Engineering Physics, The Hague University of Applied Sciences, Delft

Remote-controlled fabrication of 2D heterostructuren

December 20, 2022

Report by:
B. Klein Ikkink

Supervised by:
S. Bhattacharyya, LION
F. Galli, LION

Foreword

Add the foreword

Abstract

Create th
abstract

Contents

Used abbreviations

pagina nummer-ing pas hier be-ginnen, voor-gaande in romeinse nummer-ing

- LION : Leiden Institute Of Physics
- VDWSF: Van Der Waals Stacking Facility
- FMD : Fine Mechanical Department
- ELD : Electronics Department
- TUI : Terminal User Interface (differend from CLI in the sense that the user doesn't directly interact with the command line)
- GUI : Graphical User Interface
- SDK : Software Development Kit
- DAS : Domain Applied Science

1 Introduction

Rephrase the text so it isn't flagged a plagiaat

The discovery of graphene in 2004, awarded the Nobel Prize in Physics in 2010, has led to a completely new field of research based on 'atomically thin materials', materials of a few atomic layers thick (2D materials). Recently, the focus of the research area has shifted to fabricating completely new structures by stacking different atomically thin materials 'Van Der Waals Heterostructures'. Atomic thin materials can consist of metals, semiconductors, insulators, ferromagnets, superconductors, etc., combining these materials is a route to developing useful devices [Geim and Grigorieva].

Recently, a new research group was started under Principle Investigator Dr. Semonti Bhattacharyya who is an expert in fabricating and characterizing Van Der Waals Heterostructures and 2D materials and who recently started her new laboratory at LION, the Van Der Waals Stacking Facility (VDWSF). This project will focus on the efficient and accurate fabrication and imaging of the Van Der Waals Heterostructures and will be carried out by the Van Der Waals stacking lab at the Leiden Institute of Physics (LION). LION is one of the oldest physics research institutes in the Netherlands. The research performed at Leiden Institute of Physics covers a broad spectrum of topics, from cosmology to condensed matter physics, from quantum computers to new systems such as Van Der Waals Heterostacks. Research at LION is largely fundamental and curiosity-driven but also with an eye toward societal value.

This project will take place at the Leiden Institute of Physics (LION), which is one of the oldest physics institutes in the Netherlands. Research at the Leiden Institute of Physics covers a broad spectrum of subjects, ranging from cosmology to condensed matter physics, from the physics of jamming in granular materials to quantum computation, from protein folding to superconductivity and novel systems like van der Waals materials. Recently a new research group has been appointed to Semonti Bhattacharyya who is an expert in fabrication and characterisation of van der Waals and 2D materials and who has recently started to setup her new laboratory at LION, the Van Der Waals Stacking Facility (VDWSF). This project will focus on the efficient and accurate fabrication and imaging of the Van Der Waals Heterostructures with adaptability in mind to make the instrument more flexible and fit future research projects.

2 Project definition

Presently a commercial setup for fabricating 2D Hetero-stacks is used in the Van Der Waals Stacking Facility (VDWS) at lion. This commercial setup is prone to vibrations and has an inferior-quality microscope, wich makes it difficult to clearly identify small 2D material. Furthermore, the laboratory has a vibration level slightly above VC-C [Isolation]. For these reasons a new setup will be developed that should be less sensitive to mechanical vibrations. This setup is focused on improved rigidity, accuracy and flexibility compared to the commercially available and previous versions. The setup will be based on a commercial (Olympus BXFM) optical microscope, but to reach this goal a custom microscope stand, sample holder, mask holder and software have to be developed.

2.1 Research question

What components and control techniques should be used to make the stacking setup as least susceptible to vibration as possible and controllable at the sub-micrometer scale while producing clear images with an optical microscope?

2.2 Research goal

Building a stacking setup that is as least susceptible to vibration as possible where the base x-y-z stages move with resolution on the micrometer scale and the mask x-y-z stages on the nanometer scale, which can be remotely controlled, while being able to capture clear images with an optical microscope.

3 Background

Over the past decade, research into two-dimensional (2D) materials and van der Waals heterostructures has opened up a world of possibilities for the development of new technologies. 2D materials, such as graphene and hexagonal boron nitride, are atomically thin and have unique electrical, optical, and mechanical properties that make them interesting for a variety of applications.

3.1 van der Waals Heterostuctures

citaat invoegen

aFBEELD VAN GRAFEEN EN HEXAGONAAL BOORNITRIDE INVOEGEN

Van der Waals heterostructures are composed of two or more 2D materials that are stacked together and can be used to create electronic and optoelectronic nano devices. This research has enabled the development of a wide range of technologies, including flexible electronics, high-speed transistors, and ultra-sensitive sensors. The promise of 2D materials and van der Waals heterostructures is that they can be used to create devices that are faster, more efficient, and more reliable than existing technologies [Geim and Grigorieva].

3.2 Fabrication of van der Waals Heterostructures: Stamping

The stamping process is a method used to fabricate van der Waals heterostructures. This process involves the transfer of 2D crystals onto a substrate by pressing them together with a stamp so the van der Waals forces between the layers become strong enough to hold the layers together.

The general stamping procedure is as follows:

1. The 2D material flakes are plased on top of the substrate
2. The operator looks for suitable flakes to use for the stamping
3. The operator places the stamp on top of the 2D material flake and changes the sample temperature to stick the flake to the stamp
4. The stamp with the flake is positioned above the second 2D material flake on the substrate
5. The operator places the stamp on top of the second 2D material flake and changes the sample temperature to release the flake from the stamp
6. The operator finds a new suitable flake to place on top of the created stack and repeats the process

This process consists of serval time consuming steps, such as finding suitable flakes, placing the flakes on the stamp and substrate, and changing the sample temperature. The aim of this project is to automate the process of stamping to reduce the time needed to fabricate a stack and to increase the accuracy of the process.

3.3 Vibration isolation

Bij het ontwerpen van de stand van de microscoop moet sterk rekening gehouden worden met de rigiditeit en eigenfrequensie van de opstelling. Bij commercieel verkrijgkbare stacking opstellingen is een faseverschil tussen trilling door het monster, masker en camera een bekend en veel voorkomend probleem, dit heeft als effect dat de uitlijning van flakes en het stapelen bemoeilijkt worden en de kans op beschadigen van de flake aanzienlijk groter is.

Stuk invoegen over verandering in fase plot (bode) bij verschillend

Het doel van het in house fabriceren van de microscoop stand is dan ook het behalen va een betere performance op het gebied van trillingsisolatie en rigiditeit tegenover commercieel verkrijgbare standaarden.

Bij het bestuderen van trillings respons van een opstelling kan er gekeken worden naar kracht gedreven trilling (er word een periodieke kracht uitgeoefend op een vast punt op het instrument) of er kan gekeken worden naar amplitude gedreven trilling (trillen van het oppervlak onder het instrument). Het is belangrijk om het onderscheid tussen beide methodes te maken omdat deze wiskundig verschillend worden beschreven en ook anders dynamisch gedrag vertonen [Hesselberth, a], in het geval van dit project zal er gekeken worden naar een sterk gesimplificeerd model voor een amplitude gedreven trilling (zie figuur 1.

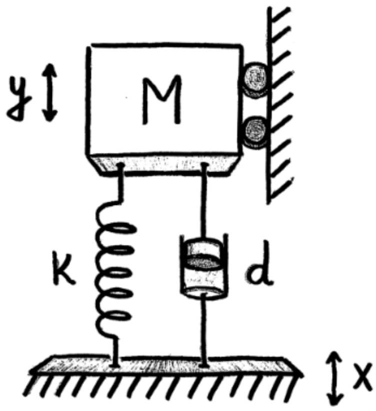

Figure 1: Schematic of a simplified microscope as a mass spring dampener system on a vibrating base. [Hesselberth, b]

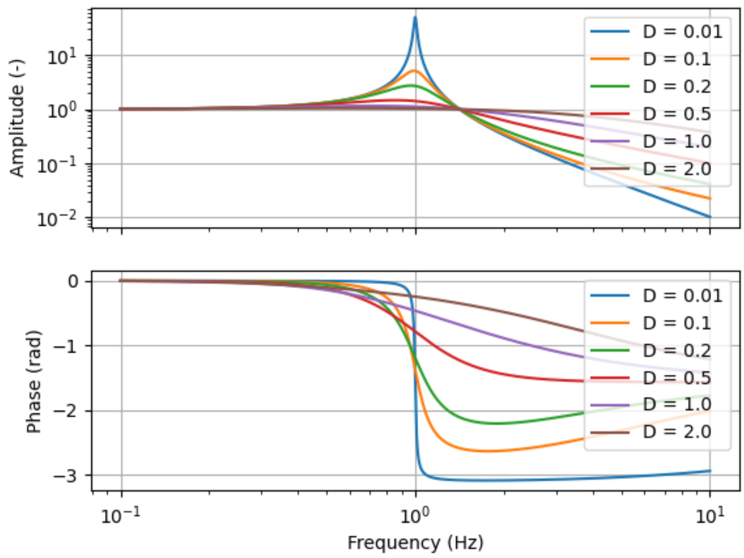

Figure 2

Door de opstelling te zien als een gedempt massa veer system (zie figuur 1)kan met regeltechniek theorie het verwachte dynamische gedrag benadert worden. Om de berekeningen te versimpelen word de opstelling verdeelt in 2 inertiaalsetelsels, stelsel Y met coorinaat y voor de uitwijking van de massa m met respect tot de evenwichtspositie van de massa en X met coordinaat x voor de uitwijking van de basis met respect tot de evenwichtspositie. Omdat het gedrag als functie van tijd voor dit systeem niet van belang alleen de overdracht als functie van frequenstie kan er gekeken worden naar de overdracht van het systeem in het Laplace domein. Het model bestaat uit de gecombineerde krachten balans in de Z richting zoals beschreven in formule 1. Door de balans vervolgens te transformeren naar het Laplace domein en te herordenen kan de overdrachtsfunctie zoals beschreven in formule 3 verkregen worden.

$$m\frac{\mathrm{d}^2 y}{\mathrm{d}t^2} + d\frac{\mathrm{d}y}{\mathrm{d}t} + ky = d\frac{\mathrm{d}x}{\mathrm{d}t} + kx \tag{1}$$

$$\omega_0 = \sqrt{\frac{k}{m}}; \quad D = \frac{d}{\sqrt{2mk}}; \quad f = \frac{\omega}{\omega_0} \tag{2}$$

$$H = \frac{Y}{X} = \frac{dj\omega + k}{-m\omega^2 + dj\omega + k} = \frac{\omega_0^2 + 2jD\omega_0^2 f}{\omega_0^2 + 2jD\omega_o^2 f - \omega^2 f^2} \tag{3}$$

$$A = \frac{\sqrt{1 + (2Df)^2}}{(1-f^2)^2 + (2Df)^2}; \quad \phi = \arctan\left(\frac{-2Df^3}{1 - f^2 + (2Df)^3}\right) \tag{4}$$

Zoals te zien in formulde 3 te zien is zal de amplitude oneindig naderen wanneer het systeem geen dempingsfactor heeft. Het instrument zelf zal geen demping hebben maar geplaats worden op een actieve vibratie isolatie tafel, voor correcte werking is het van belang dat de resonantiefrequenie van het instrument buiten de doorgelaten band van de actieve tafel valt. Voor de gebruikte active demper is dit 90% vanaf 2 Hz en 99% vanaf 10 Hz [DVI]

Aantoner dat een hoge dichtheid

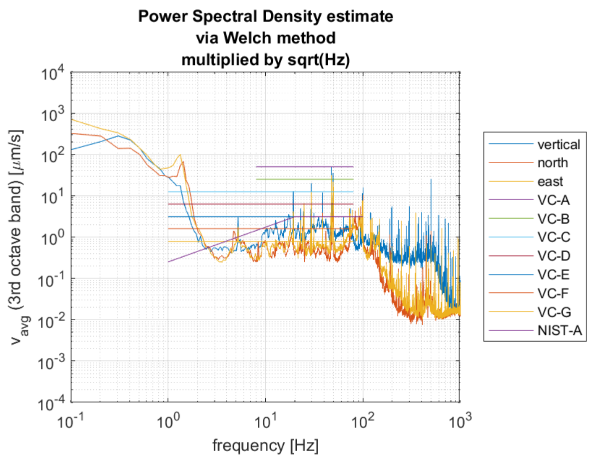

Figure 3: Measured vibration data in the Van der Waals Stacking Facility laboratory on the position of the stacking instrumentation. By J. Scheffer

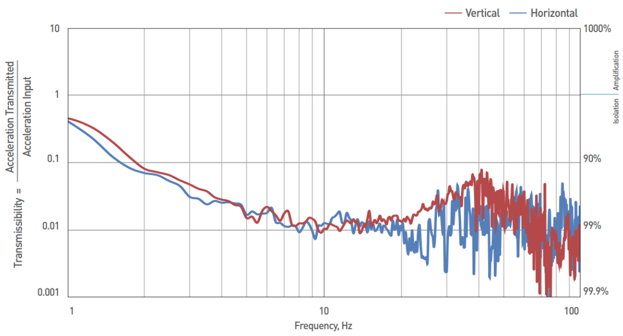

Figure 4

3.4 Criteria

Door het analyseren vcan de workflow van handmatige en gemotoriseerde commerciele stamping opstellingen in combinatie met literatuur studie is een lijst aan minimale eisen opgesteld waar het te ontwikkelen instrument aan moet voldoen.

uitzoeken aan welke vc norm wij willen voldoen

betekenis van NIST A norm uitzoeken

1. Mask holder
   (a) The mask should be motorized and able to move in the x, y and z direction with a resolution of > 30 nm.
   (b) The pitch of the mask should be able to be adjusted with a resolution of > 0.05°.
   (c) The yaw of the mask should be able to be adjusted with a resolution of > 0.05°, with a minimum range of 180°.
2. Sample holder
   (a) The sample holder should be able to hold a minimum of 2 samples.
   (b) The sample should be able to heat with a ramp speed of $2.0 - 50.0°C/s$
   (c) The sample should be able to cool with a ramp speed of $-0.1 - 30.0°C/s$
3. Microscope & stand
   (a) Due to the fact that the VDWSF laboratry has problems with external vibrations (see 3 the microscope stand should be as rigid as possible to make the instrument behave like one rigid body instead of a system of multiple bodies.
   (b) The microscope should a 20x objective with correction collar to be able to focus on the sample through the mask holder glass slide.

4 Methods & Results

4.1 Stand design

Voor het fabriceren van stacks is het belangrijk dat de opstelling zo min mogenlijk beinvloed word door trillingen uit de omgeving. . Om deze reden is er voor gekozen om in samenwerking met LION FMD een aantal uitwijkings simulaties uit te voeren om een inschatting te kunnen maken over de maximale uitwijking onder een statische kracht van de microscoop stand.

Verwijzen naar verwacht trillingen grafiek

Het complete instrument bestaat uit verschillende aspecten die in de volgende hoofdstukken behandelt zullen worden. Voor een overzicht van de gebruikte hard- en software componenten zie 13.

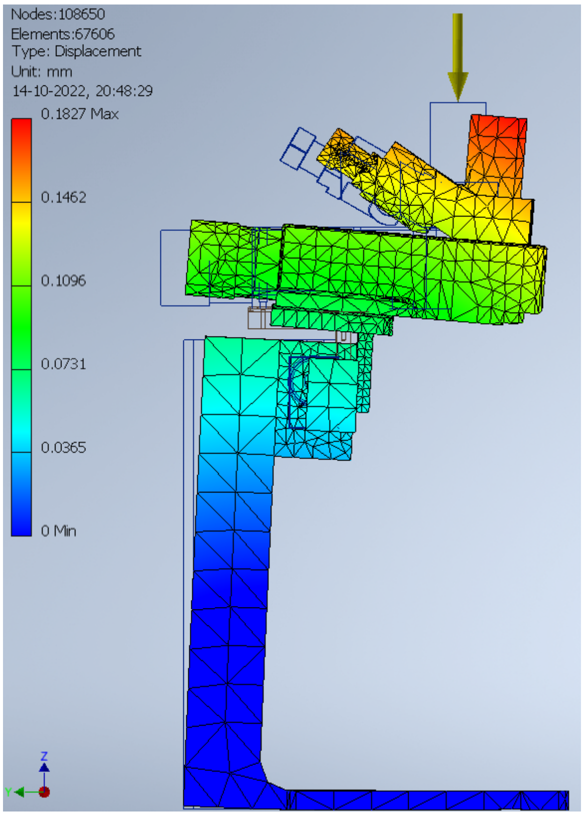

Figure 5: 100 N pressing force on a static point on the microscope. To reduce displacement a 30mm fillet was added between the base-plate and stand, reducing the maximum displacent to 183μm. By R. Stoelwinder

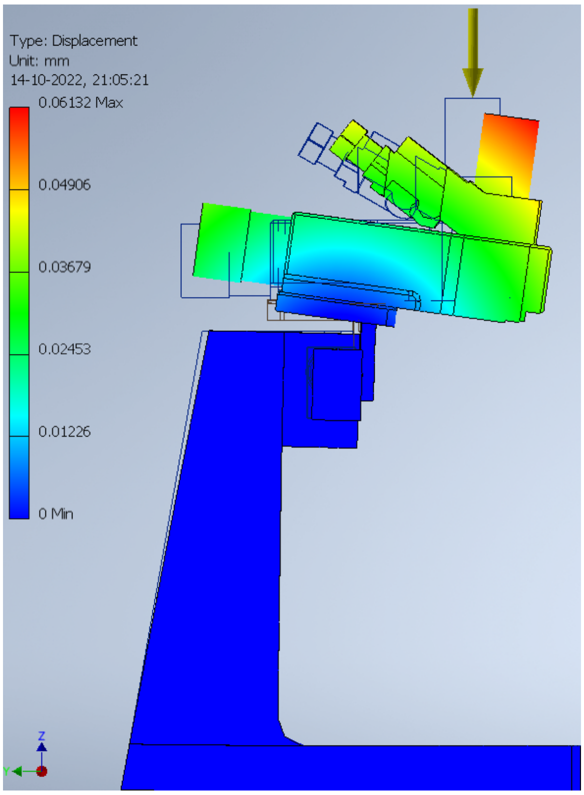

Figure 6: 100N pressing force on a static point on the microscope head. To reduce the the maximum displacement the thickness of the base-plate is increased from 20mm to 50mm, reducing the maximum displacement to 50μm. By R. Stoelwinder

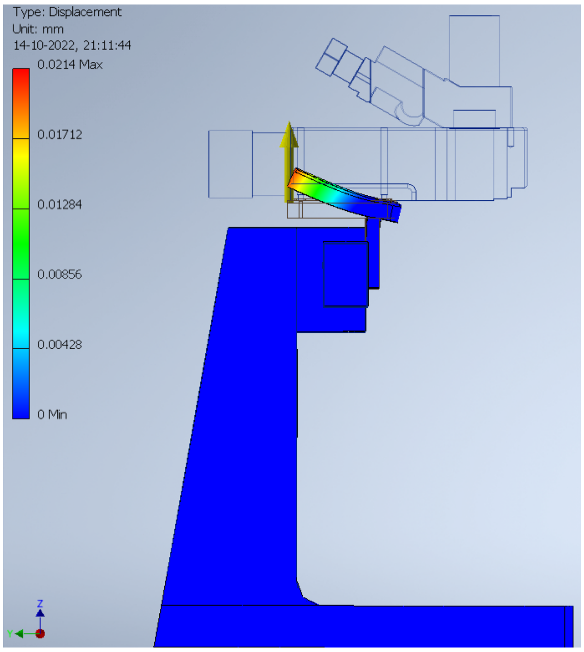

Figure 7: 100N pulling force on a static point on the corner plate causing a 21μm displacement. By R. Stoelwinder

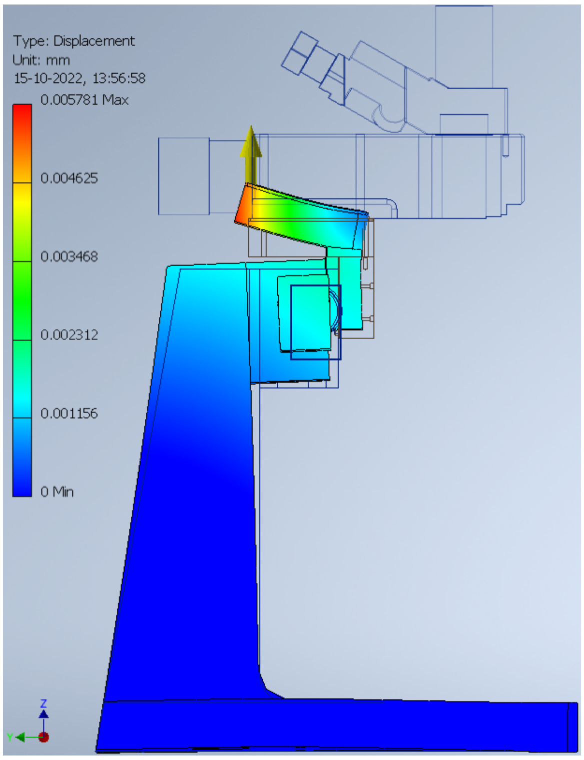

Figure 8: 100 N pulling force on a static point on the corner-plate. To reduce the maximum displacement the thickness of the corner plate is increased from 20mm to 40mm, reducing the maximum displacement to 5μm. By R. Stoelwinder

4.2 Mask design

the mask consists of two major parts: the mask holder wich holds the stamping polymer and the stages wich are used to position the mask above the sample

For the mask holder

CAD schema van de mask holder invoegen

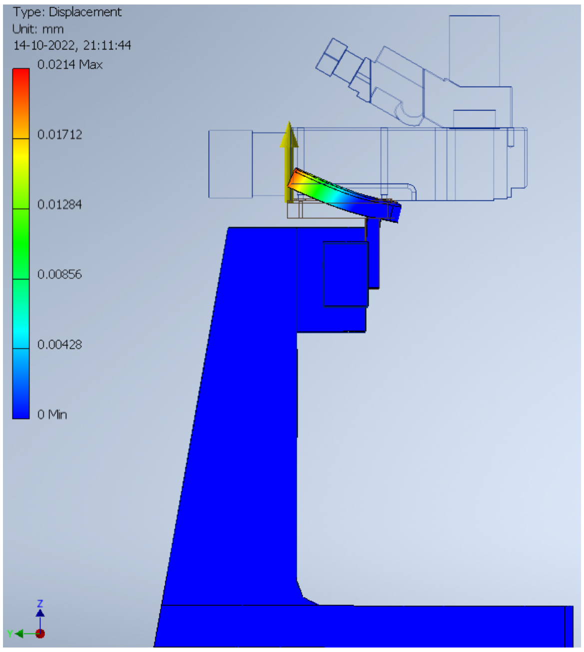

Figure 9: The mask holder schematic. By R. Stoelwinder

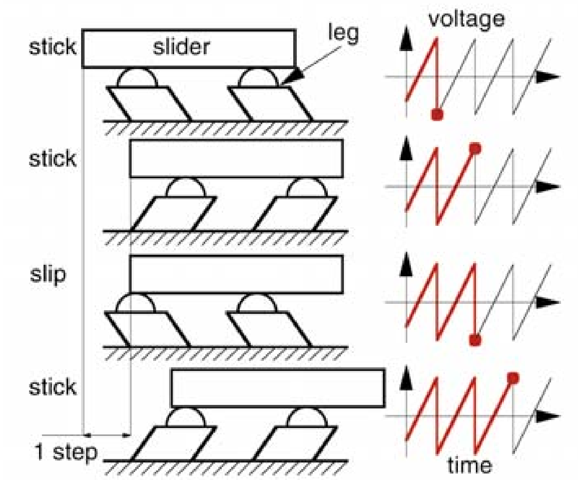

Figure 10: Operating principle of a slip-stick actuator mechanism [Mazerolle et al.]

4.3 Sample holder design

de hoofdtaak van de samplehouder is zoals de naam zegt het in positie houden van het monster waarmee gewerkt word (in dit geval een waver). Net zoals de rest van het instrument is het van belang om de sample houder zo rigide mogenlijk te maken om ervoor te zorgen dat de trillingen van de masker en de camera in fase blijven.

Tijdens de duur van dit project is het eerste prototype van de samplehouder gefabriceerd en zijn de plannen voor het vervolg ontwerp gemaakt. Het prototype (zie figuur 9) bestaat uit 3 onderdelen: Het sample bed, isolatielaag en de dump cilinder. Het sample bed bevat de benodigde componenten om de het sample naar op gewilde temperatuur tussen 0 en 200 graden C krijgen, hiervoor worden een heating coil (verwarmen) en peltier element (koelen) gebruikt. Het samplebed bevat ook een vacuum aansluiting die gebuikt kan worden om het sample stugger te bevestigen wanneer de gebruiker trillingsproblemen ondervind.

De functie van de isolatielaag is het isoleren van het samplebed en de dump cilinder om te voorkomen dat warmte vanuit de dumpcilinder teruglekt in het sampleblok (van belang tijdens koelen van het monster)

keuze voor peel en mica onderbouwen

CAD schema van de sample holder in voegen

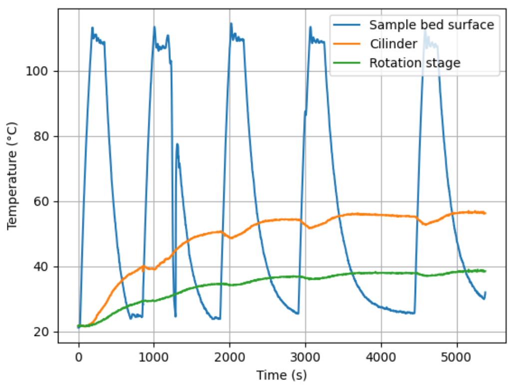

Figure 11: Temperature graph of the sample holder when simulating workload by heating to 110 degrees and cooling down to abient temperature (25 degrees) while waiting for 120 seconds when the level is reached

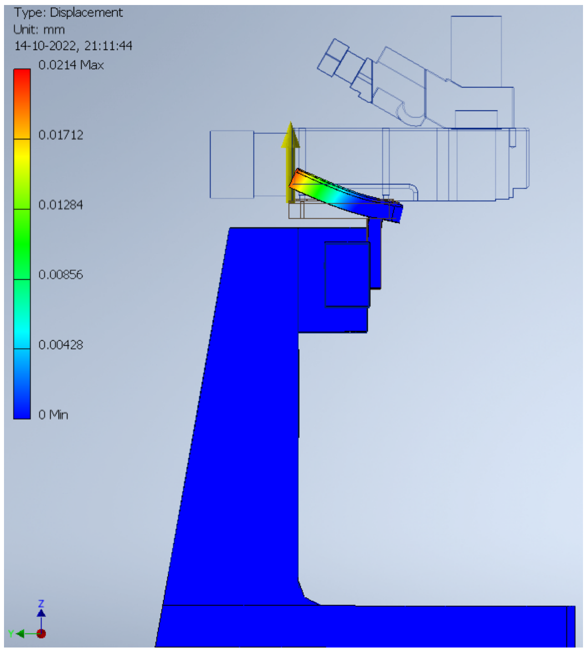

Figure 12: CAD schematic of the sample holder. By R. Stoelwinder. The sample holder prototype consists of 4 major parts. 1. The sample bed, wich contains a heating coil, peltier element, thermocouple and PT100 sensor. 2. The sample isolation layer. This is a shroud fabricated out of PEEK

4.4 Base stage automisation

for this project the base xy stage () has to be automated, this will developed in house in collaboration with LION FMD and ELD.

CAD van bracket invoegen

The stages were automated using NEMA17 stepper motors , and optical endstops. The stepper motors and endstops are controlled by the controller developed by ELD. The contents and used tecniques in the controll box are out of the scope for this report.

ref naar optosigma stage invoegen

The first prototype bracket makes it possible to test the complete system and work out implementation bugs. It is recommended to create further iterations of the brackes as the current version can flex $\pm 3mm$ and this has a detremental effect on the rigidity of the base xy stage, this because the stages are held in place by the stepper motor. Furthermore it is also recommended to replace the ball bearings in the spindles as these

citation invoegen

were damaged during fabrication and the spindle is now only held in place by the stepper motor

4.5 Stage calibration

Due to the mchanisms used in some of the actuators it is not possible to calibrate all the staged, nor is this needed for the application. The stages that cannot be calibrated are the x, y, and z axis of the mask due to the used slip-stick mechanism (see figure 10).

The slip stick mechanism makes it possible to use piezo actuators with high resolution for an unbound range. The drawback is that due to the slip stick mechanism (friction based) the actuator has a position uncertainty of and a sstep uncertainty of .

positie uncer-tainty in-voegen

step vari-atie invoe gen

4.6 Software

Because one of the main criteria of the project is flexibility and modularity this has to be taken in consideration from the beginning. This means the project needs to be created in a structured and standardised way using design patterns and a modular architecture (see appendix A for the definition of these terms).

4.6.1 Architecture

As a main architecture a web application inspired framework is used. This means that the system consist of three major components: Frontend, Middleware and backend (for more information about these definitions see appendix A). Using standard design patterns makes getting to know the code and editing it less cumbersome and error prone for future developers. An overview of the system flow can be seen in figure 13

laatste pi jlen in de correcte richting zetten

4.6.2 Frontend

The frontend is the part of the system that is visible to the user. This part of the system is responsible for the user interface and the communication with the middleware. The system has two pre made interface options: GUI and TUI. For operators of the system it is recommended to use the GUI as no knowledge of the command set is needed and safety features like thermal-runaway and forbidden areas are enabled. For developers it is recommended to use the TUI as it is more flexible and allows for more control over the system.

The GUI is created using QT6 with Pyside6 bindings to ensure interoperability between different operating systems and detailed documentation for future developers.

citation invoegen

The TUI consists of a simple loop that pushes the gcode commands to the backend and uses a thread to monitor the backend response.

4.6.3 Middleware

The middleware is a method that can be used to communicate between the frond and backend. The middleware base class (see B) contains all the base methods needed for the system to work with a middleware method. The system contains two default middleware methods: The processing pipe, wich is used for when the backend runs on the same computer as the frondend but on a seperate core (it is strongly discuraged to run the backend on the same core as the frondend as this makes the backend behavoir depend on core workload.) The second method is the serial method and is based on the PySerial library and is used when the backend runs a differend computer than the frondend. The serial middleware method can be easily adapted to work with wireless serial communication but this is dicuraged as this method is more prone to errors.

citation invoegen

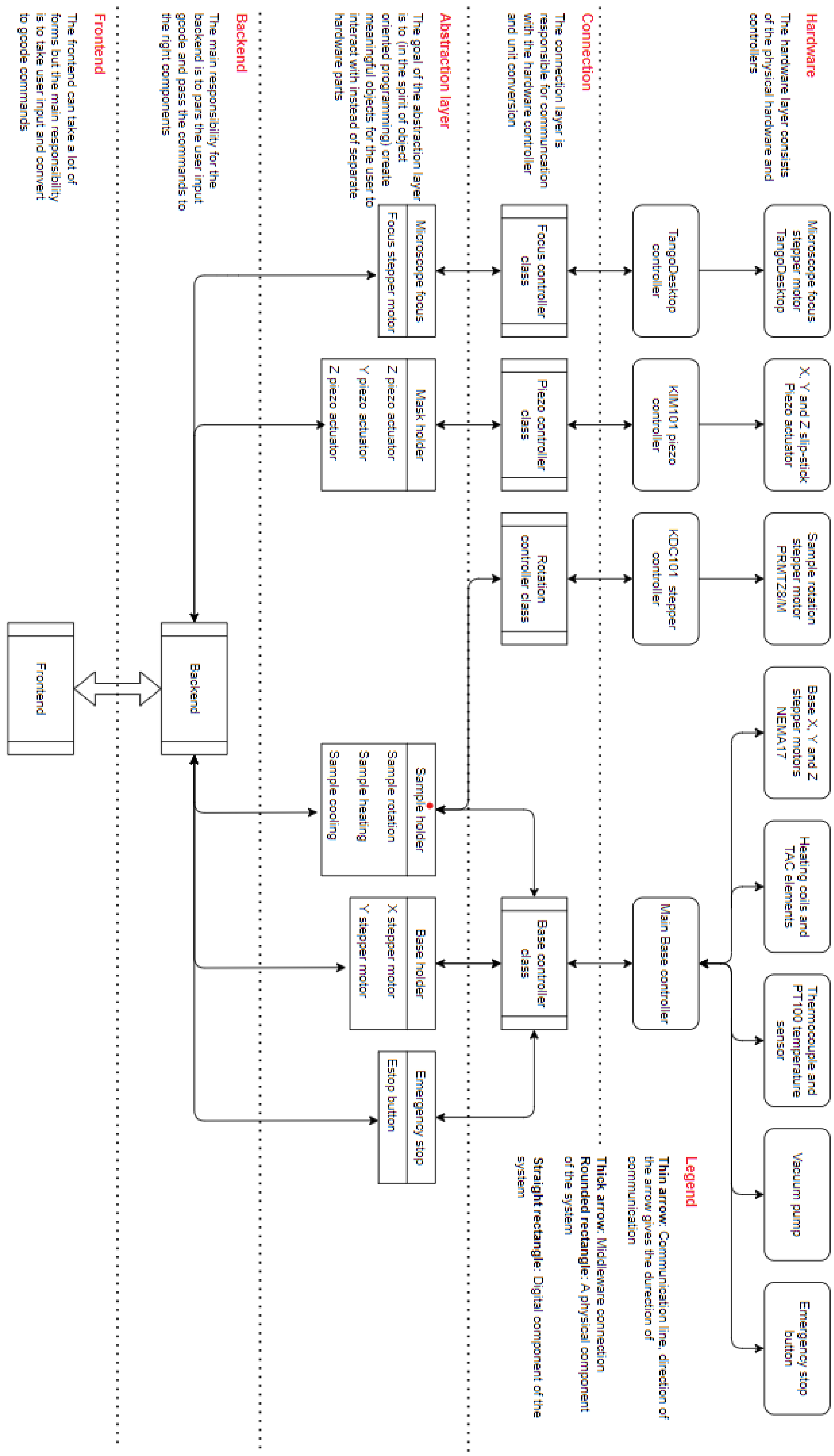

Figure 13: Software flow of the project.

4.6.4 Backend

The backend is the part of the system that is responsible for the communication with the hardware and the middleware and managed the object the user can interact with (sample holder, mask holder, etc.) The backend accepts all classes that inherit from their respective base class and abide the programming guidelines

Due to time constraints it was not possible to develop the backend to run on unix based systems

referentie naar docs toevoegen

4.6.5 Forbidden areas

stukje over FTD issue tovoegen

stukje over vo-ordeel van runnen op IO controller toevoegen

To prevent the system from damaging itself or the sample it is important to prevent the system from moving to certain areas in certain states.

Phase space variables (system attributes):

- Position of all the axis*
- Velocity of all the axis
- Acceleration of all the axix
- Sample temperature
- Sample heating and cooling ramp

This means the phase space will consist of 21 dimensions, where each dimension reperesents the current value of a system attribute.

First the phase space is bordered off by the hardware limitations like maximum velocity, minmal reproducable step size, range, etc. Further forbidden areas are studied using a fractional factorial design to determine the most important variables that influence the system.

The forbidden areas will be determined using a fractional factorial design with the following parameters (see excel file)

5 Conclusion

6 Discussion

References

DVIA-T Tabletop Active Vibration Isolation Platform | Products. URL https://www.daeilsys.com/products/active-vibration-isolation-systems/tabletop-active-vibration-isolation-platform/.

A. K. Geim and I. V. Grigorieva. Van der Waals heterostructures. 499(7459):419–425. ISSN 1476-4687. doi: 10.1038/nature12385. URL https://www.nature.com/articles/nature12385.

M. Hesselberth. Basic calculations for vibration isolators. a.

M. Hesselberth. Microscopie in beweging. b.

M. K. V. Isolation. Vibration Criterion (VC) curves | Minus K. URL https://www.minusk.com/content/technology/vc_curves_minus_k_vibration_isolation.html.

S. Mazerolle, R. Rabe, T. Varidel, and J.-M. Breguet. Positioning, Handling and Measuring inside a Scanning Electron Microscope.

A Programming terms

Design patterns

In programming, a design pattern is a general repeatable solution to a commonly occurring problem in software design. A design pattern is not a finished design that can be transformed directly into code. It is a description or template for how to solve a problem that can be used in many different situations.

Design patterns are used to describe solutions to specific design problems in object-oriented programming. They provide a way to reuse successful designs and design solutions that have been proven to work well in the past.

There are many different types of design patterns, including creational patterns, structural patterns, and behavioral patterns. Creational patterns deal with object creation mechanisms, trying to create objects in a manner suitable to the situation. Structural patterns deal with object composition, creating relationships between objects to form larger structures. Behavioral patterns focus on communication between objects, what goes on between objects and how they operate together.

Using design patterns can make it easier to design and develop software, because they provide a common vocabulary and framework for thinking about and solving design problems. They also help to make code more maintainable and scalable by providing a clear, consistent way to structure and organize it.

Modular architecture

In a web application or system, the frontend refers to the part of the application that the user interacts with, typically through a web browser. It includes all the elements that the user can see and interact with, such as the layout, user interface, and visual design of the application.

The backend, on the other hand, refers to the part of the application that handles the logic and processing behind the scenes. It is responsible for tasks such as storing and retrieving data from a database, performing calculations and processing, and interacting with other systems or APIs.

Middleware is software that sits between the frontend and the backend and helps to facilitate communication and data exchange between them. It can provide functions such as authentication, routing, and data transformation, and can help to abstract away some of the complexity of the backend from the frontend.

In a web application, the frontend and backend are typically implemented using different technologies and run on different servers or infrastructure. Middleware can be implemented using a variety of technologies, depending on the specific requirements of the application.

Multi-threading and processing

In computer science, multi-threading is a technique that allows a single process or program to have multiple threads of execution. A thread is a separate flow of execution within a process, and each thread can run concurrently or in parallel with the other threads within the same process.

Multi-threading can be used to improve the performance of a program by allowing it to perform multiple tasks concurrently. For example, a program that has a user interface and needs to perform some time-consuming calculations in the background could use multi-threading to allow the user to continue interacting with the interface while the calculations are being performed.

Multi-processing is a technique that allows a computer to run multiple processes simultaneously. A process is an instance of a program that is being executed by the operating system. Each process has its own memory space and runs in a separate environment, and multiple processes can be run concurrently on different cores or processors within a computer.

Multi-processing can be used to improve the performance of a program by allowing it to take advantage of multiple processors or cores within a computer. This can be particularly useful for programs that need to perform a lot of calculations or that are designed to run on a distributed system with multiple machines.

B Backend base classes

The base classes are used as a template for new components. The base classes contain all the functions that are required for the component to work with the rest of the system. Notice that two different errors can be raised: NotImplementedError and NotSupportedError. The functions that raise a NotImplementedError should always be implemented for the component to work correctly with the rest of the system. It is safe to raise NotSupportedError as these are caught and handled by the system.

Harware base class

```python
class NotSupportedError(Exception):
    """Exception raised when a method is not supported."""

    def __init__(self, msg=None):
        """Initialize the exception."""
        self._msg = msg

    def __str__(self):
        return self._msg


class HardwareNotConnectedError(Exception):
    """Exception raised when a hardware is not connected."""

    def __init__(self, msg=None):
        """Initialize the exception."""
        self._msg = msg

    def __str__(self):
        return self._msg


class NotCalibratedError(Exception):
    """
    Exception raised when the hardware is not calibrated for the asked function.

    F.e. trying to do an absolute move when the home point is unknown.
    """

    def __init__(self, msg=None):
        """Initialize the exception."""
        self._msg = msg

    def __str__(self):
        return self._msg


class HardwareError(Exception):
    """General exception raised when the hardware is not working properly."""

    def __init__(self, msg=None):
        """Initialize the exception."""
        self._msg = msg

    def __str__(self):
        return self._msg


class Base:
```

```
    """
    Base class for hardware.

    This class contains all the functions the StackingSetupBackend class expects
    connected hardware to have. Functions that are supported should be overridden
    in the derived class.

    .. important::
        Functions that should be supported by all hardware raise a ``NotImplementedEr
        optional functions raise ``NotSupportedError``. The ``NotSupportedError`` is
        by the ``StackingSetupBackend()`` class, so it is safe to raise it in functi
        not supported by all hardware.

    .. note:: Each function should return an exit code (0 for success, 1 for failure
        and an error message, str, or None.

    """

    _id = None
    _type = "HARDWARE BASE CLASS"

    # COMPONENT LIMITS
    _max_speed = None
    _max_acceleration = None
    _max_temperature = None

    # ATTRIBUTES
    @property
    def id(self):
        """Get the class identifier."""
        return self._id

    @property
    def type(self):
        """Get the type of the hardware."""
        return self._type

    @property
    def device_info(self):
        """Get the device info of the hardware."""
        raise NotImplementedError()

    # STEP CALIBRATION ATTRIBUTES
    """
    If the connected component can move only one of the steps per ... should be suppo
    If the component cannot move none have to be supported.
    """

    @property
    def steps_per_um(self):
        """Get the steps per um of the hardware."""
        raise NotSupportedError()

    @property
    def steps_per_deg(self):
        """Get the steps per degree of the hardware."""
        raise NotSupportedError()

    # MOVEMENT PROFILE ATTRIBUTES
    @property
    def position(self):
        """Get the position of the hardware."""
```

```python
            raise NotSupportedError()

    @property
    def speed(self):
        """Get the set speed of the hardware in um/s or deg/s."""
        raise NotSupportedError()

    @speed.setter
    def speed(self, speed):
        """Set the speed of the hardware in um/s or deg/s."""
        raise NotSupportedError()

    @property
    def acceleration(self):
        """Get the acceleration of the hardware."""
        raise NotSupportedError()

    @acceleration.setter
    def acceleration(self, acceleration):
        """Set the acceleration of the hardware."""
        raise NotSupportedError()

    # TEMPERATURE ATTRIBUTES
    @property
    def temperature(self):
        """Get the temperature of the hardware."""
        raise NotSupportedError()

    @property
    def target_temperature(self):
        """Get the target temperature of the hardware."""
        raise NotSupportedError()

    @target_temperature.setter
    def target_temperature(self, temperature):
        """Set the target temperature of the hardware."""
        raise NotSupportedError()

    # CONNECTION FUNCTIONS
    def connect(self):
        """Connect the hardware."""
        raise NotImplementedError()

    def disconnect(self):
        """Disconnect the hardware."""
        raise NotImplementedError()

    # STATUS FUNCTIONS
    def is_connected(self):
        """Check if the hardware is connected."""
        raise NotImplementedError()

    def is_moving(self):
        """Check if the hardware is moving."""
        raise NotSupportedError()

    def is_homed(self):
        """Check if the hardware is homed."""
        raise NotSupportedError()

    def get_status(self):
        """Give a status report."""
```

```
            raise NotImplementedError()

    # HOMING FUNCTIONS
    def home(self):
        """Home the hardware."""
        raise NotSupportedError()

    # MOVING FUNCTIONS
    def start_jog(self, direction):
        """Start a jog in a direction."""
        raise NotSupportedError()

    def stop_jog(self):
        """Stop a jog."""
        raise NotSupportedError()

    def move_to(self, position):
        """Move the hardware to a position."""
        raise NotSupportedError()

    def move_by(self, position):
        """Move the hardware by a position."""
        raise NotSupportedError()

    def rotate_to(self, rotation):
        """Rotate the hardware to a position."""
        raise NotSupportedError()

    def rotate_by(self, rotation):
        """Rotate the hardware by a position."""
        raise NotSupportedError()

    def stop(self):
        """Unconditionally stop the hardware."""
        raise NotImplementedError()

    def emergency_stop(self):
        """Unconditionally stop the hardware."""
        raise NotImplementedError()
```

Middleware base class

```python
import configparser
import time

SENTINEL = "SENTINEL"  # Sentinel command to close the pipe
EOM_CHAR = "EOM"  # String indicating the end of a message over a pipe


class HandshakeError(Exception):
    def __init__(self, message=None):
        self._message = message

    def __str__(self):
        return self._message


class BaseConnector:
    """
    Class to hold all the expected connection functions.

    This class does not manage the connection it is a base class that should
    be inherited.
    """

    _connection_method = None
    _role = None
    _handshake_complete = False
    _SENTINEL = SENTINEL

    # ATTRIBUTES
    @property
    def connection_method(self):
        """Return the connection method."""
        return self._connection_method

    @property
    def role(self):
        """Return the role of the connection."""
        return self._role

    @property
    def handshake_complete(self):
        """Return the handshake status."""
        return self._handshake_complete

    @property
    def is_connected(self):
        """Check if the device is connected."""
        raise NotImplementedError()

    @property
    def SENTINEL(self):
        """Return the sentinel command."""
        return self._SENTINEL

    # METHODS
    def connect(self):
        """Connect to the device."""
        raise NotImplementedError()

    def disconnect(self):
```

```python
        """Disconnect from the device."""
        raise NotImplementedError()

    def send_sentinel(self):
        """Send a sentinel to the IO controller (RPI)."""
        raise NotImplementedError()

    def send(self, command):
        """Send a command to the device."""
        raise NotImplementedError()

    def message_waiting(self):
        """Check if a message is waiting."""
        raise NotImplementedError()

    def receive(self):
        """Receive data from the device."""
        raise NotImplementedError()

    def handshake(self):
        # Depending on the role of the connector decide
        # what to send and what to receive
        if self._role == "FRONDEND":
            self._frondend_handshake()
            self._handshake_complete = True

            # Empty the buffer
            while self.message_waiting():
                _ = self.receive()

        elif self._role == "BACKEND":
            self._backend_handshake()
            self._handshake_complete = True

            # Empty the buffer
            while self.message_waiting():
                _ = self.receive()

        else:
            raise HandshakeError("Unknown role {}".format(self._role))

    def _frondend_handshake(self):
        if not self.is_connected:
            return False
        # Wait for the response
        while True:
            # Send the hello message
            self.send("Hello there.")

            if self.message_waiting():
                res = self.receive()
            else:
                continue

            if res[0] == "Hello there general Kenobi.":
                return True
            else:
                raise ValueError("Unexpected message: {}".format(res[0]))

        raise ValueError("frondend Handshake failed")

    def _backend_handshake(self):
```

```
        # Wait for the hello message
        while True:
            time.sleep(0.1)

            if self.message_waiting():
                res = self.receive()
            else:
                continue

            if res[0] == "Hello there.":
                self.send("Hello there general Kenobi.")
                break
            else:
                raise ValueError("Unexpected message: {}".format(res[0]))
```

Todo list