Readout System For Graphene-Based Gas Sensor Array

T.J. Abels, D.G. Bot, D.C. Kragt, L. Schuddeboom, T.F.M. van Teeseling

Readout System For Graphene-Based Gas Sensor Array

by

T.J. Abels, D.G. Bot, D.C. Kragt, L. Schuddeboom, T.F.M. van Teeseling

Bachelor of Science Thesis
at the Delft University of Technology,

Student numbers:	5635233 (T.J. Abels)	
	5373271 (D.G. Bot)	
	5314488 (D.C. Kragt)	
	5409810 (L. Schuddeboom)	
	4874501 (T.F.M. van Teeseling)	
Thesis committee (June 24):	Prof. Dr. Ir. J.J.A. Baselmans,	TU Delft, chair
	Dr. Ir. S. Vollebregt,	TU Delft, supervisor
	Dr. Ir. R.C. Hendriks,	TU Delft
Thesis committee (June 25):	Dr. Ir. M.A.P. Pertijs,	TU Delft, chair
	Dr. Ir. S. Vollebregt,	TU Delft, supervisor
	Dr. F. Arroyo Cardoso,	TU Delft

Abstract

This Bachelor of Science thesis presents the development of a portable readout system for a graphene-based gas sensor array, aiming to bring advanced gas sensing technology from a controlled laboratory environment to practical field applications, such as greenhouses and vineyards. The project focuses on integrating various hardware components into a single printed circuit board (PCB) equipped with a microcontroller, ensuring accurate resistance measurement of graphene strips, generating programmable waveforms for micro hotplate control, and facilitating data communication via USB and Modbus interfaces.

The design includes high-resolution analog-to-digital converters (ADCs) for precise current and voltage measurements, a digital-to-analog converter (DAC) for waveform generation, and a robust power management system supporting multiple power inputs. The system supports dual operation modes: interfacing with a desktop for laboratory use and functioning autonomously as an integrated sensor unit. Key challenges addressed include ensuring measurement accuracy, reliable sensor array connectivity, and user-friendly interfacing for both lab and field scenarios.

Comprehensive test plans are outlined to verify the system's performance, focusing on critical requirements such as resistance measurement accuracy, waveform generation, and communication interfaces. The successful implementation of a large part of this readout system demonstrates its potential for enhancing gas detection and monitoring in agricultural applications, providing a scalable solution for future deployment and development.

Preface

This thesis marks the culmination of our efforts in developing a portable readout system for a graphene-based gas sensor array. The journey has been challenging yet immensely rewarding, offering us the opportunity to apply and expand our knowledge in electrical engineering, sensor technology, embedded programming and data processing.
The project was initiated with the aim of transforming a lab-based gas sensor system into a portable device that could be used in real-world applications such as greenhouses and vineyards. This involved integrating various hardware components onto a single PCB with a microcontroller, ensuring accurate measurement capabilities, generating variable waveforms, and facilitating seamless data communication through USB and Modbus interfaces.
We would like to extend special thanks to our supervisor prof. dr. ir. Sten Vollebregt and our daily supervisor Mudassir Husain for their continuous support and guidance, Stephan van 't Hoff and the others at DEMO for their immense support in the mechanical design, and Jos van den Berg of Hoogendoorn for his insights and knowledge about the current developments in the horticultural technology sector. We would also like to thank Wouter Kragt from Texas Instruments for helping us with the component selection process and giving feedback on the early hardware design. Finally our colleagues, Tomasz Abels, Daniel Bot, Dirk-Jan Kragt, Lars Schuddeboom and Thomas van Teeseling for a productive and pleasant collaboration.

T.J. Abels, D.G. Bot, D.C. Kragt, L. Schuddeboom, T.F.M. van Teeseling
Delft, June 2024

Contents

Chapter 1

Problem Definition

1.1 Problem Scope

The client is developing a new type of gas sensor that uses microscopic graphene strips in order to measure the presence of different gases. An array of 8 graphene strips is integrated onto a micro-hotplate, which is used to heat the graphene strips to desorb gas molecules. Due to its longevity, low maintenance, and potential for high selectivity across a range of different gases, this sensor has the potential to be widely useful in agriculture and horticulture applications. Currently, the sensor is being tested in the Else Kooi Laboratory in a vacuum chamber using large and expensive hardware. An overview of this setup can be seen in Figure 1.1. This is an expensive and highly controlled environment, and currently, there is no other way of testing the sensor. Eventually, the sensor will be used in real-world applications, such as greenhouses or vineyards. This means that a portable readout system must be developed so that field tests can be performed using the sensor.

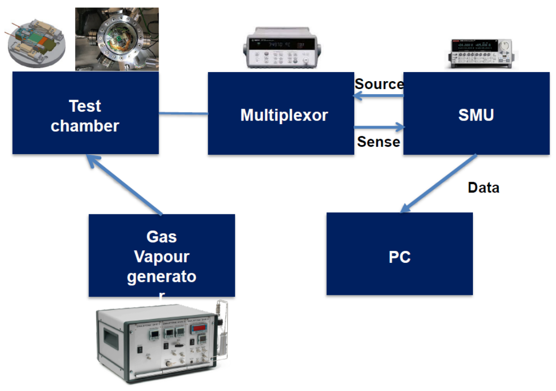


Figure 1.1: Current test setup in EKL to measure gases using sensor array

1.2 Technical Review

Measuring the presence of certain gases in greenhouses or crop fields can be highly beneficial in terms of recognizing problems and taking timely measures to prevent crop failures. Gas concentration levels, such as carbon dioxide (CO_2), ammonia (NH_3), and methane (CH_4), significantly influence the growth conditions in greenhouses and storage facilities. Optimal concentrations of these gases can lead to an increase in yield and quality of the crops. Highly sensitive gas sensors that can detect the presence of these gases early are therefore a crucial instrument in controlling the environment in horticultural and agricultural settings.

Currently, one of the most commonly used gas sensors is the metal oxide-based gas sensor. These types of sensors rely on changes in conductivity as a gas interacts with the surface of the metal oxide. They are characterized by low cost, short response time, and long lifetime, and are able to detect a wide range of gases. However, they have low sensitivity and selectivity, are sensitive to environmental factors, and have high energy consumption, as they operate at high temperatures.[1]

Optical gas sensors rely on absorption and emission spectrometry for gas detection. They rely on the light absorption properties of gas molecules at particular wavelengths to identify different gases. This results in high selectivity, sensitivity, and stability, as well as insensitivity to environmental changes. However, these types of sensors have a high cost and are not very suitable for miniaturization, resulting in their usage mainly being in high-end market applications.[5]

Another common type of gas sensor is the catalytic gas sensor, which is mainly used for the detection of combustible gases. They rely on the change in resistance of a coil in the sensor to detect combustible gases. As the gas enters the sensor, it interacts with a catalyst, igniting the gas and hereby increasing the temperature of the coil, thus increasing the resistance.[19] These types of sensors have low cost, good sensitivity, and are stable at ambient temperature, but they have the risk of catalyst poisoning and poor selectivity.[5]

One of the latest and most promising gas sensors that are being researched right now are based on the material graphene. Graphene has exceptional properties due to its molecular structure, which consists of carbon atoms in a hexagonal two-dimensional (2D) structure.[2] This 2D structure provides the graphene with a very high surface-to-volume area, where every atom is exposed to the environment. This makes the material very sensitive to the adsorption of gas molecules. The electrical conductivity of graphene changes upon interaction with the gas molecules, which can be exploited to act as a gas sensor. Because of these properties, it is theoretically possible to detect even single gas molecules adsorbed on the graphene.[9]

Integrating the graphene onto micro-hotplates allows for faster desorption of the gas molecules, giving the sensors a faster response time, providing better sensitivity, and making it possible to recover the device to its initial state.[8] Due to the small size and low thermal mass, a micro-hotplate can achieve very rapid and efficient heating and cooling.[10] Decorating the graphene strips with different types of nanoparticles, such as metal oxide, can improve the selectivity of a single graphene sensor. Multiple graphene strips can be decorated with different nanoparticles to make them sensitive to different gases. Differently decorated strips can then be combined into a sensor array to increase the selectivity even further. An array of 8 graphene strips integrated onto a micro-hotplate can be seen in Figure 1.2.

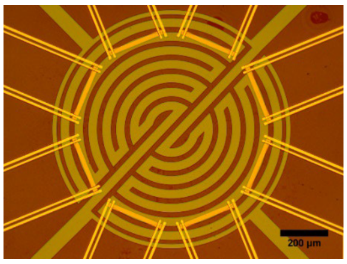


Figure 1.2: Array of 8 Graphene Strips integrated onto a Micro-Hotplate [4]

1.3 Design Requirements

Design Requirement	Importance	Units	Value
1. Accurate measurement of the resistance of 1 up to 8 graphene strips.	High	%	≤ 0.1
2. Generate a variable DC waveform to heat the hotplate. Variable voltage with different waveforms (Rectangular, sinusoid, sawtooth, and triangle)	High	mW / V	30mW at 12V
3. Connect the sensor array to the PCB. Easily enough so that someone without knowledge of the design can properly connect the array.	High	Boolean	True
4. Interface with a desktop for lab measurements	High	Boolean	True
5. Function as an autonomous sensor	Medium	Boolean	True
6. Microcontroller with ML capabilities	Low	Boolean	True

Table 1.1: Key Design Requirements

The customer has specified the goal of the project as the following: implement the hardware that is currently being used to perform measurements in the lab on a portable scale and integrate it on a single PCB with a microcontroller. After inspecting the documentation provided by the customer and multiple meetings, the requirements of the prototype were derived. The most important requirements are detailed in Table 1.1. For further clarification on how to achieve these requirements, see descriptions below.

1. Accurate measurements

As described, the PCB must be able to measure the resistance of up to 8 of the graphene strips in the sensor array with an accuracy of at least 0.1% of 10 kΩ (which is 10 ohm). In order to achieve this, it needs to:

- Generate a stable excitation voltage, programmable from the software (1V for each of the strips with an accuracy of $\pm 0.1\%$). The sensor consists of 8 slots, each of which can have a graphene strip connected to its terminals or not. The graphene strips should be under constant bias. In case one of the strips is not used, the slot should be bypassed by a short circuit.

- Measure the current through each of the strips with an accuracy of at least 0.1% for a nominal current of 0.1 mA.
- Measure the voltage over each of the strips with an accuracy of at least 0.1% for a nominal voltage of 1 V.
- Sampling Frequency: Measurements of the sensor values must be done at a rate of at least 1 Hz per strip and preferably variable.
- Measure the resistance of one up to eight strips on the sensor array.
- To provide calibration and additional data collection for verifying measurements, a humidity and temperature sensor is integrated into the system.

2. Generate a variable waveform for driving the micro-hotplate

The PCB also needs to generate a programmable waveform to control the micro-hotplate that is integrated in the sensor. In further detail, this means:

- Generate a programmable DC waveform with an amplitude of 12V at 30mW.
- The amplitude must be programmable up to 24V.
- It must be possible to control the pattern in which this power is delivered to the heating element of the sensor array - sinusoid, square wave, triangle wave, and variable outputs.

3. Connecting the Sensor Array

This sensor array needs to be easily connected to the PCB. Connecting the sensor array to the PCB such that the mechanical integrity and electrical connections are proper should be feasible without reading the documentation.

4. Interfacing with a Desktop

Furthermore, the client wants the prototype to function in two modes: both in a lab with a desktop interface for research and further development of the sensor array, and as an individual sensor that can be integrated in a robot that is being developed by the AGRARSENSE initiative so that it can detect certain molds in the agricultural and horticultural industry. When interfacing with a desktop, the prototype needs:

- To function on a 5V USB power input.
- Have a USB serial interface to communicate the data to and from the desktop.
- A Graphical User Interface for parameter entry, real-time plotting, and collection of raw data which can be downloaded.

5. Function as an Autonomous Sensor

When functioning as a sensor, the requirements of the prototype are determined by the system it will be integrated into, which dictates the following:

- Function on a 12V power input.
- Modbus serial interface for communication.

6. Microcontroller with ML Capabilities

The ML capabilities of the microcontroller were a design requirement posed by the client. The client hopes to integrate edge computing capabilities in the future to utilize the sensor to actually detect and recognize specific smells, which are characterized by different gas concentrations in the air.

System Requirements

Combining all of these requirements into a single overview of system requirements, a clear overview of the functionality that the design must fulfill can be obtained.

Table 1.2: System Requirements

#	Requirement	Values
1	**Accurate measurements**	
1.1	Resistance measurement accuracy	$\leq 0.1\%$ of 10 kΩ
1.2a	Generate stable excitation voltage	Programmable from software (1V $\pm$ 0.1%)
1.2b	Constant bias over each strip	1V $\pm$ 0.1%
1.3	Current measurement accuracy	$\pm$ 0.1%
1.4	Voltage measurement accuracy	$\pm$ 0.1%
1.5	Sampling Frequency	$\geq$ 1 Hz per strip
1.6	Bypass selected graphene strips	True/False
2	**Generate variable waveform for hotplate**	
2.1	Hotplate waveform voltage	Programmable up to 24V
2.2	Hotplate waveform power	At least 30 mW
2.3	Waveform patterns	Sinusoid, square wave, triangle wave, variable outputs
3	**Connecting the Sensor Array**	
3.1	The sensor array can be properly connected both electrically and mechanically without having to read the documentation.	True/False
4	**Interfacing with a Desktop**	
4.1	Power input	5V USB
4.2	Communication interface	USB serial
4.3	User Interface	Graphical User Interface
5	**Function as an Autonomous Sensor**	
5.1	Power input	12V
5.2	Communication interface	Modbus serial
6	**Microcontroller with ML Capabilities**	
6.1	ML capabilities	Edge computing for gas detection

Chapter 2

Design Description

2.1 Overview

The readout system is designed to detect small changes in the resistance of individual graphene strips. These changes in resistance can indicate the presence of a gas. Therefore, the system must have sufficient resolution to detect even very small resistance changes. The high-level block diagram, which illustrates the overall functionality of the system, is shown in Figure 2.1.

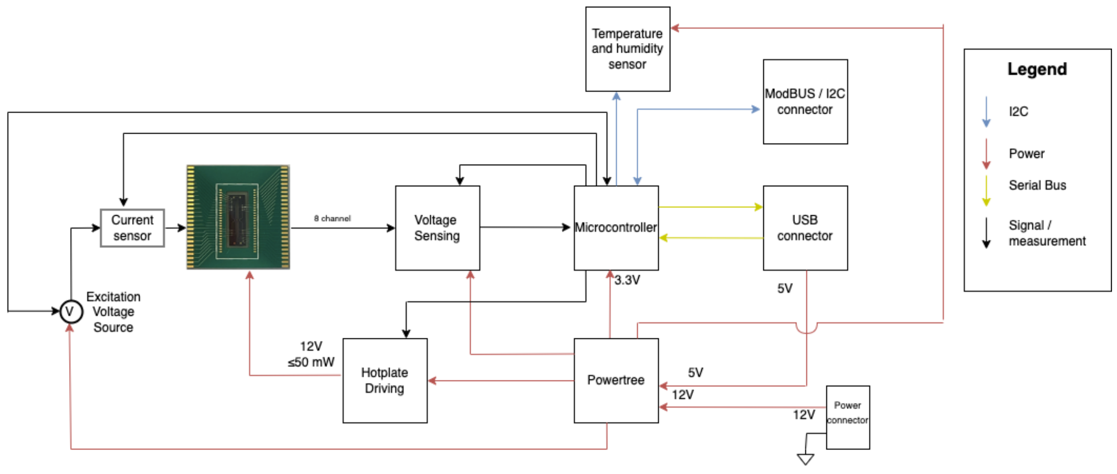


Figure 2.1: High-Level Design Overview

The operation of the circuit is as follows: an excitation voltage is applied across the graphene strips, causing a current to flow through them and through a series-connected shunt resistor. This shunt resistor is used to measure the current, as described in subsequent sections. Additionally, it is essential to know the voltage across each individual graphene strip. The method for obtaining this voltage is detailed in section 2.2.1. A multiplexer is used to select the appropriate strip for measurement, and this multiplexed signal is then forwarded to the microcontroller for further processing and, in the end, the resistance calculation. Both the variable excitation voltage and the voltage to drive the micro-hotplate are implemented using DACs on the microcontroller and operational amplifiers. The interfacing consists of a Modbus and USB interface. Lastly the components that ensure the feasibility through reliable power supply are detailed.

2.2 Detailed Description

2.2.1 Hardware

The hardware design can be broken down into several components:

- Microcontroller
- Sensor Array Connection
- Sensing
- Actuating
- Interfacing
- Powertree

The schematic in figure 2.2 shows a more detailed version of the schematic including components. In the subsequent sections all the specific components will be described in detail.

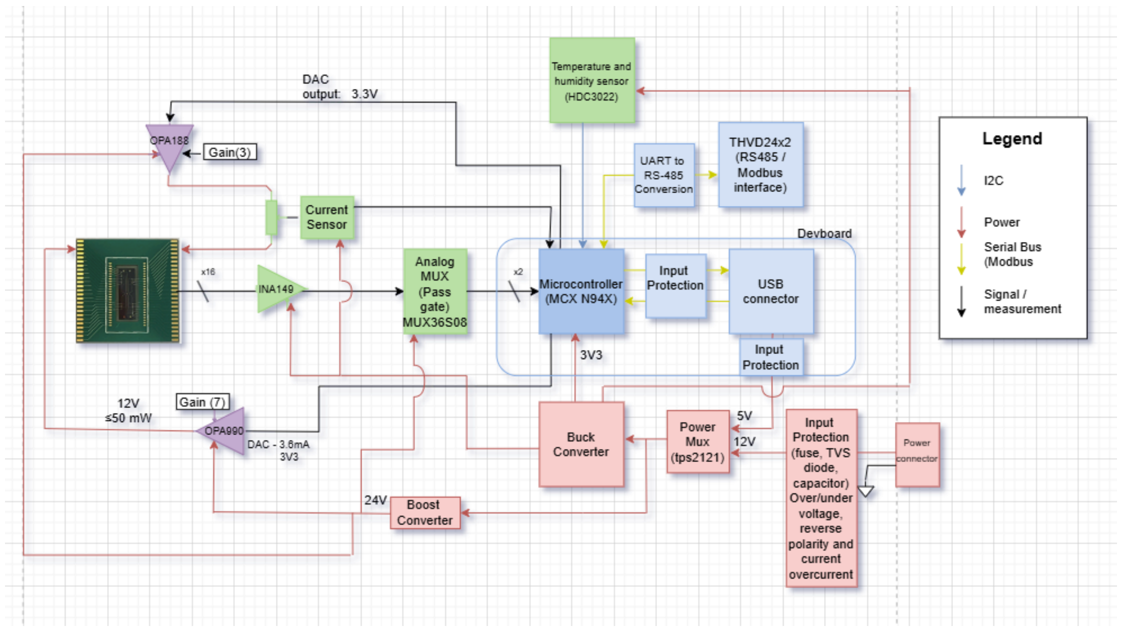


Figure 2.2: Detailed schematic

Microcontroller

One of the main requirements of the chosen microcontroller was ML capabilities. To interface with all of the components that are used in the design it also needed to be equipped with the following:

1. Two high resolution ADCs for determining the current through - and measuring the voltages over the graphene strips.
2. An I2C interface for reading out temperature and humidity readings.
3. A serial interface, UART in this case, to communicate using the modbus interface.
4. A serial interface to communicate using a USB interface.

Based on the requirements described earlier, the MCX N94X was selected from the NXP catalog. This microcontroller also has an NPU, which means that the microcontoller is able to facilitate ML capabilities. In order to reduce complexity and advance the hardware design process, the development board was integrated, such that any hardware related to interfacing with the microcontroller or input protection did not have to be designed from scratch. Male through-hole 2.54mm connectors were placed on the PCB and aligned with the positions on the development board such that the design can easily be connected to said development board. This to ensure ease of use and reliability of mechanical and electrical connections.

Sensor Array Connection

The goal is to ensure the straightforward connection of the sensor array to the readout system in a mechanically and electrically reliable fashion. To achieve this within the given project timeline, standard 0.8mm pitch headers are soldered on the sensor array PCB as shown in 2.3. The sockets are aligned with the headers on the sensor array and soldered on the readout PCB. Since the sensor array is rotational symmetric and there is only one set of connectors that the sensor array can mate with, it is straightforward to connect the sensor array correctly. For further details on the electrical connections of the sensor array, see 2.3.

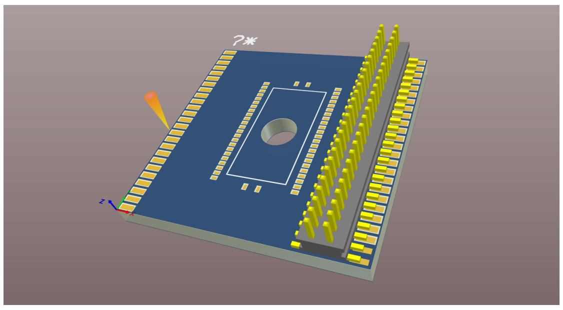


Figure 2.3: Connector Mated to Sensor Array

Interfacing

For interfacing with the prototype, there are essentially two 'communication modes' that should be supported, which stems from the requirements that the client has posed. The first mode is interfacing with a desktop, such that it can be used for data acquisition in a controlled lab setting or verification of the sensor data. In the second mode the prototype functions as an autonomous sensor, the kind of interface that is used here was also influenced by the clients' requirements for the project. For Modbus interfacing a standard RS485 interface is needed, for USB a standard USB-C interface. Hence, to support the two modes of operation, both a USB-C interface and an RS485 interface is integrated into the design.

Sensing

The sensing part of the design consist of several components:

- Temperature and humidity sensor
- Shunt resistor together with INA215 amplifier to determine the current through the sensor array.
- INA149 which together with the Analog MUX is used to measure the voltage.

For further clarification on the used components, please refer to the schematics and datasheets in the appendix.

Temperature and Humidity Sensor

The temperature and humidity sensor is connected with an I2C protocol to the microcontroller. The sensor was selected based on accuracy, input supply voltage, low power consumption and I2C interface and reliable performance across operating conditions Based on these characteristics, the HDC3022[11] was chosen. This sensor is only used for calibration of the collected data. Because the resistance of the graphene strips will vary based on different conditions, it should be known what these conditions are. Currently, this sensor is not yet implemented. However, its inclusion ensures that it will meet future requirements when it is eventually integrated in the software.

Resistance Measurement

To determine the resistance of each of the graphene strips the four-wire kelvin resistance measurement method is used. This method consists using four terminals for each resistance that is to be measured. Two of these terminals are used to supply current to bias the resistor. The other two are used to measure the voltage over the resistance. All of the graphene strips should be under constant bias and by using the four-wire sensing method and by placing all of the graphene strips in series, or daisy-chain all of the strips are under a constant voltage. Graphene strips that are not used are bypassed using a busbar jumper that shorts the input and output terminals of a graphene strip.

Placing all strips in series also means that only one excitation voltage source is necessary to supply the input power. And all graphene strips can be under constant bias. Since the resistances of the graphene strips will not vary wildly during testing (no more than a few hundred Ohms is expected) one voltage source in combination with the total resistance of all of the graphene strips is sufficient for generating this current.

Current Sensing

The method used for accurately determining the current through the graphene strips is a single shunt resistance in combination with a high precision current-sense amplifier. A small shunt resistance is placed in series with the daisy chained graphene strips. It is crucial that this resistance should be small compared to the resistance of the graphene strips, to ensure a small voltage drop and power dissipation in the shunt resistance. Two wires are placed over the shunt resistance, which are the input of a current-sense amplifier. For this amplifier the INA215 [12]is chosen. This output is fed into one of the ADCs on the microcontroller where the measured voltage is converted into a value of the current. The combination of the shunt resistance, the amplifier and the ADC will be referred to as the ”current sensor”. Calculation of the measurement resolution in relation to the current can be found in 2.7.3.

Voltage Sensing The last sensing component is the INA149[3] with the multiplexer. All 8 graphene strips have a difference amplifier with unity gain and high CMRR connected to the voltage sensing terminals. The outputs of all 8 amplifiers are connected to a pass-gate analog multiplexer, which will pass the value of the correct graphene strip to the microcontroller. Based on the 8:1 selection, GPIO interface, low on-resistance and low leakage current the MUX3608[14] was chosen.

After the multiplexer a passive low-pass filter with a cut-off frequency of 10 Hz was added to partially eliminate some of the measured noise.

Actuating

As described in the design requirements section of this report, there are two forms of actuating that the design needs to perform:

1. Generating a programmable DC waveform to actuate the micro-hotplate that is used to dissipate the gas molecules that have been adsorbed onto the graphene strips. This is essentially done to 'reset' or 'clean' the graphene strips.
2. Generate an excitation voltage over the graphene strips in order to measure the resistance of each respective strip.

To be able to create a more in depth design, these requirements can be broken down into more specific specifications so that component selection can be done and the schematics can be created. The further detailed requirements from a hardware perspective look as follows:

1. Programmable DC waveform to drive the micro-hotplate.
   (a) The DC waveform needs to be programmable with an amplitude between 0 and 23 volts. The upper limit of this voltage is determined by the client, as 24 volts and above is treated as high voltage in their environment.
   (b) This amplitude needs to be programmable from the software, preferably the GUI used for interfacing in the 'desktop interfacing mode'.
   (c) As found in previous testing of the hotplate, at a 12V input voltage, it draws around 30 mW of power, which amounts to 2.5 mA of current. Thus, the source driving the hotplate should be able to supply this power.
   (d) Several waveforms should be programmable, as the client will use it to further verify the sensor arrays' performance for different heating patterns.
2. Generate a stable and precise excitation voltage over the graphene strips.
   (a) As this excitation voltage is used to bias the graphene strips, which are the devices under test it is very beneficial for this voltage to be well known and stable. The acceptable range is 1V +-0.1 V per graphene strip.

For the first part of the actuating tasks that the design must perform, several options were considered. Generating programmable, precise DC waveforms can be implemented in an array of different approaches like digital-to-analog converter (DAC) ICs, programmable voltage sources, or signal generators. All of these implementations can be programmed using analog voltages or serial interfaces such as I2C or SPI. In order to reduce complexity and cost in the design, it was chosen to use the integrated DACs already present in the microcontroller itself. These DACs are high resolution and easily programmable using existing and well documented software implementations. The output of the designated DAC is amplified using a non-inverting voltage amplifier. Implementing the actuating voltages using simple non-inverting amplifiers reduced complexity in both hardware and software design, as well as decreasing cost and increasing robustness.

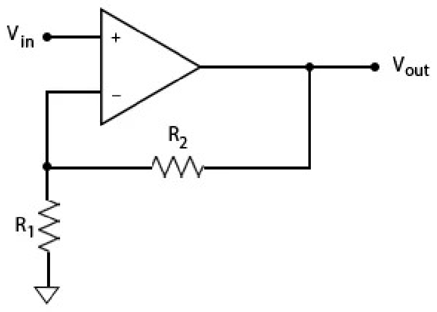


Figure 2.4: Non-Inverting Voltage Amplifier

The gain of a non-inverting voltage amplifier as shown in 2.4, is set using the resistors in the feedback circuit. The gain can be found by:

$$\text{Gain} = 1 + \frac{R_2}{R_1} \tag{2.1}$$

For the driving of the hotplate, one of the DACs is used in conjunction with the OPA990[16] low-power amplifier from Texas Instruments. This is a widely used catalog amplifier with the appropriate voltage and power ranges, has a rail-to-rail output, and very low offset voltage. As the frequencies that the output waveform will take are relatively low, frequency-related performance specifications were not considered as much. The OPA990 is used in a simple non-inverting topology with a voltage gain of 7 set by the feedback circuit. By using a gain of 7 the voltage amplitude that can be achieved using the DAC ranges from 0V to 23.1V. This is a reasonable range within the prescribed limits.

For the generation of the excitation voltage, which had more strict precision requirements, the OPA188[15] precision amplifier was chosen. It was chosen on the basis of voltage and power ratings, rail-to-rail output, small offset voltage, and high common-mode rejection ratio (CMRR). With the chosen voltage gain, the excitation voltage that can be generated ranges from 0V to 9.9V given that the input voltage of this amplifier, which is generated by one of the ADCs on the microcontroller, ranges between 0V and 3.3V.

Powertree

The powertree is the part of the design that facilitates the power of the entire system. The power tree requirements are derived from the power specifications of the selected components and the available input power.

Since there are two possible power inputs, one 5V and one of 12V, a power multiplexer is used. That is, the TPS2121[17] power multiplexer by TI. This is a priority power multiplexer, which means that it selects one of the power inputs based on the specified priority. It has an operating range form 2.8 to 22V which is in the range of the inputs. Since in the earlier stages of the use of the product will mainly be in the lab, the priority was given to the 5V power input which is supplied by the USB-C connection. The power multiplexer also has a status pin, which can be used to identify which of the two modes the system is functioning in: desktop or autonomous sensor. Finally, the power multiplexer also integrates input power protection against overvoltage, overcurrent, and reverse current. The 5V input generated on the development board is assumed to be sufficiently safe and less prone to human error. Since this is not the case for the 12V power input, which will be generated by connecting a power supply to the designated pins, an extra layer of protection has been added in the form of a simple input protection circuit protecting against overcurrent, voltage transients and electrostatic discharge (ESD).

From the output of the power multiplexer the buck and the boost converter generate the 3.3V and 24V power, respectively. These components were selected on the basis of the power requirements of the design. Looking at the power consumption of all the 3.3V and 24V components in their respective datasheets, the total power consumption can be calculated in terms of current draw, and the components can be selected. Based on these requirements, the TPSM82903[18] buck converter module and the LM27313[13]boost converter were selected.

To verify the proper generation of the necessary voltage levels, two power indication LEDs and three test points were added to the PCB for troubleshooting and diagnostics.

2.3 Use

There are essentially two use cases for the design. Both cases involve measuring the resistance of up to 8 of the graphene strips in the sensor array and driving the hotplate at given intervals. The two use cases are distinguished by the interfacing and power supply. Connecting the sensor array in both use cases is an identical procedure. The sensor array can be connected to the designated connectors shown in 2.5.

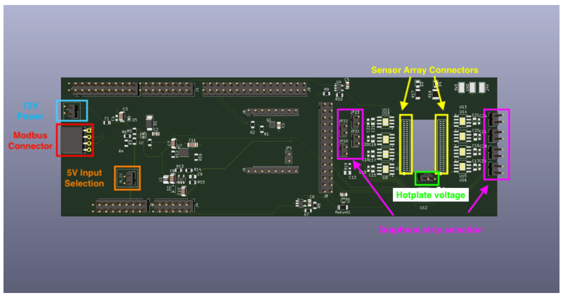


Figure 2.5: Front Facing View of the Design

Use Case 1: Desktop Interface

The first use case, likely in a lab or office, is connecting the product to a desktop using a USB-C to USB-A cable. If this is the use case, take care of connecting the power supplies and placing the appropriate busbar jumpers. The jumper used to select the 5V generated on the development board should be placed on the connector, which is the bottom left jumper indicated in orange in 2.5.

In this case, the interfacing is done using the serial port on a desktop and the graphical user interface, for which all the necessary software can be downloaded from the provided links.

Use Case 2: Autonomous Sensor

In the second use case, the jumper used to select the 5V power generated on the development board should not be placed. A power supply should be connected to the power connector, which is the top left connector indicated in blue in 2.5. The jumper should be placed on the connector next to it as well to connect the power supply to the input of the power multiplexer so that it can be used to power the sensor readout system. First connect the appropriate jumpers and the Modbus interface to the connector indicated in the image, then connect a power supply which is set to 0V and is current limited to 0A. Take care to first connect the ground connection of the power supply to the - terminal on the power connector, then connect the positive terminal. Set the voltage to 12V and slowly increase the current up until the LEDs indicating that the appropriate voltage levels are generated turn on.

Graphene Strip Selection

With the current design, it is possible to use jumpers on the pins indicated in purple in 2.5 to bypass selected graphene strips in case one of them is broken, or only a few of them are being measured. Consult the schematics and silkscreen to see which jumper should be placed to bypass each graphene strip. When not all strips in the array are used, take care to specify this in the software as well.

Testpoints and Verification

In case of verification of a sensor array or when testing the readout system, several test points have been added. This is done to simplify testing and looking for faults in case of errors.The connector marked in green in 2.5 can be used to measure the voltage over the hotplate to verify thermal behaviour and the generated waveforms. On the top right of the board testpoints for the voltage levels are added. In case of debugging, for example the serial interface, consult the schematics to determine which pins to connect each probe to.

2.4 Software Overview

This section will give a general overview of the software. It will describe the functionalities and connection between multiple Analog-to-Digital Converters (ADCs), Digital-to-Analog Converters (DACs), a multiplexer (MUX) and different communication protocols. A simplified block diagram whose main purpose is to get a clear overview can be seen in figure 2.6. One of the main principles during software development was to make the software as modular as possible, ensuring that each component handles a specific task, which enhances maintainability and scalability. The details of the different components are discussed in the subsequent section.

2.4.1 Interfacing

Communication with the microcontroller is essential. All relevant parameters can be set either through the Python GUI or ModBus. Furthermore, the results; an array containing a timestamp and resistance values of the corresponding graphene strips, can be read out by the same protocols. The Python GUI also employs a real time plot for visualization and a download option to download the raw collected data. Details about the Python and Modbus communication can be found in section 2.8 and 2.9 respectively.

2.4.2 Main Control Loop

At the heart of the software lies the main control loop. This loop is the central point of execution, continuously running and checking if there are new parameters available, and if so restarting the system using these new parameters. Furthermore, it continuously sends the results of the resistance measurements to the python GUI if connected. If the Modbus protocols is used, the results are stored in a global variable which can then be read when a request to read data is received. The parameters can also be read and written when such a request is received.

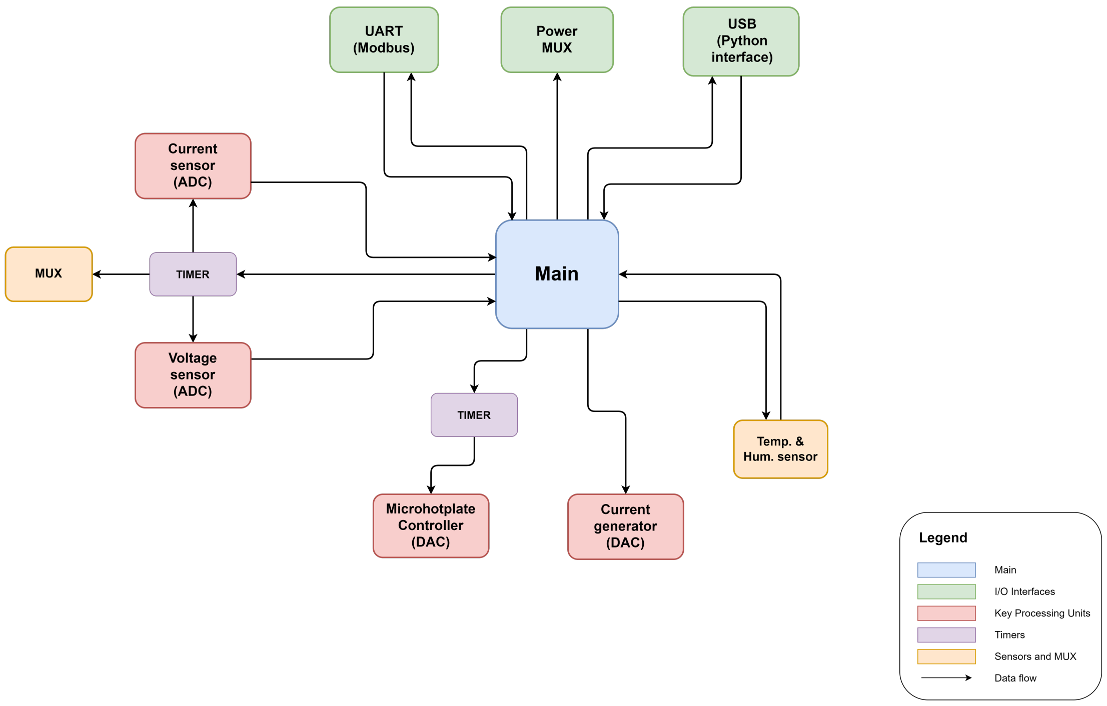


Figure 2.6: software overview

2.4.3 System Initialization

The system receives all required parameters (2.4.3) through a ModBus interface or the Python GUI, these are stored as global variable ready to be used. After components are initialized with these parameters, this includes the two ADCs, two DACs, MUX and all other necessary initializations, such as clocks, pins and the debug console.

- **Parameters**
  - Desired amplitude for the signal on the micro-hotplate.
  - Desired frequency for the signal on the micro-hotplate.
  - Type of waveform for the signal on the micro-hotplate.
  - Duty cycle of the wave
  - Excitation voltage per strip
  - Active strips to be read out

2.4.4 MUX Selection and Data Acquisition

The integration of the Multiplexer (MUX) allows the system to dynamically select the appropriate channel for data acquisition. After determining a valid channel, a graphene strip, the voltage can be read out using an ADC. This signal can then be processed as described in section 2.7. After the processing is done, an array consisting of a timestamp and resistance value is the result, which can be used to make a real-time plot in the GUI or read out via Modbus.

2.4.5 Micro-hotplate and Excitation Voltage Control

Besides the collection of data through the ADCs, two DACs are used to control the micro-hotplate and excitation voltage of the graphene strip. A detailed explanation can be found in sections 2.5 and 2.6.

2.4.6 System Monitoring and Error Handling

Throughout the execution of the code, the system continuously monitors its state and checks for the detection of any anomalies. This monitoring ensures any deviations from the expected behaviour is identified and displayed to the user through the serial monitor in the python GUI. This includes, for example, invalid parameters or failed initializing of subsystems. This ensures robustness and helps the user identify any faults in the system.

2.5 Micro-hotplate control

In order to control the micro-hotplate that is used to evaporate any residual gas molecules left in the cavity of the graphene strips, a DAC together with an operational amplifier will be used.

The DAC needs to operate in a very low frequency range in order to allow the heater to be on for multiple minutes with regular intervals. The core logic of the DAC operation is divided into two main components: one that determines the actual output value and the other that controls the timing of these outputs. The first component calculates an array of values representing one full cycle of the waveform, while the second component manages the timing for when these values are outputted.

2.5.1 Array of Values

The DAC can generate four types of waveforms: square, triangle, sawtooth, and sine waves. The values that correspond to these waves are calculated once in the setup phase; this is done to avoid real-time calculations, which could potentially slow down the process.

In order to generate a sine wave a function `generateSineWave` is used. An angle increment based on the desired frequency and sample rate is determined. Each sample is then computed using a phase shifted sine function to ensure the wave always starts from zero. The values are then adjusted to fit the entire output range of the DAC. The formula used can be seen in 2.2.

$$\text{values}[i] = \left(\sin\left(i \cdot \Delta\theta - \frac{\pi}{2}\right) + 1\right) \times 8191.5 \tag{2.2}$$

where

$$\Delta\theta = \frac{2\pi \cdot \texttt{desiredFrequency}}{\texttt{sample\_rate}} \tag{2.3}$$

The function generateSquareWave generates a square wave by filling an array with sample values based on the specified duty cycle. First, it checks if the duty cycle is within a valid range (0-100). Then, it calculates the number of samples that should be ”on” (high value) and sets those samples to the maximum DAC value. The remaining samples are set to zero. The formulas used can be seen in 2.4 and 2.5 .

$$\texttt{onSize} = \left\lfloor \frac{\texttt{size} \times \texttt{dutyCycle}}{100} \right\rfloor \tag{2.4}$$

$$\text{values}[i] = \begin{cases} 16383 & \text{for } 0 \leq i < \texttt{onSize} \\ 0 & \text{for } \texttt{onSize} \leq i < \texttt{size} \end{cases} \tag{2.5}$$

Finally, the function generateTriangleWave generates a triangle wave by calculating sample values based on the specified duty cycle. If the duty cycle is set to 100, it produces a sawtooth wave. The function first checks if the duty cycle is within a valid range (0-100). It then calculates the number of samples for the rising edge (`onSize`) and the slope of the wave. For the rising edge, values increase linearly, and for the falling edge, values decrease linearly. The formulas used can be seen in 2.6 and 2.7

$$\texttt{slope} = \frac{16383}{\texttt{onSize}} \tag{2.6}$$

$$\text{values}[i] = \begin{cases} i \times \texttt{slope} & \text{for } 0 \leq i < \texttt{onSize} \\ 16383 - (i - \texttt{onSize}) \times \texttt{slope} & \text{for } \texttt{onSize} \leq i < \texttt{size} \end{cases} \tag{2.7}$$

After an initial array has been calculated, it still needs to be scaled to the desired amplitude as specified by the user through the GUI, taking into account the gain of the operational amplifier used to amplify the signal. For this, 2.8 and 2.9 are used.

$$\texttt{maxDacValue} = \left(\frac{\texttt{desiredAmplitude}}{\texttt{opamp\_gain}}\right) \times \left(\frac{16383.0}{3.3}\right) \tag{2.8}$$

$$\text{values}[i] = \left(\frac{\text{values}[i]}{16383.0}\right) \times \texttt{maxDacValue} \tag{2.9}$$

2.5.2 Timing

To ensure the DAC output is updated at a precise interval required for a stable and precise waveform, an on-chip timer is configured to trigger interrupts at the desired rate. This process integrates hardware timer capabilities with interrupt-driven software to maintain the timing accuracy required for waveform synthesis.

The timer is thus configured to generate periodic interrupts at a frequency determined by the sample rate. The timer is attached to a 12MHz clock source which is used to setup the match value, resulting in equation 2.10.

$$\texttt{matchValue} = \frac{12000000}{\texttt{sample\_rate}} \tag{2.10}$$

When this match value is reached, an interrupt is generated. The NVIC (Nested Vectored Interrupt Controller) is configured to handle these interrupt by invoking a special interrupt service routine (ISR). This ISR clears the interrupt flag, after which it sets and increments an index that is used to determine the correct scaled value from the array and then sets the DAC to this value. This approach guarantees that the frequency will be accurate and stable.

2.6 Excitation Voltage

For the measurement of resistors, such as the array of graphene strips, it is fundamental to have a specific and known current running through the array of strips. As all graphene strips are aligned in series, this current will always be the same through all of the strips. To supply this current a Digital-to-Analog Converter (DAC) is used. A DAC is not able to generate a current directly but it produces a voltage. This voltage generates a current through the daisy chained graphene strips. As the exact current generated by the DAC will be measured with an ADC, it is not critical to have a DAC with a huge resolution and thus a very accurate output. Therefore the integrated DAC of the microcontroller, with a resolution of 12 bits and an output range between 0 and 3.3 V, suffices. This range and resolution corresponds to a DAC which can be controlled with voltage steps of:

$$\text{Voltage Step} = \frac{3.3\,\text{V}}{2^{12}-1} = \frac{3.3\,\text{V}}{4095} \approx 0.80586\,\text{mV/step} \tag{2.11}$$

A requirement of the project is to generate a stable excitation voltage of 1V (+- 0.1V) for each strip. The excitation voltage generator is responsible for meeting this requirement. The array of graphene strips can have up to 8 strips connected at any given time. Therefore, the DAC must generate enough voltage to excite 8 strips with 1.0 V each, translating to an excitation voltage of 8 V (8.8 V considering the error margin). Given the DAC's maximum output range of 3.3 V, an amplifier with a gain of 3 is placed between the DAC and the array of graphene strips. This configuration allows the DAC to excite the strips with a maximum voltage of approximately 9.9 V.

To ensure proper operation, the total voltage of the array should never exceed 8 V, and the voltage per strip should always be between 0.5 V and 1.0 V. Safety checks are implemented in the code to prevent the user from setting parameters outside their functional range. The formula to determine the correct value for which the DAC should be set is as follows:

$$\text{DAC Value} = \left(\frac{\frac{V_{\text{excitation}} \times N_{\text{resistors}}}{G_{\text{amplifier}}}}{3.3\,\text{V}} \right) \times 4095 \tag{2.12}$$

where:

$V_{\text{excitation}}$ is the excitation voltage per resistor,
$N_{\text{resistors}}$ is the number of connected graphene strips,
$G_{\text{amplifier}}$ is the amplifier gain, which is 3.

If all 8 graphene strips would be connected to the array, and per strip an excitation of 1 V is desired, the calculated DAC value to be set would be;

$$\text{DAC Value} = \left(\frac{\frac{1.0 \times 8}{3}}{3.3} \right) \times 4095 \approx 3309 \tag{2.13}$$

This ensures that the excitation voltage remains within the specified range, providing stable accurate current for the measurement of the graphene strips.

2.7 Resistance Measurement System

The resistance measurement system is composed of two coherent sensors: one sensor to measure the voltage across each graphene strip and another sensor to measure the voltage across a shunt resistor, to indicate the current through the sensor array. As resistance is the key parameter for determining the concentration of various gasses surrounding the gas sensor array, it is crucial to accurately measure and analyze the resistance values of the graphene strips.
Given that only one graphene strip can be measured at a time with the ADC voltage sensor, and considering there are up to 8 graphene strips to be measured, a multiplexer is employed to sequentially select each strip. The multiplexer operates at a user-defined frequency, ensuring that each strip is measured accurately and effectively. As one of the requirements set by the supervisor was to be able to measure the sensor values at a rate of 1 Hz per strip, the user can set this parameter to the amount of connected strips. The system uses a second ADC for measuring the voltage drop across a shunt resistor to determine the current flow through the array of graphene strips. For convenience, the ADC that measures the voltage across the shunt resistor will be referred to as the ”current sensor”.

To synchronize the voltage and current sensors with the multiplexer and ensure that correct values are measured for the corresponding graphene strip, precise timing is essential. An integrated timer is utilized to keep track of the exact measurement times for each strip. This ensures that the measurements are coherent and accurately reflect the conditions at the time for each reading.
The complete measurement system is illustrated in Figure 2.7, showing the interaction between the sensors, multiplexer, and timer, and how they work together to provide accurate resistance measurements.

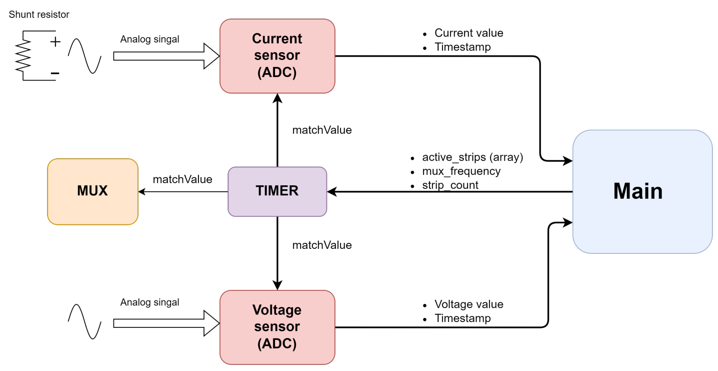


Figure 2.7: Resistance measurement system

2.7.1 Timer Setup and Interrupt Handling

The timer is a critical component in the resistance measurement system (as can be seen in section 2.7), ensuring that voltage and current measurements are synchronized, and that the measurements are done per strip without any overlap in values. This synchronization is essential for accurate calculation of the resistance of the graphene strips. The timer generates periodic interrupts at a user-defined frequency, triggering the ADCs to start new conversion cycles.
The timer uses an integrated clock frequency of 12 MHz, which can be retrieved from the microcontroller. With this clock frequency and the user-defined switching frequency between each strip, the total amount of clocks ticks per strip can be calculated using the following equation:

$$\texttt{matchValue} = \frac{12\,\text{MHz}}{\texttt{mux\_frequency}} \tag{2.14}$$

where:

- `matchValue`: The number of clock ticks per strip.
- 12 MHz: A standard integrated clock.
- `mux_frequency`: The user-defined switching frequency.

When the internal clock counter matches this `matchValue`, an interrupt is triggered, starting a new conversion cycle for the voltage and current ADCs. This ensures that the measurements of voltage and current are synchronised, providing accurate data for resistance calculation.
The timer also receives another user-defined parameter, the `active_strips` array. This array is a list of booleans that align with the graphene strips connected to the array. It informs the multiplexer which channels are active (i.e., have a graphene strip connected) and which channels are shorted, providing useful information for the ADC to convert.

A simple loop processes this array, keeping track of a full conversion cycle of all graphene strips. Each time the timer interrupt is triggered, a new active channel is measured. When all active channels are measured, a flag is set, indicating that the data is ready to be updated with the new ADC values and their corresponding timestamps, obtained during that cycle.

For better understanding, consider an example where there are 5 active graphene strips and the `active_strips` array is as follows:

```
active_strips = [true, false, true, true, false, false, true, true]
```

In this array:

- `true` indicates that the graphene strip is active and connected.
- `false` indicates that the channel is shorted and does not provide useful information.

With this configuration, the active channels are strips 0, 2, 3, 6, and 7. The multiplexer will cycle through these active channels during each timer interrupt, ensuring that only the connected strips are measured.

2.7.2 Voltage Sensor

The voltage sensor is a crucial component in the system as it is responsible for accurately converting the analog voltage signals into digital values that the microcontroller can process. The use of a high-resolution ADC (Analog-to-Digital Converter) ensures that the system can accurately measure the voltage across the graphene strips. This accuracy is essential for determining the precise resistance values of the graphene strips, making the voltage sensor a vital component that ensures reliable resistance measurements. As is previously discussed in the hardware section, the integrated ADC of the microcontroller is chosen for the voltage sensor. This sensor operates using the Succesive Approximation Register (SAR) method. This method is well suited for high-resolution applications as this method approximates the signal through a series of comparison steps until the desired resolution is achieved, which is 16 bits for this specific ADC. The reference voltage range is programmable and therefore up to the software group to decide. An ADC will provide the highest resolution when the reference voltage range is closely matched to the range of the voltage being measured. This can be best illustrated by comparing two examples, where the theoretically measured voltage is 0.9 V:

- **Example 1: Reference Voltage Range of 0 to 3.3 V**

$$\text{Voltage Step} = \frac{3.3\,\text{V}}{2^{16}} \approx 50.35\,\mu\text{V/step}$$
$$\text{ADC Value} = \frac{0.9\,\text{V}}{50.35\,\mu\text{V/step}} \approx 17870\,\text{steps}$$

- **Example 2: Reference Voltage Range of 0 to 1.2 V**

$$\text{Voltage Step} = \frac{1.2\,\text{V}}{2^{16}} \approx 18.31\,\mu\text{V/step} \tag{2.15}$$
$$\text{ADC Value} = \frac{0.9\,\text{V}}{18.31\,\mu\text{V/step}} \approx 49152\,\text{steps} \tag{2.16}$$

In these examples, it is clear that using a reference voltage range that is closer to the measured voltage (0 to 1.2 V) results in a higher ADC value and therefore better resolution compared to a wider reference voltage range (0 to 3.3 V). This demonstrates the importance of selecting an appropriate reference voltage range for achieving the highest resolution in ADC measurements. As the voltage range per strip will be in the range from 0.5 V to 1.0 V a reference voltage of 1.2 V is chosen for an added error margin. Therefore the voltage step in which the voltage sensor will operate is already calculated in equation 2.15.

The setup and initialization of the ADC are essential for precise voltage readings. At this stage, the reference voltage is adjusted to 1.2 V, and the high-resolution mode is enabled. Additionally, the microcontroller's built-in ADCs can operate in a 12-bit mode, referred to as the low-resolution mode in the settings, however this will make the readings less accurate, which is undesired. This relationship can also be demonstrated mathematically, as shown in equation 2.15. To reduce noise, hardware averaging has been implemented that averages the first 1024 converted values.

By calibrating the ADC, errors are minimized to ensure accurate measurements. During initialization, the ADC undergoes offset and auto-calibration processes. Offset calibration corrects inherent biases in the ADC readings, while auto-calibration adjusts the ADC settings to optimize performance. The ADC operates in single-ended mode, comparing values relative to the microcontroller's ground, which was a design choice made in order to simplify the hardware design.

The ADC acquires voltage readings from each connected graphene strip through the multiplexer. The conversion results are stored in a buffer and processed to determine the voltage across each connected strip. The timer triggers ADC conversions only for the connected strips, as indicated by 'true' values within the `active_strips` array. Although shorted channels are not measured, it is more efficient to process the voltage values in a standardized format. Consequently, an 8 by 2 array is used to store the ADC values, with 8 representing the maximum number of strips that can be connected. This array is always initialized with its values set to 0, ensuring a consistent structure for processing and accounting for the short-circuited channels.

The ADC itself also uses an interrupt to handle the conversion process efficiently. Shortly after the timer interrupt is triggered, the next active channel will be measured. When this conversion is complete, the ADC interrupt is triggered, and the conversion result is retrieved and stored. This interrupt ensures timely and accurate data acquisition.

Error handling mechanisms are implemented to ensure reliable ADC measurements. There is a check to validate the ADC readings to detect and remove any anomalies. This validation process involves checking the range and consistency of the readings to ensure they are within expected limits.

2.7.3 Current Sensor

The current sensor is a fundamental component in the resistance measurement system, as it enables the calculation of resistance using Ohm's Law, which states that $R = \frac{V}{I}$. Like the voltage sensor, the current sensor utilizes a 16 bit resolution ADC with a variable reference voltage, integrated within the microcontroller. Since it is not possible to directly measure current with an ADC, the current sensor operates by measuring the voltage drop across a known shunt resistor. This allows for the calculation of the current flowing through the graphene strip array using Ohm's Law again. With the shunt resistance being known and very stable at 120 Ohms, together with the voltage readings over this shunt resistor, it is possible to accurately calculate the current flowing through the entire array.

As it is very beneficial to use the exact same reference voltage for the voltage sensor ADC and the current ADC, a voltage has to be measured by the current ADC that comes close to the 1.2 V reference voltage that is also used for the voltage sensor. This is beneficial because the reference voltage will cancel out in the numerator and denominator when using Ohm's Law for the final resistance calculation, thereby simplifying the computation and reducing potential sources of error. To check if this is the case a small calculation can be made. For instance, if the current flowing through the array is approximately 0.1 mA and the shunt resistor has a resistance of 120 Ohms, the voltage drop across the resistor can be calculated as follows:

$$V_{\text{shunt}} = I \times R_{\text{shunt}} = 0.1\,\text{mA} \times 120\,\text{Ohms} = 0.012\,\text{V}$$

This voltage is then amplified by a factor of 75 to make it readable by the ADC:

$$V_{\text{amplified}} = V_{\text{shunt}} \times \text{Amplifier Gain} = 0.012\,\text{V} \times 75 = 0.9\,\text{V}$$

Given this amplified voltage, a reference voltage of 1.2 V is suitable for the ADC. This selection ensures that the ADC can accurately read the amplified voltage without hitting the upper limit of its range, providing a margin for measurement fluctuations. This can be calculated as follows:

$$\text{Reference Voltage} = 1.2\,\text{V}$$

With a 16-bit ADC, the voltage step size can be calculated as:

$$\text{Voltage Step} = \frac{1.2\,\text{V}}{2^{16}} \approx 18.31\,\mu\text{V/step}$$

The amplification is chosen to maximize the resolution of the ADC within the expected voltage range, ensuring precise current measurements.

As shown in Figure 2.7 the current sensor uses the same timer as the voltage sensor, ensuring synchronized measurements and maintaining coherence in the collected data. When the timer interrupt occurs, the ADC for the current sensor starts a new conversion cycle, measuring the voltage drop over the shunt resistor. Since the current flowing through the series-connected graphene is consistent across each strip, the switching between channels via the multiplexer is less critical for current measurement. However, by measuring the voltage over the shunt resistor every time the channel changes, the system collects new data, which can be summed and averaged to improve the precision of the current measurement. Which is fundamental for accurate resistance measurements.

The current sensor ADC, very similar to the voltage sensor, undergoes offset and auto-calibration during initialization. This ensures that any inherent biases are corrected, and the ADC settings are optimized. The current sensor also operates in single-ended mode.

2.7.4 Averaging

In order to reduce the effect of noise on the measurements, simple averaging is implemented. This is done by dividing the match value of equation 2.14 by the desired amount of samples to be averaged. For each strip, 30 samples are used in the averaging process. In total, we average over 1024x30 samples, where 1024 represents the hardware-averaged samples and 30 represents the software-averaged samples. Consequently, a total of 30,720 samples are utilized to determine the value of a single strip.

2.7.5 Resistance Calculation

The values obtained from both ADCs can now be converted to a resistance value for each strip in Ohms. The timestamps from the array containing the ADC values from the voltage across the graphene strip are preserved. The calculation can now be done using the following formula:

$$R = \frac{\left(ADCVoltage \times \frac{V_{ref}}{65535}\right)}{\left(\frac{\left(ADCCurrent \times \frac{V_{ref}}{65535}\right)}{75 \times 120}\right)} \tag{2.17}$$

Here, 75 is the gain of the operational amplifier (OpAmp) and 120 is the resistance of the shunt resistor used. This equation simplifies to:

$$R_x = \frac{ADCVoltage \times 75 \times 120}{ADCCurrent} \tag{2.18}$$

The resulting resistance is then saved in an array together with the corresponding timestamp.

2.8 Serial Communication and GUI

This chapter will explain the workings of the communication system that links the microcontroller and the Graphical User Interface (GUI). This communication is done via a UART protocol over a USB type-C connection with the microcontroller. It facilitates seamless parameter setting, data collection and real-time visualization of the results.

2.8.1 Microcontroller Firmware

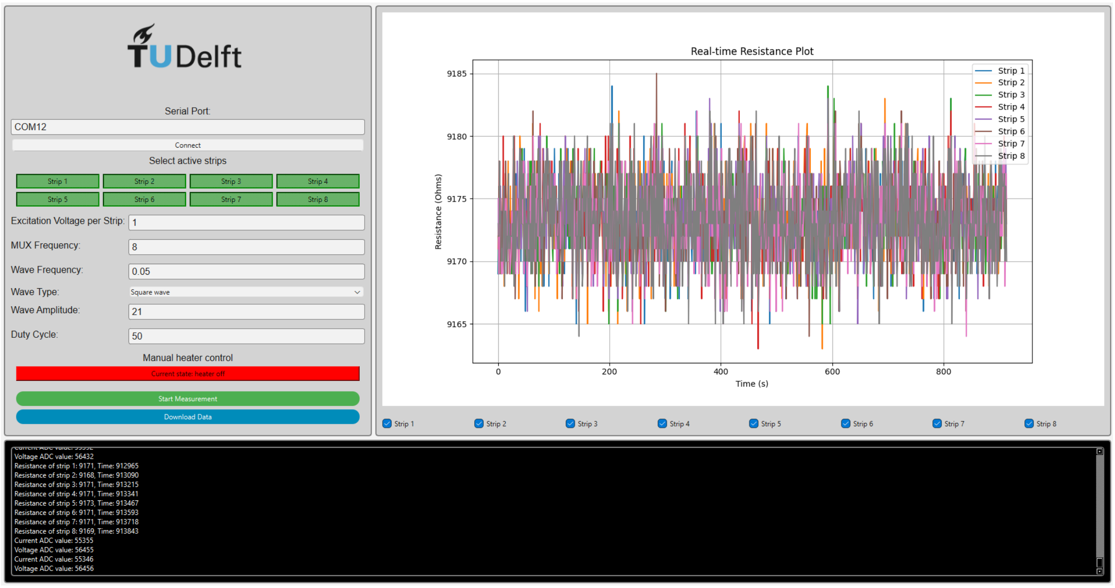
After all the subsystems are initialized, the Microcontroller waits for input from the Python GUI. This data will be in the form of a string in a specific format. This string is parsed while having multiple error handling functions that signal to the user if there is an error with the received string or parsing. After the parsing is complete the global variable representing the parameters on the microcontroller are set after which the system starts the measurements and calculates the resistance. This resistance is in the form of an array which will be communicated to the Python code for further processing.

2.8.2 Python GUI

The Python interface was built using PySide6, which allows for a high-quality modular GUI. This modularity arises from the use of three separate frames; this is done to make the Python GUI future proof and easy to work on. If certain features are to be added in the future, a frame containing these features can simply be added. This requires little to no knowledge about the rest of the code, making it easy to work on for other developers. Currently, three frames are implemented:

- A frame that mimics a serial monitor and prints all the received data from the microcontroller. This allows the user to see error messages and verify the incoming data.
- A frame that contains all the input fields for all parameters and the buttons to start the measurement and download the data.
- A frame that contains the live visualization of the received data.

A screenshot of the GUI during execution can be seen in figure 2.8.



Serial Port: COM12

Excitation Voltage per Strip: 1

MUX Frequency: 8

Wave Frequency: 0.05

Wave Type: Square wave

Wave Amplitude: 21

Duty Cycle: 50

```
Voltage ADC value: 56432
Resistance of strip 1: 9171, Time: 912965
Resistance of strip 2: 9168, Time: 913090
Resistance of strip 3: 9171, Time: 913215
Resistance of strip 4: 9171, Time: 913341
Resistance of strip 5: 9173, Time: 913467
Resistance of strip 6: 9171, Time: 913593
Resistance of strip 7: 9171, Time: 913718
Resistance of strip 8: 9169, Time: 913843
Current ADC value: 55355
Voltage ADC value: 56455
Current ADC value: 55346
Voltage ADC value: 56456
```

Figure 2.8: Screenshot of the GUI in use, displaying real-time data and control options for monitoring and analyzing the resistance measurements.

When users input their desired parameters and initiate a measurement, the Python code validates all inputs and shows a message if there is an invalid input or the serial communication setup with the microcontroller has failed. If the inputs are valid, it sends a correctly formatted string to the microcontroller with all required data.

The incoming data consisting of resistance values in Ohms and timestamps in milliseconds are then parsed and stored in a csv file, which a user can download for further research or processing if required. The GUI leverages Matplotlib's FigureCanvasQTAgg, which wraps the image, resulting in a Qt Widget compatibile with PySide6. This allows for a plot compatible with the GUI that also incorporates all features Matplotlib has to offer. This allows for high-quality plots that can easily be adapted to specific needs if required. The resulting plot is updated every 10 seconds using the stored data. Furthermore, users can toggle between the results of the individual graphene strips to enhance the visualization.

2.8.3 Threading and Asynchronous Data Handling

The system's ability to handle asynchronous data is a cornerstone that ensure the robustness of the GUI. The Python GUI employs threading and a signal-slot mechanism to manage asynchronous data communication effectively and efficiently. The serial communication is fully handled as a separate thread, which ensures that serial communication does not interfere with other GUI operations. This results in the GUI being guaranteed to be responsive even in the cause of large data streams.

The signal slot mechanism, a feature of PySide6, facilitates smooth data handling. Whenever new data is received from the microcontroller, a signal is emitted. This signal in turn is connected to slots that update the serial monitor and write data to the csv file, ensuring data handling is efficient and immediate. This approach guarantees that the GUI remains responsive even if a measurement of longer duration results in a large amount of of data points.

2.9 Modbus Protocol Implementation

In order for the sensor readout system to be applicable to actual real-world situations, there must be a communications protocol in place to be able to connect it to a server and send its' data. A, industry standard protocol that facilitates such communication is Modbus. Modbus is found in the application layer of the OSI model and consists of request and response communication between a master and a slave [7]. Every transaction is initiated by the master and requires a response from the client in the server, to whom the request was directed.

2.9.1 Modbus Data Packet

At the core of the Modbus protocol lies the protocol data unit (PDU). This consists of a byte for the function code followed by 252 possible bytes of data. The function code specifies for the slave what should be done with the rest of the PDU, since data can be either written or read from registers in the slave device. The rest of the packet is determined by the physical layer through which the communication occurs. A complete Modbus protocol unit is called the application data unit (ADU). As specified, the Modbus protocol of this sensor readout system uses RS485 as the physical layer. The ADU for this physical layer consists of an address (1 byte), followed by the PDU (253 bytes) and an error check (2 bytes) and thus consists of 256 bytes.

2.9.2 Modbus RTU

There are two modes controllers in a Modbus network using serial communication can be set up to communicate through: ASCII and RTU (Remote Terminal Unit). For this application, RTU mode is used

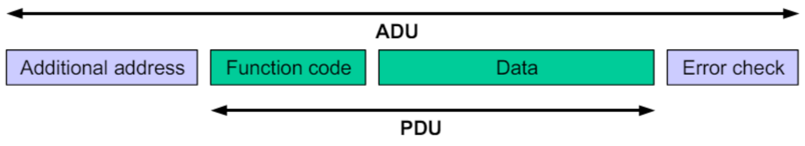


Figure 2.9: Modbus PDU and ADU [7]

as it allows for greater throughput of data. In RTU mode each 8-bit byte in contains two 4-bit hexadecimal characters, whereas ASCII mode allows for 7 bits of data per byte. RTU mode also includes a 16 bit CRC (Cyclic Redundancy Check) for error detection.

2.9.3 Modbus via UART

The MCX N94 microcontroller contains 10 low-power Flexible Communications modules called LP_FLEXCOMM. Each of these module supports Universal Asynchronous Receiver/Transmitter (UART) functions [6]. The TTL485 V-2.0 hardware module on the PCB ensures that the data packets sent from the RS485 line to the microcontroller is suitable for UART and the other way around. The UART has to be initialized using the same configuration as the master device. The configuration includes:

- Baud rate
- Number of data bits
- Stop bits
- Parity mode
- Request to send (RTS)

2.9.4 Data Model

The Modbus protocol allows for various types of data inputs and outputs. These inputs and outputs are organized into different types of registers, each serving a specific purpose:

- **Coils (Discrete Outputs)**: These are single-bit registers used to control binary output devices such as relays. Coils can be read and written by the master device.
- **Discrete Inputs**: These are single-bit read-only registers that represent the current state of binary input devices, such as switches or sensors.
- **Holding Registers**: These are 16-bit registers used to store configuration parameters, setpoints, or any other data that needs to be writable and readable. Holding registers are accessible by both read and write operations.
- **Input Registers**: These are 16-bit read-only registers that typically hold measurement data, such as sensor readings or calculated values.

For the sensor readout system, the master must be able to read and write different system parameters and read the sensor data. Therefore, the parameters will be stored in holding registers and the sensor data in input registers. The addresses of these registers have to be defined by the slave and have to be clearly documented in order for the master to know how to access the data.
Whenever the master requests sensor data, the readout system will send current the resistance values of all active graphene strips. This means that the address range of the input registers will be 1-8, with input register address 1 being strip 1 and so on. The resistance values will be send as 16 bit unsigned words.

2.9.5 Supported Function Codes

The Modbus protocol supports a variety of function codes that define the actions to be performed by the slave device. When the master sends a request to read or write data, it specifies the start address for the corresponding register and the amount of registers to be read. For the sensor readout system, the following function codes are implemented:

- **0x03 - Read Holding Registers**: This function code is used to read the contents of a contiguous block of holding registers in the slave device. If this function code is received, the system parameters will be send to the master device.
- **0x04 - Read Input Registers**: This function code allows reading from input registers. If this function code is received, the resistance values of the graphene strips will be send to the master device.
- **0x06 - Write Single Register**: This function code writes a value to a single holding register. If this function code is received, a single system parameter specified by the master can be updated.
- **0x10 - Write Multiple Registers**: This function code writes values to multiple contiguous holding registers. If this function code is received, multiple system parameters can be updated at once.

2.9.6 Error Handling

In order to ensure reliable communications within the Modbus protocol, error handling needs to be present in the slave device. Exception codes are defined in the Modbus protocol and are sent back to the master when such an error occurs. The following exception codes are relevant and implemented for the sensor readout system:

- **0x01 - Illegal Function**: Indicates that the function code in the request is not supported by the slave.
- **0x02 - Illegal Data Address**: Indicates that the data address specified in the request is not valid for the slave. For example, when the master requests data from input registers beyond the specified 8 input registers.
- **0x03 - Illegal Data Value**: Indicates that the data value in the request is not valid for the context of the operation.

Chapter 3

Evaluation

3.1 Overview

To verify that the design meets the specified requirements, a comprehensive evaluation plan and setup have been developed. Each key requirement is divided into smaller, quantifiable sub-requirements with clear target values. This hierarchical approach ensures that each key requirement is thoroughly tested and validated. A test plan has been created for each key requirement, as illustrated in Appendix D.

To assess the accuracy of the read-out system in measuring the resistances of multiple graphene strips, two distinct experimental tests will be conducted: one to demonstrate the system's capability to measure the array in various configurations, and another to evaluate the precision of measuring individual resistors.

The requirement to generate a variable waveform to heat up the micro-hotplate will be tested through an experimental setup. Using an oscilloscope, the four different waveforms will be tested and shown. To verify the system's ability to output at least 30 mW, the technical specifications of the responsible amplifier will be discussed.

The functionality of the USB and Modbus protocol interfaces will be verified individually. The USB protocol will have a Python interface in which multiple parameters can be set, and a graph with the resistance values will be shown.

The Modbus protocol can be verified by using an external program named qModMaster, in which it is also possible to set multiple parameters. These settings will be verified using a multimeter and an oscilloscope to check the ADCs and DACs.

The microcontroller's machine learning capabilities, as specified in its data sheet, will be reviewed to confirm that it can integrate machine learning models.

The following sections will detail the experimental procedures. This evaluation will confirm whether the design meets the verification of the key requirements and provide insights into the strengths and weaknesses of the design. Finally, the areas for future improvements will be discussed.

Design Requirement	Label	Target Value	Test Method
Accurately measure the resistances of an array of 1 to 8 graphene strips	**R1.1**	Accuracy within 10 Ω	Calibration with known resistors
	R1.2	Generate a stable excitation voltage of 1V ± 0.1% for each strip	Verify the DAC's voltage output accuracy using a multimeter
	R1.3	Measure resistors in any configuration using different strips	Switch off different strips and measure remaining strips
Generate a configurable waveform for the micro-hotplate excitation	**R2**	Sine wave, Square wave, Sawtooth, Triangle wave	Measurement using a power meter and oscilloscope
Interfacing through USB	**R3**	Set system parameters and plot sensor data	Functional testing using laptop
Function autonomously using Modbus protocol	**R4**	Ability to set/read system parameters and read sensor data	Connect to laptop to simulate Modbus master Device using qModMaster
Design for ease of connection, implementation, and adaptation	**R5**	User-friendly for people with no prior experience	User testing with individuals unfamiliar with the design
Support the implementation of machine learning models	**R6**	Microcontroller with NPU	N.A.

Table 3.1: Summary of key design requirements for evaluation

3.2 Prototype

At the moment of writing this report, the PCB has not yet been assembled, making it difficult to test specific requirements. While components integrated on the development board like the DAC, ADC, and serial interfaces can already be physically tested, the verification of the hardware design is not yet feasible. However, virtual simulations of some of the hardware requirements have already been done using LTSpice.

The prototype available so far is the microcontroller development board. Using this piece of hardware, some of the core functionalities of the design have been verified. Further testing of the hardware will be done in the coming week after the handing in of this report, when the PCB has arrived and the prototype can be assembled. These findings will be presented at a later stage, together with the gathered improvements on the design.

3.3 Testing and Results

3.3.1 **(R1)** Accurately measure the resistances of an array of 1 to 8 graphene strips

This requirement will be evaluated in three parts. The first sub-requirement (**R1.1**) is to verify that our read-out system can measure the resistances of the graphene strip with an accuracy of 10 Ω or less. This will be assessed using the Python GUI to visualize the measurements and confirm the system's accuracy. The second sub-requirement (**R1.2**) involves measuring the generated excitation voltage of the DAC using a multimeter. The final sub-requirement (**R1.3**) will determine if the read-out system can accurately measure the resistances of an array of 1 to 8 graphene strips. This will be achieved by measuring the GPIO pins of the microcontroller, which are responsible for switching between channels via the multiplexer.

To evaluate requirement **R1.1** of achieving accuracy within 10 Ω, multiple tests were conducted by connecting the microcontroller's current and voltage ADCs to a known voltage source. While an ideal test would involve using controlled excitation voltage across plain resistors, initial tests focused on determining the intrinsic noise of the ADCs due to the unavailability of the PCB and to simplify the procedure. This controlled input test was chosen to accurately assess the noise characteristics inherent to the ADCs. The results revealed that the accuracy was significantly compromised, with the noise band measuring approximately 80 Ω. Consequently, this noise level indicates that the requirement for accuracy within 10 Ω was not satisfied. After improving the code by implementing multiple averaging loops, the achieved accuracy is now approximately 15 Ω (shown in Figure 2.8), which is a significant improvement over the previous 80 Ω but still above the 10 Ω requirement. Further discussion on potential methods to improve this accuracy and mitigate the noise issues can be found in Section 3.5.2.

The second sub-requirement (**R1.2**) was evaluated by measuring the output of the current generator DAC using a multimeter. This DAC is indirectly responsible for the excitation voltage over the graphene strips, as discussed in Chapter 2.6. Given that this DAC can output voltage with an accuracy of approximately 0.81 mV (as shown in equation 2.11), it was clear that the requirement of generating an excitation voltage of 1.0 V $\pm$ 0.1% would be met, even after considering the amplifier gain of 3.

To verify this, the DAC output voltage was set to correspond to 8 active strips, aiming for a desired 8.0 V after amplification. This translates to a desired output of approximately 2.66 V from the DAC before amplification. Using a multimeter, the actual DAC output voltage was measured to be approximately 2.65 V, indicating an accuracy of 0.01 V. After amplification by a factor of 3, this corresponds to an accuracy of 0.03 V, well within the required $\pm$ 0.1 V range. Thus, this test confirmed that the requirement **R1.2** was met.

The final requirement (**R1.3**) aimed to verify if the read-out system could accurately measure the resistances of an array of 1 to 8 graphene strips. A test was designed to evaluate the system's capability to sequentially cycle through various configurations of the 8 channels, corresponding to different setups in which the graphene strips can be connected. To assess this, the General-Purpose Input/Output (GPIO) pins of the microcontroller, which control the multiplexer to switch between channels, were monitored.

After measuring multiple switching sequences, it was observed that each channel was activated as expected without any irregularities. Although the actual multiplexer could not be tested due to its unavailability during the testing phase, the GPIO pins reliably switched through the predefined arrays. This indicates that there should be no problems when the multiplexer is connected to the microcontroller, thereby verifying the requirement.

The evaluation of requirement **R1** demonstrates that while certain aspects of the read-out system perform well, there are areas needing improvement to meet all sub-requirements effectively. For sub-requirement **R1.1**, the system's accuracy in measuring resistances was found to be insufficient due to significant noise, which was approximately 80 Ω. This noise level far exceeds the target value of 10 Ω, highlighting the need for further refinement of the ADC setup and noise reduction. However, the system successfully meets the sub-requirements for generating a stable excitation voltage (**R1.2**) and correctly switching between channels (**R1.3**). These aspects indicate that with further improvements in the accuracy of resistance measurements, the system can fully satisfy requirement **R1**.

3.3.2 **(R2)** Generate a Configurable Waveform to Drive the Micro-hotplate

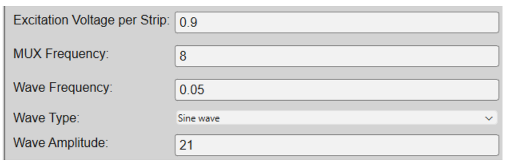
The requirement to generate a configurable waveform for the micro-hotplate excitation (**R2**) is crucial for testing the heating characteristics of the micro-hotplate under different electrical signals. Differential waveforms such as a sine wave, square wave, triangle wave, and sawtooth wave allow for different characteristics in behavior for the micro-hotplate. Together with a configurable wave frequency, wave amplitude, and duty cycle, a comprehensive analysis of the hotplate's behavior and efficiency can be made. These capabilities ensure that the hotplate can be effectively used in various experimental setups. The second requirement is to verify whether the generated waveform for the micro-hotplate can produce power up to 30 mW. Since no physical testing can be done, this is verified by LTspice simulations of the circuit. By confirming both requirements, we can validate the overall design requirement of generating configurable waveforms for the micro-hotplate (**R2**).

The evaluation of the first sub-requirement (**R2.1**) was conducted using an oscilloscope connected to the microcontroller's DAC, which generates the waveform characteristics required to control the micro-hotplate. The behavior can be evaluated by adjusting various parameters in the Python GUI, such as the waveform type, frequency, amplitude, and the duty cycle.

- Connect the microcontroller's DAC to an oscilloscope.
- Set the parameters in the user-interface to output each type of waveform sequentially.
- Record the waveform shapes and characteristics using the oscilloscope.

The results of generating different waveforms using the DAC are presented in Figures 3.1 and in Appendix A. Each waveform demonstrates the distinct pattern generated by the DAC, with the corresponding settings configured in the Python GUI.

The generated sine wave, shown in Figure 3.1, has a frequency set to 0.05 Hz. This translates to a period of 20 seconds. The oscilloscope graph accurately reflects this period, with each horizontal division representing 5 seconds. As illustrated, the sine wave completes one full cycle over 4 division blocks, confirming the correct configuration. The wave amplitude is set to 21 V in the settings. However, this value accounts for an amplifier gain of 7, resulting in an actual desired output amplitude of 3 V from the DAC. Given that each vertical division on the oscilloscope represents 1 V, it is evident that the generated sine wave has an amplitude of 3 V, also confirming the correct configuration of the DAC output.



Excitation Voltage per Strip:	0.9
MUX Frequency:	8
Wave Frequency:	0.05
Wave Type:	Sine wave
Wave Amplitude:	21

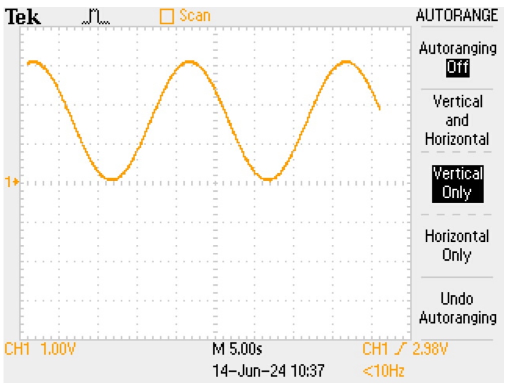


Figure 3.1: Sine Wave: Settings in the Python GUI and Generated Wave

To generate either a triangle wave or a sawtooth wave, the differences in the Python GUI settings are minimal. The setting that truly determines whether the output is a triangle wave or a sawtooth wave is the duty cycle. For a triangle wave, a duty cycle of 50 results in a peak that is at its highest point in the middle of the period and gradually returns to its lowest voltage point. This is illustrated in Figure A.1 in Appendix A.

In contrast, a sawtooth wave is characterized by a gradual increase in output voltage followed by a direct drop at the end of the period. This effect is achieved with a duty cycle of 100, as shown in Figure A.2 in Appendix A.

The triangle wave has an amplitude setting of 7, which results in an actual output of 1 V from the DAC, considering the amplifier gain. The oscilloscope division for the triangle wave is 500 mV, and the peak amplitude has a vertical length of 2 division blocks, confirming the expected output of 1 V (Figure A.1 in Appendix A).

The sawtooth wave has the same amplitude setting as the sine wave previously discussed, resulting in a desired output of 3 V. This is confirmed by the oscilloscope, where the peak amplitude reaches 3 division blocks and the vertical division per block is set to 1 V (Figure A.2 in Appendix A).

The final configurable waveform is the square wave. Figure A.3 in Appendix A illustrates that a square wave is produced by selecting the correct wave type in the Python GUI. The duty cycle is set to 75%, meaning the wave remains high for 75% of each period, which is clearly demonstrated in the graph. The frequency is configured to 0.1 Hz, resulting in a period of 10 seconds. With the oscilloscope's division blocks set to 5 seconds each, the wave spans 2 division blocks horizontally, confirming the correct period of the square wave. Additionally, the square wave's amplitude is set to 2 V, corresponding to a GUI setting of 14 (considering the amplifier gain of 7). This amplitude is correctly reflected in the graph.

For the second sub-requirement (**R2.2**), it can be shown that this easily meets the requirements. From the datasheet of the OPA990[16], the maximum delivered power is:

$$P = V \times I = 20 \times 0.08 = 1.6 \text{ Watt} \tag{3.1}$$

This value is larger than the minimal value of 30 mW. Unfortunately, at the moment of writing the thesis, the hardware test could not be done due to the fact that the PCB is not delivered yet. Therefore, the different waveforms are tested in LTSpice to check whether the expected waveforms can be generated using the current amplifier topology.

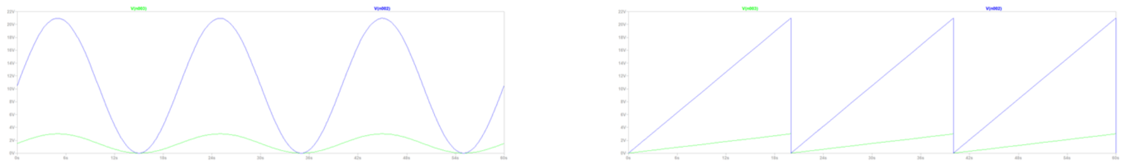


Figure 3.2: Simulation results of a Sine and a Sawtooth wave

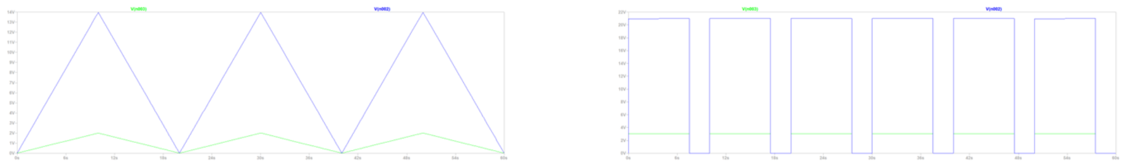


Figure 3.3: Simulation results of a Triangle and a Pulse wave

The graphs from Figures 3.2 and 3.3 show the simulated waves in LTspice. From the graphs, it can be observed that the amplifier is able to scale all the different incoming signals. One possible problem could be that clipping takes place; this is not the case in the simulation tests.

3.3.3 **(R3)** Interfacing through USB

The requirement to interface through USB (**R3**) is crucial for enabling seamless communication between the microcontroller and the Python GUI. This interface allows for parameter setting, data collection, and real-time visualization of results.

To evaluate the USB interface, a functional test was conducted using a laptop. The test was designed to verify that the parameters could be set via the Python GUI and correctly implemented in the microcontroller. Additionally, the transmission of the resistance data was assessed for responsiveness, stability, and proper display in the Python GUI. This USB interface has already been demonstrated and validated during the evaluation of the micro-hotplate controller (shown in Figure 3.1), confirming its reliability and effectiveness in setting parameters, visualizing measured resistance data (Figure 2.8), and storing data in a CSV file without delays or crashes. These results confirm that the USB interface is reliable and efficient, enabling users to interact with the system and obtain accurate, real-time measurements effortlessly.

3.3.4 **(R4)** Function Autonomously using Modbus Protocol

Fulfilling requirement **R4** ensures that the sensor readout system is able to function reliably within a network. A Modbus master device must be able to read and set system parameters, and the readout system must be able to send its data to the master when it requests the data. As specified in Section 2.9.5, the function codes that must be supported for these functionalities are: Read Holding Registers (**0x03**), Read Input Registers (**0x04**), Write Single Register (**0x06**), and Write Multiple Registers (**0x10**).

To evaluate whether the Modbus protocol functions as required, a Modbus master device must be connected to the PCB. This can be in the form of a laptop with the Modbus master simulation program, qModMaster, installed. The communication settings have to be set in the same way in the program and the firmware of the microcontroller. Each function code then has to be tested to evaluate the functionalities. The reading of available system parameters can be tested via function code 0x03. The setting of available system parameters can be tested via function code 0x06 for a single parameter and function code 0x10 for multiple parameters. The reading of the resistances of the strips can be tested via function code 0x04. Error detection can be tested by trying to read/write an incorrect range of registers, using another slave address than the one defined, or writing an incompatible data type to the holding registers (system parameters). The same method as for **R1** can then be used to evaluate the success of the setting of the system parameters.

3.3.5 **(R5)** Design for Ease of Connection, Implementation, and Adaptation

As of 14-06-2024, the PCB has not yet arrived, so the ease of connection, implementation, and adaptation cannot be tested. By design, the interfacing via USB and Modbus should be relatively easy to perform if the required software is installed. The sensor array should also be quite easy to install and remove.

3.3.6 **(R6)** Support the Implementation of Machine Learning Models

This requirement is fulfilled through the selection of the microcontroller. The MCX N94 microcontroller contains an NPU, which allows for the implementation of on-chip machine learning models [**MCXN94˙reference˙manual**].

3.4 Assessment

Design Requirement	**Label**	**Target Value**	**Assessment**
Accurately measure the resistances of an array of 1 to 8 graphene strips	**R1.1**	Accuracy within 10 Ω	Not Met
	R1.2	Generate a stable excitation voltage of 1V ± 0.1%	Met
	R1.3	Measure resistors in any configuration using different strips	Partially Met
Generate a configurable waveform for the micro-hotplate excitation	**R2**	Sine wave, Square wave, Sawtooth, Triangle wave	Met
Interfacing through USB	**R3**	Set system parameters and plot sensor data	Met
Function autonomously using Modbus protocol	**R4**	Ability to set/read system parameters and read sensor data	Met
Design for ease of connection, implementation, and adaptation	**R5**	User-friendly for people with no prior experience	Not Tested
Support the implementation of machine learning models	**R6**	Microcontroller with NPU	Met

Table 3.2: Summary of key design requirements and their assessment

As can be seen in the table above, most of the requirements that can be tested at this stage (14-06-2024: PCB has not arrived yet) have been met. The requirement that has not been met is the accuracy of the readout system, requirement **R1.1**. As accuracy is an important requirement, improvements must be made to improve the accuracy to within 10 Ω. The steps that can be taken towards meeting this requirement are explained in the next section.

3.5 Next Steps

3.5.1 Hardware Improvements

Firstly, the hardware design should be assembled and tested. Test plans have been elaborated in Appendix D. Testing is instrumental in verifying the functionality of the hardware. While some individual components of the hardware design have already been tested using the development board and simulations, verification of the full system is necessary.

Secondly, after these tests have been performed, further iteration of the design is likely beneficial in improving accuracy of measurements and ease of use. Possible improvements include using differential amplifiers, increasing CMRR in voltage measurement, and optimizing PCB layout. Finally, in a further iteration of the hardware design, all of the components, including the microcontroller and associated hardware, can be integrated on a single PCB.

3.5.2 Software Improvements

As illustrated in 2.8, the accuracy of the resistance measurements did not meet the required standards. To address this, several enhancements can be implemented to significantly improve measurement accuracy.

Most notably, a moving average filter can be applied. A moving average filter works by averaging a number of consecutive data points, which helps to smooth out random fluctuations in the data. By reducing the impact of these fluctuations, the overall accuracy of the measurements can be improved.

Furthermore, while the hardware implementation of the temperature and humidity sensor is completed, the corresponding software integration has not yet been finalized. This integration is useful to enable future calibration of the measurements when the machine learning model has been implemented.

Bibliography

[1] A. Dey. “Semiconductor metal oxide gas sensors: A review”. In: *Materials Science & Engineering B* 229 (Mar. 2018), pp. 206–217. DOI: 10.1016/j.mseb.2017.12.036. URL: https://doi-org.tudelft.idm.oclc.org/10.1016/j.mseb.2017.12.036.

[2] Novoselov Hill Vijayaragahvan. *Graphene sensors*. Dec. 2011. URL: https://ieeexplore.ieee.org/abstract/document/6016205?casa_token=kMhZt0YBYBYAAAAA:nHxWpCNVQlytXj7yzUj1Zaz4nLpTHkOeYSc3VxYsQUwB4CYpnO0Drb8esTV6WRNm2TELkrGLXA.

[3] Texas Instruments. *INA149 Precision Difference Amplifier*. 2023. URL: https://www.ti.com/lit/ds/symlink/ina149.pdf (visited on 05/21/2024).

[4] S. Vollebregt L.N. Sacco. “Graphene gas sensors monolithically integrated on microhotplates by using a transfer-free approach”. In: *Graphene Flagship* (). URL: https://www5.shocklogic.com/Client_Data/KONGRESS/al/GW2023/upload/F1-5961941-GW_2023_Abstract_SV.pdf.

[5] Xiao Liu et al. “A Survey on Gas Sensing Technology”. In: *Sensors* 12.7 (July 2012), pp. 9635–9665. DOI: 10.3390/s120709635. URL: https://doi.org/10.3390/s120709635.

[6] *MCX Nx4x Reference Manual*. Version 4th. NXP Semiconductors. Jan. 2024.

[7] Inc. Modbus Organization. *MODBUS APPLICATION PROTOCOL SPECIFICATION V1.1b*. 2006. URL: https://modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf.

[8] Leandro Nicolas Sacco, Hanxing Meng, and Sten Vollebregt. *Humidity sensor based on multilayer graphene (MLG) integrated onto a micro-hotplate (MHP)*. Oct. 2022. URL: https://ieeexplore.ieee.org/document/9967039.

[9] F. Schedin et al. “Detection of individual gas molecules adsorbed on graphene”. In: *Nature materials* 6.9 (July 2007), pp. 652–655. DOI: 10.1038/nmat1967. URL: https://doi.org/10.1038/nmat1967.

[10] Steve Semancik et al. “Microhotplate platforms for chemical sensor research”. In: *Sensors and actuators. B, Chemical* 77.1-2 (June 2001), pp. 579–591. DOI: 10.1016/s0925-4005(01)00695-5. URL: https://doi.org/10.1016/s0925-4005(01)00695-5.

[11] Texas Instruments. *HDC3022 High Accuracy, Ultra-Low Power Digital Humidity Sensor with Temperature Output*. Texas Instruments datasheet. 2022. URL: https://www.ti.com/lit/ds/symlink/hdc3022.pdf?ts=1718397933969&ref_url=https%253A%252F%252Fwww.ti.com%252Fsitesearch%252Fen-us%252Fdocs%252Funiversalsearch.tsp%253FlangPref%253Den-US (visited on 05/21/2024).

[12] Texas Instruments. *INA215 Zero-Drift, Bidirectional, Low- or High-Side Measurement, Current-Shunt Monitor*. Texas Instruments datasheet. 2023. URL: https://www.ti.com/lit/ds/symlink/ina215.pdf?ts=1718397870339&ref_url=https%253A%252F%252Fwww.ti.com%252Fsitesearch%252Fen-us%252Fdocs%252Funiversalsearch.tsp%253FlangPref%253Den-US (visited on 05/21/2024).

[13] Texas Instruments. *LM27313 High Efficiency 600-mA Boost Converter*. Texas Instruments datasheet. 2015. URL: https://www.ti.com/lit/ds/symlink/lm27313.pdf?ts=1718398052439&ref_url=https%253A%252F%252Fwww.ti.com%252Fsitesearch%252Fen-us%252Fdocs%252Funiversalsearch.tsp%253FlangPref%253Den-US (visited on 05/21/2024).

[14] Texas Instruments. *MUX36S08 Low-Resistance, 36-V, 8:1, 1-Channel Multiplexer with Enable*. Texas Instruments datasheet. 2019. URL: https://www.ti.com/lit/ds/symlink/mux36s08.pdf?ts=1718397810712&ref_url=https%253A%252F%252Fwww.ti.com%252Fsitesearch%252Fen-us%252Fdocs%252Funiversalsearch.tsp%253FlangPref%253Den-US (visited on 05/21/2024).

[15] Texas Instruments. *OPA188 Zero-Drift, Low-Noise, Rail-to-Rail Output Operational Amplifier*. Texas Instruments datasheet. 2020. URL: https://www.ti.com/lit/ds/symlink/opa188.pdf?ts=1718314092793 (visited on 05/21/2024).

[16] Texas Instruments. *OPA990 Precision, 40-MHz, Low-Noise, Low-Power Operational Amplifier*. Texas Instruments datasheet. 2021. URL: https://www.ti.com/lit/ds/symlink/opa990.pdf?ts=1718397841096&ref_url=https%253A%252F%252Fwww.ti.com%252Fsitesearch%252Fen-us%252Fdocs%252Funiversalsearch.tsp%253FlangPref%253Den-US (visited on 05/21/2024).

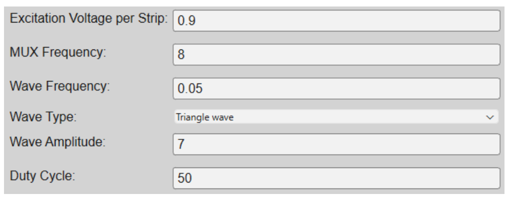
[17] Texas Instruments. *TPS2121 2.7-V to 22-V, 4.5-A, 44-m■ Power Multiplexer*. Texas Instruments datasheet. 2020. URL: https://www.ti.com/lit/ds/symlink/tps2121.pdf?ts=1718345089403.

[18] Texas Instruments. *TPSM82903 3-A, 3.7-V to 20-V Input, Synchronous Step-Down Power Module*. Texas Instruments datasheet. 2022. URL: https://www.ti.com/lit/ds/symlink/tpsm82903.pdf?ts=1718398028868&ref_url=https%253A%252F%252Fwww.ti.com%252Fsitesearch%252Fen-us%252Fdocs%252Funiversalsearch.tsp%253FlangPref%253Den-US.

[19] Z Yunusa et al. “Gas Sensors: A Review”. In: *Sensors & Transducers* 168.4 (Apr. 2014), pp. 61–75. DOI: 10.3390/s120709635. URL: https://doi.org/10.3390/s120709635.

Appendix A

Additional Figures



Excitation Voltage per Strip:	0.9
MUX Frequency:	8
Wave Frequency:	0.05
Wave Type:	Triangle wave
Wave Amplitude:	7
Duty Cycle:	50

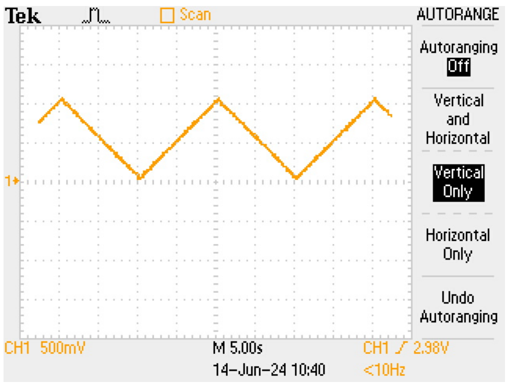
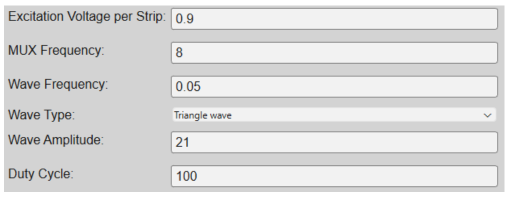


Figure A.1: Triangle Wave: Settings in the Python GUI and Generated Wave



Excitation Voltage per Strip:	0.9
MUX Frequency:	8
Wave Frequency:	0.05
Wave Type:	Triangle wave
Wave Amplitude:	21
Duty Cycle:	100

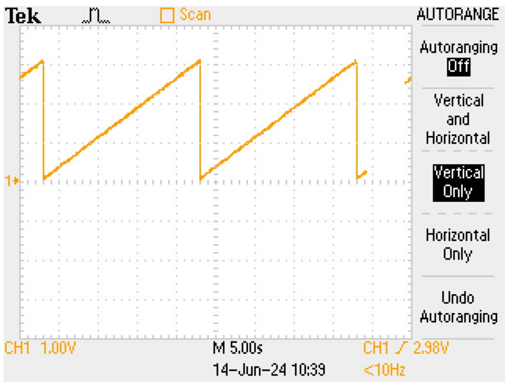
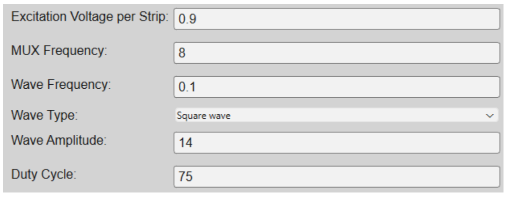


Figure A.2: Sawtooth Wave: Settings in the Python GUI and Generated Wave



Excitation Voltage per Strip:	0.9
MUX Frequency:	8
Wave Frequency:	0.1
Wave Type:	Square wave
Wave Amplitude:	14
Duty Cycle:	75

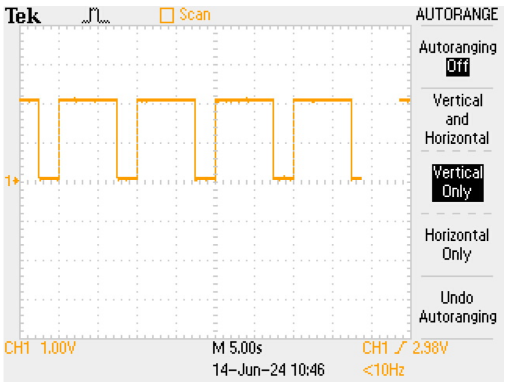


Figure A.3: Square Wave: Settings in the Python GUI and Generated Wave

Appendix B

Github Repository and Additonal Software Resources

B.1 GitHub Repository

The source code and additional resources for this project can be found on GitHub:
https://github.com/DirkJanKr/BEP.git

B.2 Additional Software Resources

The following NXP API manuals were used in this project:

- MCUXpresso SDK API Reference Manual 1
- MCUXpresso SDK API Reference Manual 2

Appendix C

PCB Design Overview

This appendix details the schematics and PCB layout of our design. The pinout of the sensor-array can be seen in figure C.1.

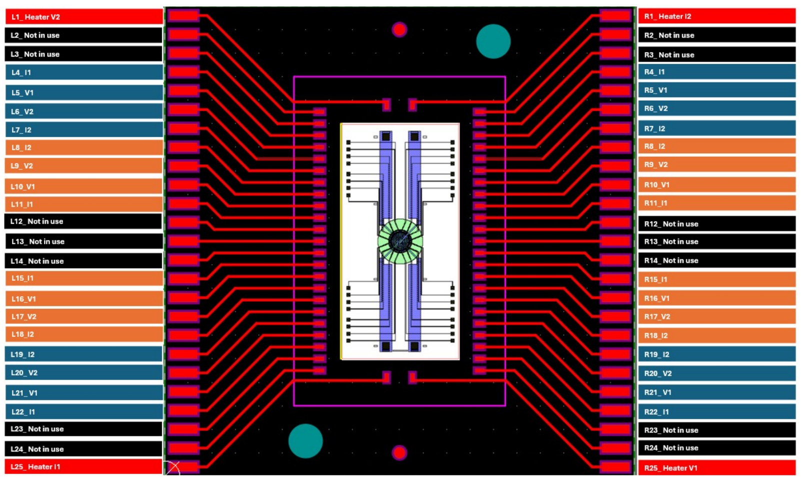
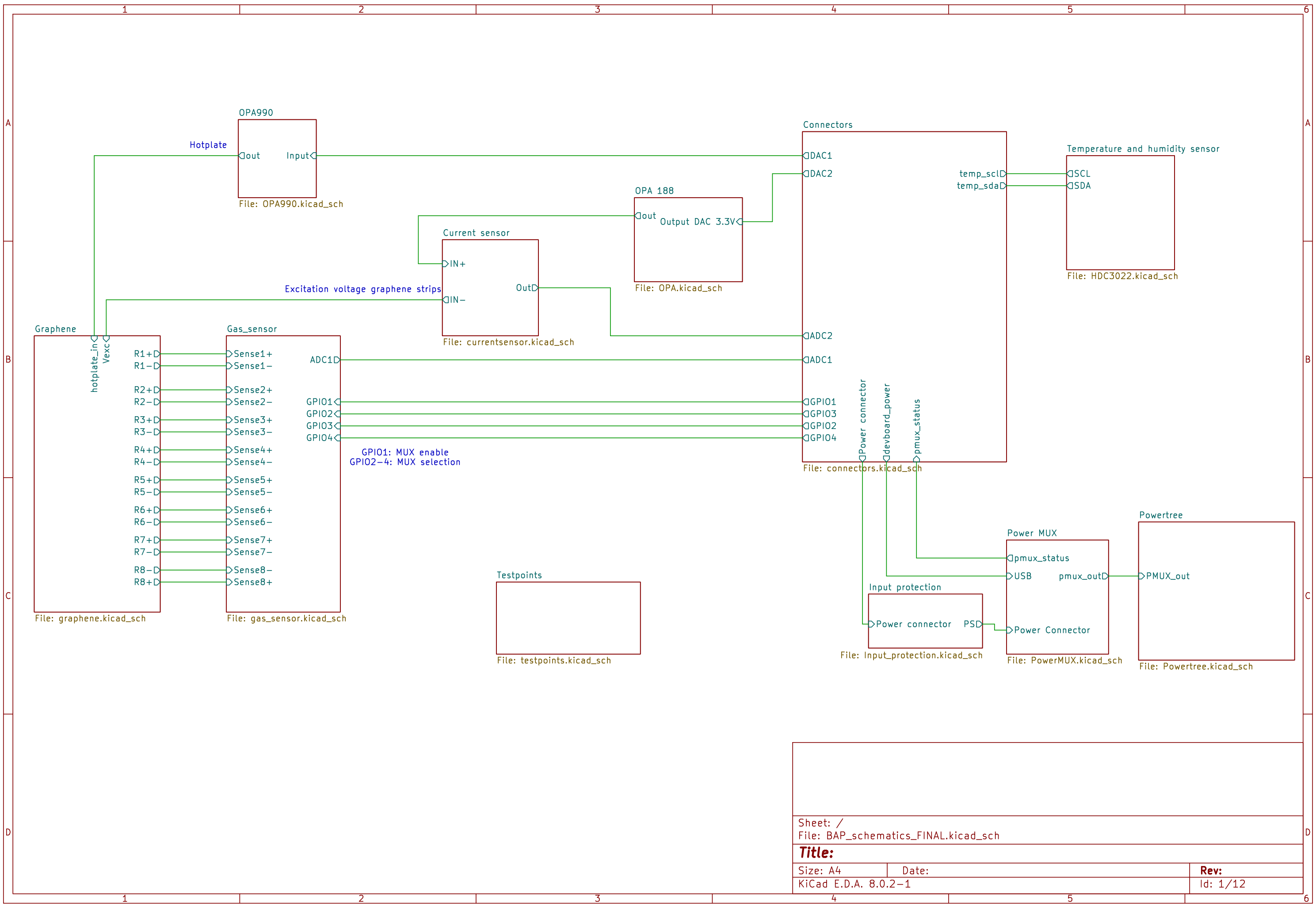
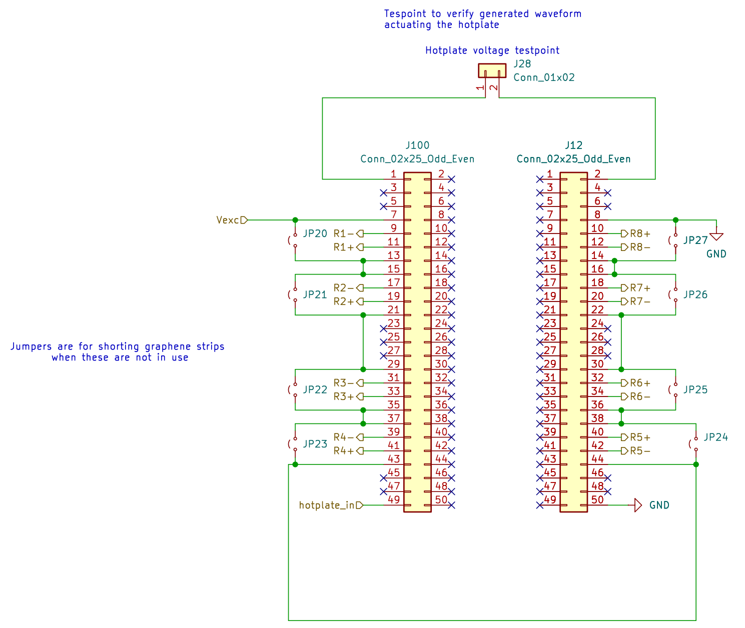


Figure C.1: Sensor Array Pinout

C.1 Schematics



OPA990
Hotplate
out
Input
File: OPA990.kicad_sch
Connectors
DAC1
DAC2
temp_scl
temp_sda
Temperature and humidity sensor
SCL
SDA
File: HDC3022.kicad_sch
OPA 188
out
Output DAC 3.3V
File: OPA.kicad_sch
Current sensor
IN+
IN−
Out
Excitation voltage graphene strips
File: currentsensor.kicad_sch
ADC2
ADC1
Graphene
hotplate_in
Vexc
R1+
R1−
R2+
R2−
R3+
R3−
R4+
R4−
R5+
R5−
R6+
R6−
R7+
R7−
R8−
R8+
File: graphene.kicad_sch
Gas_sensor
Sense1+
Sense1−
Sense2+
Sense2−
Sense3+
Sense3−
Sense4+
Sense4−
Sense5+
Sense5−
Sense6+
Sense6−
Sense7+
Sense7−
Sense8−
Sense8+
ADC1
GPIO1
GPIO2
GPIO3
GPIO4
File: gas_sensor.kicad_sch
GPIO1: MUX enable
GPIO2−4: MUX selection
GPIO1
GPIO3
GPIO2
GPIO4
Power connector
devboard_power
pmux_status
File: connectors.kicad_sch
Testpoints
File: testpoints.kicad_sch
Input protection
Power connector
PS
File: Input_protection.kicad_sch
Power MUX
pmux_status
USB
pmux_out
Power Connector
File: PowerMUX.kicad_sch
Powertree
PMUX_out
File: Powertree.kicad_sch
Sheet: /
File: BAP_schematics_FINAL.kicad_sch
Title:
Size: A4
Date:
Rev:
KiCad E.D.A. 8.0.2−1
Id: 1/12

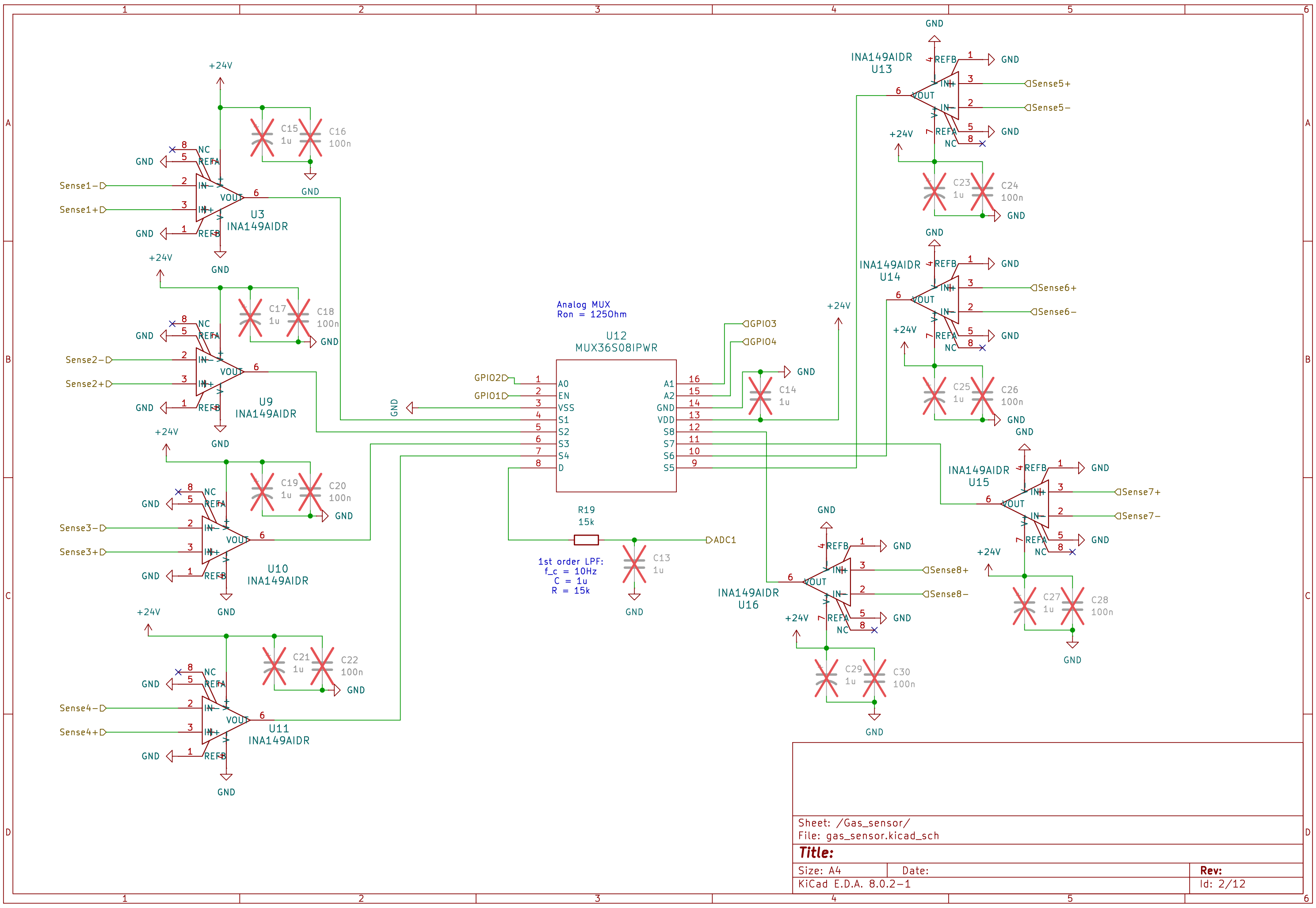


Sheet: /Graphene/
File: graphene.kicad_sch

Title:

Size: A4 Date: **Rev:**

KiCad E.D.A. 8.0.2−1 Id: 1/12

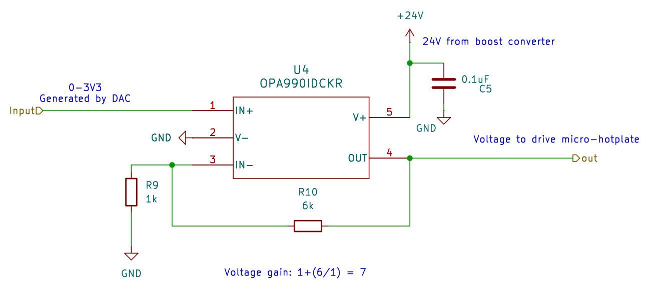


Sheet: /Gas_sensor/
File: gas_sensor.kicad_sch

Title:

Size: A4 Date: **Rev:**

KiCad E.D.A. 8.0.2-1 Id: 2/12

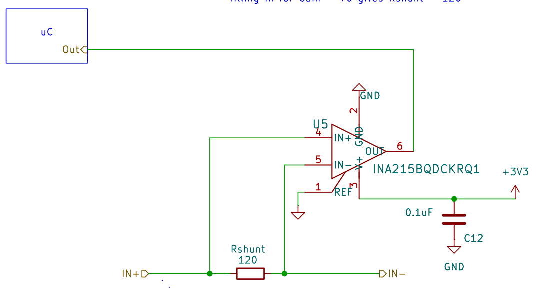


Sheet: /OPA990/
File: OPA990.kicad_sch

Title:

Size: A4	Date:	**Rev:**
KiCad E.D.A. 8.0.2-1		Id: 3/12

Rshunt = (Vout−Vref)/(Gain*Iload) −−−−− Vref = 0 & Iload = 1ma
Rshunt = 3000/ Gain
filling in for Gain = 70 gives Rshunt = 120

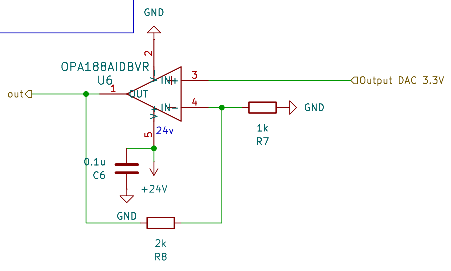


Sheet: /Current sensor/
File: currentsensor.kicad_sch

Title:

Size: A4	Date:	**Rev:**
KiCad E.D.A. 8.0.2-1		Id: 4/12

For Rin a value of 1k Ohm was recommended. So to obtain a gain of 3, where the gain is defined
1+Rf\Rin, Rin Rf should be 2k Ohm

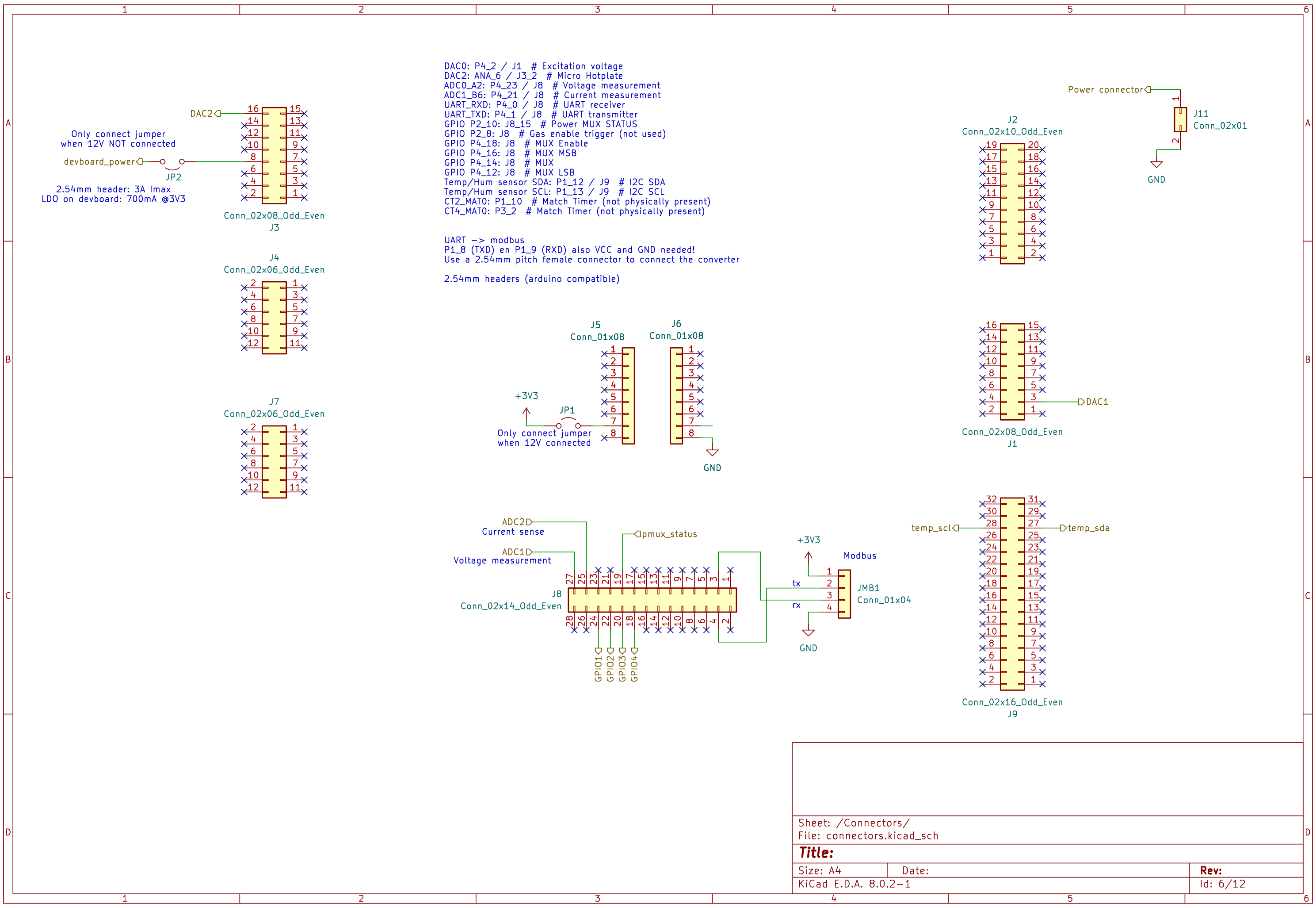


Sheet: /OPA 188/
File: OPA.kicad_sch

Title:

Size: A4 | Date: | **Rev:**

KiCad E.D.A. 8.0.2-1 | Id: 5/12



```
DAC0: P4_2 / J1  # Excitation voltage
DAC2: ANA_6 / J3_2  # Micro Hotplate
ADC0_A2: P4_23 / J8  # Voltage measurement
ADC1_B6: P4_21 / J8  # Current measurement
UART_RXD: P4_0 / J8  # UART receiver
UART_TXD: P4_1 / J8  # UART transmitter
GPIO P2_10: J8_15  # Power MUX STATUS
GPIO P2_8: J8  # Gas enable trigger (not used)
GPIO P4_18: J8  # MUX Enable
GPIO P4_16: J8  # MUX MSB
GPIO P4_14: J8  # MUX
GPIO P4_12: J8  # MUX LSB
Temp/Hum sensor SDA: P1_12 / J9  # I2C SDA
Temp/Hum sensor SCL: P1_13 / J9  # I2C SCL
CT2_MAT0: P1_10  # Match Timer (not physically present)
CT4_MAT0: P3_2  # Match Timer (not physically present)


UART -> modbus
P1_8 (TXD) en P1_9 (RXD) also VCC and GND needed!
Use a 2.54mm pitch female connector to connect the converter

2.54mm headers (arduino compatible)
```

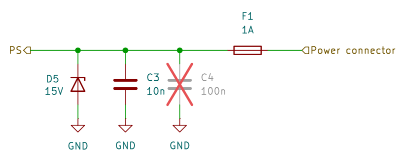
Sheet: /Connectors/
File: connectors.kicad_sch

Title:

Size: A4	Date:	**Rev:**
KiCad E.D.A. 8.0.2-1		Id: 6/12

TVS diode for overvoltage
Capacitors for voltage transients
Fuse for overcurrent

Powermux has input and reverse polarity protections as well

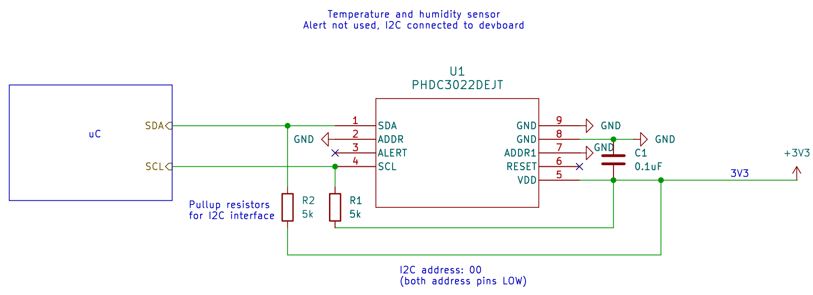


Sheet: /Input protection/
File: Input_protection.kicad_sch

Title:

Size: A4 | Date: | **Rev:**

KiCad E.D.A. 8.0.2-1 | Id: 7/12



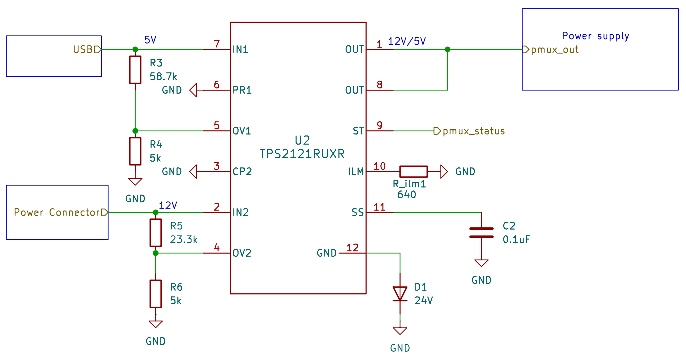
Sheet: /Temperature and humidity sensor/
File: HDC3022.kicad_sch

Title:

Size: A4	Date:	Rev:
KiCad E.D.A. 8.0.2-1		Id: 8/12

Resistive dividers used to set input voltage thresholds
Diode for reverse polarity protection
R_ilm to set output current limit

R3 = (R4*Vin)/OV1 - R4 = (5k*13.5)/1.06-5k= 58.7k
R5 = (R4*Vin)/OV2 - R4 = (5k*6)/1.06 - 5k = 23.3k
R_ilm = (65.2/Ilm)^(1/0,861)= 640 Ohm

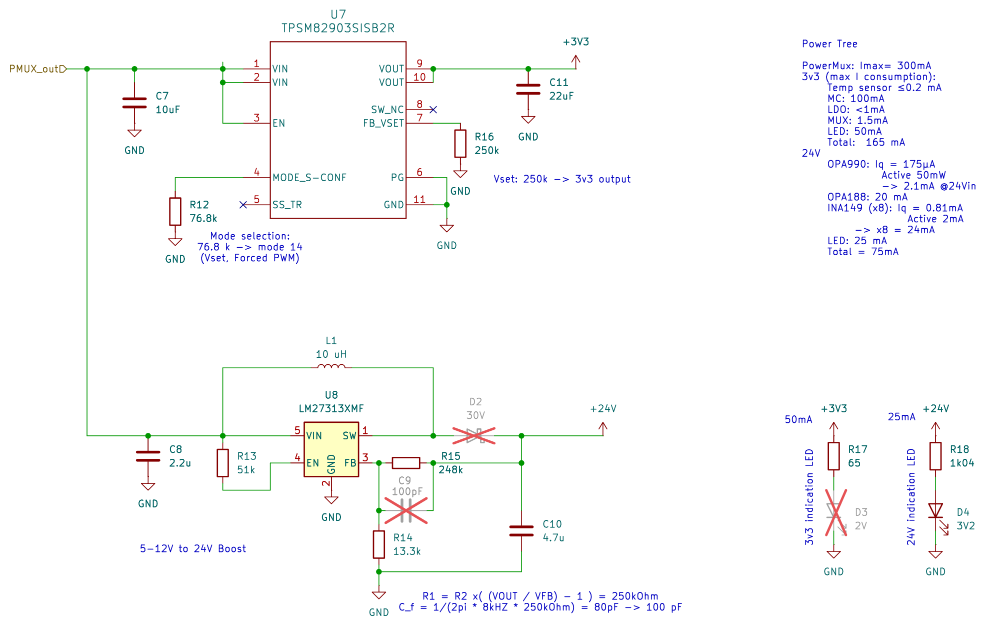


Sheet: /Power MUX/
File: PowerMUX.kicad_sch

Title:

Size: A4 | Date: | **Rev:**

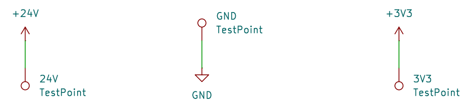
KiCad E.D.A. 8.0.2-1 | Id: 9/12



Sheet: /Powertree /
File: Powertree.kicad_sch

Title: Powertree V1

Size: A4	Date:	**Rev:**
KiCad E.D.A. 8.0.2-1		Id: 10/12

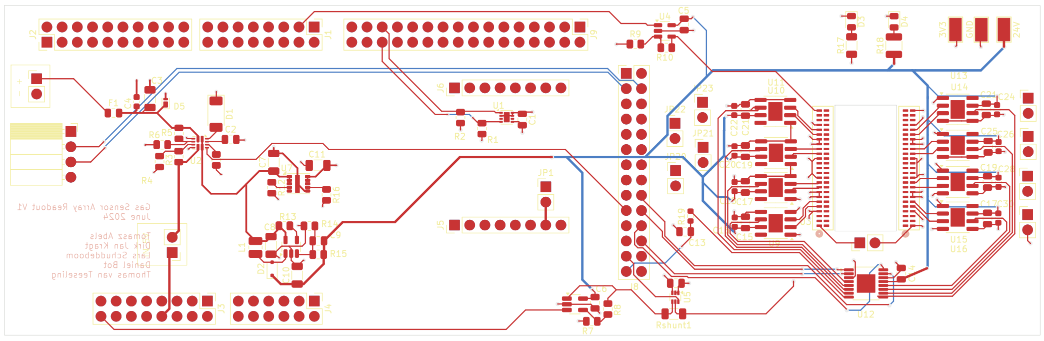


Sheet: /Testpoints/
File: testpoints.kicad_sch

Title:

Size: A4	Date:	**Rev:**
KiCad E.D.A. 8.0.2-1		Id: 12/12

C.2 PCB Layout


Gas Sensor Array Readout V1
June 2024
Tomas Abels
Dirk Jan Kraft
Lars Schuddeboom
Daniel Bot
Thomas van Teesseling
Rshunt1
U12

Appendix D

Test Plan

At the time this thesis is handed in, the prototype is yet to be assembled. To guide future verification of the design, this document will describe the necessary steps and expected results needed.

D.1 1. Accurate Measurements

Objective: Verify the accuracy of the resistance measurements of the graphene strips.
Steps:

1. Calibrate the measurement system using precision resistors (e.g., 10 kΩ).
2. Measure the resistance of known resistors and compare the measured values with the actual values.
3. Connect the resistors to the connectors used for the sensor-array. Verify the proper terminals using the schematics. Verify each pair of the terminals used to measure the voltage over the graphene strips, both combinations and individual terminals. Take care to set the excitation voltage to the appropriate values (1V for each connected resistor) and to measure this voltage using a precision voltage probe.
4. For each voltage measurement that is done, also verify the measured current and compare it to the real value. The real value can be calculated from the excitation voltage and measured resistance between the excitation voltage input and ground.

Expected Results:

- The measured resistance should be within 0.1% accuracy of the actual resistor values.
- Consistent and accurate measurements for each graphene strip in the array.
- Stable excitation voltage with minimal fluctuation.

D.2 2. Generate Variable Waveform for Hotplate

Objective: Verify the generation of variable waveforms for the micro-hotplate.
Steps:

1. Connect the microcontroller's DAC to an oscilloscope.

2. Set different waveform parameters (sine, square, triangle) using the GUI.
3. Measure and record the waveforms generated.
4. Verify the amplitude and frequency settings.

Expected Results:

- Clear and distinct waveform shapes for sine, square, and triangle waves.
- Accurate amplitude and frequency as set in the GUI.
- Ability to generate waveforms up to 24V.

D.3 3. Connecting the Sensor Array

Objective: Ensure the sensor array can be easily connected to the PCB.
Steps:

1. Attempt to connect the sensor array to the PCB without using documentation.
2. Check for secure mechanical and electrical connections.

Expected Results:

- Sensor array can be connected easily and securely.
- Proper mechanical and electrical connections are established without errors.

D.4 4. Interfacing with a Desktop

Objective: Verify the prototype's ability to interface with a desktop for lab measurements.
Steps:

1. Connect the prototype to a desktop using a USB-C cable.
2. Power the prototype via USB (5V input).
3. Use the GUI to set system parameters and read sensor data.

Expected Results:

- Successful communication between the prototype and the desktop.
- GUI can set parameters and display sensor data accurately.

D.5 5. Function as an Autonomous Sensor

Objective: Verify the prototype's functionality as an autonomous sensor.
Steps:

1. Connect the prototype to a 12V power source.
2. Set up communication via Modbus protocol using an external program (e.g., qModMaster).

3. Test the ability to read and write system parameters.

Expected Results:

- Prototype operates correctly on a 12V power input.
- Successful Modbus communication for reading and writing parameters.