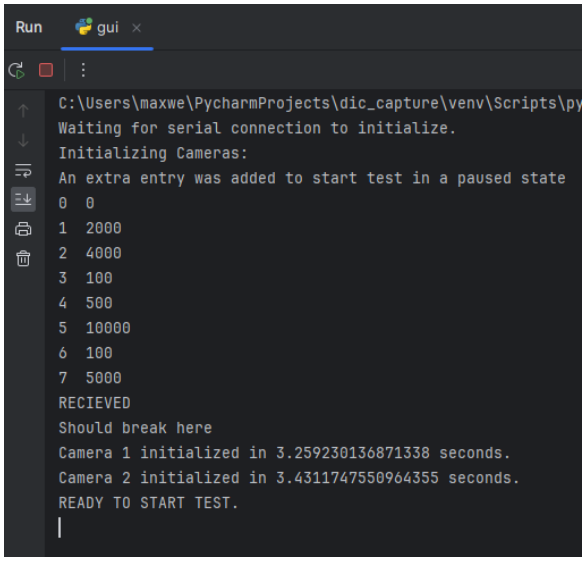



Figure 32: Pre-test consol output confirming test was initialized correctly.

6. The GSL program is then run as normal on the Gleeble. When the FPS change sequence is called in the GSL (see Figure 21) for the first time, DIC acquisition will start. The DIC images are displayed on the control computers screen as they are saved to storage. The thermal images are displayed on the Reveal IR software that is running on the computer, which is dedicated to the thermal camera.
7. At the end of the test, when the status LED on the Arduino is red, the DIC capture software is stopped and the sequence is stopped on Reveal IR. The

thermal images are then exported in .tiff format to the 'Camera 3' folder within the main test folder for the current test. The Gleeble output files can be manually transferred to the 'Raw data' folder; otherwise, this can be done when running the synchronisation program. Having a shared OneDrive folder on the DIC Capture, Gleeble and Reveal IR computer makes transferring these files to a central location slightly more efficient, but a Network Attached Storage (NAS) would be preferable due to the large files.

8. To prepare the thermal images, the *'telops_to_matchid_reformat.py'* is run. The user selects the test folder from which the images in the 'Camera 3' folder are assumed to be the images that were exported from Reveal IR. Figure 33 shows a single frame from the various cameras after being processed.

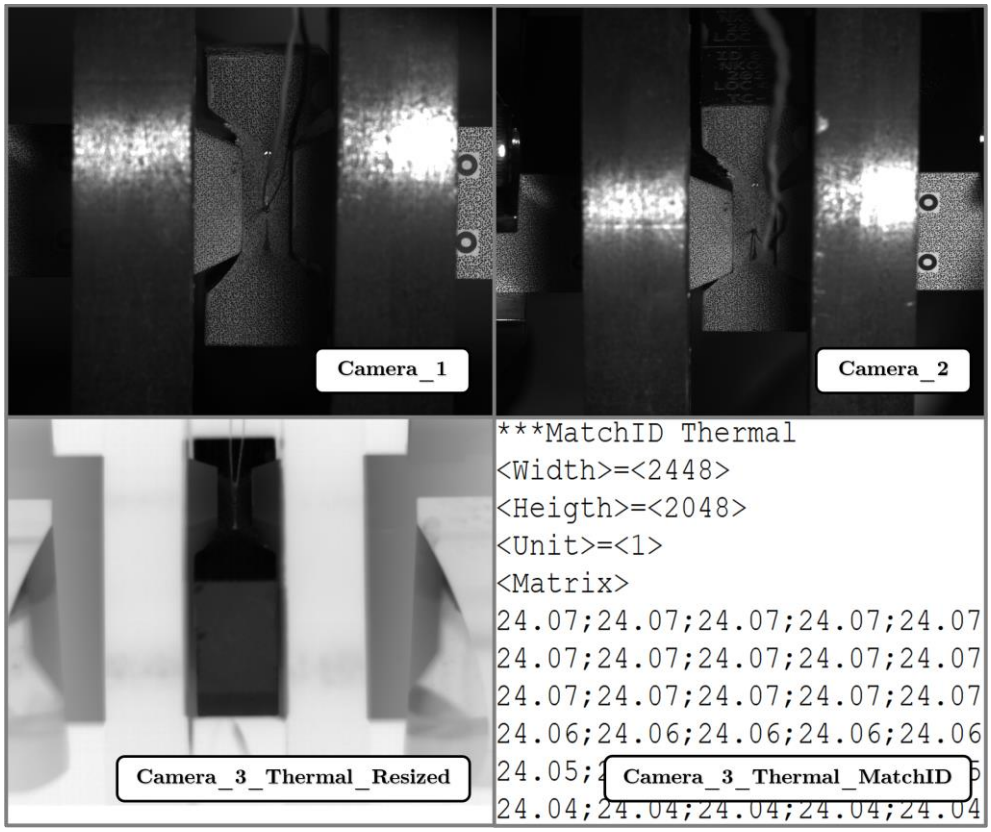



Figure 33: Comparison of the different camera outputs for the same frame number.

9. The '*post_processor_syncronization.py*' program is then run. Two .csv files are outputted; one is named '*SyncedGleebleData_testID*' and contains all the Gleeble and DIC data combined into one dataset with the Gleeble time used as reference. The other file is named *'MatchID_AcquisitionFile_testID'* and is formatted to be compatible with MatchID, this file only contains the data

associated with each image. These files are saved in the ‘Synced_Data’ folder within the main test folder.

10. At this stage, the Gleeble, DIC images, DIC ADC data and thermal images are all temporally synchronized and are formatted correctly to be used in MatchID or any open source DIC software provided that the user adheres to the formatting conventions of the software.

11. The test files are then transferred to the computer where MatchID is installed. Camera 1 and Camera 2 are first calibrated, followed by Camera 1 and Camera 3, Camera 3 is the thermal camera, see Figure 34. The images in the ‘Camera_3_Thermal_Resized’ folder are used for the calibration process as the calibration process requires both images to be in the same format and size.

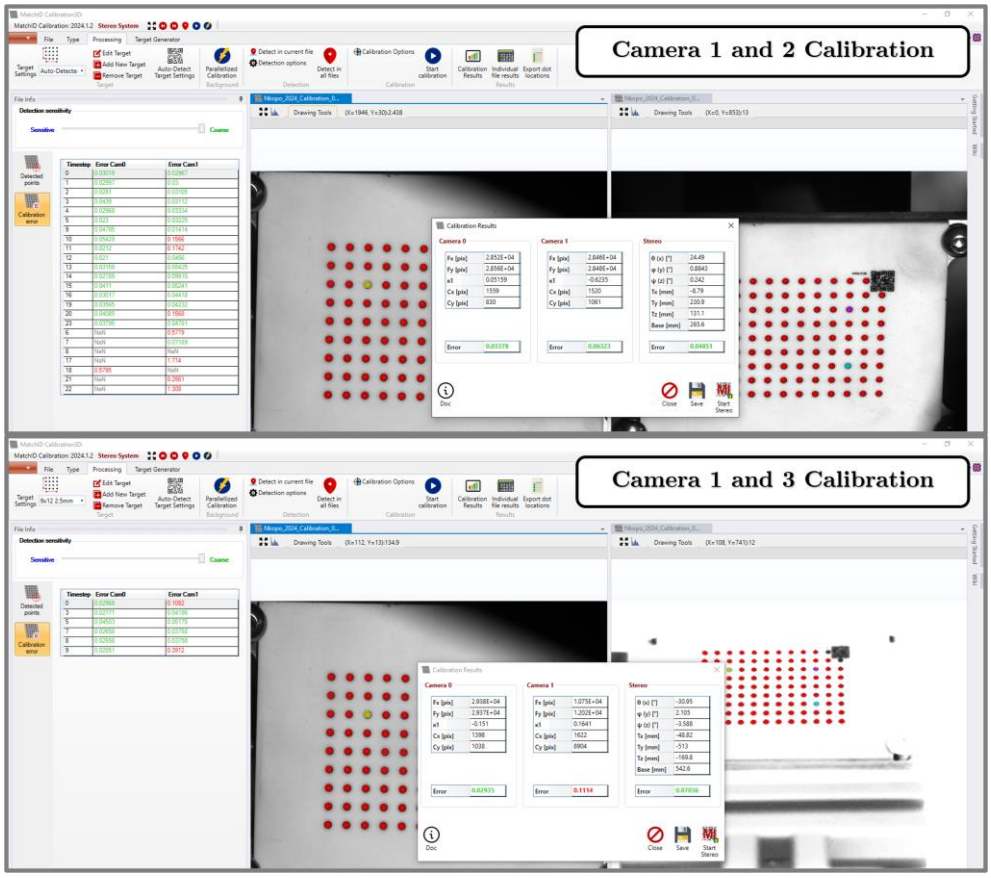



Figure 34: MatchID calibration of camera 1 and 2 (top), as well as camera 1 and 3 (bottom)

12. It is worth noting that for this test, the low resolution of the thermal camera produced a poor calibration. An alternative calibration approach, similar to what was used in Case Study 2 would probably be best, but this calibration was used

to demonstrate that it is compatible with the standard MatchID process. The thermal camera should be placed closer to the sample, or a larger calibration plate used, to improve the quality of the calibration.

13. MatchID stereo is then opened. Images from Cameras 1 and 2 are imported with a reference image set from camera 1. A region of interest is selected. Any other DIC pre-processing parameters are set by the user at this stage. To import the DAQ data for each image, click on the "Acquision Data" button and select the 'MatchID_AcquisitionFile_*testID*' within the current working test folder. Figure 35 shows an example of the preview of the data, which should be checked before proceeding.

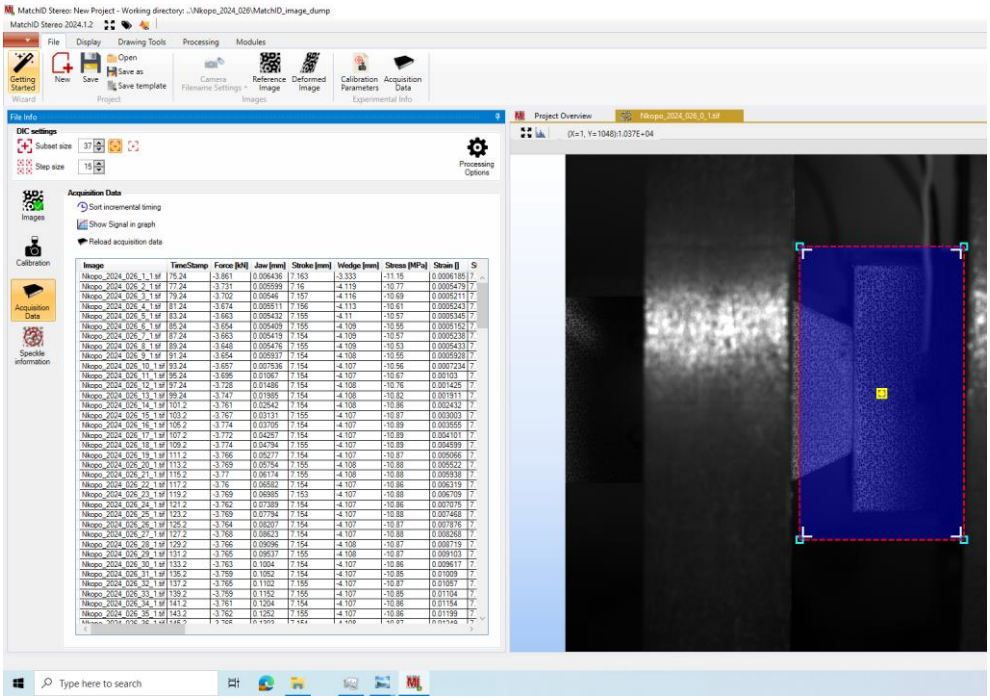


Figure 35: Preview of DAQ data generated by DIC Capture and imported into MatchID

14. After processing the DIC data, the "Temperature Import" module is run. The "Manual Upload" is used to select all the ".thermal.tiff images" in the dataset, see Figure 36. The calibration file for Cameras 1 and 3 is used for this stage when selecting "Calibration Parameters".

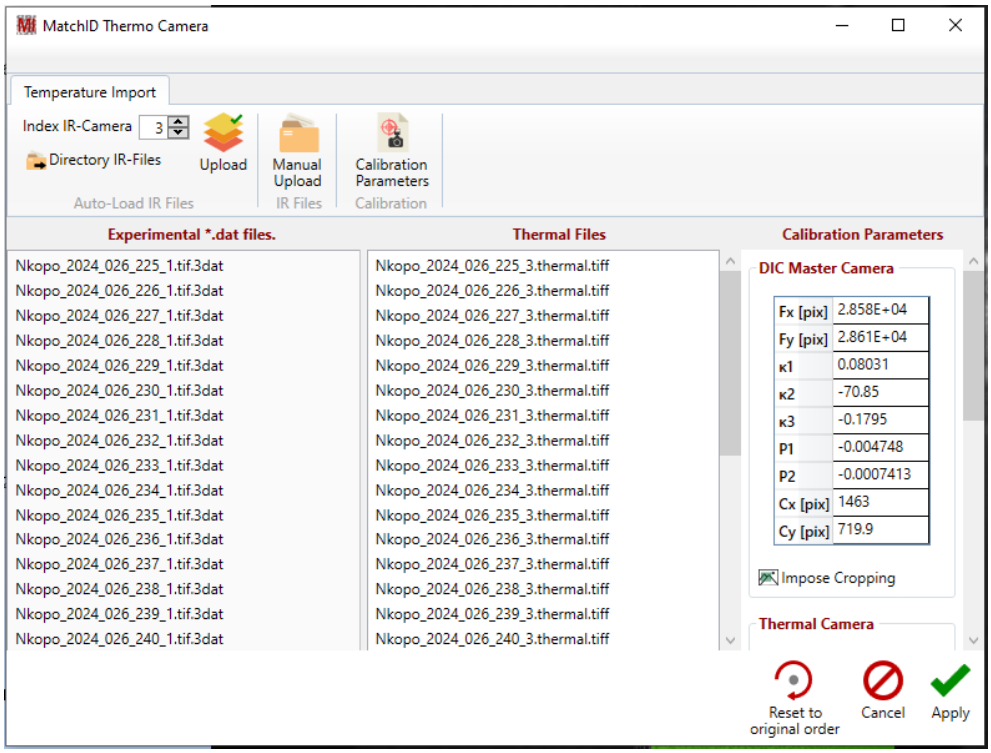



Figure 36: Import of thermal data into MatchID Temperature Import module.

15. The temperature data can then be overlayed on the DIC data. There seemed to be signs that the calibration was not sufficiently accurate for this test due to the low resolution of the thermal camera. This was not fixed as this data is simply a proof of compatibility and the ability to accurately spatially calibrate thermal data is better covered in Case Study 2.

16. A demonstration of correlation is shown in Figure 37, where the vertical displacement is shown during the quenching process. The thermocouple blocks the view of the right side of the sample and improving the correlation coverage falls outside of the scope of this project and will be part of future work.

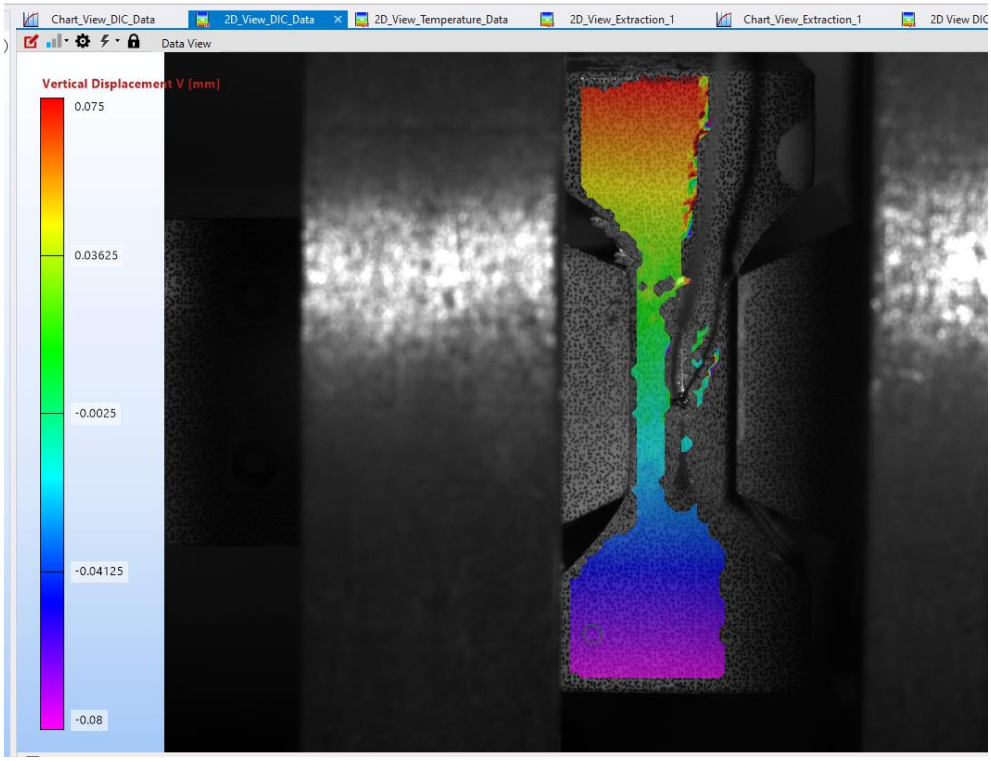



Figure 37: Vertical Displacement of PSC sample during quenching.

17. A virtual extensometer is placed on the vertical axis and plotted as a function of TC1 from the Gleeble data, see Figure 38. This produces a linear relationship which is expected as aluminium has a linear thermal expansion coefficient in these temperature ranges. Further analysis of this dataset is left as future work as, at this point, the DIC Capture system has been proven to be capable of its intended requirements.

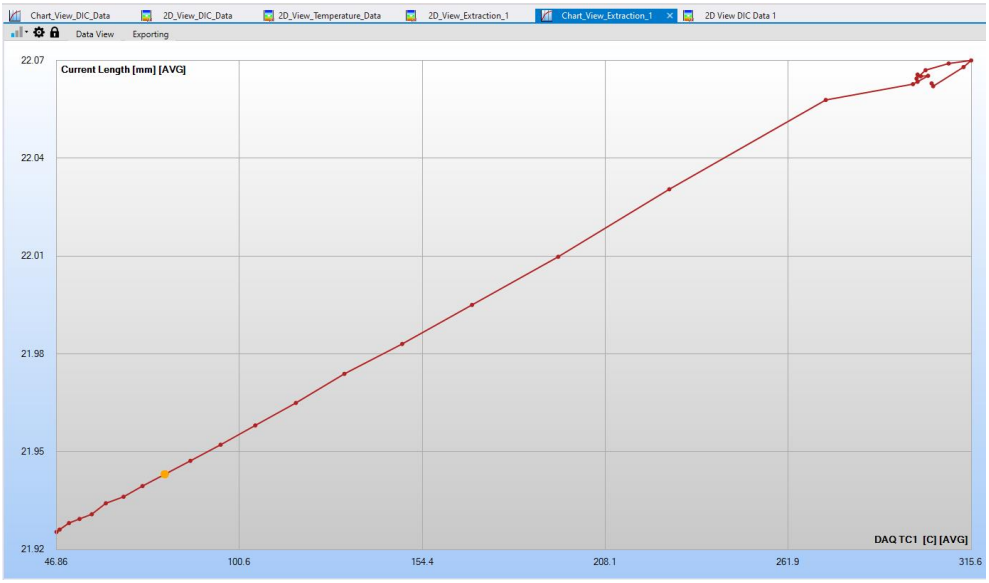



Figure 38: Vertical extension of sample vs temperature during quenching.

# 5 Discussion of overall findings

The real-world testing approach to developing the DIC Capture system was found to be extremely effective. The high rate of unexpected problems and required work arounds when integrating many different systems was the primary challenge of this project. The core methodologies of linking the systems using DIC Capture were determined through the development process of Version 1 and 2 and are listed as:

1. One system must be used as the reference system. The reference system must be capable of accurate time keeping with hardware interrupts, at least one analogue input, one digital output and a means to save or transmit basic data. The reference time of received and sent triggers is recorded for synchronisation.
2. To sync with a testing machines output data; the time of at least one, ideally two, events must be known relative to the machines output data. For testing machines that can provide an analogue voltage output for signals, the ADC on the reference system can be used as an alternative. Both sync pulses and ADC readings can be used if desired and is recommended for complex systems, such as the Gleeble 3800, as redundant synchronisation methods improve the robustness of the testing system.
3. Any Camera that can receive a hardware trigger can be used with the DIC Capture system. This is achieved by identifying the total count and time of triggers sent from the reference system and then post processing the images into a standardised format. This approach was chosen to ensure that any camera can

be used with the DIC Capture system in at least a basic, but functional, manner. One pitfall of this approach is that a single frame drop can potentially desynchronise the camera from the system. A mitigation technique used for this problem is to record the camera's local time with the image data where possible. If a discrepancy is noticed, the user can resynchronise the camera data. With this potential problem known, additional care was taken when writing the image saving algorithm in the DIC Capture Python software which has resulted in a reliable and versatile system.

4. Formatting and synchronisation of data is done after the test is completed, not during the test. The unprocessed recorded data is saved to storage as soon as possible. This greatly simplifies and improves the performance of the DIC Capture system.

With DIC Capture in an operational state and the system linking methodology established, future development of a DIC capture system is greatly simplified. Whilst factors such as cost, accessibility and reliability were important in the development of the DIC Capture system in this work; they were not explicitly optimized for, as simply getting the DIC Capture system operational and useful occupied most of the time allocation for this project. Optimising cost and accessibility by adding drivers for cheaper industrial camera brands and creating a commercially manufacturable PCB control board would be the most beneficial improvements to the DIC Capture system, especially for users new to DIC, at this point.

Version 3 of the DIC Capture system has proven itself to be a powerful research tool thus far, and is expected to be used in multiple research projects at the CME going forward.

# 6 Conclusions

The DIC Capture system successfully integrates the Gleeble 3800, Telops M350 thermal imaging camera and Match ID in a cost effective and simple manner that is versatile enough to be adapted to a wide variety of mechanical testing equipment in materials research applications.

The affordability of the DIC Capture system is achieved by using the Arduino Nano ESP32 as the reference timing system and connecting it to the ADS8584S ±10 V ADC. The DIC Capture timing accuracy is limited to the ±2 ppm timing accuracy of the ESP32

due to the control board being the reference system, this is sufficiently accurate for the majority of DIC tests.

The DIC Capture system was tested for reliability in Case Study 3, where it performed reliably and efficiently, with the only test interruptions being caused by external factors such as an unscheduled Windows OS update.

All source code for the project, along with circuit diagrams and bill of materials will be publicly available for free use from the GitHub repository:

- DIC Capture: https://github.com/MaxwellVos/dic_capture
- ADC controller: https://github.com/MaxwellVos/ADC-Triggering-System

. The DIC Capture system provides a solid foundation from which future users can build from.

The DIC Capture system meets the aims defined for this project and currently serves as a useful research tool within the CME.

# 7 References


[1] B. Pan, K. Qian, H. Xie and A. Asundi, “Two-dimensional digital image correlation for in-plane displacement and strain measurement: a review,” *Measurement Science and Technology,* vol. 20, no. 6, 2009.

[2] D. Atkinson and T. Becker, “ A 117 Line 2D Digital Image Correlation Code Written in MATLAB,” *Remote Sens.,* vol. 12, no. 2906, 2020.

[3] I. D. I. C. Society and E. a. I. M. Jones, A Good Practices Guide for Digital Image Correlation, 2018.

[4] V. Belloni, R. Ravanelli, A. Nascetti, M. Di Rita, D. Mattei and M. Crespi, “Digital Image Correlation from commercial to fos software: a mature technique for full-field displacement measurements,” *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences,* vol. XLII, no. 2, 2018.

[5] MatchID, “MatchID,” 2024. [Online]. Available: https://www.matchid.eu/en/about-us. [Accessed 20 April 2024].

[6] Dantec Dynamics, “Dantec Dynamics,” 12 May 2023. [Online]. Available: https://www.dantecdynamics.com/new-istra4d-v4-10-software-release/. [Accessed 20 April 2024].

[7] LaVision, “Calibration Plates,” 2024. [Online]. Available: https://www.lavision.de/en/applications/materials-testing/system-components/calibration-plates/index.php. [Accessed 20 April 2024].

[8] M. &. B. T. H. Van Rooyen, “High-temperature tensile property measurements using digital image correlation over a non-uniform temperature field.,” *The Journal of Strain Analysis for Engineering Design,* vol. 3, no. 53, p. 117–129, 2018.

[9] GOM, “ARAMIS 3D Camera,” Zeiss, 2024. [Online]. Available: https://www.gom.com/en/products/3d-testing/aramis-3d-camera. [Accessed 20 April 2024].

[10] Imetrum, “Mobius,” Imetrum, 2024. [Online]. Available: https://www.imetrum.com/products/mobius/. [Accessed 20 April 2024].

[11] CorreliSTC, “CorreliSTC,” CorreliSTC, 2024. [Online]. Available: https://www.correli-stc.com/. [Accessed 20 April 2024].

[12] K. Miikki, A. Karakoç, M. Rafiee, D. W. Lee, J. Vapaavuori, J. Tersteegen, L. Lemetti and J. Paltakari, “An open-source camera system for experimental measurements,” *SoftwareX,* vol. 14, no. 100688, 2021.

[13] J. C. A. de Deus Filho, L. C. da Silva Nunes and J. M. C. Xavier, “iCorrVision-2D: An integrated Python-based open-source Digital Image Correlation software for in-plane measurements (Part 1),” *SoftwareX,* vol. 19, no. 101131, 2022.

[14] J. C. A. de Deus Filho, L. C. da Silva Nunes and J. M. C. Xavier, “iCorrVision_3D,” Github, 2022.

[15] V. Belloni, R. Ravanelli, A. Nascetti, M. Di Rita, D. Mattei and M. Crespi, “py2DIC: A New Free and Open Source Software for,” *Sensors,* vol. 19, no. 18, 2019.

[16] J. Blaber, B. Adair and A. Antoniou, “Ncorr: Open-Source 2D Digital Image Correlation Matlab Software,” *Experimental Mechanics,* vol. 55, pp. 1105-1122, 2015.

[17] D. Turner, P. Crozier, P. Reu and USDOE, “Digital Image Correlation Engine v.3.0,” 2015.

[18] D. Solav, K. M. Moerman, A. M. Jaeger, K. Genovese and H. M. Herr, “MultiDIC: An Open-Source Toolbox for Multi-View 3D Digital Image Correlation,” *IEEE Access,* vol. 6, pp. 30520-30535, 2018.

[19] D. Solav and A. Silverstein, “DuoDIC: 3D Digital Image Correlation in MATLAB,” *Journal of Open Source Software,* vol. 7, no. 74, 2022.

[20] S. N. Olufsen, M. E. Andersen and E. Fagerholt, “µDIC: An open-source toolkit for digital image correlation,” *SoftwareX,* vol. 11, 2020.

[21] L. Yu and B. Pan, “Overview of High-temperature Deformation Measurement Using Digital Image Correlation,” *Experimental Mechanics,* vol. 61, pp. 1121-1142, 2021.

[22] C. Solutions, “Infrared (IR),” 2024. [Online]. Available: https://www.correlatedsolutions.com/specialized-systems-ref/infrared. [Accessed 18 May 2024].

[23] Gleeble, “Gleeble-3800,” Gleeble, 2024. [Online]. Available: https://gleeble.com/products/gleeble-systems/gleeble-3800.html. [Accessed 18 May 20024].

[24] Gleeble, “Measurement Systems,” DSI, 2024. [Online]. Available: https://gleeble.com/products/specialty-systems/measurement-systems.html. [Accessed 18 May 2024].

[25] T. W. Rob Toulson, Fast and Effective Embedded, 2nd ed., Oxford: Jonathan Simpson, 2017.

[26] Edumnd Optics, “Digital Camera Interfaces,” 2024. [Online]. Available: https://www.edmundoptics.com/knowledge-center/application-

notes/imaging/camera-types-and-interfaces-for-machine-vision-applications/. [Accessed 18 May 2024].

[27] Teledyne FLIR, “Understanding Buffer Handling,” 2024. [Online]. Available: https://www.flir.eu/support-center/iis/machine-vision/application-note/understanding-buffer-handling/. [Accessed 18 May 2024].

[28] Arduino, “Universal Asynchronous Receiver-Transmitter (UART),” 2024. [Online]. Available: https://docs.arduino.cc/learn/communication/uart/. [Accessed 18 May 2024].

[29] Electronics Notes, “BNC-Connector,” 2024. [Online]. Available: https://www.electronics-notes.com/articles/electronic_components/rf-connectors/bnc-connector.php. [Accessed 18 May 2024].

[30] Baumer, “VCXU-50M,” 2024. [Online]. Available: https://www.baumer.com/gb/en/p/23805. [Accessed 24 May 2024].

[31] Kashif, “Ceramic Resonator in Arduino,” 2022. [Online]. Available: https://linuxhint.com/ceramic-resonator-in-arduino/. [Accessed 16 November 2023].

[32] R. Borrageiro, “The relationship between the Zener-Hollomon parameter and transfer strip creep in hot rolled AA3104,” University of Cape Town, Cape Town, 2024.

[33] M. A. Blackwell, “High-temperature mechanical property characterisation of additively manufactured Inconel 718,” Stellenbosch , 2024.

[34] LaVision, “Calibration Plates,” 2024. [Online]. Available: https://www.smart-piv.com/en/products/strainmaster/system-components/calibration-plates/. [Accessed 23 May 2024].

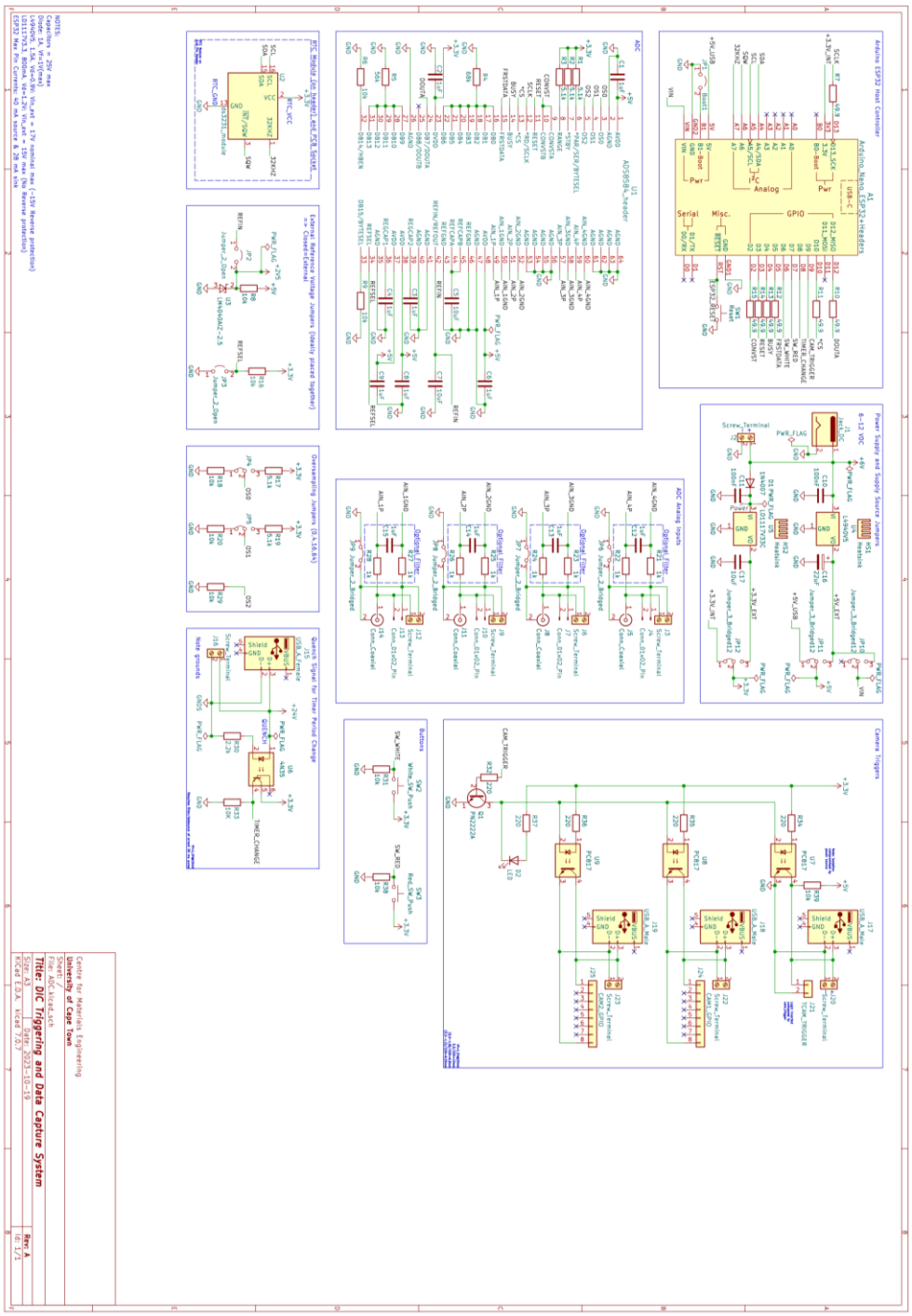
# 8 Appendix

## 8.1 Links to external resources

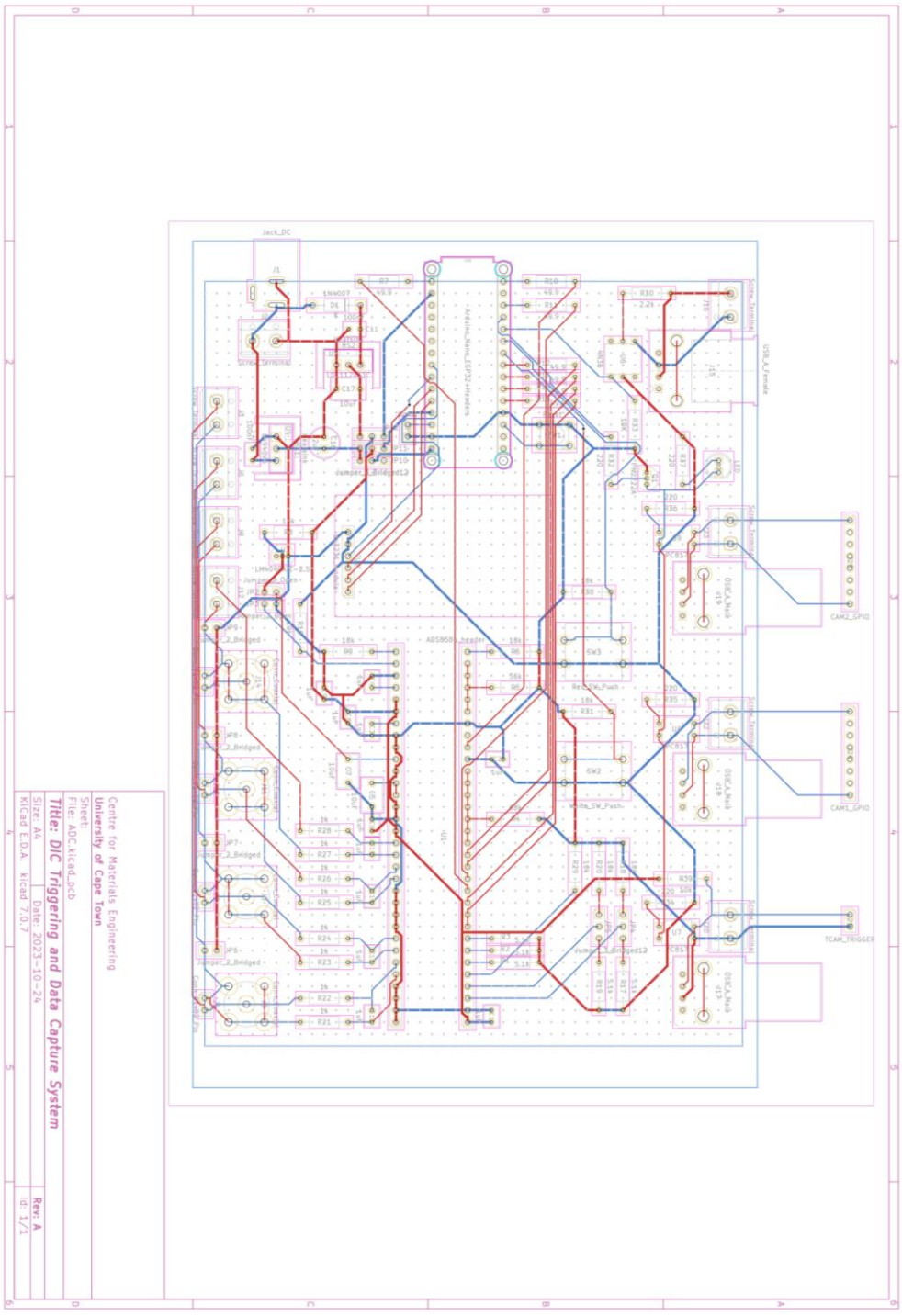
- DIC Capture: https://github.com/MaxwellVos/dic_capture
- ADC controller: https://github.com/MaxwellVos/ADC-Triggering-System

## 8.2 Circuit Diagram

Circuit diagram and PCB layout was made by David Dallas.



## 8.3 PCB layout for Veroboard



## 8.4 Example GSL code from each case study

### 8.4.1 Case Study 1 GSL

Example of the GSL used in test ID RB23_452_26 from the testing done for Roberto Borrangeiro.

```
set rampterm to 24000.0000pct
set ramiterm to 0.1000pct
set ramdterm to 120.0000pct
set forceKp to -0.0500pct
set forceKi to -0.0005pct
set maxangle to 110deg
// Heating...
// Hydrawedge Auto-load

// stress/strain setup...
set strainmode to 3
set strainsrc to Jaw.index
set strainX0 to 9.85mm
set strArea to 10.00mm*30.00mm
set planeA to 0.866
set planeB to 0.866
// end of stress/strain setup.

//Custom Variables
//==========================================(1/)
//Constants
set stroke_back_1 = 1mm
set wedge_forward_1 to 0.3mm
set stroke_back_2 = 2mm
set wedge_squish = 0.75mm
set hyd_interpass_force to -5.5kN
set step_inc to 0.001mm
set step_inc_small to 0.001mm
set step_interpass_inc to 0.004mm
set air_hold_time to 15sec


//Heat up and soak
set T_0 to 450C
set Tx_0 to 0.07mm

//Hit 1
set stroke_1 to 4.4309mm+1.0260mm+0.0000mm
set wedge_1 to 5.4191mm-9.8500mm
set compliance_1 to 0.345mm //+ve to make deformation larger, -ve to make deformation smaller
//Interpass 1
set interpass_time_1 to 276sec
set interpass_Temp_1 to 330C
set Tx_1 to 0.00mm

//Hit 2
set stroke_2 to 1.2804mm+0.0000mm+2.2859mm
set wedge_2 to 3.6627mm-9.8300mm
set compliance_2 to -0.133mm //+ve to make deformation larger, -ve to make deformation smaller
//Interpass 2
set interpass_time_2 to 576sec
set interpass_temp_2 to 300C
set Tx_2 to 0.00mm

//==========================================(/1)
```

```
acquire airrampres Force Jaw Strain Stress Stroke PTemp TC1 TC2 math PowAngle wedge PRam Pwedge
Quench3 Quench4

set mathK to 1
set math1Src to tc1.index
set math2Src to tc1.index
//set math2Src to tc2.index
set mathmode to msub

display math at 3

set TC1 to 0C
set tempmode to TC1.control

set h0 to strainX0
set stroke to 0cm
set wedge to 0cm
set heatbutton to on
set mechanical to on
delay 5sec
set hypress to on
delay 30sec

set lastruntime to systime

//=============================================(/0)
set quench4 to off // 0 fps
//=============================================(0/)

set airtc to on
set airram to on
delay 10sec
zero stroke
zero wedge
zero Jaw
set airForce = force
set wedgeHold to 0.5mm

//move wedge left until force is less than air force

set step_sum to 0mm
while force < (airForce+1.0kN)
        set step_sum to step_sum + step_inc_small
    ramp wedge to step_sum in 50msec
end

set step_sum to step_sum - wedgeHold
ramp wedge to step_sum in 1sec

ramp stroke to -step_sum in 1sec
set step_sum to 0mm
delay 100msec

while force > (airForce-1.5kN)
        set step_sum to step_sum + step_inc
    ramp stroke to -step_sum in 50msec
end

//=============================================(/1)
sample at 500Hz
delay 0.25sec
set quench4 to on // 0.5 fps
delay 0.25sec
sample at 50Hz
//=============================================(1/)
```

```
//intermediate heatup for force and seating of sample. Should be enough that sample has just barely deformed enough for full contact with anvils.

ramp stroke to -step_sum in 33sec &
ramp TC1 to 100.0000C in 33.0000sec

//Maintain full seating while reducing total deformation due to thermal expansion
set TC_Calc to (T_0 -3.0C)
ramp stroke to (-step_sum+Tx_0) in 133sec &
ramp TC1 to TC_Calc in 133sec
delay 2.7sec //dampen occelations
set TC1 to T_0

//=============================================(/2)
sample at 500Hz
delay 0.25sec
set quench4 to off // 1 fps
delay 0.25sec
sample at 50Hz
//=============================================(2/)

//Soak time at temperature before hit
delay 60.0000sec

//=============================================(/3)
sample at 500Hz
delay 0.25sec
set quench4 to on // 32 fps
delay 0.25sec
sample at 50Hz
//=============================================(3/)

delay 2sec //I suspect that there is a miniumum delay between quench triggers so adding 2 seconds for the faster hit stuff to try and compensate for this

zero stroke
zero wedge
delay 0.1sec

// Move into position for first deformation
//sample at 1000.0Hz
ramp stroke to Stroke_1 in 0.3sec
delay 0.1sec
ramp wedge to wedge_1+Tx_0+wedgeHold-compliance_1 in 0.3sec //wedge hold is compensated for because of Zeroing as zeroing wedge during loading procedure would drop the sample due to wierd 0.2mm hydraulic kick
delay 0.1sec

// final strain zero
zero Jaw
delay 0.1sec
delay 50.0000msec
set TC1 to 0C
sample at 50000.0Hz
delay 50.0000msec

ramp stroke to 4.4309mm in 0.0401sec

ramp stroke to 4.1410mm in 11.5000msec &
ramp stroke to 3.8596mm in 11.5000msec &
ramp stroke to 3.5864mm in 11.5000msec &
ramp stroke to 3.3214mm in 11.5000msec &
ramp stroke to 3.0641mm in 11.5000msec &
ramp stroke to 2.8144mm in 11.5000msec &
ramp stroke to 2.5720mm in 11.5000msec &
ramp stroke to 2.3368mm in 11.5000msec &
ramp stroke to 2.1085mm in 11.5000msec &
```

```
ramp stroke to 1.8869mm in 11.5000msec &
ramp stroke to 1.6719mm in 11.5000msec &
ramp stroke to 1.4632mm in 11.5000msec &
ramp stroke to 1.2606mm in 11.5000msec &
ramp stroke to 1.0640mm in 11.5000msec &
ramp stroke to 0.8731mm in 11.5000msec &
ramp stroke to 0.6879mm in 11.5000msec &
ramp stroke to 0.5082mm in 11.5000msec &
ramp stroke to 0.3337mm in 11.5000msec &
ramp stroke to 0.1643mm in 11.5000msec &
ramp stroke to 0.0mm in 11.5000msec &
// End segment
ramp stroke to -0.2816mm in 20.0000msec  // maxOverprogramTime=20msec

delay 10.0000msec

//INTERPASS 1
sample at 500.0000Hz
set wedge_save to wedge_1+Tx_0+wedgeHold-compliance_1-wedge_forward_1 //to simplify code
ramp stroke to stroke_back_1 in 0.1sec
ramp wedge to wedge_save in 0.1sec //air hold for post hit temp stabilisation
set TC_dip_store to T_0 - 10C //10C was chosen based on previous tests temp drop after hit
set TC1 to TC_dip_store

//=============================================(/4)
set quench4 to off // 1 fps
delay 0.25sec
sample at 10Hz
//=============================================(4/)
ramp TC1 to T_0 in 10sec //accounting for the temperature drop after the hit in an attempt to minimize
heating rates.

delay 2.0sec

sample at 50Hz

while force > hyd_interpass_force
        set wedge_save to wedge_save - step_interpass_inc
    ramp wedge to wedge_save in 25msec
end

ramp Force to -6.0kN in 2.0sec
delay 60sec //stress relaxation. 6kN compression for 15min

//=============================================(/5)
sample at 500Hz
delay 0.25sec
set quench4 to on // 0.2 fps
delay 0.25sec
sample at 10Hz
//=============================================(5/)

delay 840sec

//=============================================(/6)
sample at 500Hz
delay 0.25sec
set quench4 to off // 0 fps
delay 0.25sec
sample at 50Hz
//=============================================(6/)

zero stroke
ramp stroke to 0mm in 1sec //Wasnt happy when switching from force to stroke control and imediately
having to move. Not sure why but trying out zeroing and moving to zero before backing up for the air ram quench
hold
ramp stroke to 1mm in 1sec //air cools with air ram hold
```

```
// Cooling...
set TC1 to 0C
set quench1 to on
set quench2 to on
delay 60.0000sec
set quench1 to off
set quench2 to off
delay 60sec
// End cooling...


// Shut down...
sample off
set airram to off
set heatbutton to off
set tc1 to 0C
set hypress to off
set mechanical to off
set temptrim to 0C
delay 2sec
set wedgezero to wedgezero-1mm
```

### 8.4.2 Case Study 2 GSL

Example of the GSL used in test ID V30_T1 from the testing done with Matthew Blackwell.

```
ramp progStep to 1 in 0.0001sec &
// system setup...
set strokelimit to -150.00mm
set forcelimit to 20394kgf
set maxangle to 77deg
set forcelevelmax to 50000kgf
set forcelevelmin to 50000kgf
set limTC to TC1.control
set tempmode to TC1.control
set limTemp to 8001C
set ticoef to 1.0
set rampterm to 20000.0000pct
set ramiterm to 0.1000pct
set ramdterm to 200.0000pct
set forceKp to -0.0500pct
set forceKi to -0.0005pct

// end of system setup.

// stress/strain setup...
set strainmode to 1
set strainsrc to Stroke.index
set strainX0 to 40.00mm
set strVolume to 10.00mm*2.00mm*40.00mm
// end of stress/strain setup.

acquire airrampres chamberRead Force Force.line PowAngle PRam PTemp Quench4 Strain Stress Stroke
TC1 TC2 TC3 TC4
// store forcelevel variables until mechanical system is initialized
set forcelevelmax_store to forcelevelmax
set forcelevelmin_store to forcelevelmin
set forcelevelmax to 50000kgf
set forcelevelmin to 50000kgf
set heatbutton to on
set mechanical to on
```

```
delay 5sec
set hypress to on
delay 15sec
sync
// reset forcelevel variables after mechanical system is initialized
set forcelevelmax to forcelevelmax_store
set forcelevelmin to forcelevelmin_store

set lastruntime to systime

//Always make sure that air ram tension is engaged

delay 45sec

//set air ram to 1.0kN TENSION when loading sample before test
//loading force is the threshold force which detemines when the hydraulics engage the load train
set loading_force to 1.0kN
//set max_loading_force to 3kN
set mid_elastic_force to 2.5kN
set stroke_save to 0mm

//=============================================(/0)
set quench4 to off // 0 fps
delay 1sec
//=============================================(0/)

zero stroke
sample at 20.0Hz
delay 1sec

//Heatup and thermal soak.
//=============================================(/1)
sample at 500Hz
delay 0.25sec
set quench4 to on // 0.125 fps
delay 0.25sec
sample at 10Hz
//=============================================(1/)

ramp stroke to 0.0000mm in 45.0000sec &
ramp TC1 to 450.0000C in 45.0000sec

//Thermal soak
delay 600sec

//Prepping for deformation.
//=============================================(/2)
sample at 500Hz
delay 0.25sec
set quench4 to off // 0.125 fps
delay 0.25sec
sample at 10Hz
//=============================================(2/)

//essentially determining where the hydraulics are so that the slow strain doesnt take forever
while force < loading_force
        set stroke_save to stroke_save + 0.001mm
    ramp stroke to stroke_save in 50msec
end

//unloads the sample just enough so that the test starts from the air ram force
set stroke_save to stroke_save - 0.15mm
ramp stroke to stroke_save in 1sec


//Deformation start.
//=============================================(/3)
```

```
sample at 500Hz
delay 0.25sec
set quench4 to on // 0.125 fps
delay 0.25sec
sample at 10Hz
//=============================================(3/)
set safety_travel to 0mm
while (force < mid_elastic_force) .and. (TC1 >  350C) .and. (TC1 < 1200C)
        set stroke_save to stroke_save + 0.001mm
        set safety_travel to safety_travel + 0.001mm
    ramp stroke to stroke_save in 300msec
        if safety_travel > 5mm
                        set mid_elastic_force to 0kN
        end
end

//after going through most of the elastic region, carries on at the slow strain rate
//safety stroke is set to stop while loop if the sample doesnt break or there is an error with the thermocouples
set safety_stroke to stroke_save + 12.5mm
//If the sample breaks, the temperature will drop which will break the while loop

while (stroke_save < safety_stroke) .and. (TC1 < 1200C) .and. (TC1 > 350C)
        //ramping stroke slowly at 0.2mm/min
        set stroke_save to stroke_save + 0.1mm
        ramp stroke to stroke_save in 30sec
        sync
end

//delay 60sec
delay 5sec

//set stroke_save to stroke_save - 0.1mm
//ramp stroke to stroke_save in 1sec

//Deformation end.
//=============================================(/4)
sample at 500Hz
delay 0.25sec
set quench4 to off // 0 fps
delay 0.25sec
sample at 10Hz
//=============================================(4/)
sync
delay 1sec
set TC1 to 0C
delay 15sec

delay 1sec
sync
sample off
set heatbutton to off
set tc1 to 0C
set temptrim to 0C
set hypress to off
set mechanical to off
```

### 8.4.3 Case Study 3 GSL

Example of the GSL used in test ID Nkopo_2024_Group_J_040 from the testing done for Nkopo Chaole

```
// Author and operator: Maxwell Vos (2024)
// Multi-hit PSC for AA3104 with DIC and thermal imaging
```

```
// Part of Nkopo 2024 testing supervised by Dr Sarah George (UCT)
// Set AirRam force to 2.5kN compression
// Using Induction Heating
// Operate with extreme caution.

//Press the Auto button on the booster pump after powering up the Gleeble for the first time
//Changes to timer mode on induction system.
//If timer not set to 9999sec or counter mode, induction will not heat.
//Trigger is defined in the custom setup section and simply sets the induction timer to 9999 seconds at the start of each test.

//custom_setup.gsl should include:
//---------------------------------
//      display DIC.trigger at 11
//      display trigger at 1
//      set Induction.debug to on
//      event testfunction if trigger
//          go Indcom_com::send "1,timer,9999"
//          say "Make sure test is shorter than 9999seconds"
//          go com_delay
//          sync
//          delay 500msec
//                  sync
//                  set trigger to off
//      end
//---------------------------------

set servoMode to 3
set RunTimeInhibit to on

// stress/strain setup...
set strainmode to 3
set strainsrc to Jaw.index
set strainX0 to 12.00mm
set strArea to 10.00mm*30.00mm
set planeA to 0.866
set planeB to 0.866
set strain.blanking to 0kN
// end of stress/strain setup.

//Custom Variables
//Constants
set wedgeHold to 0.5mm

//Heat up and soak
set T_0 to 360C

//Hit 1
set stroke_1 to 4.2173mm+2.9371mm
set wedge_1 to 7.7827mm-12.0000mm
set compliance_1 to 0.39mm //+ve to make deformation larger, -ve to make deformation smaller
//Interpass 1
set interpass_time_1 to 20sec
//set interpass_Temp_1 to 330C

//Hit 2
set stroke_2 to 3.0534mm+0.0000mm+2.9028mm
set wedge_2 to 4.8799mm-12.0000mm
set compliance_2 to 0.39mm + 0.12mm //+ve to make deformation larger, -ve to make deformation smaller
//Interpass 2
set interpass_time_2 to 499sec
//set interpass_temp_2 to 300C

//Hit 2
set stroke_3 to 3.6796mm+0.0000mm+2.4307mm
set wedge_3 to 2.4493mm-12.0000mm
```

```
	set compliance_3 to 0.39mm + 0.12mm + 0.23mm //+ve to make deformation larger, -ve to make
deformation smaller
	//Interpass 2



	//Can acquire a maximum of 16 variables
	acquire Force Jaw Stroke wedge Stress Strain Stroke.setpoint Wedge.setpoint Wedge.abs.filter
Stroke.abs.filter TC1 TC2 TC1.setpoint HeatPower Quench1 DIC.trigger

	set TC1 to 0C
	set tempmode to TC1.control

	set h0 to strainX0
	set stroke to 0cm
	set wedge to 0cm
	set heatbutton to on
	wait thermal.ready
	set mechanical to on
	wait mechanical.ready
	delay 5sec
	set hypress to on
	delay 30sec
	wait mechanical.ready

	set airtc to on
	set airram to on
	delay 10sec
	zero Jaw
	set airForce = force

	delay 0.1sec
	sync
	set wedge to 0mm
	set stroke to 0mm
	sync

	set lastruntime to systime
	sample at 10Hz
	delay 0.1sec
	sync

	//Loading procedure using Maxwells PSC loading system
	repeat 1
		//move wedge left until force is less than air force
		//force check of 8kN is just a safety precaution
		//rough movement of the wedge
		while (Wedge.abs.filter < 0mm) .and. (force > -8kN)
			set wedge.zero to wedge.zero-0.02mm
		  delay 50msec
			sync
		end
		delay 0.5sec
		sync
		//fine movement of wedge to absolute position
		while (Wedge.abs.filter < 0.2mm) .and. (force > -8kN)
			set wedge.zero to wedge.zero-0.005mm
		  delay 50msec
			sync
		end
		//fine movement of wedge to specified load
		while force < (airForce+1.0kN)
			//note negative is because we are using the zero to move so the direction is inversed
		  set wedge.zero to wedge.zero-0.001mm
		  delay 50msec
			sync
		end
```

```
        //reduces chances of dropping sample. Wedge hold must be compensated for as is effectively changes
the value of the stroke zero
        ramp wedge to -wedgeHold in 1sec

        //rough movement of the stroke to absolute position
        while (Stroke.abs.filter > 2mm) .and. (force > -8kN)
                set stroke.zero to stroke.zero+0.2mm
           delay 50msec
                sync
        end
        delay 1sec
        sync
        //fine movement of stroke to absolute position
        while (Stroke.abs.filter > 0.2mm) .and. (force > -8kN)
                set stroke.zero to stroke.zero+0.005mm
           delay 50msec
                sync
        end
        //fine movement of stroke until specified load
        while force > (airForce-1.5kN)
                //note positive is because we are using the zero to move so the direction is inversed
           set stroke.zero to stroke.zero+0.001mm
           delay 50msec
                sync
        end
        sync
end

//=============================================(/0)
// Start of heating
// Frame Period: 2000ms
repeat 1
        sample at 1000Hz
        sync
        delay 100msec
        sync
        set DIC.trigger to on
        sync
        delay 130msec
        sync
        sample at 20Hz
        sync
end
//=============================================(0/)

say acquire "TC1.kp is %TC1.kp"
say acquire "TC1.ki is %TC1.ki"
say acquire "stroke.kp is %stroke.kp"
say acquire "stroke.ki is %stroke.ki"
say acquire "stroke.kd is %stroke.kd"
say acquire "AirRam force set to 2.5kN"

set Trigger to on

//move everything into position for hit_1 befor heating
ramp stroke to Stroke_1 in 1sec
delay 0.1sec
sync
ramp wedge to wedge_1+wedgeHold-compliance_1 in 1sec
delay 0.1sec
sync

//heating to soak temperature
set TC_Calc_1 to (T_0 -10.0C)
set TC_Calc_2 to (T_0 -20.0C)
ramp TC1 to TC_Calc_2 in 120.0000sec
```

```
sync
ramp TC1 to TC_Calc_1 in 10.0000sec
sync
ramp TC1 to T_0 in 20sec
sync

//=============================================(/1)
// Start of soak
// Frame Period: 4000ms
repeat 1
        sample at 1000Hz
        sync
        delay 100msec
        sync
        set DIC.trigger to on
        sync
        delay 130msec
        sync
        sample at 20Hz
        sync
end
//=============================================(1/)

//Soak time at temperature before hit
delay 90.0000sec //soak time
sync

//make sure that stroke and wedge are in the correct locations
ramp stroke to Stroke_1 in 1sec
delay 0.1sec
sync
ramp wedge to wedge_1+wedgeHold-compliance_1 in 1sec
delay 0.1sec
sync

//=============================================(/2)
// Start of Hit 1
// Frame Period: 100ms
repeat 1
        sample at 1000Hz
        sync
        delay 100msec
        sync
        set DIC.trigger to on
        sync
        delay 130msec
        sync
        sample at 20Hz
        sync
end
//=============================================(2/)

// final strain zero
zero Jaw
delay 0.1sec
delay 50.0000msec
set TC1 to 0C
sample at 50000.0Hz
delay 50.0000msec

ramp stroke to 4.2173mm in 0.0087sec


ramp stroke to 3.9603mm in 0.6250msec &
ramp stroke to 3.7088mm in 0.6250msec &
ramp stroke to 3.4626mm in 0.6250msec &
ramp stroke to 3.2218mm in 0.6250msec &
```

```
ramp stroke to 2.9861mm in 0.6250msec &
ramp stroke to 2.7555mm in 0.6250msec &
ramp stroke to 2.5298mm in 0.6250msec &
ramp stroke to 2.3089mm in 0.6250msec &
ramp stroke to 2.0928mm in 0.6250msec &
ramp stroke to 1.8813mm in 0.6250msec &
ramp stroke to 1.6743mm in 0.6250msec &
ramp stroke to 1.4717mm in 0.6250msec &
ramp stroke to 1.2735mm in 0.6250msec &
ramp stroke to 1.0796mm in 0.6250msec &
ramp stroke to 0.8898mm in 0.6250msec &
ramp stroke to 0.7040mm in 0.6250msec &
ramp stroke to 0.5223mm in 0.6250msec &
ramp stroke to 0.3444mm in 0.6250msec &
ramp stroke to 0.1703mm in 0.6250msec &
ramp stroke to 0.0mm in 0.6250msec &
// End segment
ramp stroke to -4.0000mm in 14.8371msec   // maxOverprogramTime=20msec

delay 10.0000msec

sample at 10.0000Hz
ramp stroke to stroke_2 in 0.5sec
delay 0.1sec
sync
ramp wedge to wedge_2+wedgeHold-compliance_2 in 0.5sec
sync

//==========================================(/3)
// End of Hit 1 and post hit heating
// Frame Period: 500ms
repeat 1
        sample at 1000Hz
        sync
        delay 100msec
        sync
        set DIC.trigger to on
        sync
        delay 130msec
        sync
        sample at 20Hz
        sync
end
//==========================================(3/)

set TC1 to 360C
delay 0.1sec
sync
ramp TC1 to 345C in 4.4sec
ramp TC1 to 335C in 6.03sec
ramp TC1 to 330C in 7.6sec

//==========================================(/4)
// End of cool down period
// Frame Period: 10 000ms
repeat 1
        sample at 1000Hz
        sync
        delay 100msec
        sync
        set DIC.trigger to on
        sync
        delay 130msec
        sync
        sample at 20Hz
        sync
end
```

```
//=============================================(4/)

//Interpass 1
set airram to off
delay interpass_time_1
sync
set airram to on
delay 1sec
sync

//=============================================(/5)
// Hit 2
// Frame Period: 100ms
repeat 1
        sample at 1000Hz
        sync
        delay 100msec
        sync
        set DIC.trigger to on
        sync
        delay 130msec
        sync
        sample at 20Hz
        sync
end
//=============================================(5/)

//HIT 2

delay 50.0000msec
set TC1 to 0C
sample at 50000.0Hz
delay 50.0000msec

ramp stroke to 2.9028mm in 0.0090sec


ramp stroke to 2.7233mm in 0.3850msec &
ramp stroke to 2.5479mm in 0.3850msec &
ramp stroke to 2.3765mm in 0.3850msec &
ramp stroke to 2.2091mm in 0.3850msec &
ramp stroke to 2.0456mm in 0.3850msec &
ramp stroke to 1.8858mm in 0.3850msec &
ramp stroke to 1.7297mm in 0.3850msec &
ramp stroke to 1.5773mm in 0.3850msec &
ramp stroke to 1.4283mm in 0.3850msec &
ramp stroke to 1.2828mm in 0.3850msec &
ramp stroke to 1.1406mm in 0.3850msec &
ramp stroke to 1.0017mm in 0.3850msec &
ramp stroke to 0.8661mm in 0.3850msec &
ramp stroke to 0.7335mm in 0.3850msec &
ramp stroke to 0.6040mm in 0.3850msec &
ramp stroke to 0.4775mm in 0.3850msec &
ramp stroke to 0.3539mm in 0.3850msec &
ramp stroke to 0.2332mm in 0.3850msec &
ramp stroke to 0.1152mm in 0.3850msec &
ramp stroke to 0.0000mm in 0.3850msec &
// End segment
ramp stroke to -4.0000mm in 13.5216msec  // maxOverprogramTime=20msec

delay 10.0000msec
sync



sample at 10.0000Hz
ramp stroke to stroke_3 in 0.5sec
```

```
sync
ramp wedge to wedge_3+wedgeHold-compliance_3 in 0.5sec
sync

//=============================================(/6)
// Cooling to interpass 2
// Frame Period: 10000ms
repeat 1
        sample at 1000Hz
        sync
        delay 100msec
        sync
        set DIC.trigger to on
        sync
        delay 130msec
        sync
        sample at 20Hz
        sync
end
//=============================================(6/)


//Cool to interpass 2 temp in as close to 40 seconds as possible
set TC1 to 330C
delay 0.1sec
sync
ramp TC1 to 315C in 8sec
ramp TC1 to 305C in 12sec
ramp TC1 to 300C in 20sec
sync


set airram to off
delay interpass_time_2
sync
set airram to on
delay 0.1sec
sync

//=============================================(/7)
// Hit 3
// Frame Period: 100ms
repeat 1
        sample at 1000Hz
        sync
        delay 100msec
        sync
        set DIC.trigger to on
        sync
        delay 130msec
        sync
        sample at 20Hz
        sync
end
//=============================================(7/)

delay 50.0000msec
set TC1 to 0C
sample at 50000.0Hz
delay 50.0000msec

ramp stroke to 2.4370mm in 0.0114sec


ramp stroke to 2.2710mm in 0.2667msec &
ramp stroke to 2.1107mm in 0.2667msec &
ramp stroke to 1.9558mm in 0.2667msec &
```

```
ramp stroke to 1.8061mm in 0.2667msec &
ramp stroke to 1.6616mm in 0.2667msec &
ramp stroke to 1.5220mm in 0.2667msec &
ramp stroke to 1.3872mm in 0.2667msec &
ramp stroke to 1.2569mm in 0.2667msec &
ramp stroke to 1.1311mm in 0.2667msec &
ramp stroke to 1.0095mm in 0.2667msec &
ramp stroke to 0.8921mm in 0.2667msec &
ramp stroke to 0.7787mm in 0.2667msec &
ramp stroke to 0.6692mm in 0.2667msec &
ramp stroke to 0.5634mm in 0.2667msec &
ramp stroke to 0.4612mm in 0.2667msec &
ramp stroke to 0.3624mm in 0.2667msec &
ramp stroke to 0.2671mm in 0.2667msec &
ramp stroke to 0.1749mm in 0.2667msec &
ramp stroke to 0.0860mm in 0.2667msec &
ramp stroke to 0.0mm in 0.2667msec &
// End segment
ramp stroke to -4.0000mm in 12.6268msec  // maxOverprogramDist

delay 100msec
sync

sample at 10.0000Hz
ramp stroke to stroke_3 in 0.5sec
sync
ramp wedge to wedge_3+wedgeHold-compliance_3 in 0.5sec
sync

//=============================================(/8)
// Start of Quench
// Frame Period: 2000ms
repeat 1
        sample at 1000Hz
        sync
        delay 100msec
        sync
        set DIC.trigger to on
        sync
        delay 130msec
        sync
        sample at 20Hz
        sync
end
//=============================================(8/)


// Cooling...
set TC1 to 0C
set quench1 to on
set quench2 to on
delay 90.0000sec
sync
set quench1 to off
set quench2 to off
delay 1sec
// End cooling...

//=============================================(/9)
// End of Quench
// Frame Period: END TEST
repeat 1
        sample at 1000Hz
        sync
        delay 100msec
        sync
        set DIC.trigger to on
```

```
        sync
        delay 130msec
        sync
        sample at 20Hz
        sync
end
//=============================================(9/)
delay 1sec
sync
sample off

//reposition for next test
repeat 1
        //move wedge out of the way
        ramp wedge to -2mm in 3sec
        sync
        //move stroke to position where sample can fit through induction coils
        ramp stroke to 5mm in 3sec
        sync
        //set air ram to tension to track strok position
        set airtc to off
        delay 1sec
        sync
        //turn air ram off so that stroke can move independantly
        set airram to off
        delay 1sec
        sync
        //move stroke further back to give an air ram buffer for when loading the sample with wedge
        ramp stroke to 10mm in 3sec
        //put air ram into compression mode for loading next sample
        set airtc to on
end


// Shut down...
set heatbutton to off
set tc1 to 0C
set hypress to off
set mechanical to off
set temptrim to 0C
delay 2sec
```