Technische Universität Braunschweig

Institut für Mikrotechnik

Master thesis

Development of a universal test environment for flexible and stretchable sensor systems

from cand. mach. Yang Wang

Registernumber 4994598

Technische Universität Braunschweig

Faculty of Engineering

Prepared on 10.11.2021

Institute of Microtechnology

Examinant: Prof. Dr. Andreas Dietzel

Supervisor: Dr.-Ing. Eugen Koch

Submitted on 02. May 2022

Declaration

I , Yang Wang, hereby confirm that I have written the present thesis independently and without illicit assistance from third parties and using solely the mentioned sources.

Braunschweig, 02.05.2022

Yang Wang

Abstract

In this work, a test environment is built based on existing stretching and bending test setups for the study of the deformation-resistance characteristics in stretching and bending of printed sensor elements. Retaining the original mechanical structure and electrical motion system, the measurement system is redesigned, and a sensor readout is implemented with the help of the Cypress PSoC platform, which enables a dynamic configuration of the measurement circuit. The evaluation result of the test environment shows that the new measurement system has high static accuracy and dynamic readout speed, providing reliable test result for the characterization of the printed sensor elements.

Keywords: printed flexible and stretchable sensors, stretching setup, bending setup, Cypress PSoC

Assignment of tasks

Table of contents

List of illustrations

List of tables

List of code blocks

List of abbreviations

IMT	Institute of Microtechnology, TU Braunschweig
PSoC	Programmable System-on-Chip
PC	Personal Computer
MCU	Microcontroller Unit
PWM	Pulse Width Modulation (PWM)
UART	Universal Asynchronous Receiver/Transmitter
I2C	Inter-Integrated Circuit
SPI	Serial Peripheral Interface Bus
CPU	Central Processing Unit
SRAM	Static Random Access Memory
EEPROM	Electrically Erasable Programmable read only memory
GPIO	General Purpose Input/Output
USB	Universal Serial Bus
PCB	Printed Circuit Board
SMD	Surface-Mounted Device
ADC	Analog to Digital Converters
3D	3 Dimensions
AMux	Analog Multiplexer
SPS	Samples per Second
GUI	Graphical User Interface
SPS	Samples Per Second
LSB	Least Significant Bit
FSR	Full Scale Range

1 Introduction

1.1 Background

The Institute for Microtechnology (IMT) at Technical University of Braunschweig has developed a flexible sensor that can convert the bending or stretching deformation of the surface into voltage signals. The sensor consists of multiple units in Wheatstone bridge configuration. Each Wheatstone bridge consists of four strain gauges (see Figure 1.1). When the flexible sensor is bent or stretched, the strain gauges are deformed, resulting in a change of the output voltages of the Wheatstone bridges. This kind of sensor can be applied for neonatal respiratory monitoring. With the help of geometric or artificial intelligence algorithms, the surface shape of the flexible sensor can be reconstructed digitally, which can be applied on objects to detect their shape [1].

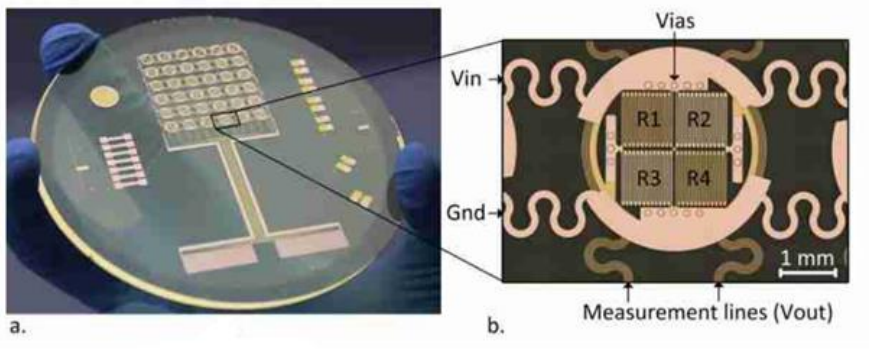


Figure 1.1: (a) The flexible sensor on a glass wafer (b) The single sensor unit of the flexible sensor consisting of four strain gauges in Wheatstone bridge configuration [1]

In the latest research, a new generation of flexible sensors is being developed to replace the older flexible sensor designs mentioned above. New structures, new materials and new manufacturing processes are applied into the development of the new sensor. The new generation is expected to have better mechanical stretchability, lower cost, lower manufacturing complexity and higher manufacturing scalability.

Figure 1.2 shows the structure of the new sensor patch with 4x4 sensing units. The sensor patch consists of resistors, conduction tracks and insulating layers. Unlike the earlier sensor units consisting of a Wheatstone bridge with four strain gauges, resistors made of piezoelectric material are used, making the sensor structure simpler. The contact pad at the bottom of the sensor makes it possible to select one sensing unit from the array and connect it to an external

measurement circuit consisting of a reference resistor and a DC power supply. The sensing unit is connected in series with the reference resistor. When the sensor patch is deformed, the resistance of the sensing unit changes due to the piezoresistive effect, resulting in the change of output voltage of the sensing unit, thus realizing the conversion of deformation of the sensor surface into a voltage signal.

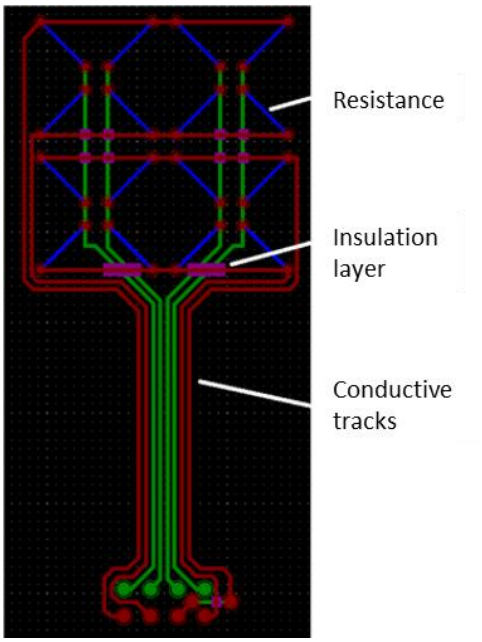


Figure 1.2: The structure of the new flexible sensor with 4x4 sensing units [2]

In terms of materials, the new sensor uses Polyurethan (DuPont Intexar TE-11C) as the substrate material, Silber paste (DuPont Intexar PE874) as the track material, carbon paste (DuPont Intexar PE671) as the resistance material, and Isolations-Paste (DuPont Intexar PE773) as the isolator material.

Regarding the manufacturing process, screen printing is used to make sensors. Figure 1.3 shows the principle of screen printing. Screen printing consists of five elements: blade, paste, screen, template, substrate and base. A Screen is net-like fabric, and a template is a layer structure fabricated on the screen using photolithography with openings for the pattern to be printed. In the manufacturing, the screen, template, substrate, and base are tightly overlapped and fixed together. Afterwards, liquid paste is applied to the screen. The screen is pressed with the blade. Then, the paste penetrates down through the porous screen under pressure and continues through the vacant areas of the template to the object surface. As a result, pattern on the template is then transferred onto the substrate [3].

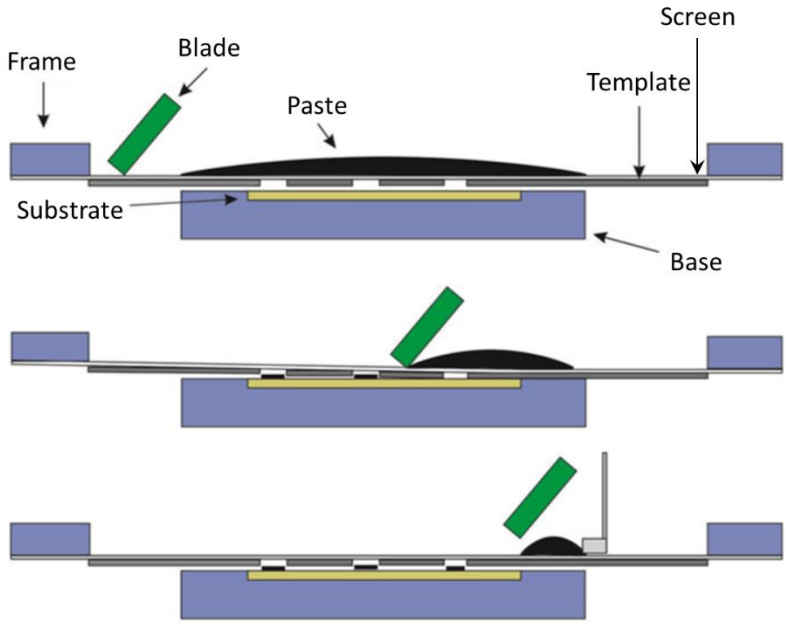


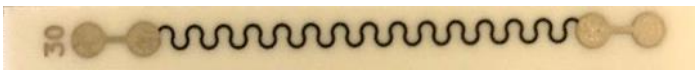
Figure 1.3: Schematic diagram of the components for screen printing [4]

In the latest study, a sensor mask is manufactured as the template. Sensor structures made of different materials is sequentially brushed onto the substrate material with the help of the template, eventually forming the complete sensor.

1.2 Motivation

To study the electrical characteristics of the sensing resistor and the conductor when subjected to deformation under different manufacturing parameters and material parameters, two types of samples have been designed as shown in the Figure 1.4. The sample in Figure 1.4 (a) is used to test the electrical characteristics in the case of being stretching. The sample in Figure 1.4 (b) is used to test the electrical characteristics in the case of being bending.

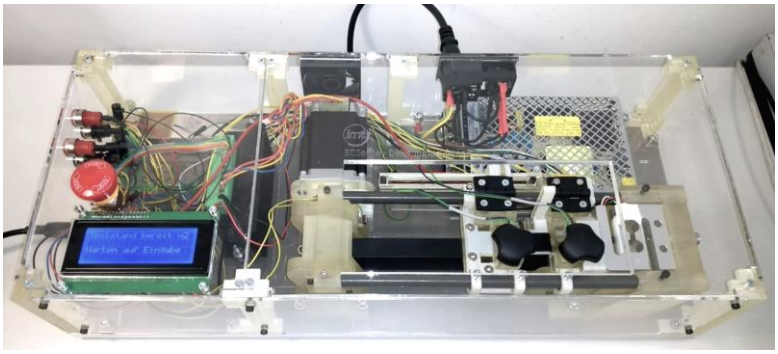
(a)



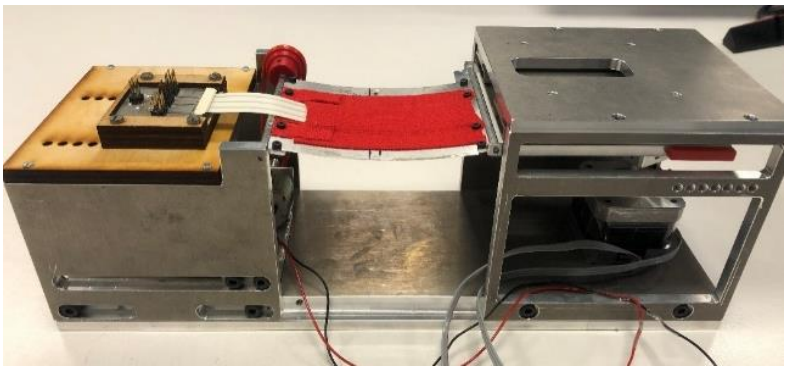
(b)

Figure 1.4: Photograph of the sample for (a) stretching test and (b) bending test

The stretching and bending test are planned to be performed on dedicated test setups. We find two existing test setups which were developed in other projects. Figure 1.5 (a) shows the stretching test setup, which is designed and built by Vincent Gottwald in his bachelor work "Aufbau eines Dehnungsprüfstands für Leiterbahnelemente mit integrierter Widerstandsmessung" [5]. The setup can stretch the sample and measure its resistance. Figure 1.5 (b) shows the bending test setup, which is designed and built by Philip Hoop and Jimmy Retzlaff in their project work "Entwicklung eines automatisierten Biegemessstandes für ein flexibles Sensorpatch" [6]. The setup can bend the sample and measures its resistance.



(a)



(b)

Figure 1.5: Photograph of (a) original stretching test setup [5] and (b) bending test setup [6]

However, these two test setups cannot be put into use directly because their functionality have the following limitations so that they cannot completely meet the requirement of the test.

- In terms of the stretching test setup, the mechanical and electrical systems used for motion can still work properly, but the sensor measurement system is no longer functional. There are also some other problems, such as slow measurement data acquisition rate, high measurement noise, and temporal asynchrony of measurement and motion.

- In terms of the bending test platform, there are problems of low motion reliability and inefficient measurement process.

To solve the above-mentioned problem to achieve better test quality, the two existing setups need to be upgraded. The focus of the upgrade is on the measurement electrical system, which determines whether reliable experimental results can be obtained. With traditional solutions, we must redesign the circuit diagram of the measurement system, select the right components and order components and PCBs from suppliers and manufacturers. If the results are not as expected, we would have to repeat the process again, which would cause a lot of time and financial costs. To increase the flexibility of the project, we decided to use the cypress PSoC, which means “Programmable system-on-chip”, as the MCU for the new data acquisition system. PSoC is a microcontroller of integrated circuit， which contains a lot of configurable components like MCU, timer, amplifier, ADC, PWM, UART, I2C, SPI, GPIO etc [7]. We only need to use the official integrated development environment (IDE), which provides an easily used graphical interface, on a computer to create some configuration files and flash them into the chip, then a circuit is built inside the PSoC chip. In layman's terms, it's like installing and uninstalling components on a breadboard, except that it all happens inside the PSoC (see Figure 1.6). If requirements change or there is a problem with the design, the solution can be changed simply by modifying the configuration, thus increasing development flexibility, and reducing costs.

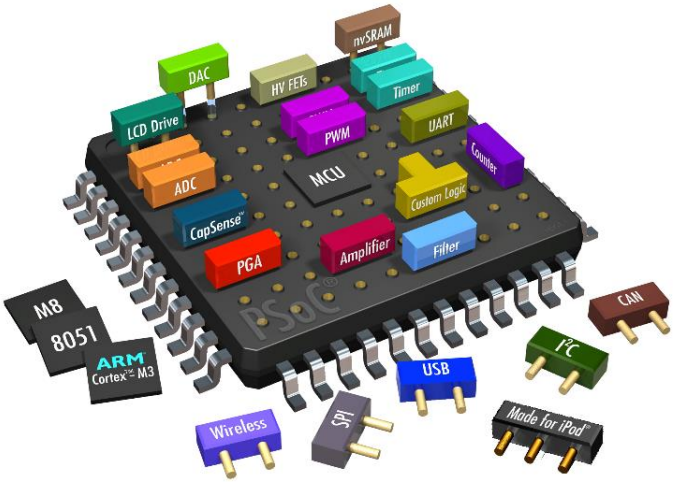


Figure 1.6: Illustration explaining the concept of PSoC: a single chip system integrated with various dynamically configurable components [8]

Therefore, in this work, we upgrade the original stretching and bending test setup. While retaining the original mechanical and part of the electrical system, the old measurement electronics is replaced with a new measurement system using PSoC. In addition, the software compatible with the new system is redeveloped. Finally, real sample testing will be conducted on the upgraded platform to evaluate the reliability of the test platform and the electrical characteristics of the samples.

2 Basics of PSoC Development

2.1 What is PSoC

PSoC is a product of microcontroller integrated circuits developed by Cypress Semiconductor. The two concepts, “programmable” and “system on a chip” in the PSoC name indicate its two main features [7]:

- “System on a chip”: System on a chip (SoC) integrates the microcontroller with a variety of commonly used analog and digital components in a single chip. These components include amplifiers, digital-to-analog converter (ADC), pulse width modulation (PWM), UARTs, timers, SPI, GPIO, I2C etc.
- “Programmable”: The internal components of a PSoC can be configured through programming, which means that people can dynamically configure them by modifying program stored in its memory to change the functionality of the internal components and the analog connections between components to achieve the desired functionality.

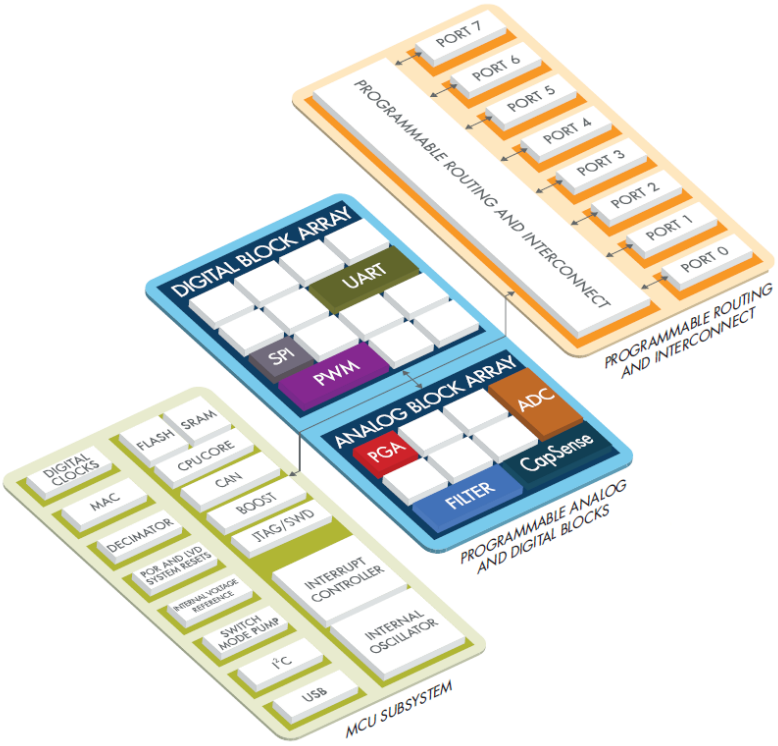


Figure 2.1: System architecture of PSoC [8]

Figure 2.1 shows the system architecture of PSoC, which can be divided into three levels [8]:

- The top level is the programmable routing and analog interface, which provides dynamically changeable internal line connections and input and output interfaces.
- The second level is the programmable analog and digital blocks. The analog block contains various switched capacitors, operational amplifiers, comparators, ADCs, DACs, and digital filter. The digital block consists of UART, SPI, and PWM. The user can connect blocks to the desired port in the top level through programmable routing.
- The lowest level is a MCU subsystem, which provides a complex CPU subsystem with SRAM, EEPROM and Flash memory, multiple core options and various basic system resources.

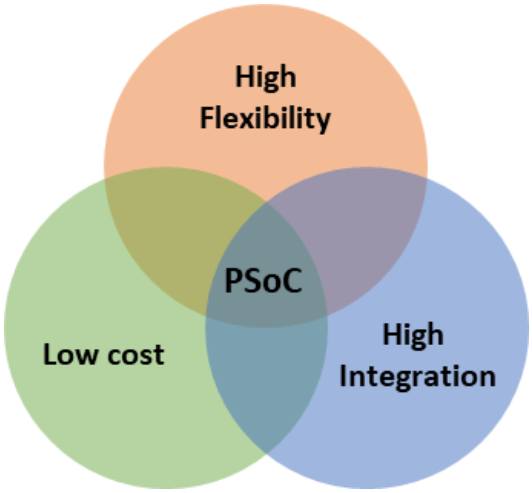


Figure 2.2: Advantages of PSoC: high flexibility, low cost and high integration

As shown in Figure 2.2, PSoC combines the three main advantages of high flexibility, low cost and high integration. In terms of flexibility, people can use the accompanying software for PSoC to dynamically design and implement your circuit in a matter of hours: Firstly, create the desired combination of peripherals. Next, select the pins to be used. Then generate and import the configuration file into PSoC. Finally, a new design is complete. Benefit from the flexibility of PSoC, the development time and economic cost of the project is reduced. In addition, the high integration level of PSoC can significantly reduce the size, weight and power of the product.

So far, Cypress has launched five different device families in total. Each family is built around a different microcontroller core. They are [7]:

- PSoC 1 - CY8C2xxxx family - M8C core.
- PSoC 3 - CY8C3xxxx family - 8051 core.

- PSoC 4 - CY8C4xxxx family - ARM Cortex-M0 core.
- PSoC 5/5LP - CY8C5xxxx family - ARM Cortex-M3 core.
- PSoC 6 - CY8C6xxxx family - ARM Cortex-M4 core, with the addition of an ARM Cortex-M0+ core.

As of February 2022, PSoC 6 is the latest PSoC product.

2.2 PSoC 6 BLE Pioneer Kit

In this work, we use a development board of PSoC 6 BLE Pioneer Kit (CY8CKIT-062-BLE) as the hardware platform to develop PSoC. This board contains a PSoC 6 series MCU and some for development necessary peripheral components such as IO headers, KitProg Programmer/Debugger, power supply, etc., so that users can learn and apply PSoC fast and easily.

Figure 2.3 shows the top view of the development board. Components 31 is the MCU of this board, Cypress PSoC 6 BLE CY8C6347BZI-BLD53, which is one of the PSoC 6 series chips mentioned in the previous section.

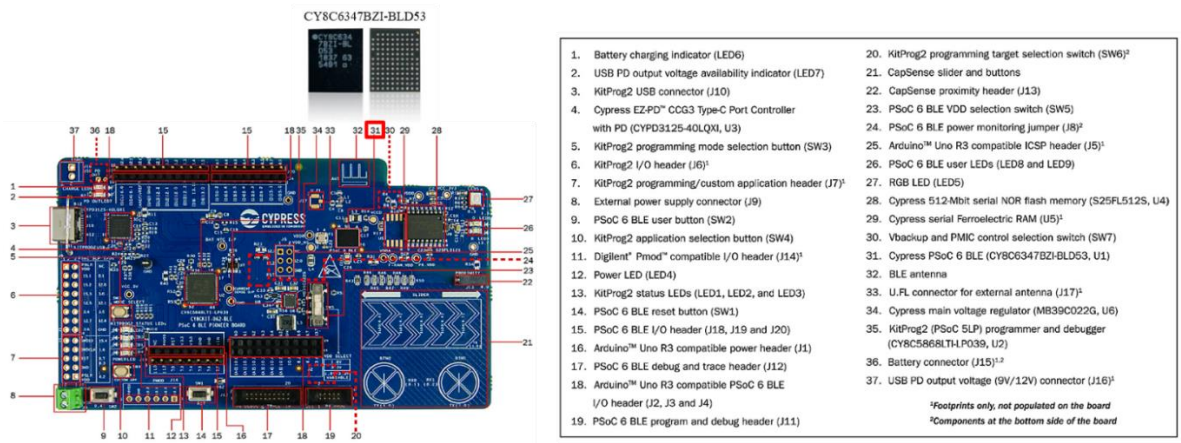


Figure 2.3: Components of PSoC 6 BLE Pioneer development board [9,10]

Table 2.1 shows the main parameters of the PSoC 6 MCU. The MCU contains two cores, the more powerful main core ARM Cortex-M4 and the less powerful second core ARM Cortex-M0. The memory contains a 1024kB Flash and 288kB SRAM. For communication, the MCU support serial communication such as I2C, UART and SPI, and Bluetooth communication. In addition, the MCU also contains two operational amplifiers, a 12-bit ADC, a 12-bit DAC, and 78 GPIOs [11].

Table 2.1: Specification of Cypress PSoC 6 BLE CY8C6347BZI-BLD53 [11]

Component	Parameter
Core	Main core: ARM Cortex-M4 (32-bit, up to 150 MHz) Secondary core: ARM Cortex-M0+ (32-bit, up to 100 MHz)
Memory	Flash: 1024 kByte SRAM: 288 kByte
GPIO	78
Communication	I²C, SPI, UART, BLE
Operational Amplifier	2
ADC/DAC	1 SAR ADC (12-bit, up to 1 MSPS) 1 DAC (12-bit, 200 kSPS)

2.3 Development of PSoC: PSoC creator

A PSoC project is usually developed in a windows software called PSoC creator, which is an Integrated Design Environment (IDE) developed by Cypress. Using this IDE, people can edit PSoC hardware configurations, write applications, compile and debug the applications, and flash them into PSoC chip with the help of the Kitprog programmer pre-installed on the development board. The dynamic hardware-software combination allows people to test their project in the hardware environment while viewing and debugging device activity in the software environment [12]. Figure 2.4 shows a typical workflow for developing a PSoC project using PSoC creator.

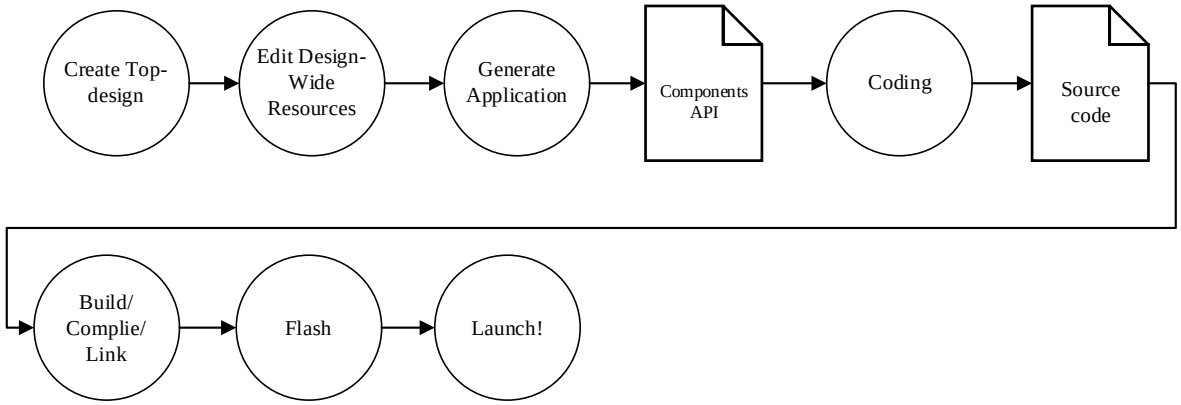


Figure 2.4: Flow chart of the development of a PSoC project

The first step is to create the Top-design, in which people design the circuit schematic, add the required components and configure their functionalities the connections. Then Design-wide resources is edited. in this step people manage the resources of PSoC, like Pin, Interrupt, analog routes, clock, etc. Then People use Generate Application to generate the APIs of the components automatically and dynamically. People can use these APIs to control the components conveniently in their own applications. The source files are converted into binary files through building, compiling, and linking. Finally, binary files are flashed into the PSoC and the PSoC is ready to launch.

Next, the step of top-design, Design wide resources and coding are described in detail.

2.3.1 Top-design

The Top-design Editor allows people to create schematics and insert components into the it. Figure 2.5 shows the interface of Top design. After opening the Top design tab in the browser, a blank canvas will be opened in the workspace. Components can be found in the built-in, pre-verified, production-ready peripheral component library on the right side. Components are placed in the library catalog represented by different symbols. Components are divided into Cypress and Off-chip. The former type exists really in the chip, and the latter type is external and only used for documentation. People can select the desired component from the library and drag and drop it into the canvas [12].

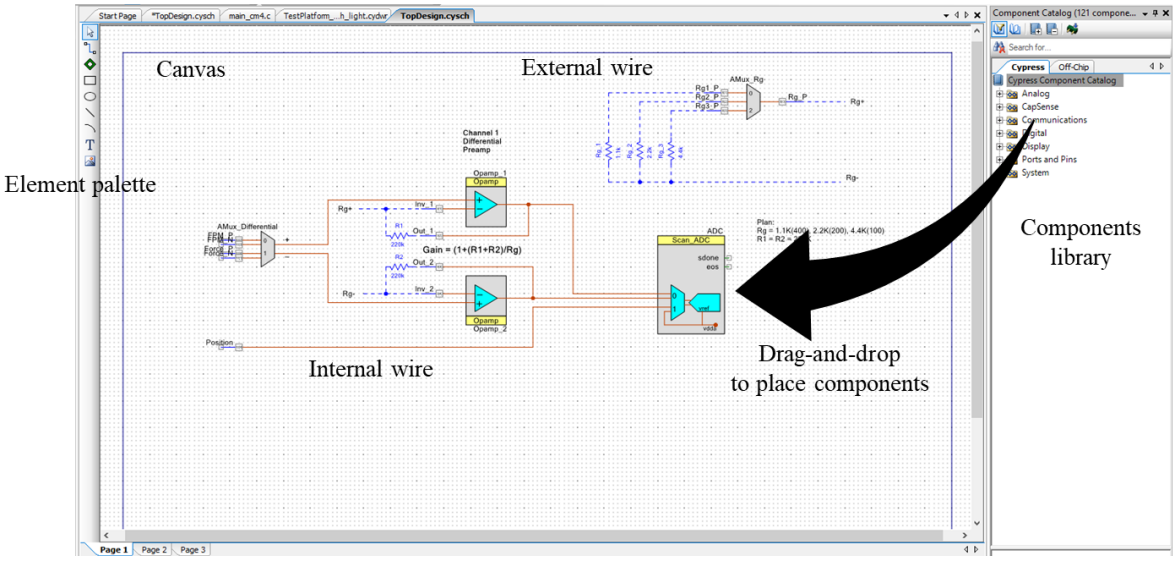


Figure 2.5: Top-desgin interface of PSoC creator

Connections can be created between components using the wire object in the Element Palette or the shortcut key “w”. When Cypress components are connected to Cypress components, the connection lines are shown in red, which means that these connections are present within the chip. When an on-chip component is connected to an off-chip component or between off-chip components, the connection lines are shown in blue, which means that these connections are for documentation purposes only. By double-clicking on the component symbol, people can also open the component configuration window to set the specific parameters of the component.

2.3.2 Design wide resources

In Design wide resources, people configure the resources in the PSoC, including pins, interrupts, clocks, analog routes, etc. In this section, we mainly focus on the assignment of Pin, which is the most important.

After the top-design, people need to assign the input and output signals of the added components to the GPIO pins of the chip. People can open the pin assignment window through tab “Pins” of Design wide resources in the browser (see Figure 2.6).

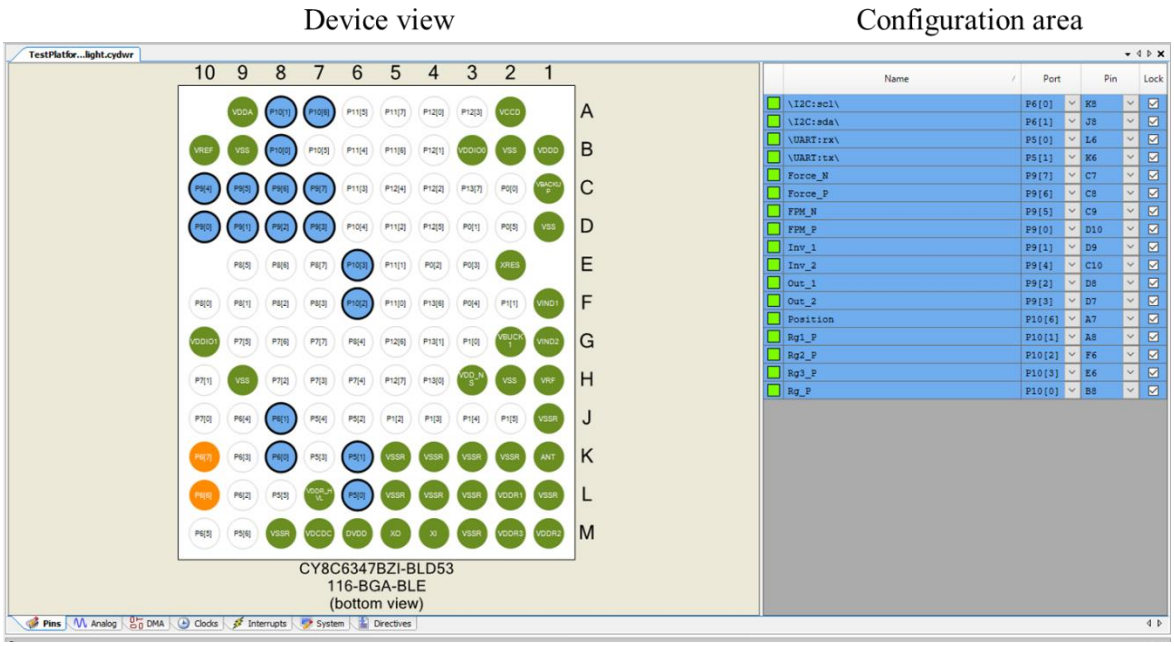


Figure 2.6: Design wide resources interface of PSoC creator

On the left side of the workspace is the device view, which shows the physical position of the pins on the chip. The right side of the workspace is the configuration area, which consists of a table. The first column of the table is the signal names of all components in top-design, the second column is the names of assigned pins, the third column is the number of assigned pins, and the fourth column is the assignment lock. People can assign the signals of components to the pins from combo box “Port” or “Pin”. These tow combo boxes are equivalent and the only difference between them is the representation.

The pin assignment is not completely free, some components can only use a fixed pin. PSoC creator indicate the pin availability via different colors. Green indicates that the assigned pin is working and is the best choice, yellow indicates that the assigned pin is working but not the best. Gray indicates that the pin is not available for the current component.

2.3.3 Application development

After finishing the Top-design and pin assignment, the basic configuration work is done. People can click button “Generate application” to generate API files, configuration files, database and so on. These files will be listed in the browser. There are four folders in the explorer where these files are stored, “*CM0P*”, “*CM4*”, “*Shared Files*” and “*Generated source*”. Then, people

can write their application in C in the code editor and call the variables and functions in the generated files (see Figure 2.7).

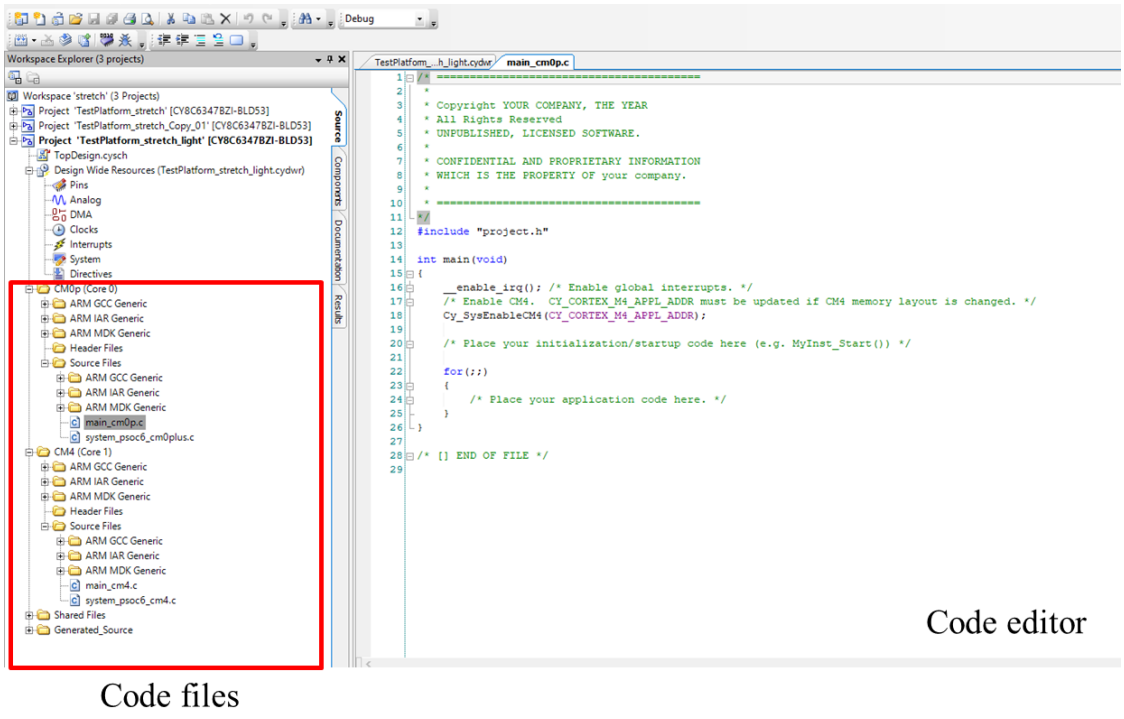


Figure 2.7: Application development interface of PSoC creator

In the following part, a brief introduction of the different code folders is given.

CM0P and CM4

Folder *CM0P* and *CM4* are used to store the main program. Since PSoC is a dual-core architecture, the two cores can execute different programs in parallel. Programs in folder *CM0P* are executed by the ARM Cortex-M0 core and programs in folder *CM4* are executed by the ARM Cortex-M4. There are two files called *main_cm0.c* and *main_cm4.c* in the two folders, which are the entry points of the programs for both cores. They are the best starting point for your program.

Generated Sources

Folder *Generated sources* holds the APIs of the components added to the design. These folders are automatically included in the file *main_cm0.c* and *main_cm4.c*. So, people can use the variable and functions of these APIs in their program without any configuration.

Share Files

In folder *Share Files* public code files are stored, such as the standard input and output library *stdio_user.c* and the macro callbacks declaration *cyapicallbacks.h* for calling user's code from the generated codes of components.

2.4 Consideration of design

2.4.1.1 Analog routing

In PSoC creator, we have the flexibility to create connections and assign interfaces to components. But in practice the connections and assignments are not completely free. Because the analog circuits inside the chip are limited. When people finish the configuration of the circuit and builds the project, PSoC creator routes the internal circuits of PSoC according to the user defined connections. Once a conflict occurs between routes, analog routing error is reported. Therefore, it is important to understand the structure of the analog routes inside the PSoC to design the schematic properly.

Figure 2.8 shows the analog routes inside the PSoC 6. There are two ring analog buses AMUXBUS A and AMUXBUS B.

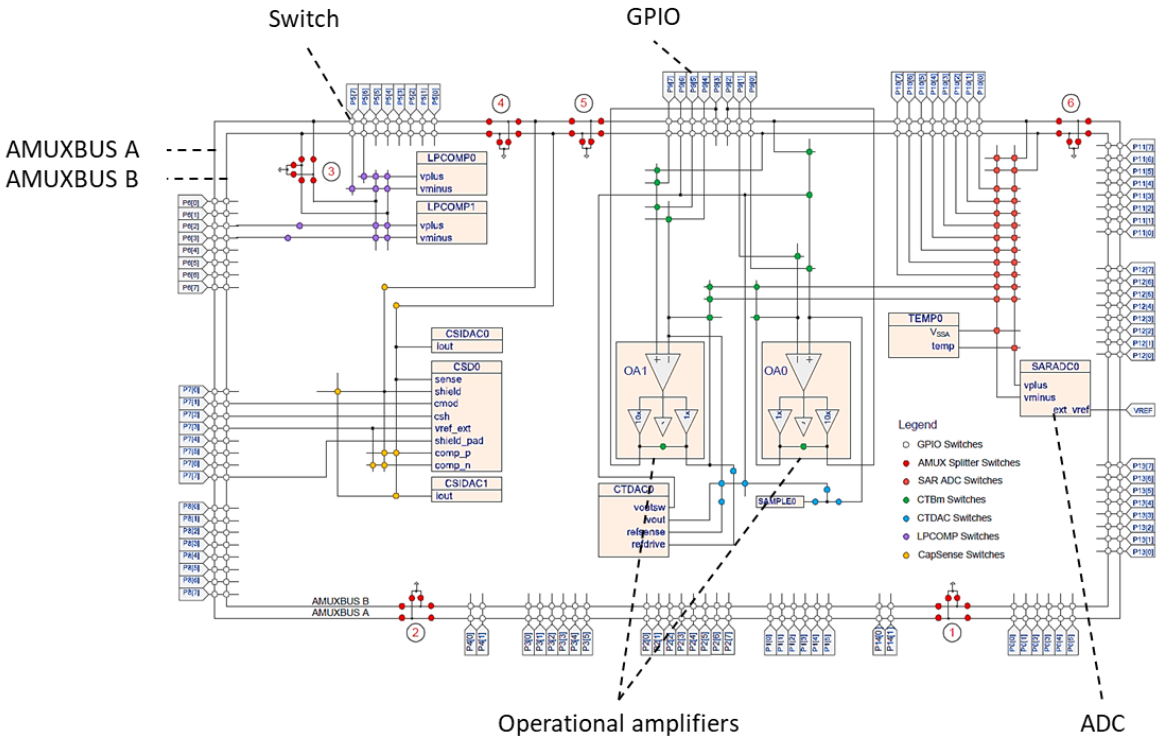


Figure 2.8: PSoC 6 MCU analog routing Diagram [13]

The two analog buses are connected to the GPIO pins and the analog components via switch, which are indicated by circles. It is the on/off states of these switches that implement the connections between the components in the top-design. In addition, analog multiplexer (AMux) components are also implemented via switches.

Figure 2.9 shows the principle of PSoC AMux component. An AMux with two input channels, *IN0* and *IN1*, and one output channel *OUT* is given. Signal *IN0*， *IN1* and *OUT* is assigned to pin P10.2, P10.1and P10.0. The selection of different channels of the multiplexer is achieved by controlling the switch between these three pins and the bus:

1. As shown in Figure 2.9(a), when the switch between pins *IN0* and *IN1* and AMUXBUS are disconnected, no signal is output to pin *OUT*, which is equivalent to AMux disconnecting all channels.
2. As shown in Figure 2.9(b), when the switch of pin *IN0* is connected to the AMUXBUS but *IN2* not, the signal of IN0 pin is output to pin *OUT*, which is equivalent to AMux selecting channel 0.
3. As shown in Figure 2.9(c), when the switch of pin *IN1* is connected to the AMUXBUS but *IN1* not, the signal of *IN1* pin is output to pin *OUT*, which is equivalent to AMux selecting channel 1.

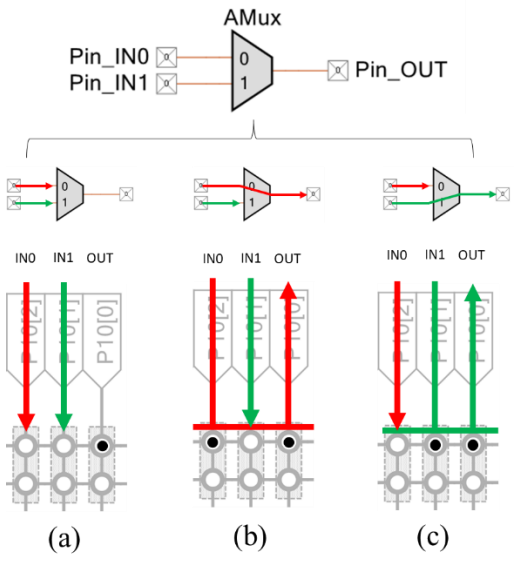


Figure 2.9: Schematic of the principle of the multiplexer of PSoC

Among the switches there is a special class of switches called AMUXBUS splitter switch, which is represented by the red circle. As shown in Figure 2.10, by controlling the connection status of the splitter, we can divide each bus into several segments [13]. On both buses of PSoC6,

there can be at most two analog signals transmitted in parallel at the same time. By dividing them with splitters , it is possible to have multiple analog signals transmitted in parallel without interfering with each other.

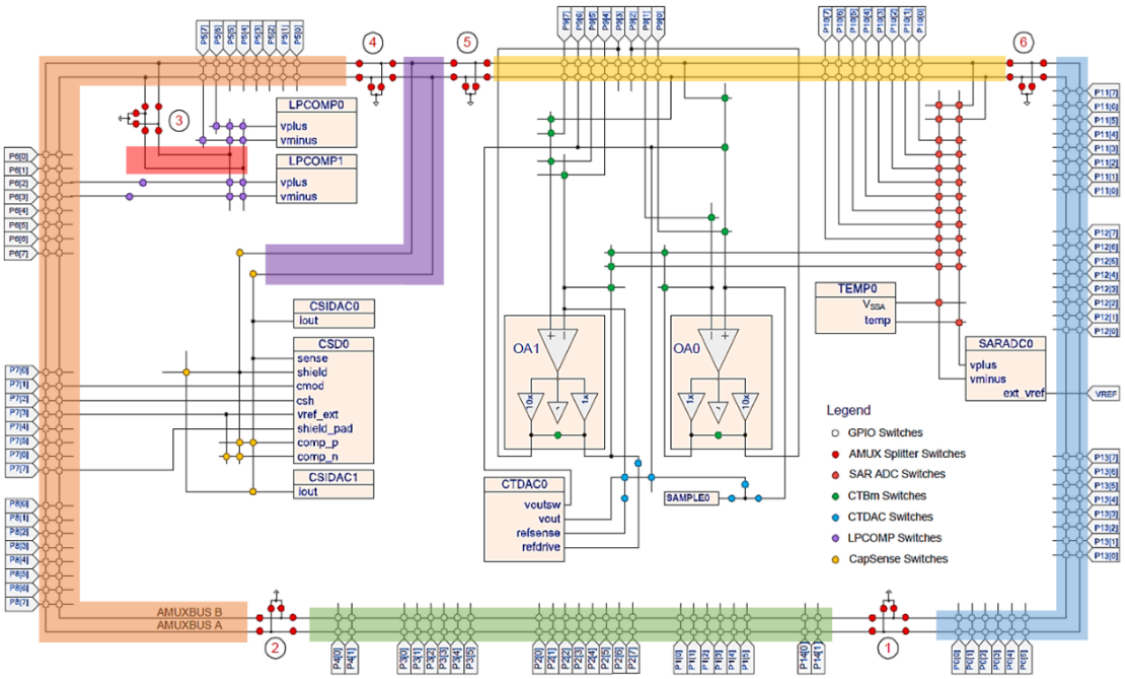


Figure 2.10: Splitting of PSoC 6 analog bus system using AMUXBUS splitter

2.4.1.2 Pin selection

PSoC 6 has 78 GPIO pins for analog signal input and output. Some of these pins have dedicated connections to the internal components. Assigning dedicated pins to components has two advantages.

First, the dedicated connection allows the signal to be transmitted without occupying additional AMUXBUS, thus maximizing the use of analog route resources.

Second, the dedicated connection reduces the number of switches along the analog route, resulting in less wire resistance.

Therefore, it is important to assign GPIOs according to the component type. For example, the priority of GPIO pins selection for ADC and operational amplifier are introduced in Table 2.2. And the selection of other components can be referred to documentation “PSoC 6 MCU Hardware Design Considerations”.

Table 2.2: Pin selection for ADC and operational amplifier of PSoC 6 [14]

Component	Signal	First priority	Second priority	Third priority
ADC	Input	P10[0:6]	P9[0:7]	others
Operational amplifier	Positive input	P9[0], P9[6], P9[5],P9[7]	others	-
	Negative input	P9[1], P9[4]	others	-
	Output	P9[2], P9[3]	others	-

ADC

P10[0:6] is the most suitable pins for ADC input because P10[0:6] has the dedicated connection to ADC input.(see Figure 2.11) P9[0:7] is the second most suitable pin for ADC input but has a slight disadvantage of switch resistance compared to P10[0:6].

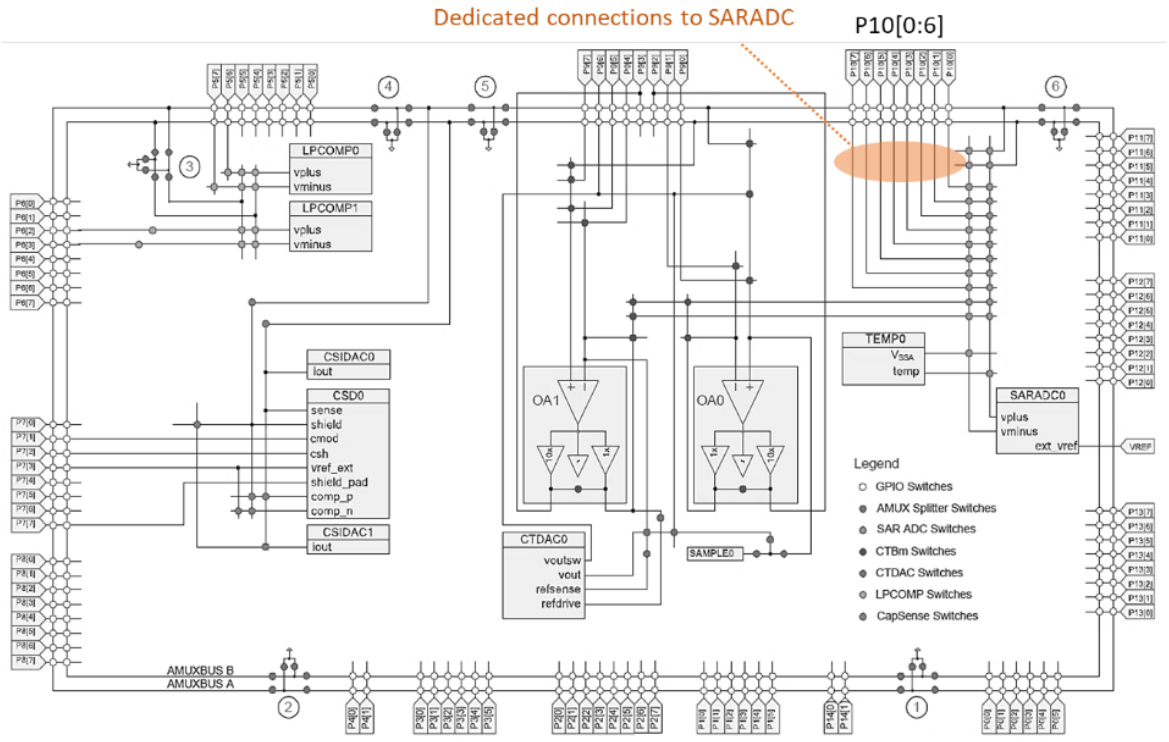


Figure 2.11: The dedicated connections between GPIOs and SARADC of PSoC 6

Operational amplifier

P9[0:7] is the most suitable GPIO pins for operational amplifiers. They gave dedicated connections to the input sand outputs of operational amplifiers (see Figure 2.12). P9[0], P9[6], P9[5], P9[7] are connected to the positive input, P9[1], P9[4] are directly connected to the negative input and P9[2], P9[3] are directly connected to the output.

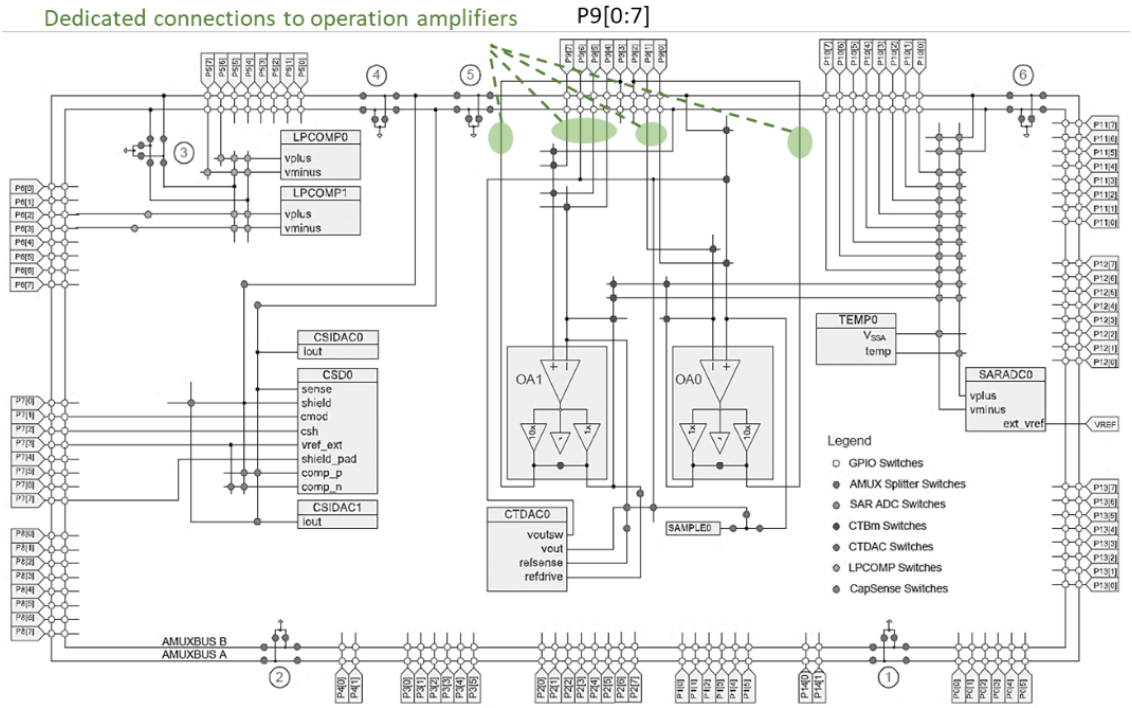


Figure 2.12: The dedicated connections between GPIOs and operational amplifier of PSoC 6

2.4.1.3 Board pin header

The GPIO pins of PSoC 6 can be accessed through the corresponding I/O header on the development board. For example, the GPIO pins P9, P10 of PSoC can be accessed with headers P9 and P10 on the board. However, people must note that not all headers can access its corresponding pins, since some header not connected to corresponding PSoC GPIO pins, but to peripherals on the board. Sometimes, by soldering jumper resistors in some specific reserved solder pads on the board or removing some jumper resistors, can change the connection of the headers and make the corresponding GPIO pins accessible again. The number of these resistors on the board can be obtained by consulting “PSoC 6 BLE Pioneer Kit Guide”.

Take header P9.3 as an example, according to the manual, the primary function of this header is TRACEDATA[0] input, which means that the header is connected to TRACEDATA[0] by default. If people want to use it to access GPIO pin P9.3, people need to populate R131 with a 0 Ω resistor to connect the header to the pin. (See Figure 2.13)

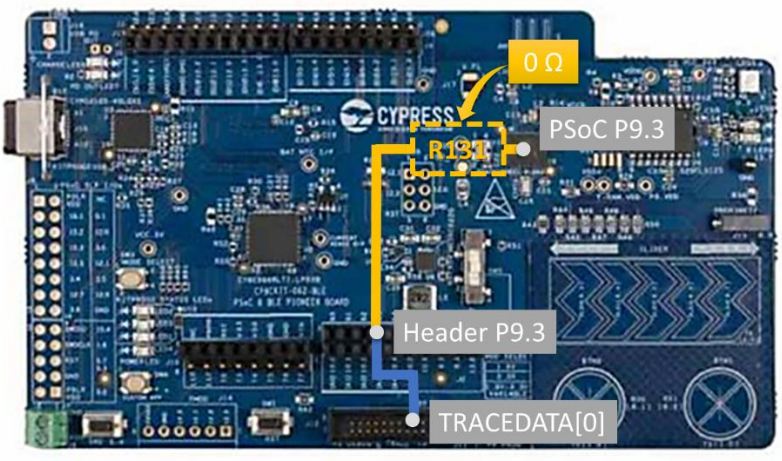


Figure 2.13: How to connect board header P9.3 and PSoC GPIO P9.3: populate R131 with a 0 Ω resistor

2.4.1.4 Application example

To better help readers understand the above-mentioned points, an example is given in this section. Figure 2.14 shows a measurement circuit of an array sensor that can measure the voltage across any one of the resistors in the array.

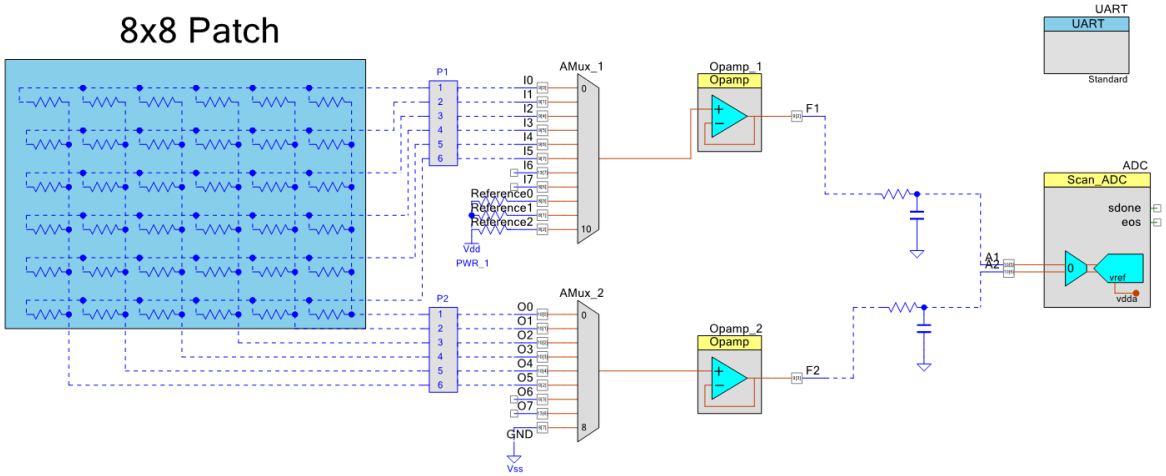


Figure 2.14: Top-design diagram of an example of a measurement circuit for an 8x8 sensor resistor array

By controlling the input channels *I0* to *I7* and *O0* to *O7* of multiplexer *AMux_1* and *AMux_2*, to measured resistance in the array can be loaded into the measurement circuit consisting of the reference resistor, a DC power supply and ground. At the same time, the voltage of the loaded resistor is input into PSoC via the two outputs of multiplexers. Then, the two poles of the voltage signal are input into two voltage followers to isolate the pre- and post-stage circuits. The signal is then output to an off-chip low-pass filter to reduce the noise. The filtered signal is re-input into the chip and converted to a digital value by the ADC. In addition, by selecting the reference0, reference1 and referenc3 channels of *AMux_1*, the suitable reference resistance can be selected according to the resistance to be measured, thus making full use of the measurement range of the ADC.

Now, let us try to assign ports to the pins in this circuit. The assignment result is shown in the Figure 2.15.

Name	Port	Pin	Lock
\UART:rx\	P5[0]	L6	☑
\UART:tx\	P5[1]	K6	☑
A1	P10[5]	B7	☑
A2	P10[6]	A7	☑
F1	P9[2]	D8	☑
F2	P9[3]	D7	☑
GND	P8[7]	E7	☑
I0	P9[0]	D10	☑
I1	P9[1]	D9	☑
I2	P9[4]	C10	☑
I3	P9[5]	C9	☑
I4	P9[6]	C8	☑
I5	P9[7]	C7	☑
I6	P13[7]	C3	☑
I7	P8[6]	E8	☑
O0	P10[0]	B8	☑
O1	P10[1]	A8	☑
O2	P10[2]	F6	☑
O3	P10[3]	E6	☑
O4	P10[4]	D6	☑
O5	P6[2]	L9	☑
O6	P6[3]	K9	☑
O7	P13[6]	F4	☑
Reference0	P8[0]	F10	☑
Reference1	P8[1]	F9	☑
Reference2	P8[2]	F8	☑

Figure 2.15: Pin assignment of the component signals for the example

The following reasons for the choice of pin is given:

1. First, for two multiplexers, they are implemented by connect input pins and output pins together with the help of analog bus. Therefore, the two multiplexers occupy two analog buses in total.
2. The output of two voltage followers has dedicated connections to GPIO pin 9[2], 9[3]. To avoid analog bus occupation as much as possible, these two pins are assigned for the outputs *F1*, *F2* of the voltage followers.
3. The filtered signal needs to be input to the PSoC and converted into a digital signal by the ADC. Here, P10[5] and P10[6], which has a dedicated connection to the ADC, is assigned to inputs *A1* and *A2* of the ADC to avoid bus occupancy.
4. So far, only two analog buses are used, so we do not need to consider additional splitting of the buses. We can choose any of the remaining GPIO pins to assign to the inputs of *AMux_1* and *AMux_2*. It is better to choose the GPIO pins that can be accessed directly by the headers on the board. According to "PSoC 6 BLE Pioneer Kit Guide", these pins include P9[0:7] (except P9[3]), P10[0:6], part of the P6 and part of the P13. Since P9 and P10 have a lower route impedance to the operational amplifier, P9 and P10 have a higher priority in the assignment.
5. After all the pins mentioned above have been assigned, there are still some component signals that are not assigned. In this case, we select headers from the headers that is connected to the GPIO pin as the secondary function, such as P8[0:7].

3 Design of stretching test setup

The main body of the stretching test setup is designed by Vincent Gottwald in his Bachelor work "Aufbau eines Dehnungsprüfstands für Leiterbahnelemente mit integrierter Widerstandsmessung". For reader's better understanding and to avoid chaos article structure, in this thesis the old original part and the new upgrade part are not reviewed separately. But a complete description of the final setup is given.

3.1 Test principle

3.1.1 Sample structure

Figure 3.1 shows the structure of the sample used for stretching test. The sample is fabricated based on a PDMS substrate. The tested object and contact pads are printed on the substrate using screen printing. At both end of the tested object, a pair of contact pads is placed for the connection to the measurement electronics.

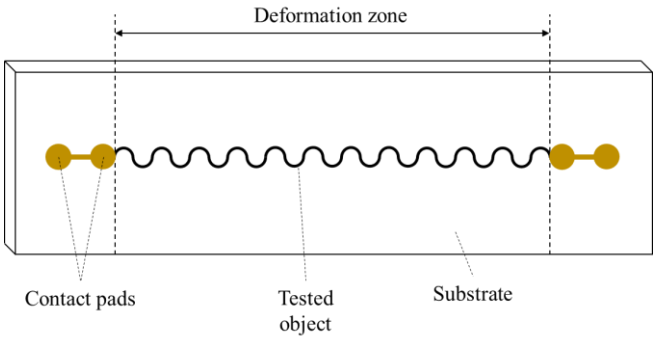


Figure 3.1: Schematic of the structure of the sample for stretching test

3.1.2 Test process

Figure 3.2 shows the principle of the stretching test. One end of the sample is fixed in some way to the setup to ensure that only the deformation area is stretched, and the contact pads is not deformed, thus avoiding poor contact between the measuring electronics and the sample. The other end of the sample has the freedom of movement along the axial direction. During the test, the free end is pulled by some mechanics to stretch the sample. Similarly, to prevent deformation of the contact pad, the applied force should be on the boundary of the deformation zone. Once the sample reaches the specified elongation, the setup moves in the opposite

direction until the sample returns to the initial length. In the process of the sample being stretched, the sample elongation and resistance are measured at regular intervals. Finally, we obtain the strain-resistance characteristics of the sample along the longitudinal direction in the case of elongation and shortening for subsequent studies.

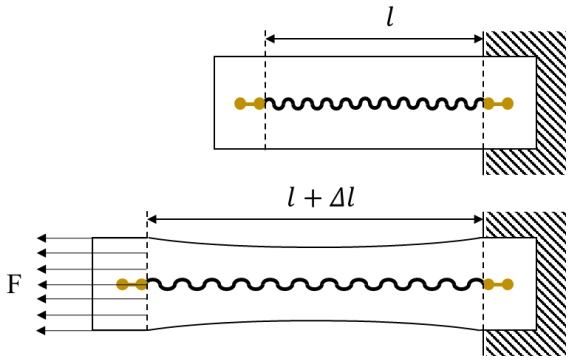


Figure 3.2: Schematic of the principle of the stretching test

3.1.3 Measurement principle

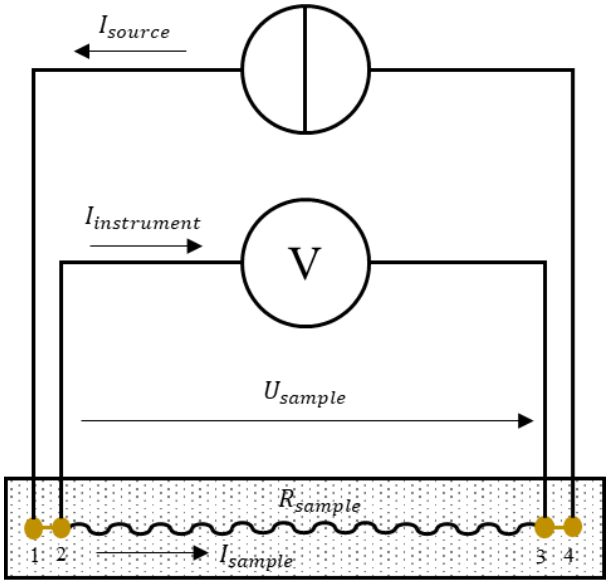


Figure 3.3: Schematic of the principle of four-point measurement method [5]

F In the stretching test, four-point measurement method is used to measure the resistance of the tested object. our-point measurement method is based on Ohm's law. Figure 3.3 shows the principle of the four-point measurement method. A current source is connected to the sample via contact pad 1 and 4, which provides a constant current I_{source} to generate excitation signal for the measurement. A voltmeter is connected to contact pad 2 and 3 of the sample to measure

the output voltage U_{sample} of the sample. Since the voltmeter has a very high impedance, the current flowing through the branch of the voltmeter is nearly zero, i.e., $I_{instrument} \approx 0$. So, the current flowing through the sample is almost exactly equal to the total current, i.e., $I_{sample} \approx I_{source}$. According to Ohm's law, the resistance of the sample is can be calculated by the following formula

$$R_{Sample} = \frac{U_{Sample}}{I_{Source}} \qquad \text{formula 3-1}$$

Strictly speaking, the resistance of the contact pad 2, 3 is also included into the calculated result. However, since the resistance of the samples is much larger than the contact pad, the error is ignorable.

3.2 System overview

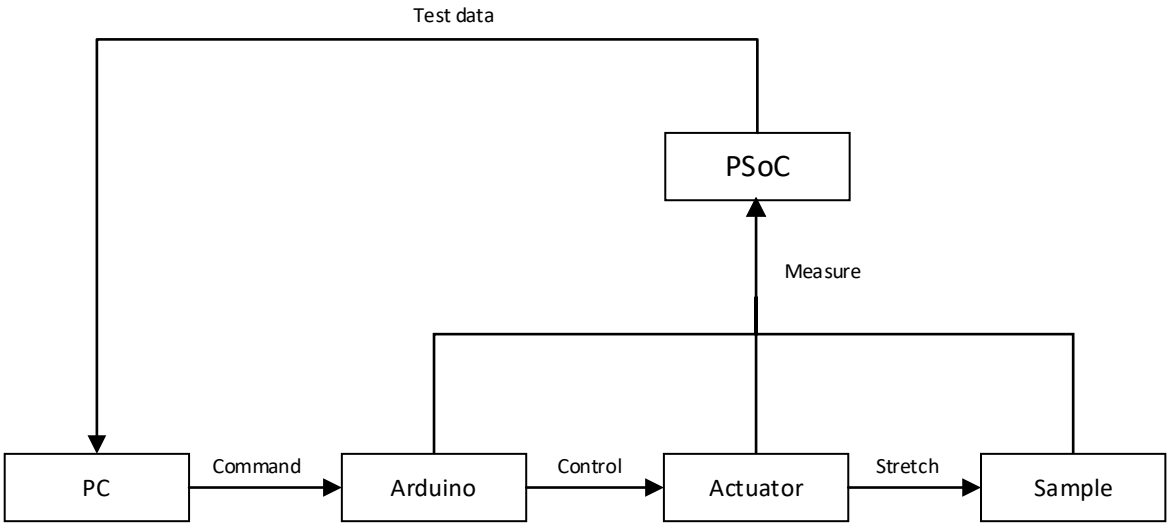


Figure 3.4: Schematic overview of the stretching test setup

Figure 3.4 shows the overview of the stretching test setup, which consists of a PC, an Arduino UNO development board, a PSoC 6 development board, actuators and a sample. The PC is the master of the whole test system. It provides a graphical interface (GUI) program for the user to send test commands and receive and visualize measurement data. The Arduino receives the commands from the PC and converts them into control signals. The actuator including mechanical components such as motors, transmission structures, etc., converts the control signals into mechanical motion to stretch the sample. The PSoC measures the resistance of the

sample and other signals like fore and position of the actuator. Finally, the test data is sent back to the PC to be shown to the user.

3.3 Mechanics

Figure 3.5 shows the kinematic model of the actuator. One end of the sample is fixed to a fixture and the other end is fixed to a sliding block. The sliding block is connected to a toothed belt. A stepper motor applies a torque to the toothed belt pulley to drive the toothed belt counterclockwise. Under the pull of the toothed belt, the slider translates to the left. Then the sample is stretched in the direction of elongation. Similarly, when the stepper motor is turned clockwise, the sample is stretched in the shortening direction.

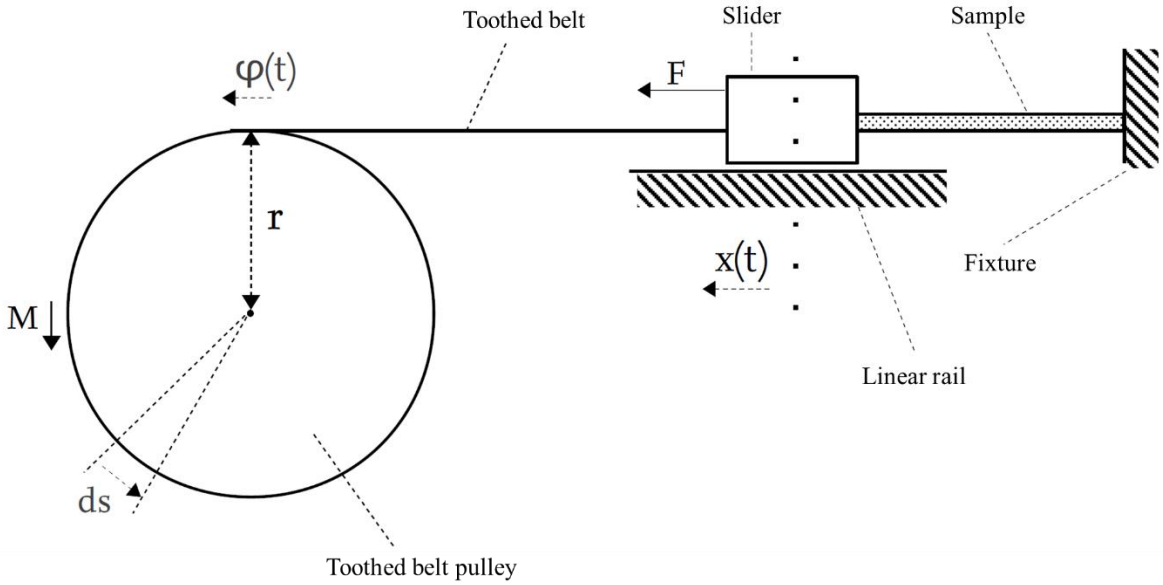


Figure 3.5: Kinematic diagram of the actuator of the stretching test setup [5]

In the case of the drive mode of 3600 microsteps, the slider translates 13.69 µm for one motor step [5]. Thus, the relation between the length of the sample being stretched and the number of the motor steps can be expressed by the following formula:

$$\Delta L = 13.69\ \mu m \cdot N_{step} \qquad \text{formula 3-2}$$

Figure 3.6 shows a 3D model of the mechanical part of the stretching test setup. The bottom of the slider is connected to a toothed belt with holes on both sides that are crossed by two linear

rails, limiting the slider to move in linearly. The fixture is attached to the frame by means of an S-type load cell for force measuring. It also has holes on both sides that are crossed by identical linear rails, ensuring that its central axis is aligned with the central axis of the Slider. A hinge structure is installed on each of the slider and fixture. People can fix the sample to the setup by closing the sample cover downward.

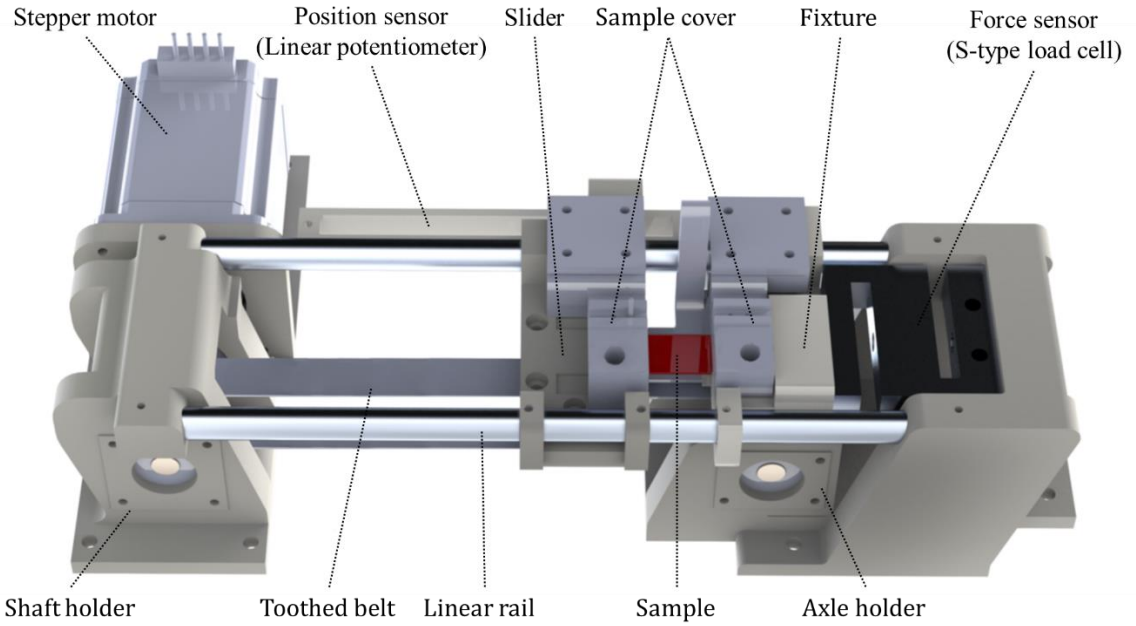


Figure 3.6: 3D model of the mechanical structure of the stretching test setup [5]

3.4 Electronics

Figure 3.7 shows the schematic diagram of the electrical system. According to the functionality of the components, the electrical system can be divided into three subsystems, data processing system, motion control system and measurement system:

- Data processing system consists of a PC, which is responsible for human-computer interaction, test command sending and data receiving.
- Motion control system consists of a stepper motor, an Arduino UNO development board, a motor driver and a 12V DC power supply. The Arduino controller controls the movement of the motor through the motor driver.
- Measurement system consists of a PSoC 6 development board, a PSoC shield board, a current source providing excitation signals for the sensors, a four-point measurement unit, a force sensor and a position sensor. The PSoC shield board is used to carry off-

board external components for the measurement circuit. The four-point measurement unit is used to measure the resistance of the sample, the force sensor is used to measure the tensile stress on the sample, and the position sensor measures the strain of the sample. The PSoC collects the output signal of the four-point measurement and sensors.

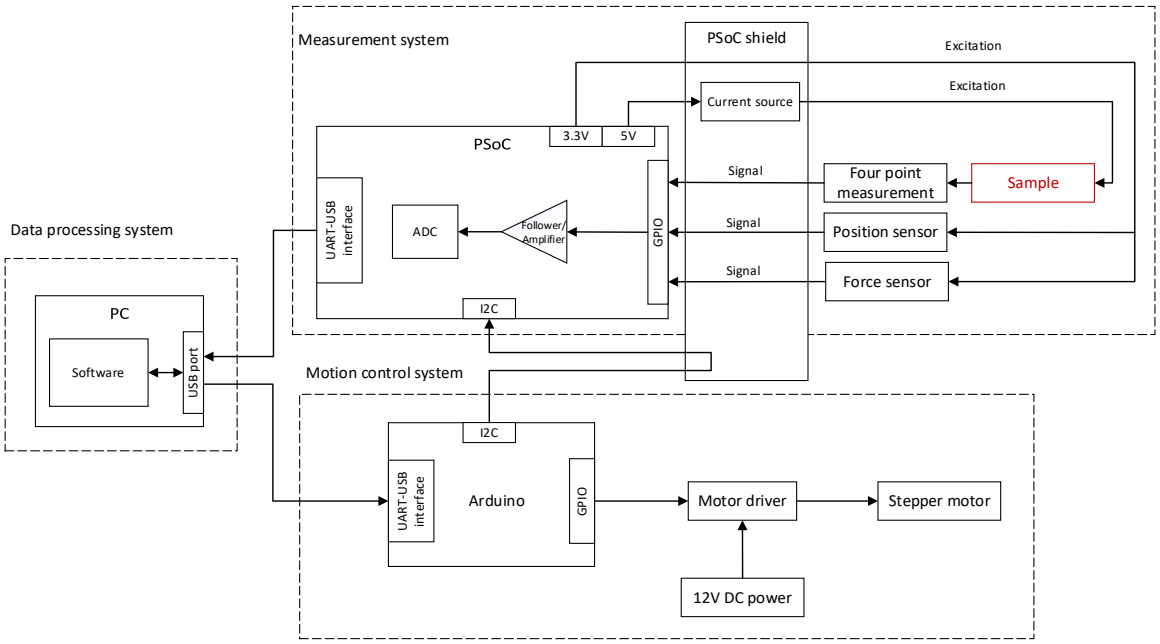


Figure 3.7: Schematic overview of the electronics part of the stretching test setup consisting of three subsystems

3.4.1 Data processing system

The PC is connected to the PSoC of the measurement system and the Arduino of the motion system via USB cables. People can configure the test on the PC and send commands to the Arduino via serial port. The PC also receives the measurement data from the PSoC via serial port.

3.4.2 Motion control system

The motion control system consists of an Arduino UNO development board, a motor driver, a stepper motor, and a 12V DC power supply. Table 3.1 shows the models of these components.

Table 3.1: Components of motion control system of stretching test setup [5]

Component	Model
Arduino	Arduino UNO
Motor driver	TB6600
stepper motor	Sanyo Denki 103H7126
12V DC power supply	SRS150-12

The components are connected in the following configuration:

1. Port ENA-, DIR- and PUL- of the motor driver is connected to port 4, 2, 3 of the Arduino board.
2. Port ENA+, DIR+ and PUL+ of the driver is connected to +5 V of the Arduino board.
3. Port A+, A-, B+, B- of the driver is connected to the winding of the stepper motor
4. Port VCC of the driver is connected to the 12 V output of the DC power supply.
5. Port GND of the driver, GND of the Arduino are connected to the GND of the DC power supply.

The Arduino drives the motor by setting the pin level of port ENA-, DIR- and PUL-. Port DIR- controls the rotation direction of the stepper, and port ENA- controls the sleep and wake of the stepper motor. Every time the Arduino sends a falling edge signal to the port PUL-, the stepper motor takes a step. The TB6600 motor driver also features a microstep drive for smaller step resolution. in this work, the microstep is set to 3600 steps/r.

3.4.3 Measurement system

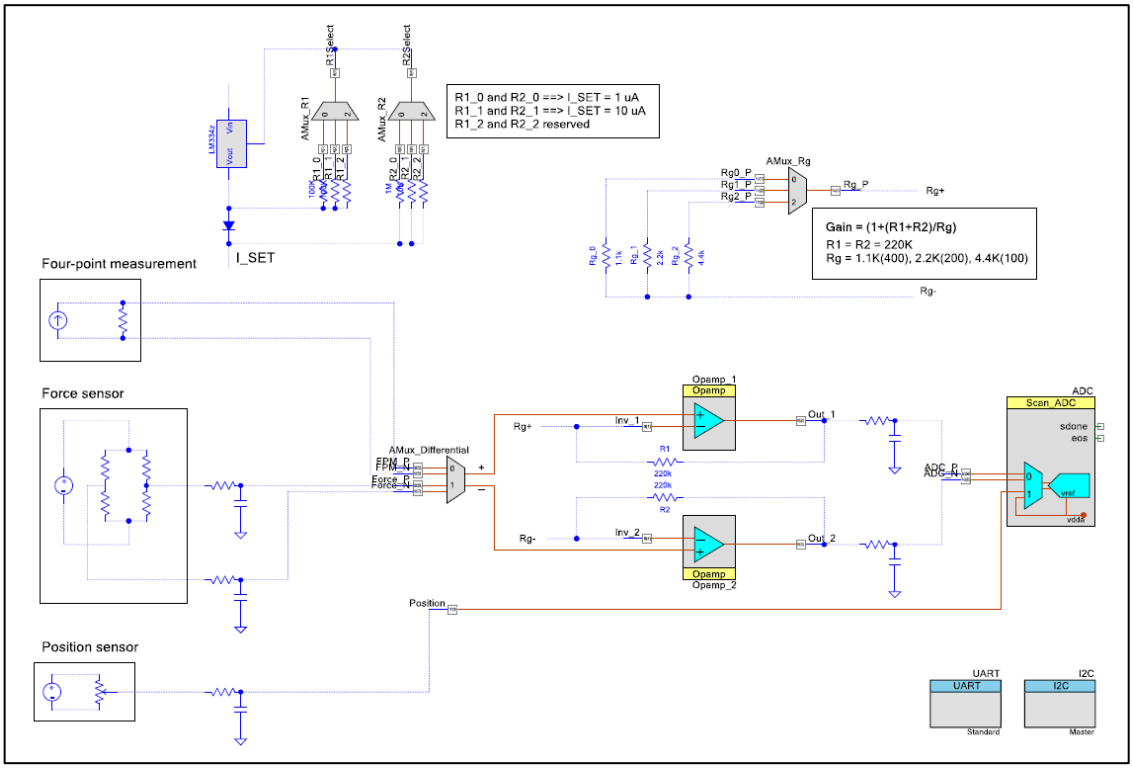


Figure 3.8: Top-design of the PSoC based measurement system of the stretching test setup

Figure 3.8 shows the top-design of the PSoC, which is also the schematic of the measurement system. The red lines are the on-chip circuit, and the blue lines is the external circuit. The system consists of sensor units including a four-point measurement unit, a force sensor and a position, and a PSoC-based measurement circuit including primary filtering circuit, a signal input circuit, a pre-amplifier circuit, a secondary filtering circuit, an ADC, a UART component and an I2C component.

Name	Port	Pin	Lock
\I2C:scl\	P6[0]	K8	☑
\I2C:sda\	P6[1]	J8	☑
\UART:rx\	P5[0]	L6	☑
\UART:tx\	P5[1]	K6	☑
ADC_N	P10[5]	B7	☑
ADC_P	P10[4]	D6	☑
Force_N	P9[7]	C7	☑
Force_P	P9[6]	C8	☑
FPM_N	P9[5]	C9	☑
FPM_P	P9[0]	D10	☑
Inv_1	P9[1]	D9	☑
Inv_2	P9[4]	C10	☑
Out_1	P9[2]	D8	☑
Out_2	P9[3]	D7	☑
Position	P10[6]	A7	☑
R1_0	P8[1]	F9	☑
R1_1	P8[2]	F8	☑
R1_2	P8[3]	F7	☑
R1Select	P8[0]	F10	☑
R2_0	P8[5]	E9	☑
R2_1	P8[6]	E8	☑
R2_2	P8[7]	E7	☑
R2Select	P8[4]	G6	☑
Rg0_P	P10[1]	A8	☑
Rg1_P	P10[2]	F6	☑
Rg2_P	P10[3]	E6	☑
Rg_P	P10[0]	B8	☑

Figure 3.9: Pin assignment of the PSoC for the measurement system of the stretching test setup

Figure 3.9 shows the pin assignments of the PSoC for component signals. Here, PSoC pin P9.3 is assigned to signal Out2 to connect the output of voltage follower Opamp_2 to the external circuit. As shown in Table 3.2, by default, on a newly purchased PSoC 6 BLE Pioneer Kit development board the P9.3 pin of the PSoC chip is not connected to the P9.3 header but to pin TRACEDATA[0] of the Embedded Trace Module (ETM) on the board . So, for a new PSoC 6

development board, we need to solder a 0 Ω resistor to position R131 on the board to establish the connect of the PSoC pin and board header, so that external circuits can access the PSoC P9.3 pin through the header P9.3.

Table 3.2: On-board connection and connection change of PSoC 6 pin P9.3 [9]

PSoC pin	Default on-board connection	Optional on-board connection	Connection change
P9.3	TRACEDATA[0]	GPIO on header P9.3	Populate R131 to implement optional on-board connection

3.4.3.1 Measurement units

Four-point measurement

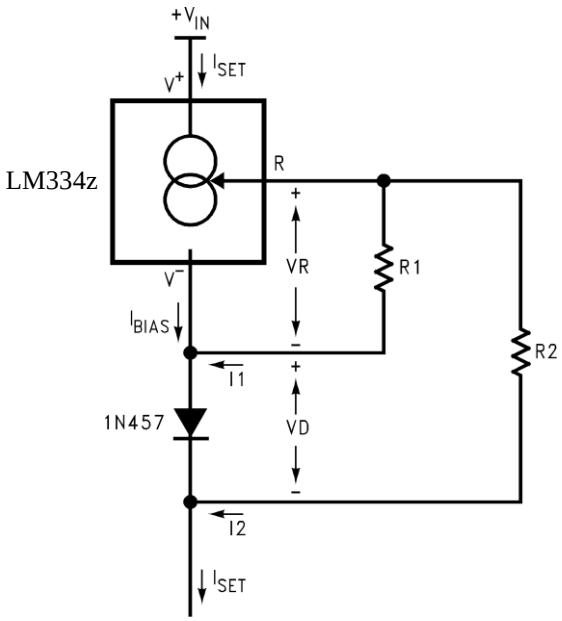


Figure 3.10: Schematic of a temperature-independent current source based on LM334z 3-Terminal adjustable current source [15]

The principle of the four-point measurement is described in Chapter 3.1.3. A current source is required to provide excitation current to the sample. Figure 3.10 shows the schematic of a temperature-independent current source based on LM334z. The circuit consists of an LM334z adjustable current source, a 1N457 diode and two resistors R1,R2. The advantage of this current source is that the output current is not affected by environment temperature.

When $R_2 = 10R_1$, the output constant current I_{SET} can be determined by the following formula [15]:

$$I_{SET} \approx \frac{0.134V}{R_1} \qquad \text{formula 3-3}$$

In the practice, people usually need different current to measure different resistance so that the output voltage of the sample locate in the optimal measurement range of the ADC. When the measured resistance is larger (e.g. 1 MΩ), smaller current is more suitable since larger current leads to the output voltage too high for the ADC. When the measured resistance is small (e.g. 1 kΩ), large current is better to make the measurement more sensitive.

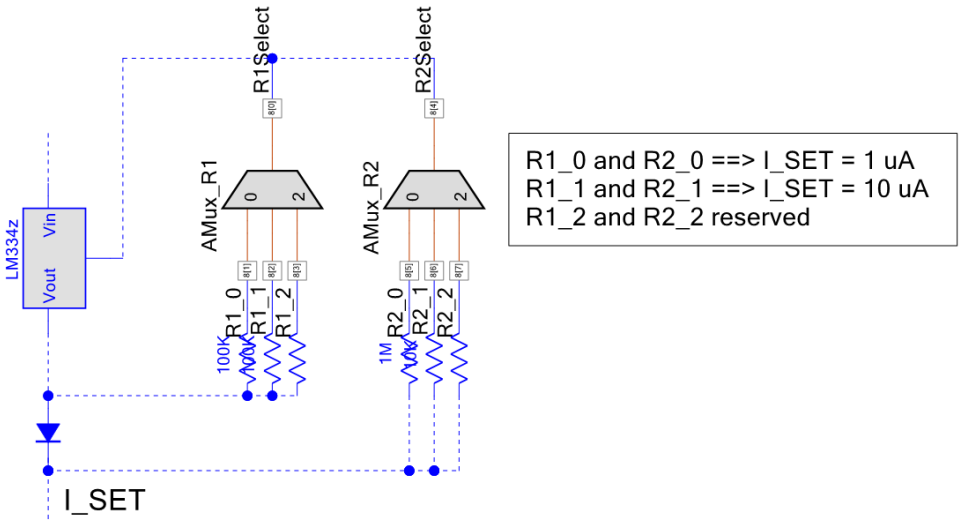


Figure 3.11: Schematic of the current source configuration circuit

Therefore, as shown in Figure 3.11, two multiplxers *AMux_R1* and *AMux_R2*, which are implemented by PSoC, is used to change the resistance of R_1 and R_2 in Figure 3.10 to obtain different output current I_{SET}. Here, both multiplxers have three resistance channels, so three different output current is available. Table 3.3 shows the different multiplexer configurations and its corresponding output current of the current source.

Table 3.3: Available output current of the current source in the case of different multiplexer configurations

AMux_R1 input channel	AMux_R2 input channel	R1	R2	Theoretical Output current	Actual Output current
0	0	100 kΩ	1 MΩ	1 µA	1.15 µA
1	1	10 kΩ	100 kΩ	10 µA	12.12 µA
2	2	Reserved	Reserved	-	-

Force Sensor

A "LCT LAS-A" S-type load cell is used as the force sensor (see Figure 3.12(a)). A Wheatstone bridge consisting of strain gauges is installed on the load cell, which outputs a voltage signal when there is tension or pressure along the longitudinal direction of the load cell (see Figure 3.12(b)).

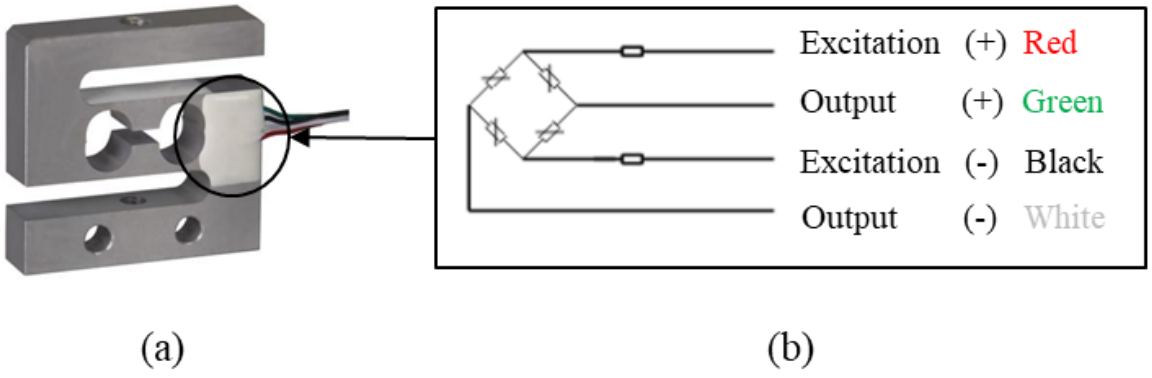


Figure 3.12: (a) Photograph of a "LCT LAS-A" S-type load cell (b) Schematic of the wire configuration of the S-type load cell [16]

Since the output voltage of the bridge is usually exceedingly small, the signal needs to be amplified. The theoretical relationship between the measured voltage and the applied longitudinal force can be determined by the following formula:

$$U_{measured} = \frac{C \cdot U_{Exc}}{F_{max}} \cdot F_x \cdot G \qquad \text{formula 3-4}$$

Where $U_{measured}$ is the measured voltage, C is the Rated Output of the sensor in mV/V, which means the output voltage (mV) per unit excitation voltage (V) for a maximum load indicating the sensitivity of the sensor, F_{max} is the maximum load. U_{Exc} is the excitation voltage of the sensor bridge. F_x is the load. G is the gain of the amplifier circuit. According to the datasheet of LCT LAS-A load cell, C is $2\ mV \pm 10\%$ and F_{max} is 9.81 N (1 kg).

In practice, considering the error of the system like zero-point drift and sensitivity errors of the sensor, errors in the amplification circuit, etc., , experiments are necessary to calibrate the sensor. Since, in this work, the force measurement is not urgently required, only a position in the system is reserved for the force sensor. The calibration will be performed in the future.

Position Sensor

For the position sensor, a PS100-1B1BR10K linear potentiometer is used. (See Figure 3.13) This potentiometer has a total resistance of 10 kΩ. Its resistance varies linearly with the position of the slider. Similarly to the force sensor, the output voltage of the position sensor also needs to be calibrated experimentally. Since, in this work, the position sensor is also not urgently needed, and the stretching length information can be calculated from the number of steps taken by the stepper motor. So only a position is reserved in the system, and the calibration will be done in the future.

Figure 3.13: Photograph of a PS100-1B1BR10K linear potentiometer [17]

3.4.3.2 Primary filter

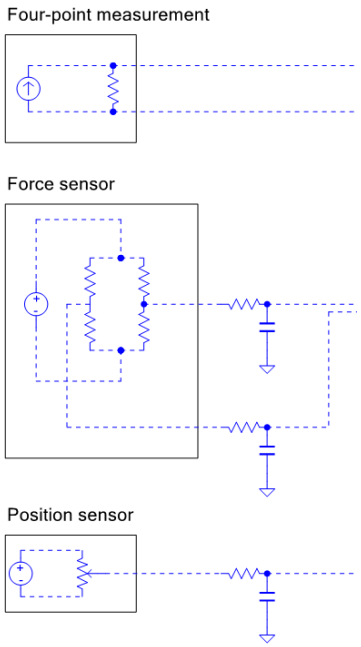
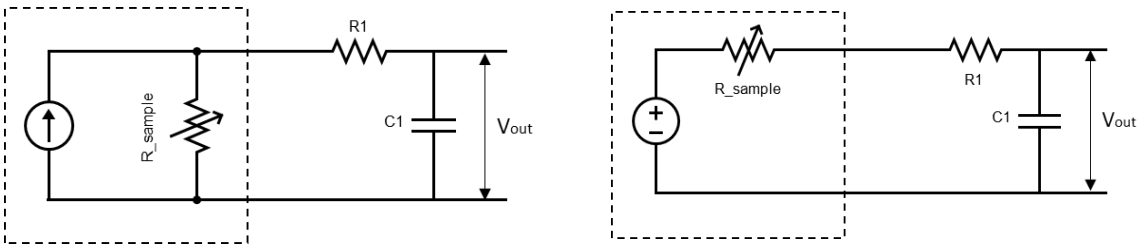


Figure 3.14: Schematic of the primary filter circuit of the measurement system of the stretching test setup

As shown in Figure 3.14, the primary filter circuit consists of multiple RC low-pass filters to remove high-frequency noise from the signal. The cut-off frequency of the low-pass filter should be as low as possible, and the delay of the output of the filter should be within a reasonable range to ensure the dynamic response performance of the system. In the normal test speed range, the drive interval of each motor step is 50ms in minimum, so the output settling time of the filter should be within 50 ms. Here, we use a 20 kΩ resistor and a 0.1 μF capacitor. the cutoff frequency of the RC filter is about 79.5 Hz and the time constant of the filter is about 2 ms.



(a) (b)

Figure 3.15: Explanation of the effect of sample resistance on the filter in a four-point measurement

For four-point measurements, low-pass filters are not used. The reason is that, in the four-point measurement a current source is used as the power supply. In this case, the time constant of the filter increases with the sample resistance, resulting in the increase the delay time of the filter output voltage. Figure 3.15 explains the reason for this phenomenon. The dashed box in Figure 3.15(a) is a simplified circuit for the four-point measurement method. According to Norton's theorem, the circuit is equivalent to a voltage source with a series resistor (see Figure 3.15(b)). So, the actual time constant of the RC filter is $\tau = (R_{sample} + R1)C$, which means that the time constant increase with the sample resistance. When the resistance of the sample reaches 1 MΩ, the time constant of the filter reach 102 ms, which is 50 times the design value. Therefore, to ensure the reliability of the measurement results , it is decided not to filter for the four-point measurement. Adding a low-pass filter after the voltage follower is a solution.

3.4.3.3 Signal input

The sensor signals are fed into the PSoC via GPIOs and a multiplexer. Figure 3.16 shows the schematic of the signal input circuit.

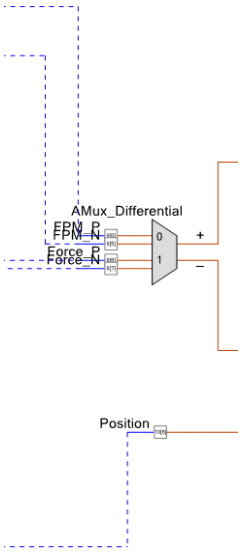


Figure 3.16: Schematic of the signal input circuit of the measurement system of the stretching test setup

The four-point measurement has two input signals *FPM_P* and *FPM_N*. The force sensors also have two signal inputs *Force_P* and *Force_N*. The position sensor only has one input signal *Position*.

The signal of the position sensor is individually directly input to the ADC for digital-to-analog conversion. The signal of the four-point measurement and force sensor are passed through a pre-amplifier circuit before being fed to the ADC. Since there is only one pre-amplifier circuit, the input signal is switched using the multiplexer *AMux_Differential*. When the multiplexer selects channel 0, the signal of the four-point measurement is input, and when the multiplexer selects channel 1, the signal of the force snsor is input.

3.4.3.4 Pre-amplifier

As shown in Figure 3.17, the pre-amplifier circuit consists of two operational amplifiers inside the PSoC, two external feedback resistors R1 and R2 (Upper part), and a gain selection circuit (Lower part).

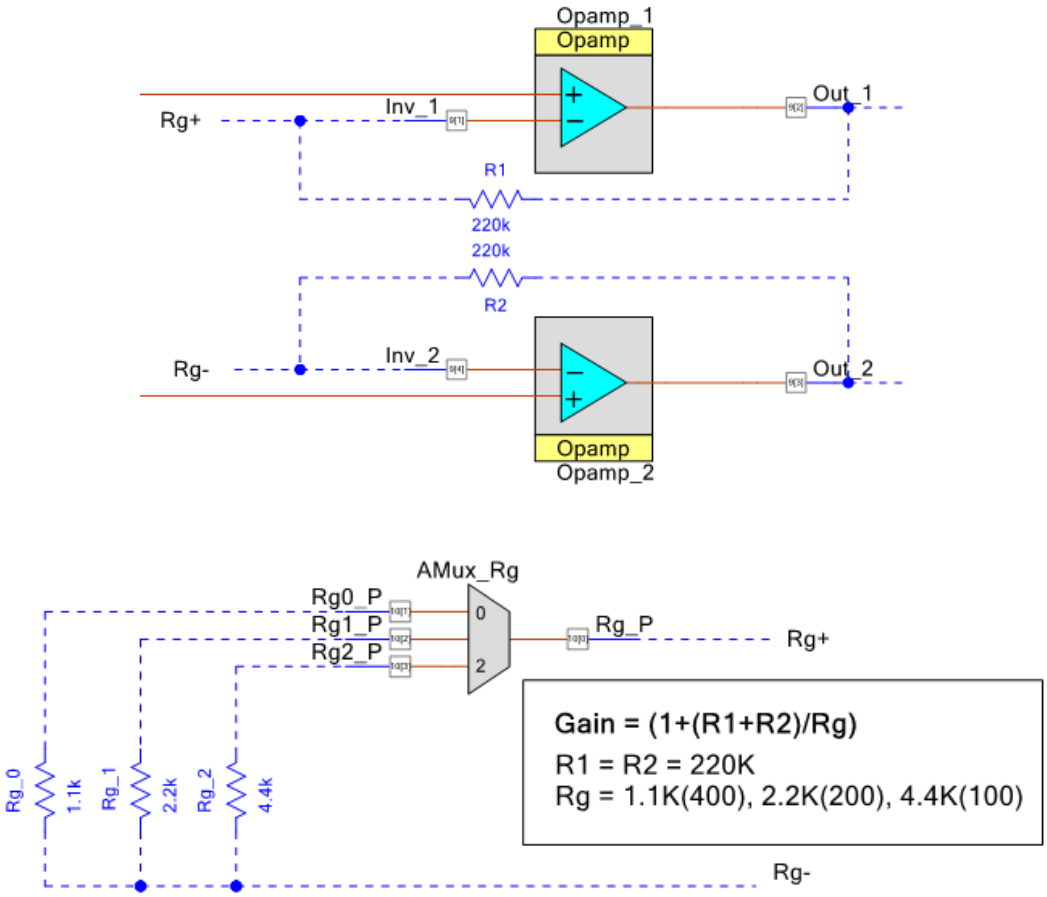


Figure 3.17: Schematic of the pre-amplifier circuit of the measurement system of the stretching test setup

The outputs of the operational amplifiers are connected to the negative terminals of the amplifiers via pins *Out_1* and *Out_2* with feedback resistors R1 and R2. Between the negative

terminals of the two operational amplifiers, there is a gain selection circuit consisting of a PSoC three-channel multiplexer *AMux_Rg*, and three resistors *Rg_0*, *Rg_1* and *Rg_2*.

By setting the operating state of the *AMux_rg*, two operating modes of the pre-amplifier circuit, follower mode and amplification mode, and the selection of the gain in the amplification mode can be realized.

When the output signal of the force sensor is measured, the pre-amplifier operates in amplification mode. Because the output voltage of the bridge is small. To fully use the conversion range of the ADC, the signal is to be amplified.

When measuring the output signal of a four-point measurement, the pre-amplifier operates in voltage following mode. Because the resistance of the sample is usually very large, which can be up to several MΩ, while the order of the input impedance of the ADC is several kΩ. Such significant difference causes the current to enter the measurement branch, resulting in inaccurate measurement results. So, in this case, voltages followers are necessary. The voltage follower is characterized by a very high input impedance and a very low output impedance, which allows the ADC to be isolated from the four-point measurement circuit.

Amplification mode

When any input channel is selected by the multiplexer *AMux_RG*, terminals *Rg+* and *Rg-* of the pre-amplifier circuit are connected through a resistor, and the entire pre-amplifier circuit is then equivalent to a differential amplifier. (See Figure 3.18)

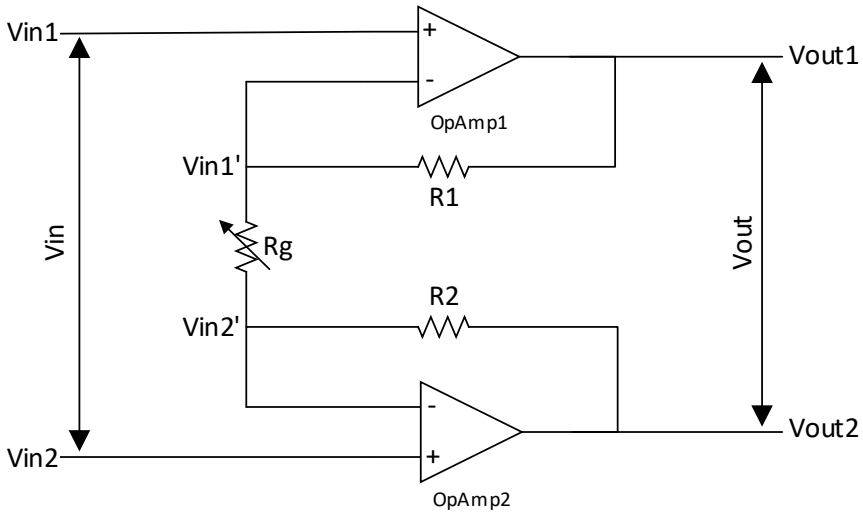


Figure 3.18: Schematic of the pre-amplifier circuit in amplification mode

Based on the virtual short characteristic of the op-amp input, we know $Vin1 = Vin1'$, $Vin2 = Vin2'$. Using Ohm's law, the output voltage of the pre-amplification can be derived:

$$Vout = (R1 + R2 + Rg) \cdot \frac{Vin1' - Vin2'}{Rg} = (1 + \frac{R1 + R2}{Rg}) \cdot (Vin1 - Vin2)$$

formula 3-5

So, the gain of the pre-amplifier circuit can be determined by the following formula

$$Gain = 1 + \frac{R_1 + R_2}{R_g}$$

formula 3-6

By setting the input channel of *AMux_rg*, the gain can be changed dynamically. Table 3.4 shows the available gain.

Table 3.4: Available gain of the in the pre-amplifier circuit in amplification mode

Multiplexer input channel	R1	R2	Rg	Gain (theoretical)
0	220 kΩ	220 kΩ	Rg_0 = 1.1 kΩ	400
1			Rg_1 = 2.2 kΩ	200
2			Rg_2 = 4.4 kΩ	100

In addition, *AMux_rg* can even select multiple input channels at the same time to connect different *Rg* in parallel, thus extending extra 4 different gains.

The gain in Table 3.4 are only theoretical values. In practice, it is also necessary to consider the internal resistance of the analog routes. As shown in Figure 3.19, the internal resistance is marked as red resistors.

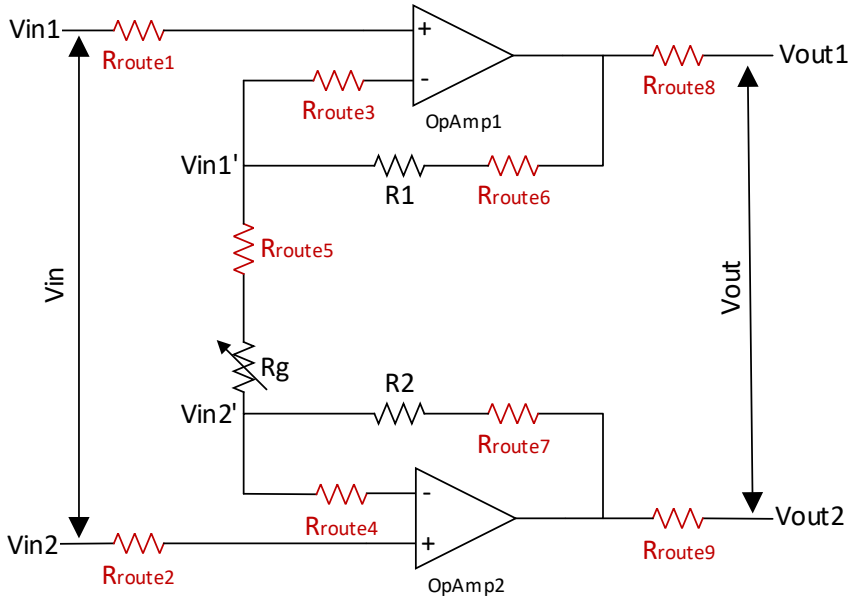


Figure 3.19: Schematic of internal route resistances in the pre-amplifier circuit in amplification mode

Using the ohmmeter in the analog interface of PSoC creator (see Figure 4.14), the resistance of these resistors can be measured.

- Rroute1, Rroute2, Rroute3, Rroute4 is about 500 Ω.
- Rroute5 is about 550 Ω.
- Rroute6, Rroute7 is about zero.
- Rroute8, Rroute9 is about 700 Ω.

Since the input impedance of the operational amplifier is infinite, Rroute1, Rroute2, Rroute3, Rroute4 can be ignored. Rroute8, Rroute9 have no effect on the output voltage, which can be also ignored. Only Rroute5 influences the actual amplification. The evaluation of actual gain is described in the Chapter 5.3.2.

Follower mode

When the multiplexer *AMux_RG* disconnects all input channels, the connection between terminal *Rg+* and *Rg-* is broken. The outputs of the two operational amplifiers are directly fed back to the negative inputs. In this case, the pre-amplifier circuit is equivalent to a differential voltage follower. (See Figure 3.20)

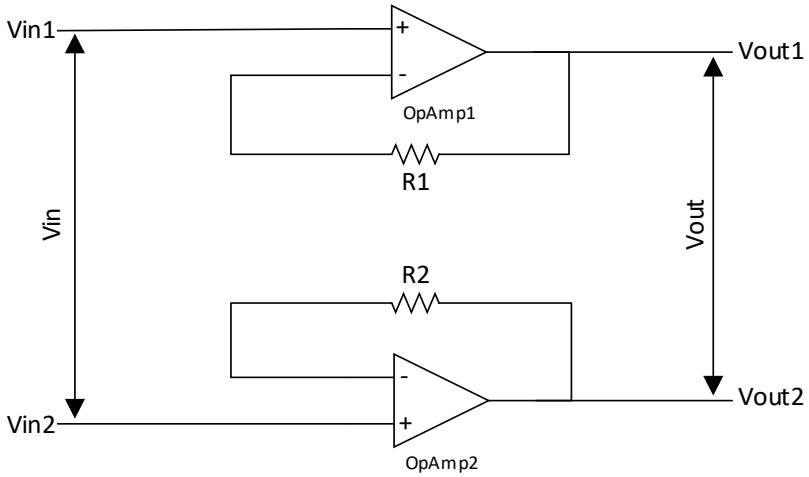


Figure 3.20: Schematic of the pre-amplifier circuit in follower mode

Similar to the amplification mode, the resistance of internal switches and routes also needs to be considered in practical situations. (See Figure 3.21) According to the analysis for the amplification mode in the previous section, Rroute1-Rroute8 has no effect on the output voltage.

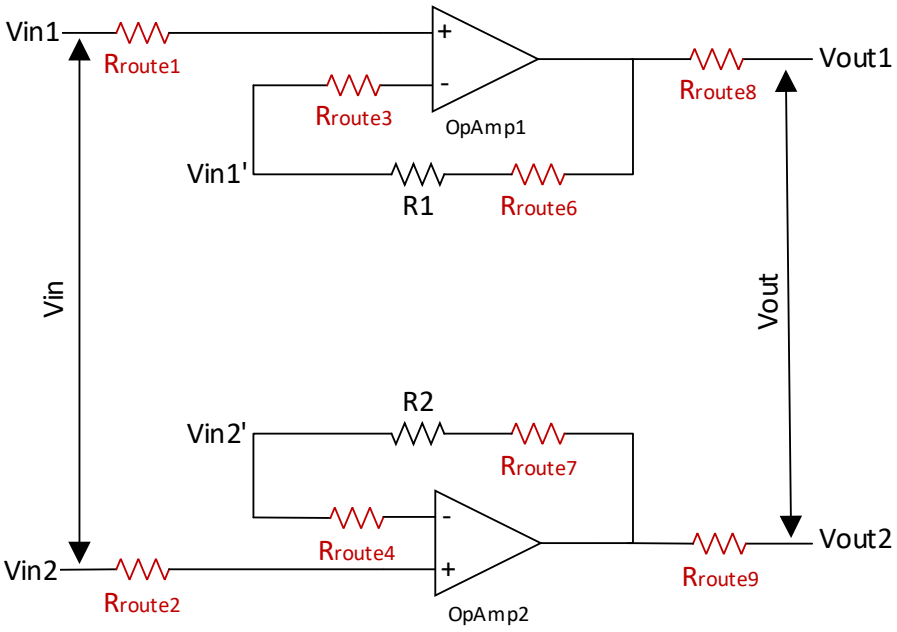


Figure 3.21: Schematic of internal route resistances in the pre-amplifier circuit in follower mode

3.4.3.5 Secondary filter

After the pre-amplifier, the signal is fed into a secondary filter consisting of an RC low-pass filter to further remove noise before the ADC. (See Figure 3.22)

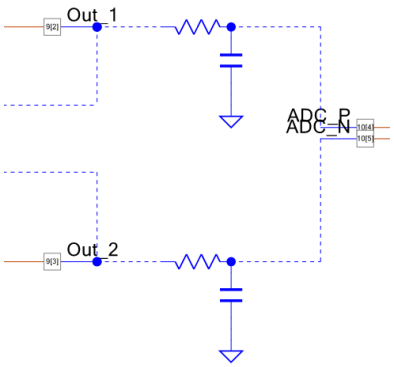


Figure 3.22: Schematic of the secondary filter circuit of the measurement system of the stretching setup

Here, a 20 kΩ resistor and a 0.01 µF capacitor is used with a cutoff frequency of 795 Hz.

3.4.3.6 ADC

The ADC has two input channels, channel 0 and channel 1. Channel 0 is configured as differential input for four-point measurements or force sensor signals. Channel 1 is configured as single-ended input for the input of a single signal from a position sensor.

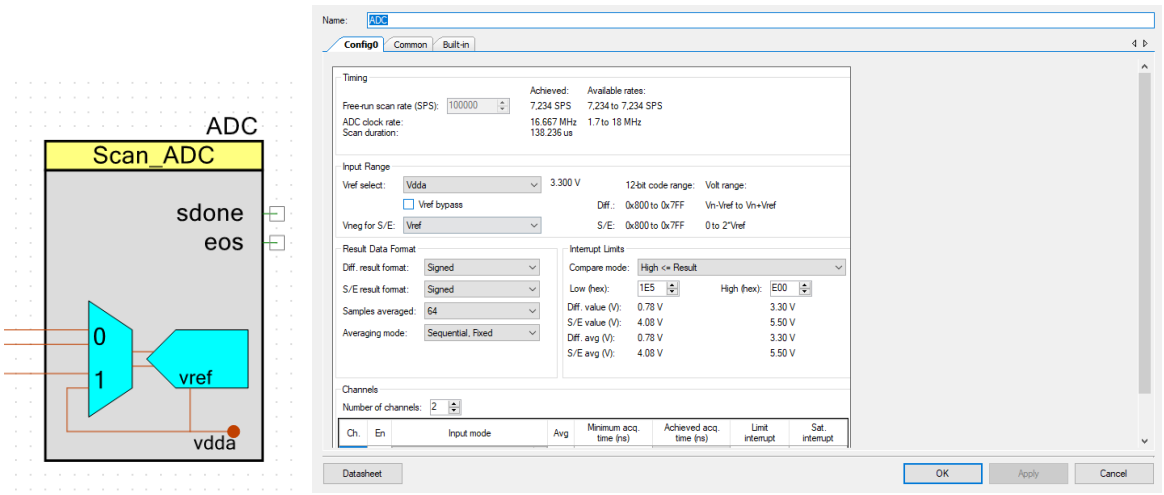


Figure 3.23: PSoC ADC configuration of the measurement circuit of the stretching test setup

Figure 3.23 shows the detailed configuration of the ADC. 3.3 V Vdda is selected as the reference voltage of the ADC and the format of conversion results configured as signed. In this case, the measurement range of the differential signal is -3.3 to 3.3 V. For single-ended signals, the measurement range is 0 - 6.6 V. However, in practice, ,since the maximum input voltage of

the ADC cannot exceed Vdda, the actual single-ended measurement range is 0 - 3.3 V. To reduce noise, sampling averaging is used. The result of a single measurement is averaged over 64 consecutive samples.

With the above configuration, the ADC can sample at a rate of 7232 samples per second (SPS) and take 138 µs for a complete measurement of both channels. Since the four-point measurement shares channel 0 with the force sensor, two complete measurements are required to get all sensors data. So, the actual single measurement time is about 276 µs, which is much less than the 10 ms of the old measurement system.

3.4.3.7 I2C

PSoC collects the number of motor steps since test start from Arduino via I2C to calculate the elongation of the sample. For the I2C component, the default configuration is used, as shown in Figure 3.24. PSoC is master, Arduino is slave and communication rate is 100 kbps. Since the pull-up resistors are preset on both PSoC and Arduino development boards, there is no need to add pull-up resistors.

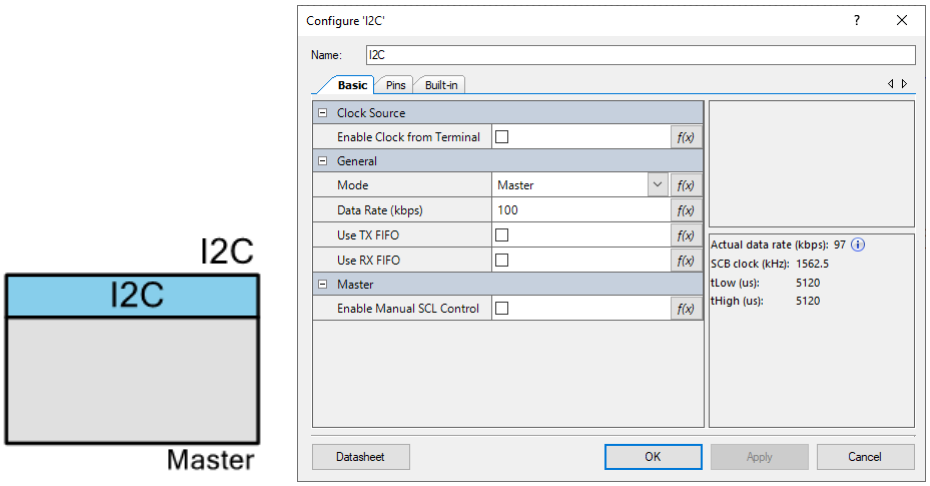


Figure 3.24: PSoC I2C component configuration of the measurement circuit of the stretching test setup

3.4.3.8 UART

The PSoC receives commands from the PC and sends the collected data to the PC via UART. For the UART component, the default configuration is used as shown in Figure 3.25. The baud rate is 115200bps, the data length is 8 bits, the bit order is LSB first, and the stop bits is 1.

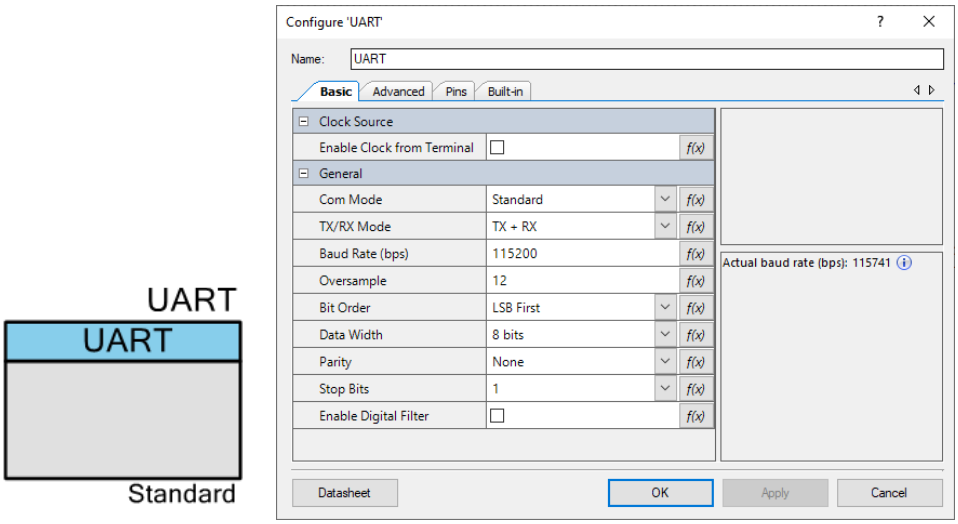


Figure 3.25: PSoC UART component configuration of the measurement circuit of the stretching test setup

3.4.3.9 PSoC shield

To improve the integration of the system, a shield board is designed for the PSoC to hold all the external components out of PSoC. Annex A shows the schematic of PSoC shield for stretching test setup. Figure 3.26 shows the 3D model of the shield board in top view.

In the current source area, the current source required for the four-point measurement is placed. R6, R7 and R8, and R9, R10, R11 are resistors in the current source.

J8 (Rg1), J9 (Rg2), J10 (Rg3) and J11 (Rf1), J12 (Rf2) are the resistors in the pre-amplifier circuit. Also considering possible requirement changes in future, slots are used instead of SMD resistors. By inserting the correct standard resistors in the slots, the pre-amplifier is ready to work.

In the upper part of the extension board there are four measurement connection headers, which are used to provide an interface for measuring power and receiving measurement signals. J5 is used for four-point measurements, J6 for force sensors and J7 for position sensors. Above the headers there are texts to indicate the function of each socket, where “I” represents the output of the current source, “G” represents ground, “+” and “-” are the positive and negative pole of the voltage signal, S is the measured single-ended voltage signal.

Below the measurement headers there are several sets of resistors and capacitors that form the low-pass filters in the measurement circuit.

The lower right slot is the I2C connection headers used to connect to the Arduino's I2C port for step acquisition.

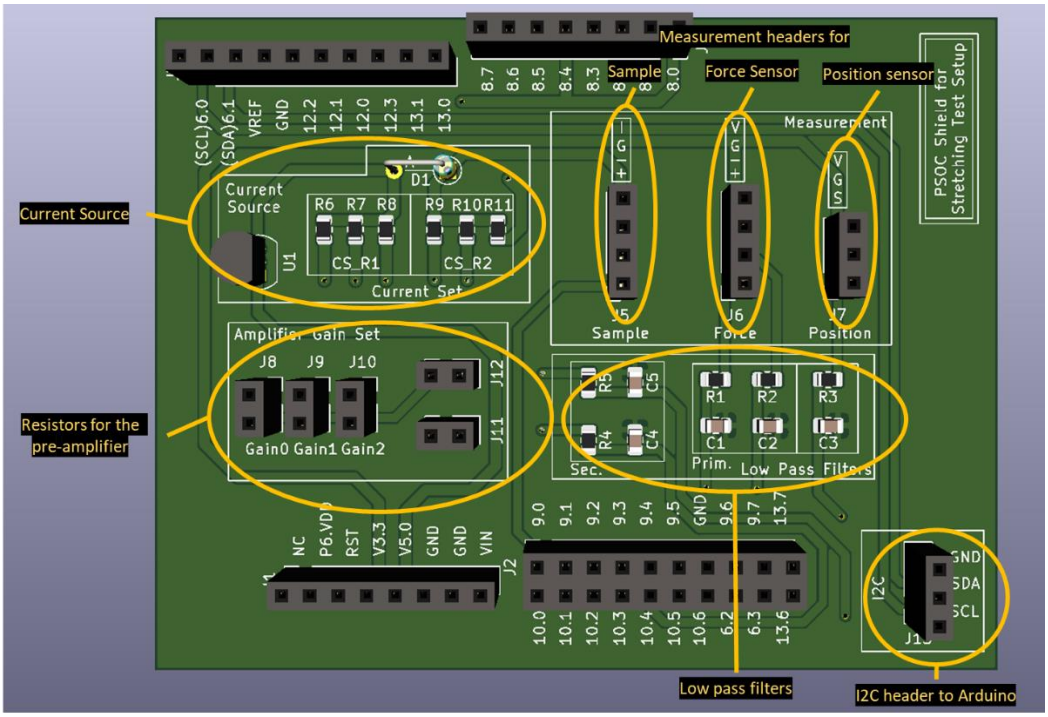


Figure 3.26: 3D model of the PSoC shield board of the stretching test setup in top view

3.5 Software

According to the platform, the software system of the setup can be divided into three parts:

1. PC program: It is an application witch GUI, which enables user to configure the test parameters, send the test commands, monitor and save the measurement data.

2. Arduino program: It is mainly responsible for receiving and parsing the test commands from the PC and driving the motor.

3. PSoC program: It is mainly responsible for the acquisition of measurement data and sending measurement data to the PC.

3.5.1 Communication protocol

As a whole system, it is necessary for the programs of different platform to share and exchange information. Figure 3.27 shows the communications existing between three platforms.

1. PC-Arduino communication: The purpose of communication between PC and Arduino is to control and monitor the test process.
2. PC-PSoC communication: The communication between PC and PSoC is mainly used for the control and acquisition of measurement.
3. PSoC-Arduino communication: The communication between PSoC and Arduino is for PSoC to collect the step data from Arduino.

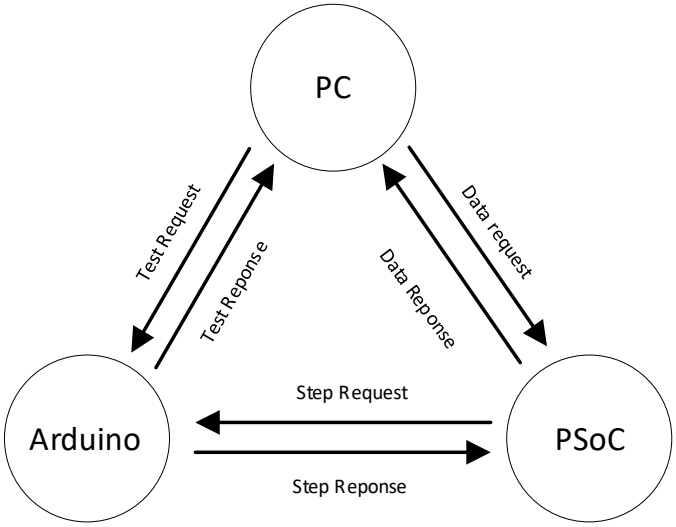


Figure 3.27: Schematic of the communication system of the stretching test setup

The communication between the three platforms has a uniform protocol. Table 3.5 shows a general telegram layout. A telegram is bytes consisting of type, parameters, delimiter, and terminator. The telegram type is the assigned code for the telegram indicating its purpose and direction. Each parameter is directly translated from the literal value of the associated value, e.g., the value 42 is directly represented by "42" in the telegram. Different parameters are separated by the delimiter ",". The telegram is terminated with the terminator "\r\n".

Table 3.5: General layout of a communication telegram

Type	,	Parameter 1	,	Parameter 2	,	Parameter 3	,	…	,	Parameter N	\r\n

The telegram has an indeterminate number of bytes, and its length is related to the value of each parameter. When reading telegrams, splitting of continuous telegrams is often implemented with the help of terminator.

To better understand the organization of the telegram, an example is given. There are the following variables whose values need to be sent in order:

- int a = 100
- float b = 3.14
- char c = ‘A’
- string e = “Hello”

Assuming the telegram type is ‘00’, the layout of the corresponding message is bytes *"00,100,3.14,A,Hello\r\n"*.

The communication is achieved by request and response telegrams. A request is a telegram sent by the master to the slave. Once the slave receives the request, it send a telegram called response the master. Figure 3.28 shows the two basic communication models in the system. In Figure 3.28(a) is the one-to-one model. The host sends a request telegram once. Then the slave sends a respond telegram back to report that the request from the master is successfully received. This kind of model is mainly used for the task of process control and setting. In Figure 3.28(b) is the one-to-many model. The master sends a request telegram once. Then the slave continuously sends responses to the host. This kind of model is mainly used for the task of process monitoring or data acquisition.

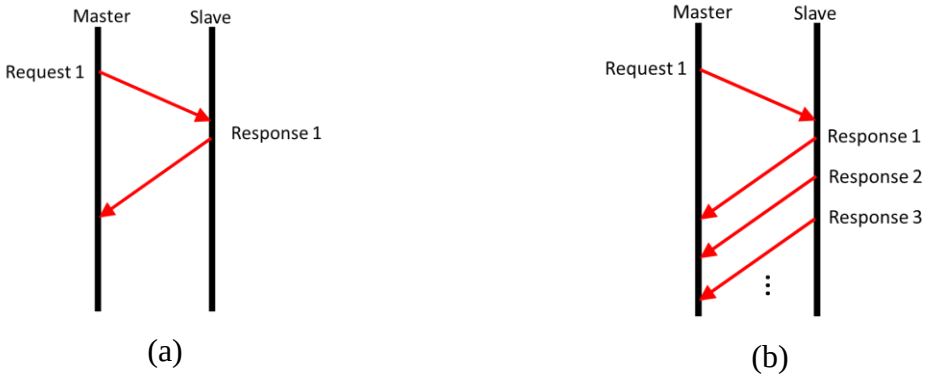


Figure 3.28: Sequence diagram of the two communication models: one-to-one (a) and one-to-many (b)

Next, the communication protocols between each pair of platforms are described in detail.

3.5.1.1 PC – Arduino communication

The task of the communication between PC and Arduino is mainly the control of the test and the monitoring of test process. The communication task can be divided into four parts: launch-test, test process control, test parameter setting and error telegram.

Launch-test

The purpose of the communication task launch-test is to command the Arduino to start a new test.

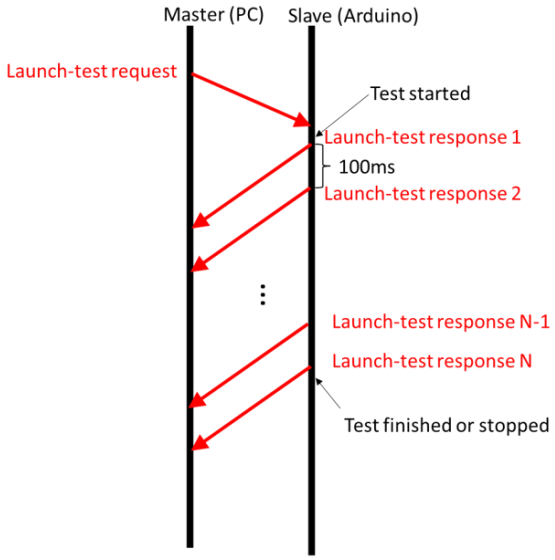


Figure 3.29: Sequence diagram of PC-Arduino launch-test communication task of the stretching test setup

As shown in Figure 3.29, PC sends a *launch-test request* to the Arduino, once the Arduino successfully receives the request, it starts a test and reply launch-test response to the PC to monitor the test progress every 100 microseconds. The reply is not stopped until the test is finished or stopped by the user.

Launch-test request

Table 3.6 shows the layout of the *Launch-test request* and Table 3.7 explains each parameter in the telegram in detail.

Table 3.6: Layout of PC-Arduino launch-test request telegram of the stretching test setup

Type(00)	\r\n

Table 3.7: Parameter description of PC-Arduino launch-test request telegram of the stretching test setup

Place	Fieldname	Format	Unit	Range	Description
0	Type	string	-	-	Telegram type: “00” = Launch-test request

Launch-test response

Table 3.8 shows the layout of the Launch-test request and Table 3.9 explains each parameter in the telegram in detail.

Table 3.8: Layout of PC-Arduino launch-test response telegram of the stretching test setup

Type(01)	,	Progress status	,	Steps	,	Total steps	,
Step delay	,	Stretch length	,	Stretch speed	,	Cycles	\r\n

Table 3.9: Parameter description of PC-Arduino launch-test response telegram of the stretching test setup

Place	Fieldname	Format	Unit	Range	Description
0	Type	string	-	-	Telegram type: “01” = Launch-test response
1	Progress status	int	-	0,1,2	0: Test stopped or finished 1: Test started 2: Test paused
2	Steps	int	-	> 0	The number of steps taken by the stepper motor since the test starts.

3	Total steps	int	-	> 0	Total number of steps of stepper motor required for the sample to reach maximum strain. Calculated with stretching length and motor step resolution
4	Step delay	int	-	> 0	The delay between two steps of stepper motor. Calculated with stretching speed and motor step resolution
5	Stretch length	int	μm	> 0	The received maximum length of the sample to be stretched
6	Stretch speed	int	μm/s	> 0	The received speed at which the sample is stretched after setting
7	cycles	int	-	> 0	The received number of repetitions of Back and forth stretching

Test progress control

The purpose of the communication task test progress control is to control the test progress to pause, start and stop after the test is launched. As shown in Figure 3.30, the PC sends a *Test progress control request* to the Arduino, once the Arduino successfully receives the request, it sends a respond to the PC to report that it receives the command and then follows the command to control the progress.

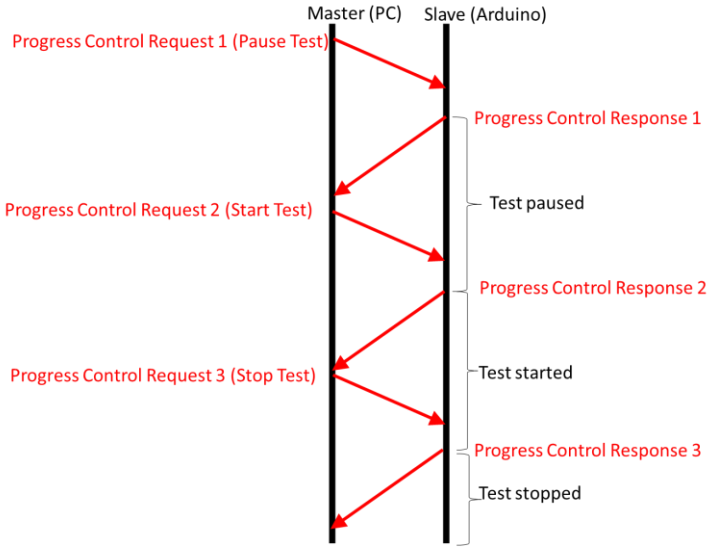


Figure 3.30: Sequence diagram of PC-Arduino test progress control communication task of the stretching test setup

Test progress control request

Table 3.10 shows the layout of the *Test progress control request* and Table 3.11 explains each parameter in the telegram in detail.

Table 3.10: Layout of PC-Arduino test progress control request telegram of the stretching test setup

Type(02)	,	Control flag	\r\n

Table 3.11: Parameter description of PC-Arduino test progress control request telegram of the stretching test setup

Place	Fieldname	Format	Unit	Range	Description
0	Type	string	-	-	Telegram type: “02” = Progress control request
1	Control flag	int	-	0,1,2	0: Stop test 1: Start test 2: Pause test

Test progress control response

Table 3.12 shows the layout of the *Test progress control response* and Table 3.13 explains each parameter in the telegram in detail.

Table 3.12: Layout of test PC-Arduino progress control response telegram of the stretching test setup

Type(03)	,	Control flag	\r\n

Table 3.13: Parameter description of PC-Arduino test progress control response telegram of the stretching test setup

Place	Fieldname	Format	Unit	Range	Description
0	Type	string	-	-	Telegram type: “03” = Progress control response
1	Control flag	int	-	0,1,2	0: Stop test 1: Start test 2: Pause test

Test parameter setting

The purpose of the communication test parameter setting is to configure the test parameter. As shown in Figure 3.31, the PC sends a test *parameter setting request* to the Arduino, once the Arduino successfully receives the request, it replies to the PC to report that it receives the command and then change the test parameter.

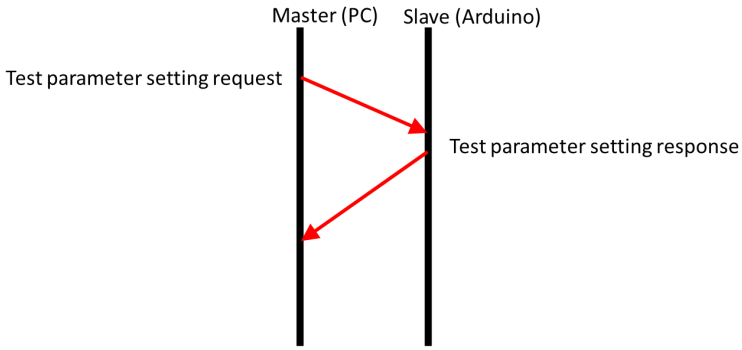


Figure 3.31: Sequence diagram of PC-Arduino test parameter setting communication task of the stretching test setup

Test parameter setting request

Table 3.14 shows the layout of the *test parameter setting request* and Table 3.15 explains each parameter in the telegram in detail.

Table 3.14: Layout of PC-Arduino test parameter setting request telegram of the stretching test setup

Type(04)	,	Stretch length	,	Stretch speed	,	Cycles	\r\n

Table 3.15: Parameter description of PC-Arduino parameter setting request telegram of the stretching test setup

Place	Fieldname	Format	Unit	Range	Description
0	Type	string	-	-	Telegram type: “04” = Parameter setting request
1	Stretch length	int	µm	> 0	The maximum length of the sample to be stretched.
2	Stretch speed	int	µm/s	> 0	The speed at which the sample is stretched
3	cycles	int	-	> 0	The number of repetitions of Back and forth stretching

Test parameter setting response

Table 3.16 shows the layout of the *test parameter setting response* and Table 3.17 explains each parameter in the telegram in detail.

Table 3.16: Layout of PC-Arduino test parameter setting response telegram of the stretching test setup

Type(05)	,	Stretch length	,	Stretch speed	,	Cycles	\r\n

Table 3.17: Parameter description of PC-Arduino parameter setting response telegram of the stretching test setup

Place	Fieldname	Format	Unit	Range	Description
0	Type	string	-	-	Telegram type: "05" = Parameter setting response
1	Stretch length	int	µm	> 0	The received maximum length of the sample to be stretched
2	Stretch speed	int	µm/s	> 0	The received speed at which the sample is stretched after setting
3	cycles	int	-	> 0	The received number of repetitions of Back and forth stretching

Error-telegram

As shown in Figure 3.32, once the user sends an undefined request or the layout and parameters of the request doesn't match the rule, the Arduino just sends an Error-telegram to the PC and the request doesn't have any effect.

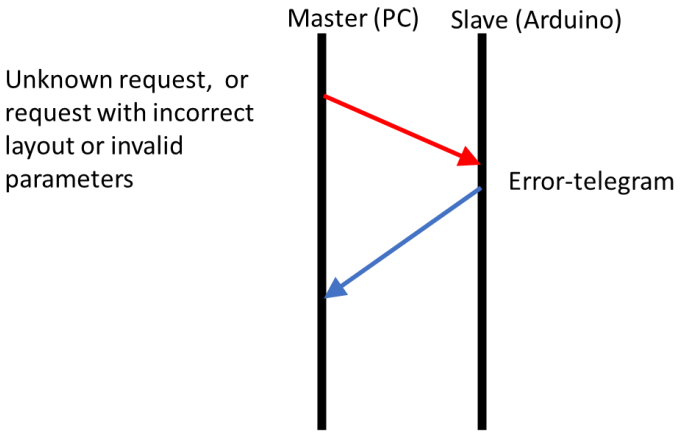


Figure 3.32: Sequence diagram of PC-Arduino error telegram of the stretching test setup

Table 3.18 shows the layout of the *Error-telegram* and Table 3.19 explains each parameter in the telegram in detail.

Table 3.18: Layout of PC-Arduino error telegram of the stretching test setup

Type(99)	\r\n

Table 3.19: Parameter description of PC-Arduino error telegram of the stretching test setup

Place	Fieldname	Format	Unit	Range	Description
0	Type	string	-	-	Telegram type: “99” = Error-telegram

3.5.1.2 PC – PSoC communication

The task of the communication between PC and PSoC is mainly the control of the transmission of the test data. The communication task can be divided into three parts: data subscription, measurement setting and error telegram.

Data subscription

The purpose of the communication task data subscription is to command the PSoC to send measured data to the PC. As shown in Figure 3.33, PC sends a *data subscription request* to the Arduino, once the Arduino successfully receives the request, it starts sending measured data to the PC continuously. The data transfer is not stopped until a *data subscription request* for unsubscription is sent from the PC by the user.

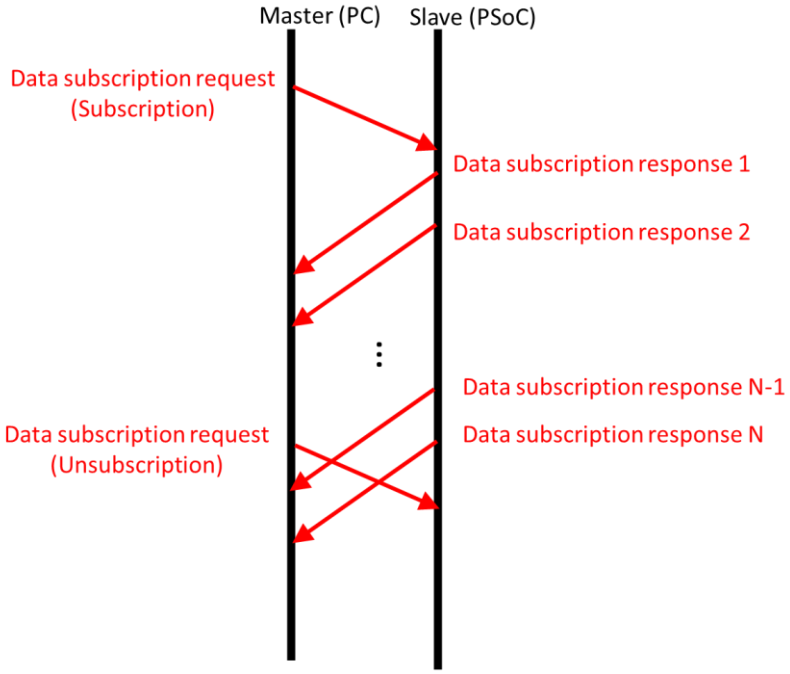


Figure 3.33: Sequence diagram of PC-PSoC data subscription communication task of the stretching test setup

Data subscription request

Table 3.20 shows the layout of the *data subscription request* and Table 3.21 explains each parameter in the telegram in detail.

Table 3.20: Layout of PC-PSoC data subscription request telegram of the stretching test setup

Type(00)	,	Subscription status	\r\n

Table 3.21: Parameter description of PC-PSoC data subscription request telegram of the stretching test setup

Place	Fieldname	Format	Unit	Range	Description
0	Type	string	-	-	Telegram type: “00” = Data subscription request
1	Subscription status	int	-	0,1	0: Unsubscription 1: Subscription

Data subscription response

Table 3.22 shows the layout of the *data subscription response* and Table 3.23 explains each parameter in the telegram in detail.

Table 3.22: Layout of PC-PSoC data subscription response telegram of the stretching test setup

Type(01)	,	Timestamp	,	Steps	,	Elongation	,	Resistance	,
v_4pm	,	v_force	,	v_position	,	Gain	,	Current	\r\n

Table 3.23: Parameter description of PC-PSoC data subscription response telegram of the stretching test setup

Place	Fieldname	Format	Unit	Range	Description
0	Type	string	-	-	Telegram type: “01” = Data subscription response
1	Timestamp	int	ms	-	The number of microseconds since PSoC is powered on.
2	steps	int	-	-	The number of steps taken by the stepper motor from the starting position driven by the Arduino.

3	Elongation	int	mm	-	The length change of the sample under stretching. It is calculated by multiplying the number of steps by the step resolution of the actuator. .
4	Resistance	int	Ω	-	The resistance of the sample. It is calculated by multiplying the voltage of four-point measurement by the constant current.
5	v_4pm	float	V	-	The measured output voltage of four-point measurement.
6	v_force	float	V	-	The measured amplified output voltage of the force sensor.
7	v_position	float	V	-	The measured output voltage of the position sensor
8	Gain	int	-	0,1,2	The selected Rg channel of the pre-amplifier. 0: Gain = 400 1 : Gain =200 2: Gain =100
9	Current	int	-	-1, 0 , 1	The Selected output current of the current source -1: Auto set 0: Current = 1 μA 1: Current = 10 μA

Measurement setting

The purpose of the communication task measurement setting is to configure the measurement parameter of the PSoC. As shown in Figure 3.34, the PC sends a t *measurement setting request* to the Arduino, once the Arduino successfully receives the request, it replies to the PC to report that it receives the command and then change the measurement parameter.

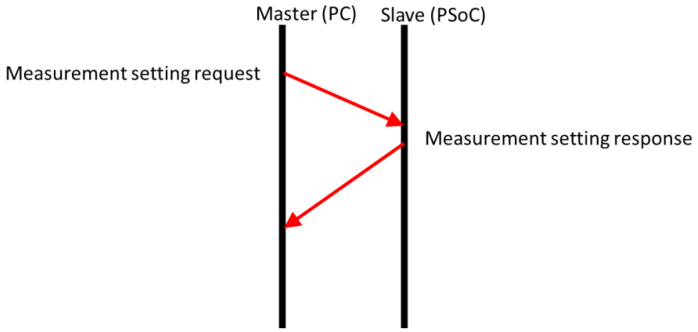


Figure 3.34: Sequence diagram of PC-PSoC measurement setting communication task of the stretching test setup

Measurement setting request

Table 3.24 shows the layout of the *measurement setting request* and Table 3.25 explains each parameter in the telegram in detail.

Table 3.24: Layout of PC-PSoC measurement setting request telegram of the stretching test setup

Type	,	Sample rate	,	Downsample	,	Gain	,	Current	\r\n

Table 3.25: Parameter description of PC-PSoC measurement setting request telegram of the stretching test setup

Place	Fieldname	Format	Unit	Range	Description
0	Type	string	-	-	Telegram type: “02” = Measurement setting request
1	Sample rate	int	Hz	> 0	indicates how often the signal is measured

2	Downsample	int	-	> 0	indicates how often the data is sent to PC. If downsample is 5 then one data is sent for every 5 measurements.
3	Gain	int	-	0,1,2	Selecting Rg of pre-amplifer to set the gain for force measurement 0: Gain = 400 1 : Gain =200 2: Gain =100
4	Current	int	-	-1, 0 , 1	Set the output current of the current source -1: Auto set 0: Current = 1 μA 1: Current = 10 μA

Measurement setting respond

Table 3.26 shows the layout of the *measurement setting respond* and Table 3.27 explains each parameter in the telegram in detail.

Table 3.26: Layout of PC-PSoC measurement setting response telegram of the stretching test setup

Type(03)	,	Sample rate		,	Downsample	,	Gain	,	Current	\r\n

Table 3.27: Parameter description of PC-PSoC measurement setting response telegram of the stretching test setup

Place	Fieldname	Format	Unit	Range	Description
0	Type	string	-	-	Telegram type: “03” = Measurement setting respond

1	Sample rate	int	Hz	> 0	indicates how often the signal is measured
2	Downsample	int	Hz	> 0	indicates how often the data is sent to PC. If downsample is 5 then one data is sent for every 5 measurements.
3	Gain	int	-	0,1,2	Selecting the channel of Rg of pre-amplifer to set the gain for force measurement Channel 0: 400 Channel 1 : 200 Channel 2: 100
4	Current	int	-	-1, 0 , 1	Set the output current of the current source -1: Auto 0: Current = 1 µA 1: Current = 10 µA

Error-telegram

As shown in Figure 3.35,once the user sends an undefined request or the layout and parameters of the request doesn't match the rule, the PSoC just sends an *Error-telegram* to the PC and the request doesn't have any effect.

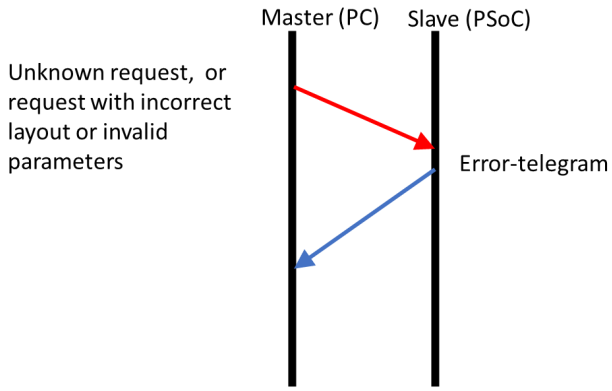


Figure 3.35: Sequence diagram of PC-PSoC error telegram of the stretching test setup

Table 3.28 shows the layout of the *Error-telegram* and Table 3.29 explains each parameter in the telegram in detail.

Table 3.28: Layout of PC-PSoC error telegram of the stretching test setup

Type(99)	\r\n

Table 3.29: Parameter description of PC-PSoC error telegram of the stretching test setup

Place	Fieldname	Format	Unit	Range	Description
0	Type	string	-	-	Telegram type: "99" = Error-telegram

3.5.1.3 PSoC – Arduino communication

The purpose of the I2C communication is to transfer the step information from the Arduino to the PSoC. The PSoC is the master, and the Arduino is the slave. As shown in Figure 3.36, the PSoC sends a I2C request to the Arduino. Once the Arduino receives the request, it sends the response to the PSoC.

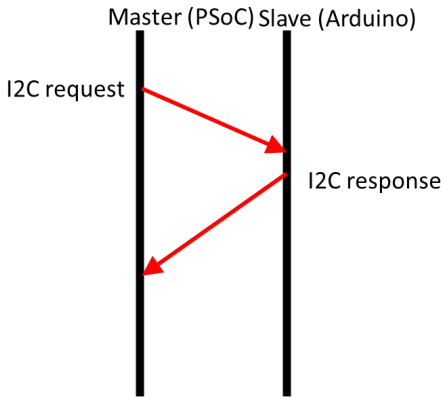


Figure 3.36: Sequence diagram of PSoC-Arduino I2C communication of the stretching test setup

I2C Request

The I2C request contains the address of the slave. The request does not need to be explicitly defined by the user, so it is not specified here.

I2C Response

Table 3.30 shows the format of the I2C response telegram sent from the Arduino to the PC. This telegram only contains the step data of the stepper. Since I2C requests and responses are single-to-single , terminator is not used here to distinguish between continuous telegrams.

Table 3.30: Layout of PSoC-Arduino I2C response telegram of the stretching test setup

Steps

Table 3.31 explains each parameter in the telegram in detail.

Table 3.31: Parameter description of PSoC-Arduino I2C telegram of the stretching test setup

Place	Fieldname	Format	Unit	Range	Description
0	Steps	int	-	> 0	The number of steps taken by the stepper motor from the starting position.

3.5.2 PC program

The PC program provides a GUI for the user to set test parameters, send test commands and view measurement data. Figure 3.37 shows the GUI of the PC program.

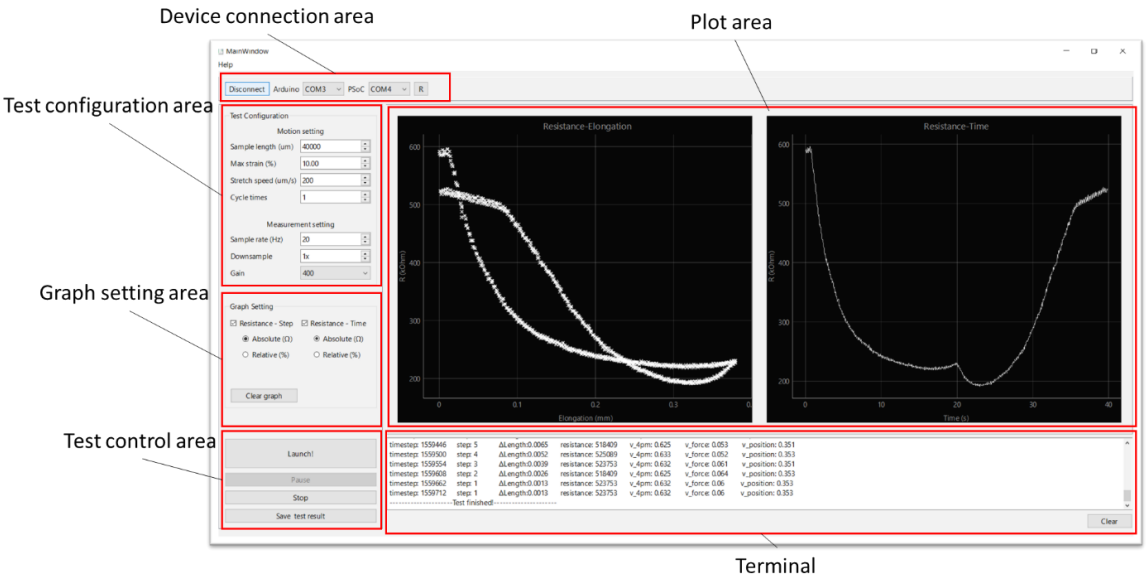


Figure 3.37: GUI of the PC program of the stretching test setup

1. Device connection area: in this area, the user can select the port of the Arduino and PSoC and establish the connection with them, which is the prerequisite for a test.
2. Test configuration area: In this area, the user can adjust the test parameters including the configuration of the motion and the measurement.
3. Plot area: in this area, the measurement data is visualized as real-time graphs. There are two types of graphs, resistance-step graph and resistance-time graph. The resistance-step graph shows the relationship between steps of the stepper motor, which indicates the progress of the bending, and resistance. The resistance-time graph describes the sample resistance with the time. So, there are two graphs in the plot area in total.
4. Graph setting area: In this area, the user can configure the graphs in plot area, like showing/hiding of the specific graph and changing the resistance format. If radio box absolute is checked, the graph y-axis shows the resistance value. If relative is selected, the graph y-axis shows percentage of the resistance relative to initial resistance
5. Terminal: In this area, application notifications and the measurement data is shown in text form.

3.5.3 Arduino program

The Arduino program receives test commands from the PC and sends control signals to the motor driver. The Arduino program is based on the Vincent Gottwald's original program and

removes unnecessary functions such as the LED display, the data measurement etc. Figure 3.38 shows the flow chart of the program.

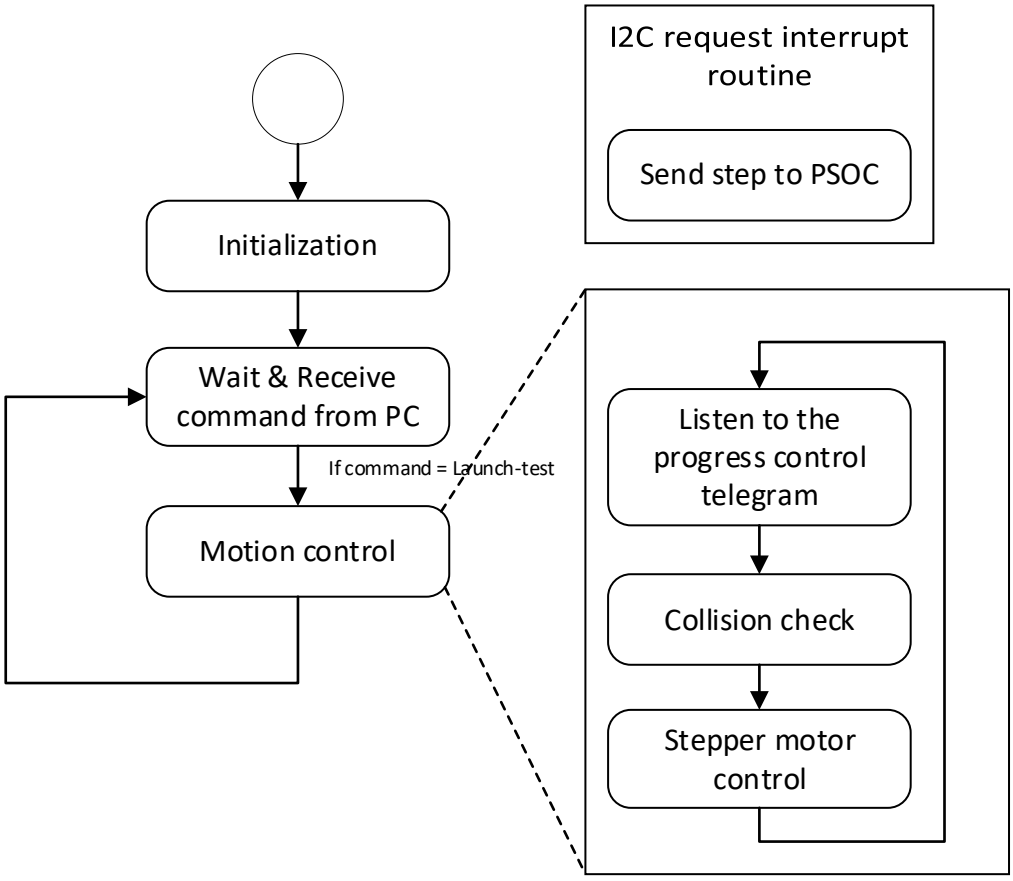


Figure 3.38: Flow chart of the Arduino program of the stretching test setup

3.5.3.1 Initialization

After the Arduino is powered on, the program first executes function *setup* to initialize the Arduino, including the initialization of the UART and I2C, and the configuration of the GPIO working mode.

3.5.3.2 Wait & Receive command from PC

Once initialization is complete, the program executes function *loop* to enters an infinite loop. At the start of the loop, the program blocks to listen to the serial port status and waits for the command from PC. When a request telegram is detected in the UART input buffer, the telegram is received and parsed according to the communication protocol. The detection and reading of the serial port are realized by Arduino library *Serial*.

3.5.3.3 Motor control

After receiving the launch-test request, the program controls the motor movement according to the test parameters. The stepper motor is controlled by setting the level of the pins connected to the driver. Code block 3.1 shows the implementation of this process. The code block mainly consists of two *for* loops. The first for loop controls the slider to move in the direction of stretching, and the second for loop controls the slider to return to the initial position.

Code block 3.1: Implementation of function launch_test for motor control of the stretching test setup

```
1.  void launch_test() {
2.   steps = 0;
3.    for (int j = 0; j < testParameter.cycles; ++j)
4.      {
5.        for (int i = 0; i < caculateStepsToGo(); i++) // GO
6.        {
7.
8.          do {
9.            if(Serial.available()) {
10.               MsTimer2::stop();
11.               receive_command();
12.               MsTimer2::start();
13.           }
14.           if(progressStatus == STOPPED) return;
15.         }
16.         while(progressStatus != STARTED);
17.
18.
19.         if (digitalRead(PIN_COLLISION_PROTECTION) == LOW)
20.           return;
21.         else
22.         {
23.           digitalWrite(PIN_DIR, LOW);
24.           digitalWrite(PIN_ENA, HIGH);
25.           digitalWrite(PIN_PUL, HIGH);
26.           delayMicroseconds(20);
27.           digitalWrite(PIN_PUL, LOW);
28.           delayMicroseconds(20);
29.           delay(calculateStepDelay());
30.           ++steps;
31.         }
32.       }
33.
34.       //delay(100);
35.
36.       for (int i = 0; i < caculateStepsToGo(); i++) // BACK
37.       {
38.         do {
39.           if(Serial.available()) {
40.               MsTimer2::stop();
41.               receive_command();
42.               MsTimer2::start();
43.           }
44.           if(progressStatus == STOPPED) return;
45.         }
46.         while(progressStatus != STARTED);
47.
48.         if (digitalRead(PIN_COLLISION_PROTECTION) == LOW)
49.           return;
50.         else
51.         {
```

```
52.           digitalWrite(PIN_DIR, HIGH);
53.           digitalWrite(PIN_ENA, HIGH);
54.           digitalWrite(PIN_PUL, HIGH);
55.           delayMicroseconds(20);
56.           digitalWrite(PIN_PUL, LOW);
57.           delayMicroseconds(20);
58.           delay(calculateStepDelay());
59.           --steps;
60.         }
61.       }
62.     }
63.
64.
```

Listen to the progress control telegram

At the beginning each motor step, the program checks the UART input buffer to listen to the progress control request telegram from PC. If a progress control request telegram is received, the program pauses, starts or stops the motor motion according to the command. If the parsed command is a stop command, the program exits the function and stop the test. If the command is a pause-command, the program wait using a while loop until a start-command is received.

Collision check

A collision check is performed, which is implemented by reading the level of the GPIO to which the collision detection switch is connected. If a collision occurs, the test is terminated. If there is no collision the program starts driving the motor.

Stepper motor control

The motor steps are controlled by the number of loops, which is calculated by the *caculateStepsToGo* function. Pin DIR controls motor rotation direction, pin ENA controls motor sleeping and waking up and pin PUL controls the motor step. Whenever the level of the PUL pin goes from high to low, the motor takes a step. Finally, the speed of movement is controlled by the delay function, which generates an interval between each loop. The control code for the return process is almost the same as that for the forward process, The difference is that due to the return error of the actuator, a correction for the number of steps for the return trip is required. The number of steps for the return trip should be 0.69% more than the lengthening trip, which is determined experimentally in Vincent Gottwald’s thesis.

3.5.3.4 Send step to PSoC

The PSoC collects the necessary data from the Arduino via I2C. In the I2C communication, PSoC is the host, and the Arduino is the slave. On the Arduino side, the program needs to send a response back to the PSoC after receiving a request from it. It is implemented by the I2C interrupt routine using class *Wire* of the Arduino libray.

3.5.4 PSoC program

The PSoC program is responsible for measuring the sensor signal and sending the measurement data back to the PC via the serial port. Figure 3.29 shows the flow chart of the program.

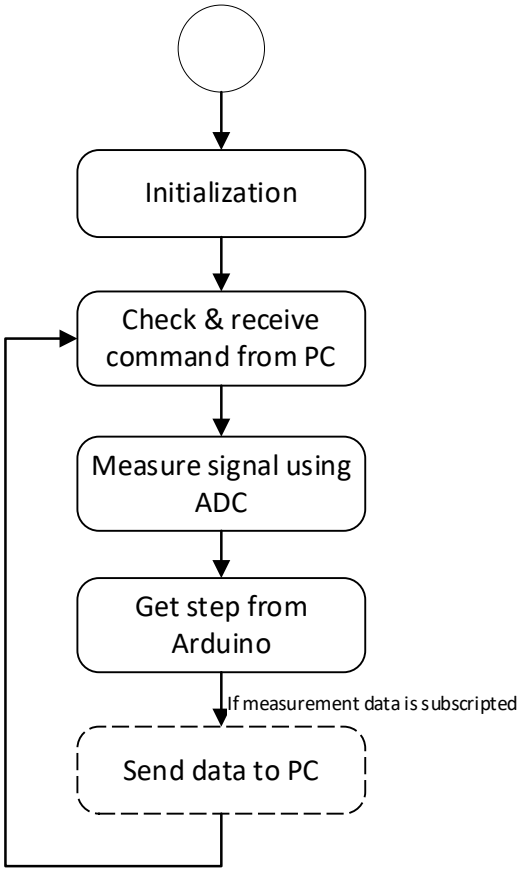


Figure 3.39: Flow chart of the PSoC program of the stretching test setup

3.5.4.1 Initialization

After the PSoC is powered on, the initialization is first executed, including enabling the interrupt, starting the operational amplifiers, the multiplexers, the ADC ,the UART component, the I2C component, etc.

3.5.4.2 Check & receive command from PC

After initialization is completed, the program enters an infinite loop. At the beginning of the loop, the program checks the UART buffer to check if the PC sends any command to PSoC. If so, the command is received and parsed according to the communication protocol. If not, the rest of the program is continued.

3.5.4.3 Measure signals using ADC

Next, the program controls the ADC to sample the sensor signal. The following code block shows the implementation.

Code block 3.2: Implementation of function read_ADC for signal measurement of the stretching test setup

```
1.  void readADC(void) {
2.
3.      AMux_Differential_FastSelect(0);
4.      AMux_Rg_DisconnectAll();
5.      CyDelayUs(1000);
6.      ADC_StartConvert();
7.      while(!ADC_IsEndConversion(CY_SAR_WAIT_FOR_RESULT));
8.      ADC_StopConvert();
9.      v_FPM = ADC_CountsTo_Volts(0, ADC_GetResult16(0));
10.
11.
12.     AMux_Differential_FastSelect(1);
13.     AMux_Rg_FastSelect(0);
14.     CyDelayUs(1000);
15.     ADC_StartConvert();
16.     while(!ADC_IsEndConversion(CY_SAR_WAIT_FOR_RESULT));
17.     ADC_StopConvert();
18.     v_force_sensor = ADC_CountsTo_Volts(0, ADC_GetResult16(0));
19.
20.
21.     v_position_sensor = ADC_CountsTo_Volts(1, ADC_GetResult16(1));
22.
23.
24. }
```

Firstly, the API function of the multiplexer *AMux_Differential_FastSelect* is used to control multiplexer *AMux_Differential* to connect channel 0 so that the voltage signal of four-point measurement is fed into the PSoC, then using the API function *AMux_Rg_DisconnectAll* disconnects all channels of multiplexer *AMux_Rg* so that the pre-amplifier circuit operate in voltage follower mode. A delay of 1000 microseconds is set after the channel switching. The reason is that a secondary filter is placed in front of the ADC. After the channel is switched, the output of the low-pass filter needs some time to be stable. It is determined experimentally that with a delay of 1000 microseconds, the stability and reliability of the ADC measurement results can be guaranteed. The ADC conversion is then started using the API function

ADC_StartConvert. The program waits in a loop until API function *ADC_IsEndConversion* returns true value and the conversion is stopped with API function *ADC_StopConvert*. The API function *ADC_CountsTo_Volts* is used to convert the by ADC measured count value is to voltage values according to the reference voltage and result format. The measurement result is stored in variable *v_FPM*. For force sensors, the same operation is repeated for the measurement. For position sensors, the voltage is directly measured via channel 1 of the ADC.

3.5.4.4 Get step from Arduino

After measuring the analog signal using ADC, the program collects motor step number from the Arduino via I2C to calculate the length of the sample being stretched.

In I2C communication, the PSoC is the master, and the Arduino is the slave. On the master side, the PSoC program needs to send requests to the Arduino and wait for the Arduino to respond. This process is implemented through the API function of the I2C component provided by the PSoC.

3.5.4.5 Send data to PC

Finally, if the PC sends the data subscription request in the second step, the PSoC sends all the collected data to the PC via UART. The implementation of the sending is very simple, people only need map the I/O functions of the C standard library to UART component, then data can be sent to PC by calling function *printf*.

4 Design of bending test setup

The main body of the bending test setup is designed by Jimmy Retzlaff and Philip Hoop in project work "Entwicklung eines automatisierten Biegemessstandes für ein flexibles Sensorpatch". For reader's better understanding and to avoid chaos article structure, in this thesis the old original part and the new upgrade part are not reviewed separately. But a complete description of the final setup is given.

4.1 Test principle

4.1.1 Sample structure

Figure 4.1 shows the schematic of the structure of the sample. There are two tested resistors on the sample that intersect at 90 degrees to each other to study the influence of the direction of the screen printing on the resistor characteristics. At the end of resistors, there are three measurement lines with contact pads at the end for connection to measurement circuit to measure the resistance of the tested resistor.

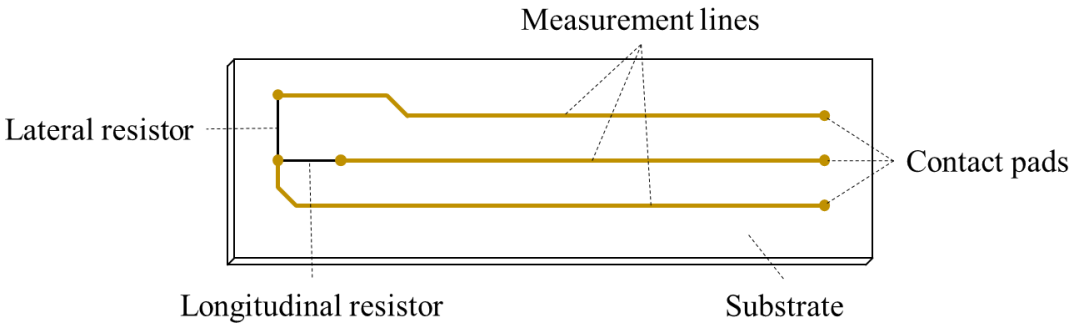


Figure 4.1: Schematic of the structure of the sample for bending test

4.1.2 Test process

Figure 4.2 describes the principle of bending test. As shown in Figure 4.2(a), the sample is in a naturally flat state before the test. After the test starts, the sample is bent by the setup around the lateral direction. There are two bending directions: positive bending and negative bending. In positive bending, the bending axis is above the sample (see Figure 4.2(b)) and in negative bending, the bending axis is below the sample (see Figure 4.2(c)). During the test, the longitudinal and lateral resistances are deformed with the bending progress, resulting in the

change of resistance. The bending progress and sample resistance are measured and recorded periodically for study.

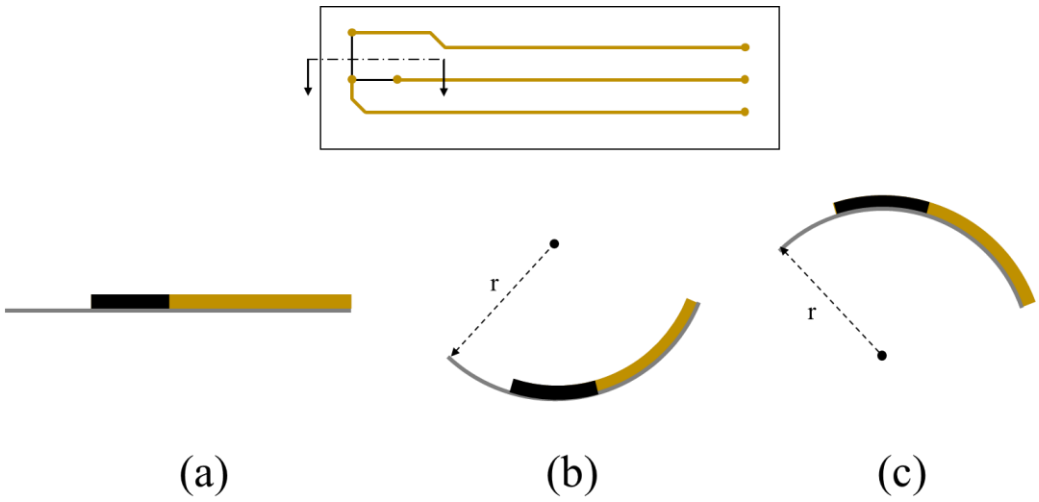


Figure 4.2: Schematic of the principle of the bending test. (a) Sample in flat state. (b) Positive bending. (c) Negative bending.

4.1.3 Measurement principle

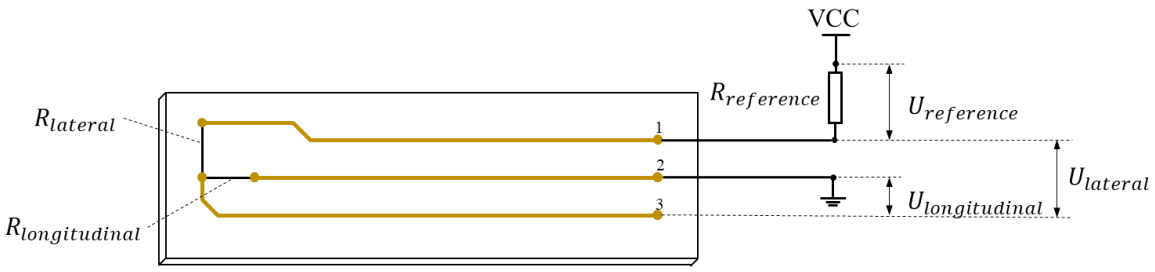


Figure 4.3: Schematic of the principle of four-point measurement method

Figure 4.3 shows the principle of resistance measurement. Contact pad 1 is connected to a DC voltage source VCC with a reference resistor $R_{reference}$. Contact pad 2 is connected to ground. The voltage of $R_{reference}$ is obtained by measuring the voltage between the VCC and pad 1. The voltage of $R_{lateral}$ can be obtained by measuring the voltage between pad 1 and pad 3. The voltage of $R_{longitudianl}$ can be obtained by measuring the voltage between plate 2 and pad 3.

According to the voltage division law of series circuit, the voltages ratio of the two resistors is equal to their resistance ratio. Because the resistance of the reference resistor is known, the resistance of $R_{lateral}$ and $R_{longitudianl}$ can be obtained by the following formula:

$$R_{longitudinal} = \frac{U_{longitudinal}}{U_{reference}} R_{reference} \qquad \text{formula 4-1}$$

$$R_{lateral} = \frac{U_{lateral}}{U_{reference}} R_{reference} \qquad \text{formula 4-2}$$

4.2 System overview

Figure 4.4 shows the overview of the bending test setup, which consists of a PC, an Arduino UNO development board, a PSoC 6 development board, actuators and a sample. The PC is the master of the test system. It provides a graphical interface program for the user to send test commands and receive and visualize measurement data. The Arduino receives the commands from the PC and converts them into control signals. The actuator including mechanical components such as motors, transmission structures, etc., converts the control signals into mechanical motion to bend the sample. The PSoC measures the resistance of the sample. Finally, the test data is sent back to the PC to be shown to the user.

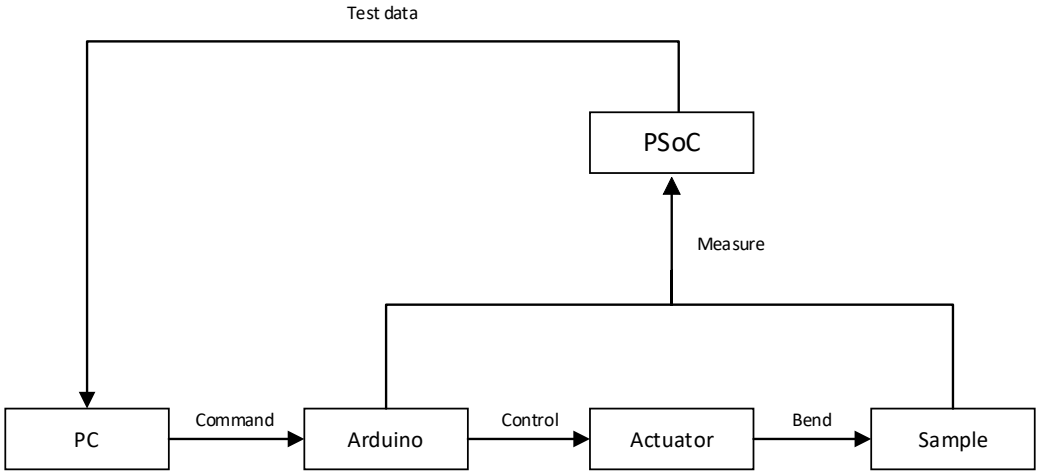


Figure 4.4: Schematic overview of the bending test setup

4.3 Mechanics

Figure 4.5 shows the kinematic schematic of the actuator. The sample is fixed to a bendable flexible metal sheet. The left side of the sheet is attached to a rolling support and the right side to a slider. Both ends of the connection are free to rotate. A crank-slider mechanism is used to translate the rotational motion of the motor into translation of the slider. By the drive of the slider, the elastic metal sheet is bending with the sample on it.

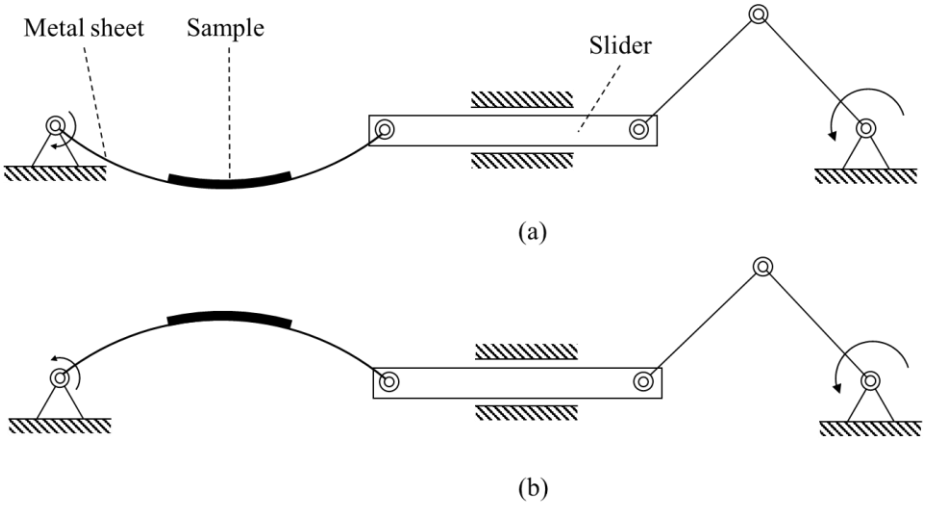


Figure 4.5: Kinematic diagram of the actuator of the bending test setup. (a) Positive bending. (b) Negative bending.

To control the bending direction of the sheet, a torque is applied around the rolling support on the left side. The direction of the torque will be reversed when the sheet reaches its flat position to change the bending direction of the sheet. When the torque is clockwise, positive bending is applied and when counterclockwise, negative bending is applied.

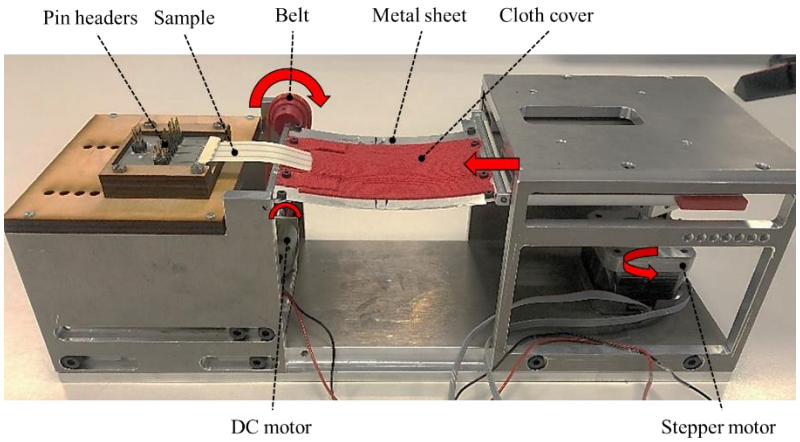


Figure 4.6: Photograph of the mechanical structure of the bending test setup

Figure 4.6 shows the photograph of the mechanical structure bending test setup. The crank-slider mechanism is actually arranged horizontally, which is different from Figure 4.5. Stepper motors are used to drive the crank-slider mechanism. The bending direction control mechanism is driven by a DC motor via a belt drive. The end of the sample with the resistance is fixed to the metal sheet by means of a cloth cover, which is flexible and can apply continuous pressure to the sample during the bending process to ensure that the sample remains rigidly attached to the sheet. The end of the sample with the pads is fixed to an electrical connector fixed on the

frame. The output voltages between three measurement lines of the sample can be easily measured via the pin header.

4.4 Electronics

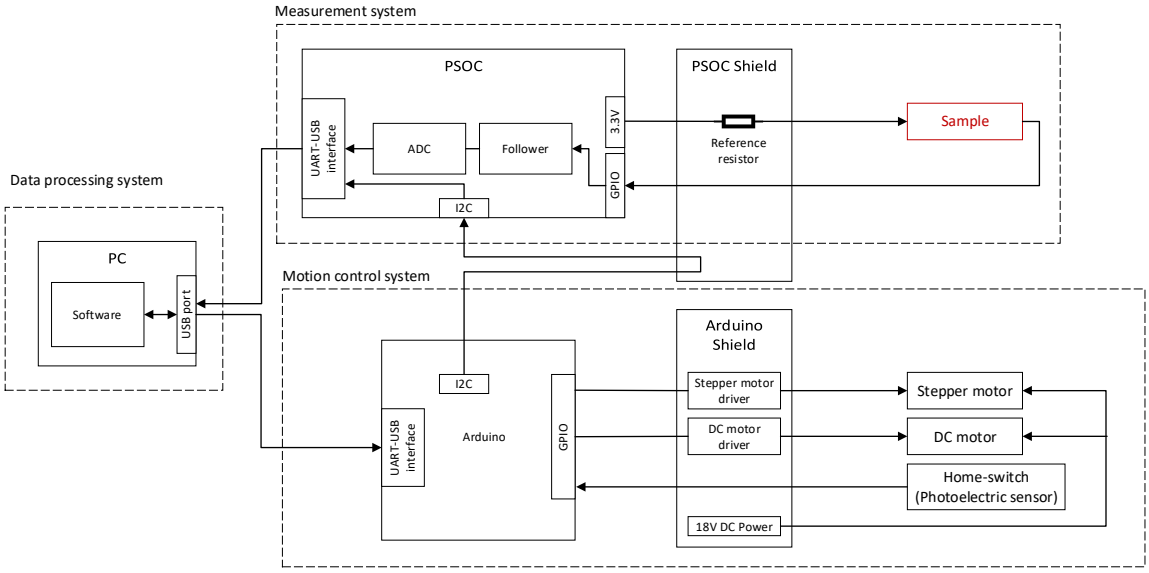


Figure 4.7: Schematic overview of the electronics part of the bending test setup

Figure 4.7 shows the schematic diagram of the electrical system. According to the functionality of the components, the electrical system can be divided into three subsystem, data processing system, motion control system and measurement system.

- Data processing system consist of a PC, which is responsible for human-computer interaction, test command sending and data receiving.
- Motion control system consists of a stepper motor, an Arduino UNO development board, an Arduino shield board with a motor driver and a 18V DC power supply. The Arduino controller controls the movement of the motor through the motor driver.
- Measurement system consists of a PSoC 6 development board and a PSoC shield board. The PSoC shield board is used to carry off-board external components for the measurement circuit including reference resistors. The PSoC collects the output signal of the sample.

4.4.1 Data processing system

The PC is connected to the PSoC of the measurement system and the Arduino of the motion system via USB. People can configure the test on the PC and send commands to the Arduino via serial port. The PC also receives the measurement data from the PSoC via serial port.

4.4.2 Motion control system

The motion control system consists of an Arduino UNO development board, an Arduino shield board, a 9 - 24 V adjustable DC power supply, a stepper motor, a DC motor and a Photoelectric sensor.

Figure 4.8 shows the photograph of the circuit of the measuring system. The Arduino is connected to PC via a USB cable to receive commands and convert them into control signals for the motors. A shield board is mounted on top of the Arduino with a DC motor driver, a stepper motor driver, a power input for motors and a wire connector. The power input is connected to a 9 - 24 V adjustable DC power supply. It is determined experimentally that the output voltage of the DC supply needs to be above 18 V, for DC the motor to generate sufficient torque to change the bending direction. Wire connector is connected to the stepper motor, the DC motor and the home-switch circuit.

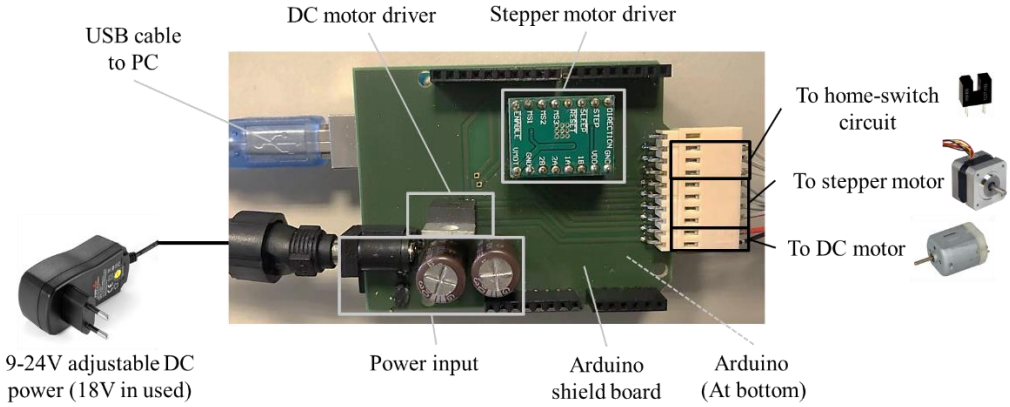


Figure 4.8: Photograph of the motion control system of the bending test setup

The home-switch circuit consists of a photoelectric sensor. It is mounted on the housing of the stepper motor and is used to detect whether the crank is correctly in the initial position before the test starts. Figure 4.9 shows the photograph of the home-switch circuit and its basic principle. The photoelectric sensor consists of a light emitting diode and an optical transistor. When the optical transistor receives light from the light emitting diode, the transistor conducts and the output pin outputs a low-level voltage signal, otherwise a high-level signal is output. Based on

this principle, a stop bar is installed on the crank handle. When the crank is in the initial position, the stop bar is located exactly in the gap of the photoelectric sensors. Before the test, if the sensor outputs a high electrical level, it can be confirmed that the crank is in the initial position.

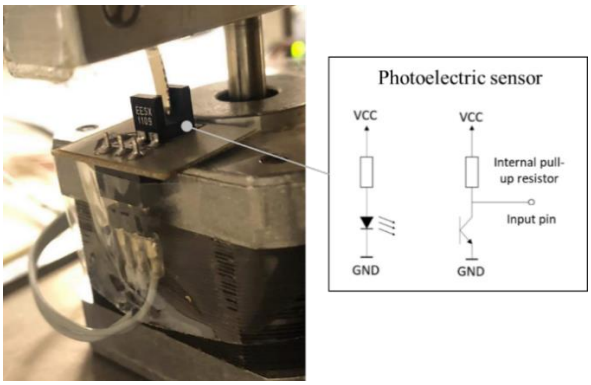


Figure 4.9: Photograph of the home-switch circuit and its basic principle [6]

4.4.3 Measurement system

Figure 4.10 shows the top-design of the PSoC, which is also the schematic of the measurement system. The red wires are the on-chip circuit configuration, and the blue line is the external circuit. The system contains a PSoC-based measurement circuit including a signal excitation circuit, a signal filtering circuit, a pre-voltage follower circuit, a signal selection circuit, a reference selection circuit, an ADC a UART component, and a I2C component.

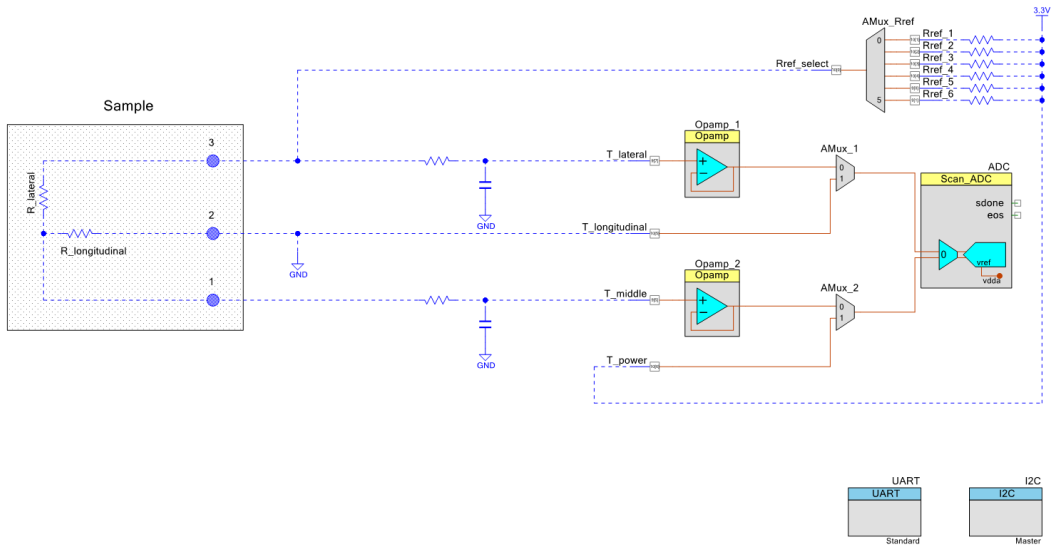


Figure 4.10: Top-design of the PSoC based measurement system of the bending test setup

Figure 4.11 shows the pin assignments of the PSoC for component signals.

	Name	Port	Pin	Lock
	\I2C:scl\	P6[0]	K8	☑
	\I2C:sda\	P6[1]	J8	☑
	\UART:rx\	P5[0]	L6	☑
	\UART:tx\	P5[1]	K6	☑
	Rref_1	P10[1]	A8	☑
	Rref_2	P10[2]	F6	☑
	Rref_3	P10[3]	E6	☑
	Rref_4	P10[4]	D6	☑
	Rref_5	P9[0]	D10	☑
	Rref_6	P9[1]	D9	☑
	Rref_select	P10[0]	B8	☑
	T_lateral	P9[7]	C7	☑
	T_longitudinal	P10[5]	B7	☑
	T_middle	P9[5]	C9	☑
	T_power	P10[6]	A7	☑

Figure 4.11: Pin assignment of the PSoC for the measurement system of the bending test setup

4.4.3.1 Excitation circuit

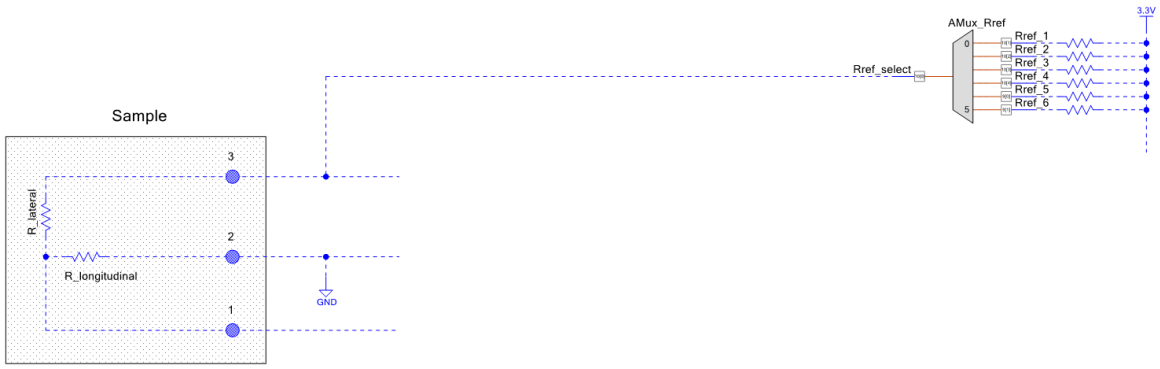


Figure 4.12: Schematic of the excitation circuit of the measurement system of the bending test setup

Figure 4.12 shows the schematic of the excitation circuit. According to the measurement principle, which is described in Chapter 4.1.3, contact pad 2 of the sample is connected to ground and pad 3 is connected to a 3.3 V voltage source via a switchable reference resistor. At each contact pad and the output 3.3 V voltage source a measurement wire is extended for signal acquisition.

4.4.3.2 Reference selection circuit

In test, the difference between the measured resistance and the reference resistance can influence the accuracy of the measurement. When the measured resistance is much larger than

the reference resistance, the voltage of the reference voltage is extremely low. On the one hand, the signal is susceptible to noise, on the other hand, the voltage is hard measured accurately due to limitation of the ADC. When the measured resistance is much smaller than the reference resistance, the voltage of the measured resistance is so low that similar effects can arise. Therefore, a switchable reference resistor circuit is designed to implement the selection of a suitable reference resistor according to the range of the measured resistance.

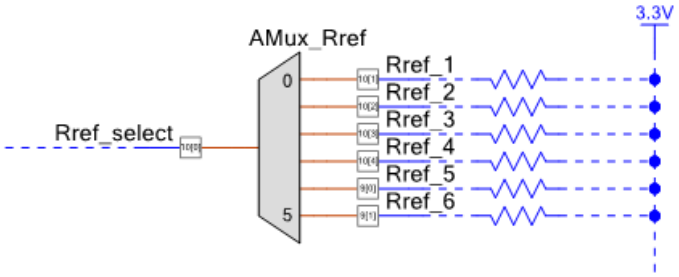


Figure 4.13: Schematic of the reference selection circuit of the measurement system of the bending test setup

Figure 4.13 shows the schematic of the reference selection circuit. Six different reference resistors are connected to the inputs of multiplexer *AMux_Rref*. By setting the connected input channel of *AMux_Rref* , one of these reference resistors is selected to be connected to the measurement loop. The resistance of reference *Rref_1*, *Rref_2*, *Rref_3*, *Rref_4* is1 kΩ, 10 kΩ, 100 kΩ, 1 MΩ, respectively. *Rref_5* and *Rref_6* is reserved for user-defined resistance.

However, in practice, the internal resistance of the routes of the multiplexer also needs to be considered. The internal resistance can be measured with the ohmmeter in the interface analog of PSoC creator, Figure 4.14 shows the measurement result of the route resistance between *Rref_select* (P10[0]) and *Rref_1* (P10[1]). The resistance is approximately 550 Ω. So, the actual resistance of the reference resistor needs to be corrected by adding 550 Ω.

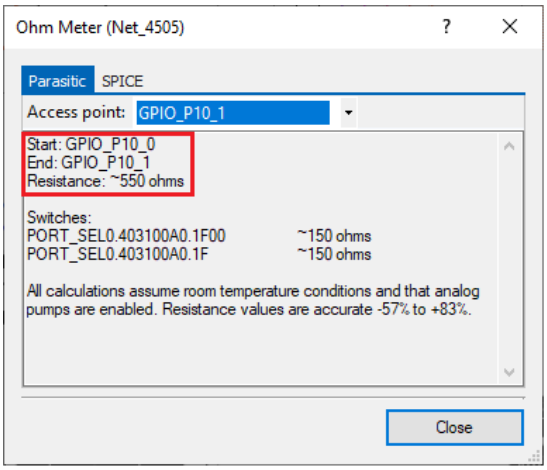


Figure 4.14: Internal route resistance of the multiplexer measured with the ohmmeter of IDE PSoC creator

Table 4.1 shows the reference resistances for each channel of the multiplexer and their corresponding optimal resistance ranges of the sample. The method of the decision of the optimal measuring range is to keep the resistance ratio of the reference sample in the range of 1:10 and 1:1. In this range, the voltage of the reference is between 0.3 V and 1.65 V, which can be accurately converted by the 12-bit ADC.

Table 4.1: Available references of the measurement system of the bending test setup and their corresponding measurement range for the sample

AMux_Rref	Channel	Resistance	Measurement range (Total resistance of R_lateral and R_longitudianl)
1	0	1 kΩ (+550 Ω)	1 kΩ – 10 kΩ
2	1	10 kΩ (+550 Ω)	10 kΩ – 100 kΩ
3	2	100 kΩ (+550 Ω)	100 kΩ – 1 MΩ
4	3	1 MΩ (+550 Ω)	1 MΩ – 10 MΩ
5	4	Reserved	-
6	5	Reserved	-

4.4.3.3 Signal filtering circuit

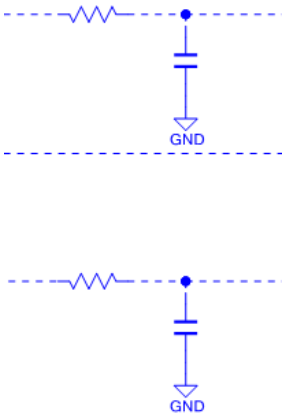


Figure 4.15: Schematic of the signal filtering circuit of the measurement system of the bending test setup

Figure 4.15 shows the schematic of the signal filtering circuit. The signals from contact pads 1 and 3 are fed into two RC low-pass filters. The resistors have a resistance value of 20 kΩ, and the capacitance size is 0.01 μF. So, the cut-off frequency of the filters is about 79.5 Hz. It is experimentally found that the noise of the signal from pad 2, which is connected to the ground and from power supply is so small that there is almost no difference before and after the low-pass filter was added, so no filtering was used for these two measurements.

The choice of resistance and capacitance of the RC-filter is based on the response time of the filter. For the bending test setup, the stepper motor speed is within 15 r/min at normal bending speed, i.e., the interval between each step of the stepper motor is approximately 1.1 ms. To ensure that the filter outputs an accurate voltage signal in time, the output settling time of the filter should be within 1.1 ms. With the above resistors and capacitors, the output voltage delay is approximately 1 ms, meeting the dynamic requirements.

4.4.3.4 Voltage follower circuit

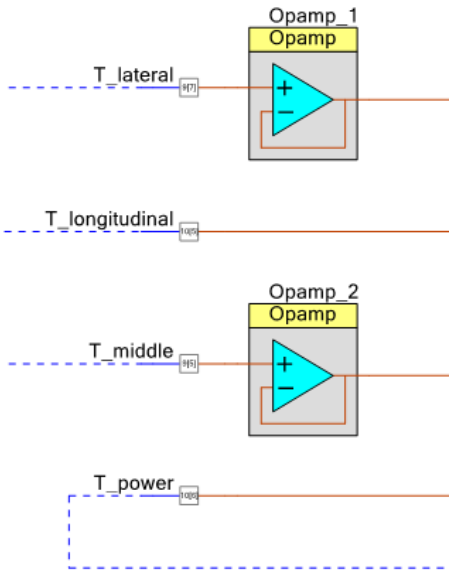


Figure 4.16: Schematic of the voltage follower circuit of the measurement system of the bending test setup

As shown in Figure 4.16, after being filtered, the signals from pad1 and pad3 are fed into a voltage following circuit via pin *T_lateral* and *T_middle*. The reason for using voltage followers is that, since the resistance of the resistor under test can be several MΩ while the input impedance of the ADC is much lower, voltage measurement errors occur. With the help of the voltage follower, whose input impedance is infinite, the isolation front and rear stage circuits can be achieved. For signal form pad 2 and power, the signal is directly fed into the PSoC via pin *T_longitudianl* and *T_power* without voltage followers, the stable 0 V and 3.3 V signals is measured, and the internal resistance of the GPIO port is much smaller than the ADC input impedance, which can be ignored.

4.4.3.5 Signal selection circuit

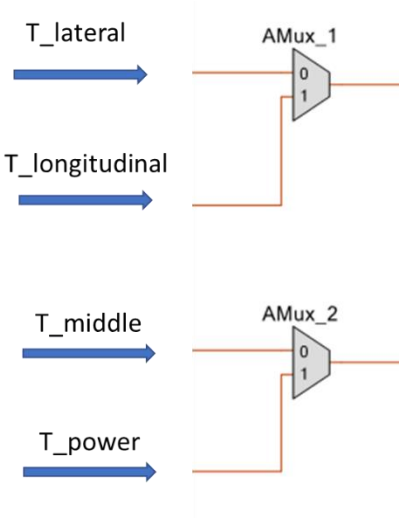


Figure 4.17: Schematic of the signal selection circuit of the measurement system of the bending test setup

Due to conflicting routes, it is not possible to connect all signals to the ADC in parallel, so a signal selection circuit is necessary to switch the signals to be measured. As shown in Figure 4.17, the signals input via pin *T_power*, *T_middle*, *T_lateral*, , *T_longitudinal* and *T_power* is connected to two multiplexers, forming a signal selection circuit.

The two multiplexers enable the selection of the measurement of the different resistors' voltage signals. Table 4.2 shows the selected input signals corresponding to the combination of the two multiplexers' input channels.

Table 4.2: Multiplexer channel configuration and its corresponding measurement signal

Input channel		Selected signal	By ADC measured resistor
AMux_1	AMux_2		
0	0	T_lateral, T_middle	R_lateral
0	1	T_lateral, T_power	R_reference
1	0	T_middle, T_longitudinal	R_longitudinal

When multiplexer *AMux_1* selects channel 0 and *AMux_2* selects channel 0, the signals from pin *T_lateral* (Pad 3) and *T_middle* (Pad 1) are selected. Thus, the ADC measures the voltage

of *R_lateral*. When multiplexer *AMux_1* selects channel 0 andAMux_2 selects channel 1, the voltage signals from pin *T_lateral* (Pad 3) and *T_power* (3.3 V) are selected. Thus, the ADC measures the voltage of *R_reference*. When multiplexer *AMux_1* selects channel 1 and *AMux_2* selects channel 0, the voltage signals from pin *T_middle* (Pad 1) and *V_longitudinal* (Pad 2) are selected. Thus, the ADC measures the voltage of *R_longitudinal*.

4.4.3.6 ADC

During a single sampling period, the signal of each resistor is input to the ADC for measurement in turn with the signal selection circuit. Figure 4.18 shows the configuration of the ADC component. The ADC has one input channel, channel 1, which is configured as differential input. 3.3 V Vdda is selected as the reference voltage of the ADC and the format of conversion results configured as signed. In this case, the measurement range of the differential signal is -3.3 V to 3.3 V. To reduce noise, sampling averaging is used. The result of a single measurement is averaged over 64 consecutive samples.

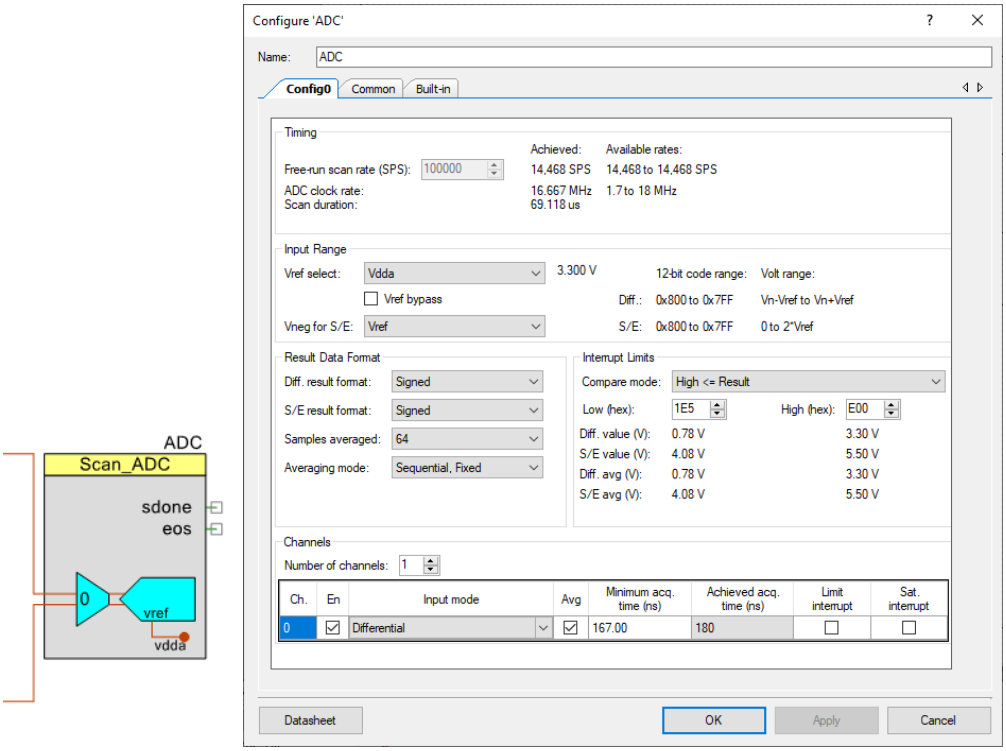


Figure 4.18: PSoC ADC configuration of the measurement circuit of the bending test setup

With the above configuration, the ADC can sample at a rate of 14468 SPS and it costs 69 µs to finish the measurement. Since the signal selection circuit needs to be switched 3 times to

measure the voltage of all resistors. So, the complete measurement time of a period is about 207 µs.

4.4.3.7 I2C

PSoC collects the number of motor steps since test start from Arduino via I2C to calculate the elongation of the sample. For the I2C component, the default configuration is used, as shown in Figure 4.19. PSoC is master, Arduino is slave and communication rate is 100kbps.

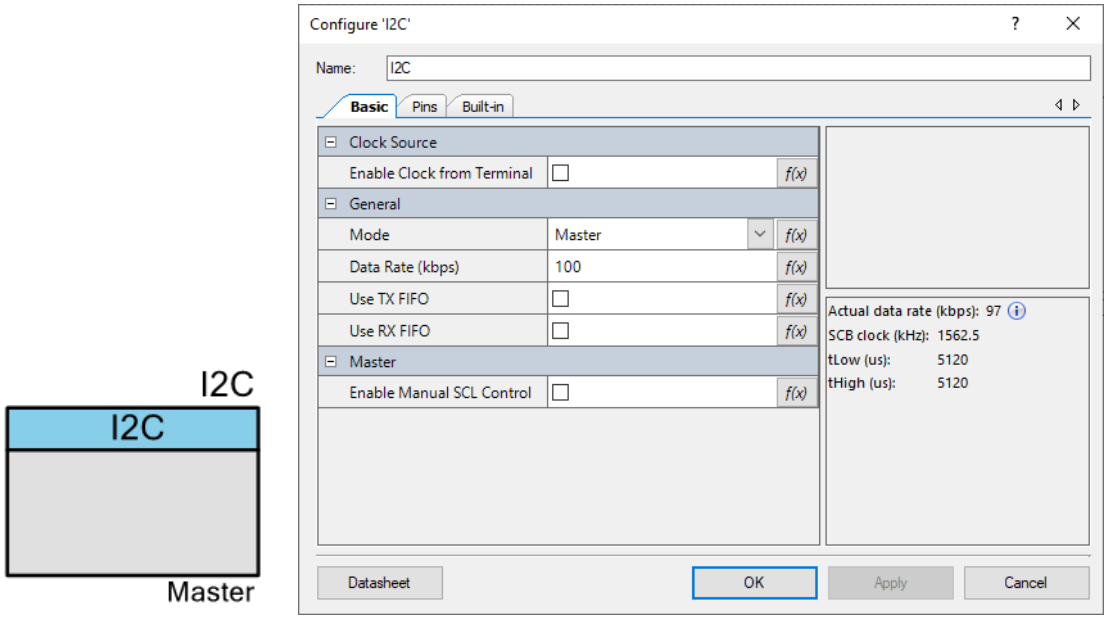


Figure 4.19: PSoC I2C component configuration of the measurement circuit of the bending test setup

Since the pull-up resistors are preset on both PSoC and Arduino development boards, there is no need to add pull-up resistors.

4.4.3.8 UART

The PSoC receives commands from the PC and sends the measured data to the PC via UART. For the UART component, the default configuration is used as shown in Figure 4.20. The baud rate is 115200 bps, the data length is 8 bits, the bit order is LSB first, and the stop bits is 1.

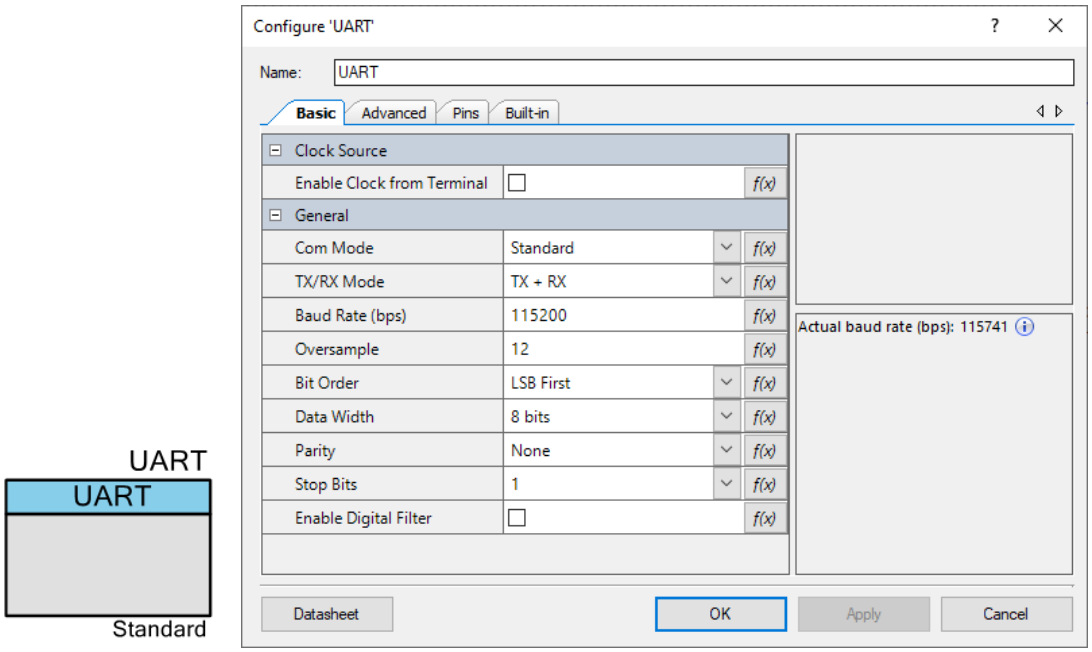


Figure 4.20: PSoC UART component configuration of the measurement circuit of the bending test setup

4.4.3.9 PSoC shield

To improve the integration of the system, a shield board is designed for the PSoC to hold the external off-chip components of the measurement system. Annex B shows the schematic of PSoC shield for bending test setup. Figure 4.21 shows the 3D model of the shield board in top view.

The left side of the shield is the area for the reference resistor. In consideration of possible future changes, connection slots are used instead of fixed SMD resistors.

In the upper part of the extension y measurement connectors, these slots are used to provide an interface between the measurement power and the received measurement signal.

On the upper side of the shield are connection headers for measurement, providing the interface for excitation source and the reading of signals. The top of the headers has text indicating the connection destination of each header. The Mid. header on the left side is connected to the contact pad 3 of the sample, the Long. header is connected to the contact pad 2, and the Lat. header connects to the contact pad 1. (Pad number is shown in Figure 4.3)

Below the measurement headers there are several sets of resistors and capacitors that form the low-pass filters in the measurement circuit.

The lower right slot is the I2C connection headers used to connect to the Arduino's I2C port for step acquisition.

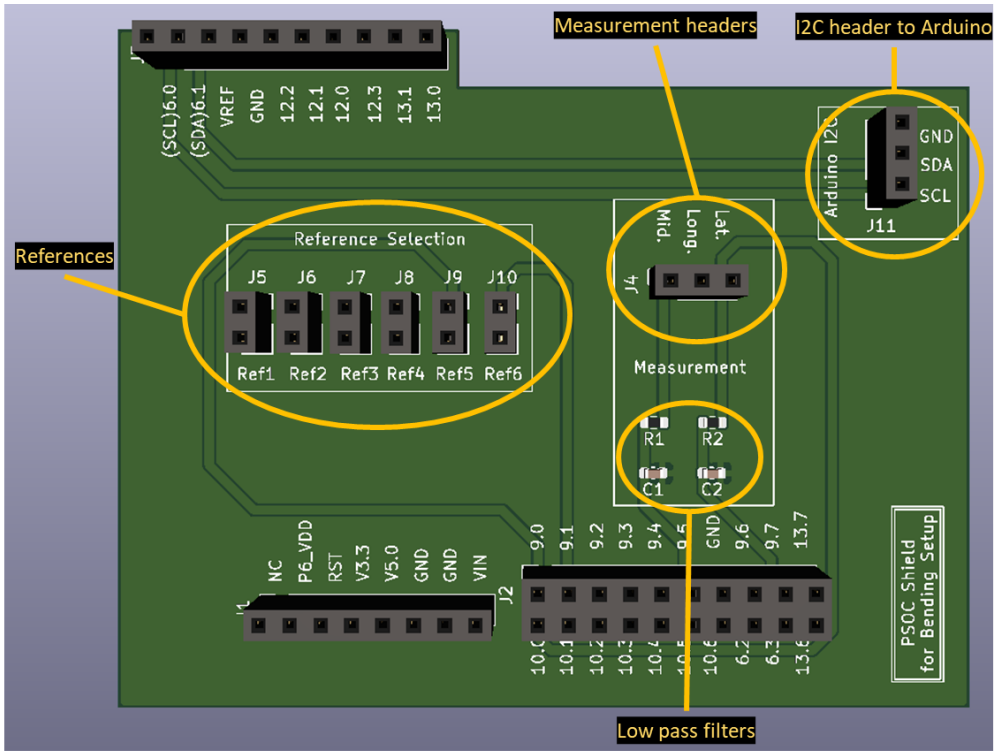


Figure 4.21: 3D model of the PSoC shield board of the bending test setup in top view

4.5 Software

According to the platform, the software system of the setup can be divided into three parts:

1. PC program: It is an application witch GUI, which enables user to configure the test parameters, send the test commands, monitor and save the test result.

2. Arduino program: It is mainly responsible for the receiving and parsing the test commands from the PC and driving the motor.

3. PSoC program: It is mainly responsible for the acquisition of measurement data and sending measurement data to the PC.

4.5.1 Communication protocol

As a whole system, it is necessary for the programs of different platform to share and exchange information. The communications existing between three platforms:

1. PC-Arduino communication: The purpose of communication between PC and Arduino is to control and monitor the test process.
2. PC-PSoC communication: The communication between PC and PSoC is mainly used for the control and acquisition of measurement.
3. PSoC-Arduino communication: The communication between PSoC and Arduino is for PSoC to collect the step data from Arduino.

Communication is achieved by request and response telegrams. The layout of the telegram of the bending setup is the same as the stretching test setup. Please see Chapter 3.5.1 for detailed description.

4.5.1.1 PC – Arduino communication

Launch-test

The purpose of the communication task launch-test is to command the Arduino to start a new test. As shown in Figure 4.22, PC sends a *launch-test request* to the Arduino, once the Arduino successfully receives the request, it starts a test and reply launch-test response to the PC to monitor the test progress every 100 microseconds. The reply is not stopped until the test is finished or stopped by the user.

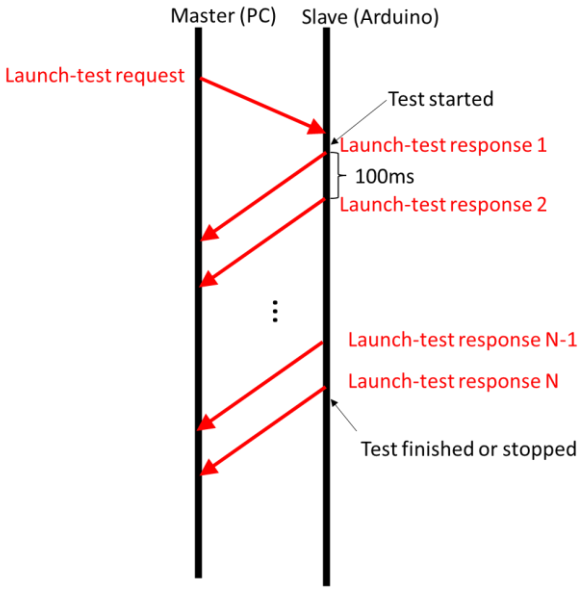


Figure 4.22: Sequence diagram of PC-Arduino launch-test communication task of the bending test setup

Launch-test request

Table 4.3 shows the layout of the *Launch-test request* and Table 4.4 explains each parameter in the telegram in detail.

Table 4.3: Layout of PC-Arduino launch-test request telegram of the bending test setup

Type(00)	\r\n

Table 4.4: Parameter description of PC-Arduino launch-test request telegram of the stretching test setup

Place	Fieldname	Format	Unit	Range	Description
0	Type	string	-	-	Telegram type: “00” = Launch-test request

Launch-test response

Table 4.5 shows the layout of the Launch-test request and Table 4.6 explains each parameter in the telegram in detail.

Table 4.5: Layout of PC-Arduino launch-test response telegram of the bending test setup

Type(01)	,	Progress status	,	Step	,	Bending direction	,
Speed	,	Cycles	\r\n				

Table 4.6: Parameter description of PC-Arduino launch-test response telegram of the stretching test setup

Place	Fieldname	Format	Unit	Range	Description
0	Type	string	-	-	Telegram type: “01” = Launch-test response

1	Progress status	int	-	0,1,2	0: Test stopped or finished 1: Test started 2: Test paused
2	Step	int	-	> 0	The number of steps taken by the stepper motor since the test starts.
3	Bending direction	int	-	0,1,2	The bending direction of the sample 0: Positive (Sample compressed) 1: Negative (Sample stretched) 2: Alternate
4	Bending speed	int	r/min	> 0	The received number of repetitions of Back and forth stretching
5	Cycles	int	-	> 0	The number of cycles of the bending test

Test progress control

The purpose of the communication task test progress control is to control the test progress to pause, start and stop after the test is launched. As shown in Figure 4.23, the PC sends a *Test progress control request* to the Arduino, once the Arduino successfully receives the request, it sends a respond to the PC to report that it receives the command and then follows the command to control the progress.

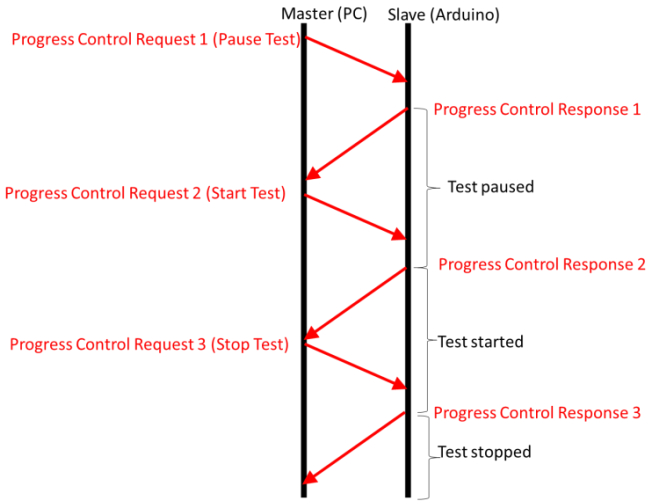


Figure 4.23: Sequence diagram of PC-Arduino test progress control communication task of the bending test setup

Test progress control request

Table 4.7 shows the layout of the *Test progress control request* and Table 4.8 explains each parameter in the telegram in detail.

Table 4.7: Layout of PC-Arduino test progress control request telegram of the bending test setup

Type(02)	,	Control flag	\r\n

Table 4.8: Parameter description of PC-Arduino test progress control request telegram of the bending test setup

Place	Fieldname	Format	Unit	Range	Description
0	Type	string	-	-	Telegram type: “02” = Progress control request
1	Control flag	int	-	0,1,2	0: Stop test 1: Start test 2: Pause test

Test progress control response

Table 4.9 shows the layout of the *Test progress control response* and Table 4.10 explains each parameter in the telegram in detail.

Table 4.9: Layout of PC-Arduino test progress control response telegram of the bending test setup

Type(03)	,	Control flag	\r\n

Table 4.10: Parameter description of PC-Arduino test progress control response telegram of the bending test setup

Place	Fieldname	Format	Unit	Range	Description
0	Type	string	-	-	Telegram type: “03” = Progress control response
1	Control flag	int	-	0,1,2	0: Stop test 1: Start test 2: Pause test

Test parameter setting

The purpose of the communication test parameter setting is to configure the test parameter. As shown in Figure 4.24, the PC sends a test *parameter setting request* to the Arduino, once the Arduino successfully receives the request, it replies to the PC to report that it receives the command and then change the test parameter.

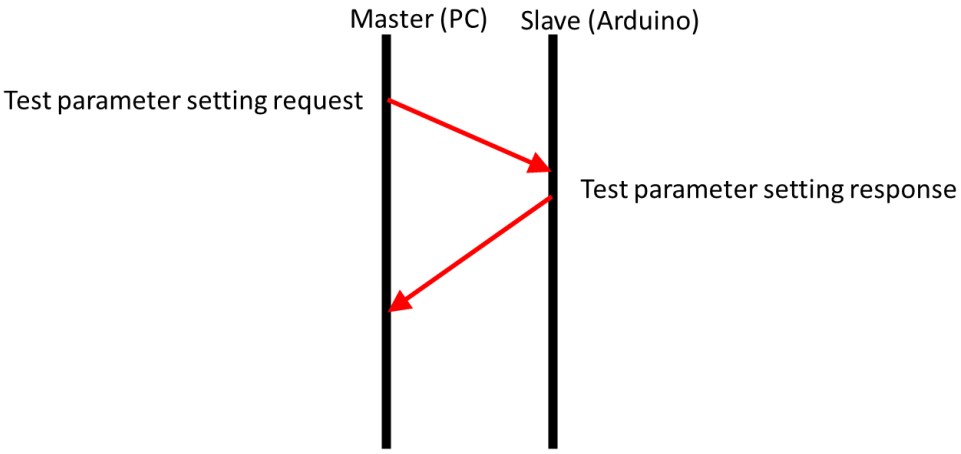


Figure 4.24: Sequence diagram of PC-Arduino test parameter setting communication task of the bending test setup

Test parameter setting request

Table 4.11 shows the layout of the *test parameter setting request* and Table 4.12 explains each parameter in the telegram in detail.

Table 4.11: Layout of PC-Arduino test parameter setting request telegram of the bending test setup

Type(04)	,	Bending direction	,	Bending speed	,	Cycles	\r\n

Table 4.12: Parameter description of PC-Arduino parameter setting request telegram of the bending test setup

Place	Fieldname	Format	Unit	Range	Description
0	Type	string	-	-	Telegram type: “04” = Parameter setting request
1	Bending direction	int	-	0,1,2	The bending direction of the sample 0: Positive (Sample compressed) 1: Negative (Sample stretched) 2: Alternate

2	Bending speed	int	r/min	> 0	The received number of repetitions of Back and forth stretching
3	Cycles	int	-	> 0	The number of cycles of the bending test

Test parameter setting response

Table 4.13 shows the layout of the *test parameter setting response* and Table 4.14 explains each parameter in the telegram in detail.

Table 4.13: Layout of PC-Arduino test parameter setting response telegram of the bending test setup

Type(05)	,	Bending direction	,	Bending speed	,	Cycles	\r\n

Table 4.14: Parameter description of PC-Arduino parameter setting response telegram of the bending test setup

Place	Fieldname	Format	Unit	Range	Description
0	Type	string	-	-	Telegram type: “05” = Parameter setting response
1	Bending direction	int	-	0,1,2	The bending direction of the sample 0: Positive (Sample compressed) 1: Negative (Sample stretched) 2: Alternate
2	Bending speed	int	r/min	> 0	The received number of repetitions of Back and forth stretching
3	Cycles	int	-	> 0	The number of cycles of the bending test

Error-telegram

As shown in Figure 4.25, once the user sends an undefined request or the layout and parameters of the request doesn't match the rule, the Arduino just sends an Error-telegram to the PC and the request doesn't have any effect.

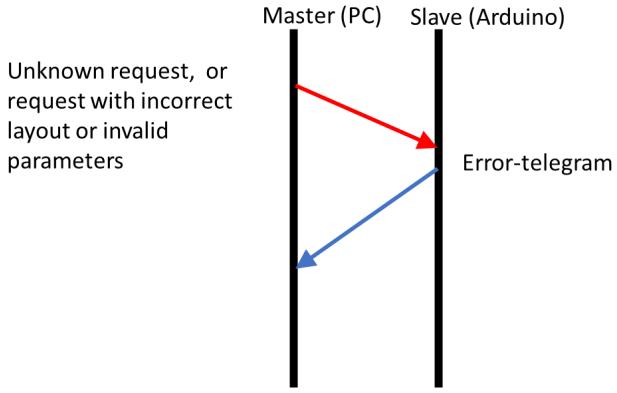


Figure 4.25: Sequence diagram of PC-Arduino error telegram of the bending test setup

Table 4.15 shows the layout of the *Error-telegram* and Table 4.16 explains each parameter in the telegram in detail.

Table 4.15: Layout of PC-Arduino error telegram of the bending test setup

Type(99)	\r\n

Table 4.16: Parameter description of PC-Arduino error telegram of the bending test setup

Place	Fieldname	Format	Unit	Range	Description
0	Type	string	-	-	Telegram type: "99" = Error-telegram

4.5.1.2 PC – PSoC communication

Data subscription

The purpose of the communication task data subscription is to command the PSoC to send measured data to the PC. As shown in Figure 4.26, PC sends a *data subscription request* to the Arduino, once the Arduino successfully receives the request, it starts sending measured data to the PC continuously. The data transfer is not stopped until a *data subscription request* for unsubscription is sent from the PC by the user.

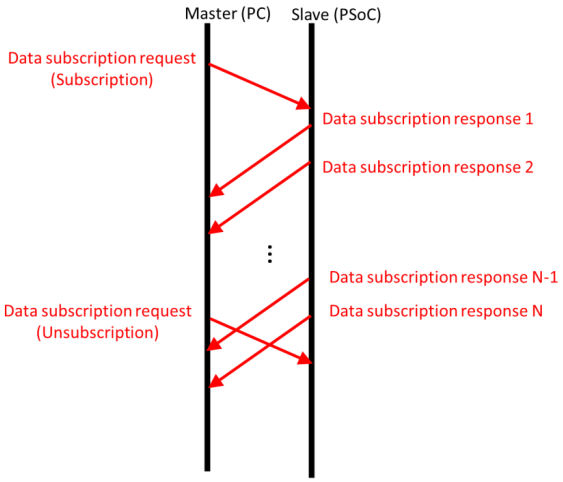


Figure 4.26: Sequence diagram of PC-PSoC data subscription communication task of the bending test setup

Data subscription request

Table 4.17 shows the layout of the *data subscription request* and Table 4.18 explains each parameter in the telegram in detail.

Table 4.17: Layout of PC-PSoC data subscription request telegram of the bending test setup

Type(00)	,	Subscription status	\r\n

Table 4.18: Parameter description of PC-PSoC data subscription request telegram of the bending test setup

Place	Fieldname	Format	Unit	Range	Description
0	Type	string	-	-	Telegram type: “00” = Data subscription request

1	Subscription status	int	-	0,1	0: Unsubscription 1: Subscription

Data subscription response

Table 4.19 shows the layout of the *data subscription response* and Table 4.20 explains each parameter in the telegram in detail.

Table 4.19: Layout of PC-PSoC data subscription response telegram of the bending test setup

Type(01)	,	Timestamp	,	Step	,	R_longitudinal	,	R_lateral	,
V_ longitudinal	,	V_ lateral	,	V_reference	,	Reference channel	\r\n		

Table 4.20: Parameter description of PC-PSoC data subscription response telegram of the bending test setup

Place	Fieldname	Format	Unit	Range	Description
0	Type	string	-	-	Telegram type: “01” = Data subscription response
1	Timestamp	int	ms	> 0	The number of microseconds since PSoC is powered on.
2	steps	int	-	> 0	The number of steps taken by the stepper motor from the starting position driven by the Arduino.
3	R_longitudinal	int	Ω	-	The resistance of the longitudinal resistor of the sample
4	R_lateral	int	Ω	-	The resistance of the lateral resistor of the sample

5	V_ longitudinal	float	V	-	The measured voltage of the longitudinal resistor of the sample
6	V_ lateral	float	V	-	The measured voltage of the lateral resistor of the sample
7	V_ref	float	V	-	The measured voltage of the reference resistor
7	Reference channel	int	-	0 - 5	The current selected channel reference 0: Channel 0 (1 kΩ) 1: Channel 1 (10 kΩ) 2: Channel 2 (100 kΩ) 3: Channel 3 (1 MΩ) 4: Channel 4 (Reserved) 5: Channel 5 (Reserved)

Measurement setting

The purpose of the communication task measurement setting is to configure the measurement parameter of the PSoC. As shown in Figure 4.27, the PC sends a t *measurement setting request* to the Arduino, once the Arduino successfully receives the request, it replies to the PC to report that it receives the command and then change the measurement parameter.

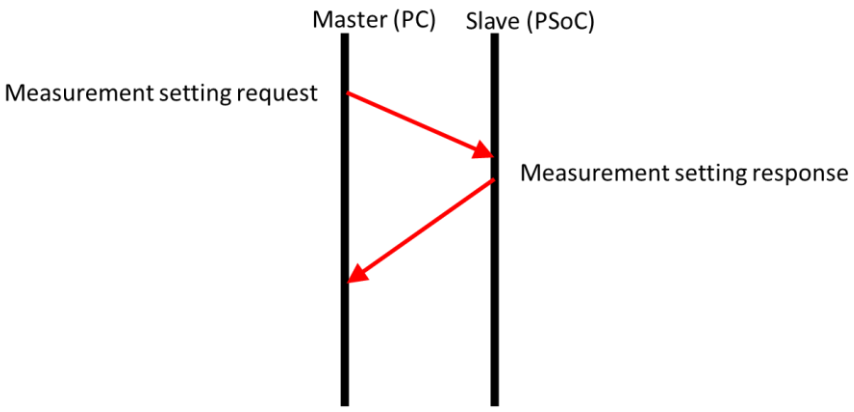


Figure 4.27: Sequence diagram of PC-PSoC measurement setting communication task of the bending test setup

Measurement setting request

Table 4.21 shows the layout of the *measurement setting request* and Table 4.22 explains each parameter in the telegram in detail.

Table 4.21: Layout of PC-PSoC measurement setting request telegram of the bending test setup

Type(02)	,	Sample rate	,	Downsample	,	Reference Selection	\r\n

Table 4.22: Parameter description of PC-PSoC measurement setting request telegram of the bending test setup

Place	Fieldname	Format	Unit	Range	Description
0	Type	string	-	-	Telegram type: “02” = Measurement setting request
1	Sample rate	int	Hz	> 0	indicates how often the signal is measured
2	Downsample	int	-	> 0	indicates how often the data is sent to PC. If downsample is 5 then one data is sent for every 5 measurements.
3	Reference Selection	int	-	-1-5	Selection of the channel of reference -1: Program selects the optimal reference automatically 0: Channel 0 (1kΩ) 1: Channel 2 (10 kΩ) 2: Channel 2 (100 kΩ) 3: Channel 3 (1 MΩ)

					4: Channel 4 (Reserved) 5: Channel 5 (Reserved)

Measurement setting respond

Table 4.23 shows the layout of the *measurement setting respond* and Table 4.24 explains each parameter in the telegram in detail.

Table 4.23: Layout of PC-PSoC measurement setting response telegram of the bending test setup

Type(03)	,	Sample rate		,	Downsample	,	Reference Selection	\r\n

Table 4.24: Parameter description of PC-PSoC measurement setting response telegram of the bending test setup

Place	Fieldname	Format	Unit	Range	Description
0	Type	string	-	-	Telegram type: “03” = Measurement setting respond
1	Sample rate	int	Hz	> 0	indicates how often the signal is measured
2	Downsample	int	Hz	> 0	indicates how often the data is sent to PC. If downsample is 5 then one data is sent for every 5 measurements.
3	Reference Selection	int	-	-1-5	Selection of the channel of reference -1: Program selects the optimal reference automatically 0: Channel 0 (1 kΩ) 1: Channel 2 (10 kΩ) 2: Channel 2 (100 kΩ) 3: Channel 3 (1 MΩ)

					4: Channel 4 (Reserved) 5: Channel 5 (Reserved)

Error-telegram

As shown in Figure 4.28, once the user sends an undefined request or the layout and parameters of the request doesn't match the rule, the PSoC just sends an *Error-telegram* to the PC and the request doesn't have any effect.

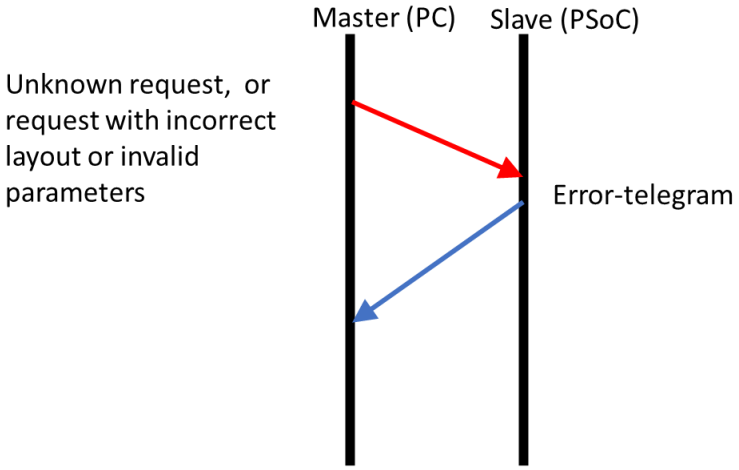


Figure 4.28: Sequence diagram of PC-PSoC error telegram of the bending test setup

Table 4.25 shows the layout of the *Error-telegram* and Table 4.26 explains each parameter in the telegram in detail.

Table 4.25: Layout of PC-PSoC error telegram of the bending test setup

Type(99)	\r\n

Table 4.26: Parameter description of PC-PSoC error telegram of the bending test setup

Place	Fieldname	Format	Unit	Range	Description
0	Type	string	-	-	Telegram type: "99" = Error-telegram

4.5.1.3 PSoC – Arduino communication

The purpose of the I2C communication is to transfer the step information from the Arduino to the PSoC. The PSoC is the master, and the Arduino is the slave. As shown in Figure 4.29, the PSoC sends a I2C request to the Arduino. Once the Arduino receives the request, it sends the response to the PSoC.

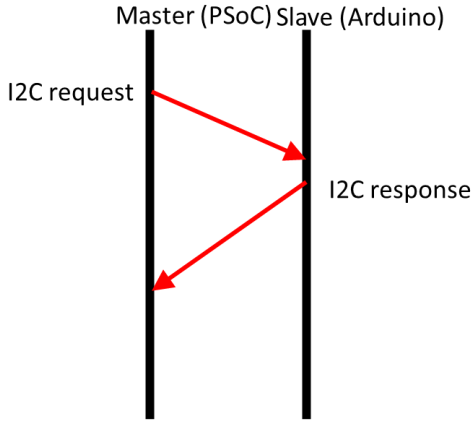


Figure 4.29: Sequence diagram of PSoC-Arduino I2C communication of the bendings test setup

I2C Request

The I2C request contains the address of the slave. The request does not need to be explicitly defined by the user, so it is not specified here.

I2C Response

Table 4.27 shows the layout of the I2C response telegram sent from the Arduino to the PC. This telegram only contains the step data of the stepper. Since I2C requests and responses are one-to-one, terminator is not used here to distinguish between continuous telegrams.

Table 4.27: Layout of PSoC-Arduino I2C response telegram of the bending test setup

Steps

Table 4.28 explains each parameter in the telegram in detail.

Table 4.28: Parameter description of PSoC-Arduino I2C telegram of the bending test setup

Place	Fieldname	Format	Unit	Range	Description
0	Steps	int	-	> 0	The number of steps taken by the stepper motor from the starting position.

4.5.2 PC program

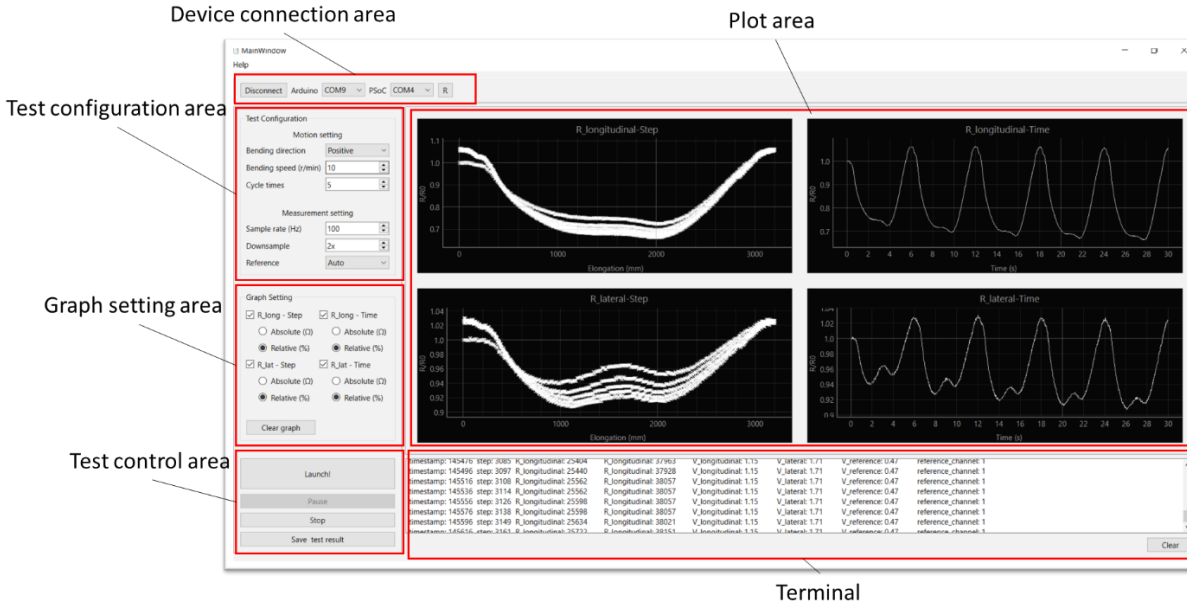


Figure 4.30: GUI of the PC program of the bending test setup

The PC program provides a GUI for the user to set test parameters, send test commands and view measurement data. Figure 4.30 shows the GUI program. According to the functionality, the interface can be divided into five areas:

1. Device connection area: in this area, the user can select the port of the Arduino and PSoC and establish the connection with them, which is the prerequisite for a test.
2. Test configuration area: In this area, the user can adjust the test parameters including the configuration of the motion and the measurement.
3. Plot area: in this area, the measurement data is visualized as real-time graphs. There are two types of graphs, resistance-step graph and resistance-time graph. The resistance-step graph shows the relationship between steps of the stepper motor, which indicates

the progress of the bending, and resistance. The resistance-time graph describes the sample resistance with the time. For both types, there are graphs of the longitudinal and lateral resistance. So, there are four graphs in the plot area in total.

4. Graph setting area: In this area, the user can configure the graphs in plot area, like showing/hiding of the specific graph and changing the resistance format. If radio box absolute is checked, the graph y-axis shows the resistance value. If relative is selected, the graph y-axis shows percentage of the resistance relative to initial resistance
5. Terminal: In this area, application notifications and the measurement data is shown in text form.

4.5.3 Arduino program

The Arduino program receives test commands from the PC and sends control signals to the motor driver. The new Arduino program is based on the Philip Hoop's original program. Figure 4.31 show the flow chart of the program.

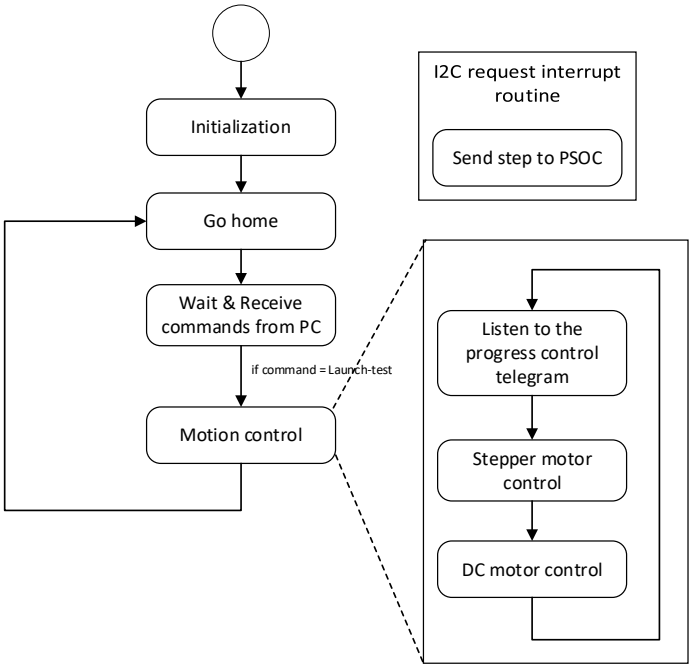


Figure 4.31: Flow chart of the Arduino program of the bending test setup

4.5.3.1 Initialization

After the Arduino is powered on, the program first executes function *setup* to initialize the Arduino, including the initialization of the UART and I2C, and the configuration of the GPIO working mode.

4.5.3.2 Go home

After the initialization is completed, the program enters function *loop*， which is executed infinitely. At the beginning of the loop, the program executes function *goHome* to drive the stepper motor to the initial position. The program continuously sends motion commands to the stepper motor. When it is detected that the photoelectric sensor is blocked using method *H_switch->isHome*, which indicates that the stepper motor reaches the initial position, the process is stopped.

4.5.3.3 Wait & Receive command from PC

After the mechanism is initialized, the program to listen to the serial port status and waits for the command from PC. When data is detected in the UART input buffer, the data is received parsed according to the communication protocol. The detection and reading of the serial port are realized by Arduino library Serial.

4.5.3.4 Motion control

Once launch-test request telegram is received, the program controls the motor movement according to the set test parameters. The control of the stepper motor and DC motor are implemented using class *DC_Engine* and Micro*Stepper* in the original program [6]. The following code block shows the process.

Code block 4.1: Implementation of function launch_test for motor control of the bending test setup

```
1.   void launchTest() {
2.    stepper->setSpeed(testParameter.stepperSpeed);
3.    for (long i = 0; i < testParameter.cycles; ++i)
4.    {
5.      for(int j = 0; j < 3200; ++j)
6.      {
7.        do {
8.          if(Serial.available()) {
9.              MsTimer2::stop();
10.             receiveCommand();
```

```
            MsTimer2::start();
          }
          if(progressStatus == STOPPED) return;
        }
        while(progressStatus != STARTED);

        switch (testParameter.bendingDirection) {
          case BENDING_DIRECTION_POSITIVE:
            if (j <= 1600 && i == 0) dc_engine->setSpeed(-cos(PI/3200*j)*20);
            else dc_engine->setSpeed(0);
            break;

          case BENDING_DIRECTION_NEGATIVE:
            if (j <= 1600 && i == 0) dc_engine->setSpeed(cos(PI/3200*j)*100);
            else dc_engine->setSpeed(0);
            break;

          case BENDING_DIRECTION_DOUBLE:
            if (j <= 1600 && i%2 == 0) dc_engine->setSpeed(-cos(PI/3200*j) * 20);
            else if (j > 1600 && i%2 == 0) dc_engine->setSpeed(-cos(PI/3200*j)*50);
            else if (j <= 1600 && i%2 != 0) dc_engine->setSpeed(cos(PI/3200*j)*100);
            else if (j > 1600 && i%2 != 0) dc_engine->setSpeed(cos(PI/3200*j) * 20);
            break;
        }
        stepper->takeSteps(1);
      }
    }

}
```

Listen to the progress control telegram

Before controlling the motor, the program first checks the UART input buffer to listen to the progress control telegram. If the parsed command is a stop command, the program exits the function and stop the test. If the command is a pause-command, the program wait using a while loop until a start-command is received.

Stepper motor control

The program consists of two *for* loops. The external loop controls the number of bend cycles, and the internal loop controls the step of the stepper in a single bend cycle. to complete a motion cycle, the stepper motor needs to take 3200 microsteps. Thus, the cycle times of the internal loop is set to 3200, and for each execution of the internal loop, the program uses fucntion *takeStep(1)* to drive the stepper motor to take one step. The speed of the stepper motor is implemented by the interval between each step, which is achieved by a delay operation inside class *MicroStepper*.

DC motor control

In the process of the control of the stepper motor for sample bending, it is also necessary to control the control DC motor to change the bending direction.

The DC motor is controlled by function *setSpeed* of class *DC_Engine*. The function has one argument, which is ranged from -127 to 128, indicating the speed level of the motor. The sign of the argument indicates the direction of the DC motor. When it is positive, the DC motor turns counterclockwise. When it is negative, the DC motor turns clockwise. The absolute value of the argument indicates the rotation speed of the motor. The larger the absolute value, the faster the DC motor rotate.

As shown in Code block 4.1, when the bending direction is set to positive (case BENDING _DIRECTION_POSITIVE) or negative (case BENDING_DIRECTION_NEGATIVE), the program drives the DC motor in the first cycle to force the metal sheet to bend in the corresponding direction. In the subsequent cycles, the sheet maintains the previous bending direction under the effect of residual deformation. Therefore, the motion of the DC motor is not necessary, and its speed is set to 0.

To avoid the vibrations caused by sudden starts and stops of the DC motor， a speed function is used to smoothly adjust the DC motor speed.

formula 4-3 shows speed curve function of the DC motor in the case of positive bending

$$\begin{cases} y = -a \cdot \cos\left(\frac{\pi}{3200} j\right), when\ j < 1600 \\ y = 0, when\ j \geq 1600 \end{cases} \quad \text{formula 4-3}$$

formula 4-4 shows speed curve function of the DC motor in the case of negative bending

$$\begin{cases} y = b \cdot \cos\left(\frac{\pi}{3200} j\right), when\ j < 1600 \\ y = 0, when\ j \geq 1600 \end{cases} \quad \text{formula 4-4}$$

where j is the steps number of the stepper motor, y is the speed level of the DC motor, a and b is the correction coefficients, a, b $\geqslant$ 0. It can be determined experimentally that when the correction coefficient a = 20 and b = 100, the metal sheet can bend in the desired direction steadily at test start. Figure 4.32 shows the speed curve of the DC motor when the above correction coefficients are used.

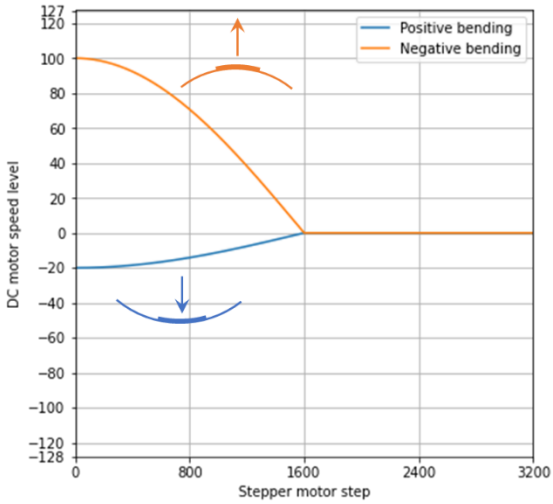


Figure 4.32: The speed curve of the DC motor in the case of single direction bending in relation to the step number of the stepper motor

When the bending direction is set to alternate (case *BENDING_DIRECTION_DOUBLE*), the DC motor rotates continuously during the test. It changes the rotation direction at specific time to force the metal sheet to overcome the residual deformation and change the bending direction when it reaches the critical position.

Like the case of single bending direction, the speed of the DC motor is also dynamically adjusted, to reduce the undesired deformation and mechanical vibration caused by the change of direction

formula 4-5 shows speed curve function of the DC motor in the case of alternate direction

$$\begin{cases} y = -a \cdot \cos\left(\frac{\pi}{3200} j\right), when\ j < 1600\ , i\ is\ even \\ y = -b \cdot \cos\left(\frac{\pi}{3200} j\right), when\ j \geq 1600\ , i\ is\ even \\ y = c \cdot \cos\left(\frac{\pi}{3200} j\right), when\ j < 1600\ , i\ is\ odd \\ y = d \cdot \cos\left(\frac{\pi}{3200} j\right), when\ j \geq 1600\ , i\ is\ odd \end{cases}$$ formula 4-5

where y is the speed level of the DC motor, j is the step number of the stepper motor, i is the cycle number of bending, a, b, c, d ≥ 0 is the correction coefficients. In the mode of alternating bending direction, the first bending direction is always positive. So, when i is an even number, the bending direction is positive and when i is an odd number, the bending direction is negative. It can be determined experimentally that when the correction coefficients a = 20, b = 50, c = 100, d = 20, the mechanical vibration is relatively small, and the metal sheet can pass the critical

position stably. Figure 4.33 shows the speed curve function of the DC motor when the above correction coefficients are used.

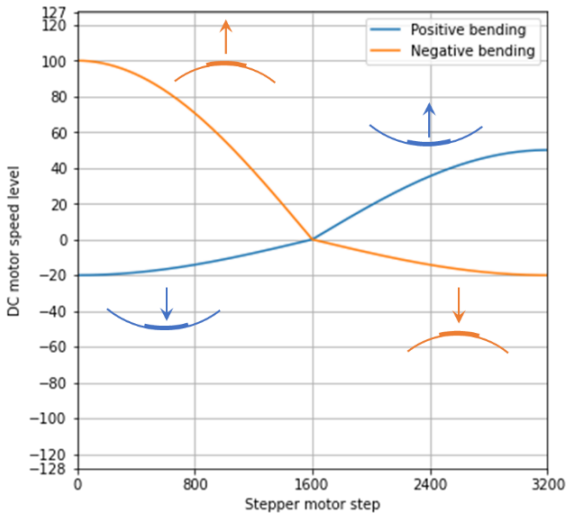


Figure 4.33: The speed curve of the DC motor in the case of alternate direction bending in relation to the step number of the stepper motor

In each bending cycle, when the step number of the stepper motor (j) is within 1600 steps, i.e., during the half cycle when the metal sheet deforms from the flat state to the bent state, the speed of the DC motor becomes smaller and smaller, thus reducing the undesired deformation.

When the stepper motor rotates from 1800 to 3600 steps, i.e., during the half cycle when metal sheet deforms from the bent state to the flat state, the DC motor's speed increases, thus forcing the metal sheet to pass the critical position.

4.5.3.5 Send step to PSoC

The PSoC collects the steps data from the Arduino via I2C. In the I2C communication, PSoC is the master, and the Arduino is the slave. On the Arduino side, the program needs to send a response back to the PSoC after receiving a request from it. It is implemented by the I2C interrupt routine using *Wire* class of the Arduino library.

4.5.4 PSoC program

The PSoC program is responsible for measuring the sensor signal and sending the measurement data back to the PC via the serial port. Figure 4.34 shows the flow chart of the program.

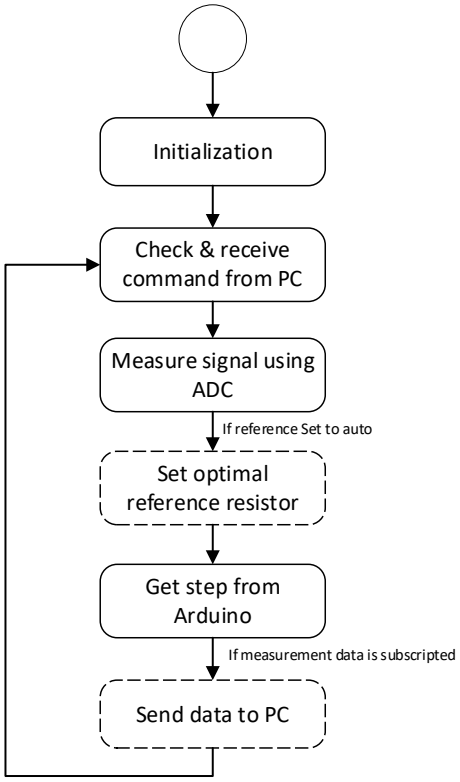


Figure 4.34: Flow chart of the PSoC program of the bending test setup

4.5.4.1 Initialization

After the PSoC is powered on, the program first initializes the hardware, including enabling the interrupt, initializing and starting the operational amplifiers, the multiplexers, the ADC, ,the UART component, the I2C component, etc.

4.5.4.2 Check & receive command from PC

After initialization is complete, the program enters an infinite loop. At the beginning of the loop, the program checks the UART buffer to check if the PC sends command to PSoC. If so, the command is received and parsed it according to the communication protocol. If not, the rest of the program is continued.

4.5.4.3 Read ADC

Next, the program controls the ADC to sample the sensor signal.

The code block shows measurement function *readADC*. The program switches the channels of *AMux_1* and *AMux_2* to measure the voltage of all resistors.

First, *AMux_1* and *AMux_2* both select channel 0. According to Table 4.2, the voltage of *R_lateral* is measurable. After the channel switching is completed, a delay of 100 µs is set to wait for the input voltage of the ADC to be stable to reduce the influence of the switching operation of the multiplexers on the measurement results.

The ADC conversion is then started using the API function *ADC_StartConvert*. The program waits in a loop until API function *ADC_IsEndConversion* returns true value, and the conversion is stopped with API function *ADC_StopConvert*.

Using API function *ADC_CountsTo_Volts*, the by ADC measured count value is convert to voltage values according to the reference voltage and result format. The measurement result is stored in variable *v_lateral*.

Next, the same operation is repeated for the other signals. Resistors' voltage *v_lateral*, *v_longitudianl* and *v_reference* can be obtained.

Finally, the resistances can be calculated according to formula 4-1 and formula 4-2 and stored in variable *R_lateral* and *R_longitudianl*.

Code block 4.2: Implementation of function read_ADC for signal measurement of the bending test setup

```
1.  void readADC(void)
2.  {
3.
4.      AMux_1_FastSelect(0);
5.      AMux_2_FastSelect(0);
6.      CyDelayUs(100);
7.      ADC_StartConvert();
8.      while (!ADC_IsEndConversion(CY_SAR_WAIT_FOR_RESULT))
9.          ;
10.     ADC_StopConvert();
11.     v_lateral = ADC_CountsTo_Volts(0, ADC_GetResult16(0));
12.
13.     AMux_1_FastSelect(1);
14.     AMux_2_FastSelect(0);
15.     CyDelayUs(100);
16.     ADC_StartConvert();
17.     while (!ADC_IsEndConversion(CY_SAR_WAIT_FOR_RESULT))
18.         ;
19.     ADC_StopConvert();
20.     v_longitudinal = -ADC_CountsTo_Volts(0, ADC_GetResult16(0));
21.
22.     AMux_1_FastSelect(0);
23.     AMux_2_FastSelect(1);
24.     CyDelayUs(100);
25.     ADC_StartConvert();
26.     while (!ADC_IsEndConversion(CY_SAR_WAIT_FOR_RESULT))
27.         ;
28.     ADC_StopConvert();
```

```
    v_ref = -ADC_CountsTo_Volts(0, ADC_GetResult16(0));

    // current = v_ref / (12.97e3 + 400);  //12.97, 558.9
    float current = v_ref / (referenceValue[referenceChannel] + 450);
    R_longitudinal = v_longitudinal / current;
    R_lateral = v_lateral / current;
}
```

4.5.4.4 Set optimal reference

If parameter reference is set to auto. The program self-adaptively selects the optimal reference resistor according to the voltage of the reference resistor, when the difference between the resistance of the measured resistor and the resistor is too large.

Code block 4.3 shows the implementation of the functions to select the best reference. Variable *v_ref* stores the voltage of the reference resistor measured by the ADC. Variable *referenceChannel* stores the currently selected channel of *AMux_Rref*. When the voltage of the currently selected reference is less than 0.3V, the program changes the channel of *AMux_Rref* to select the next larger reference, resulting the increase of *v_ref*. When the voltage of the currently selected reference is larger than 1.65V, the program changes the channel of *AMux_Rref* to select the last smaller reference, resulting the decrease of *v_ref*. This function is executed after each ADC measurement. By trying a larger or smaller reference resistance step by step, the voltage of the reference is quickly adjusted to the expected range.

Code block 4.3: Implementation of function setOptimalReference for automatic reference selection of the bending test setup

```
void setOptimalReference(void)
{
    if (v_ref < 0.3)
    {
        referenceChannel = (referenceChannel + 1) % (sizeof(referenceValue) / sizeof(referenceValue[0]));
        AMux_Rref_FastSelect(referenceChannel);
    }
    else if (v_ref > 1.65) {
        referenceChannel = (referenceChannel -1) % (sizeof(referenceValue) / sizeof(referenceValue[0]));
        AMux_Rref_FastSelect(referenceChannel);
    }
}
```

4.5.4.5 Get step from Arduino

After measuring the analog signal using ADC, the program collects motor step number from the Arduino via I2C to indicate the progress of the bending.

In I2C communication, the PSoC is the master, and the Arduino is the slave. On the master side, the PSoC program needs to send requests to the Arduino and wait for the Arduino to respond. This process is implemented through the API function of the I2C component provided by the PSoC.

4.5.4.6 Send data to PC

Finally, if the PC sends the data subscription request in the second step, the PSoC sends all the collected data to the PC via UART. The implementation of the sending is very simple, people only need map the I/O functions of the C standard library to UART component, then data can be sent to PC by calling function *printf*.

5 Measurement characteristics of stretching test setup

5.1 Overview

The measurement characteristics of the stretching test setup include the characteristics of four-point measurement, force measurement and position measurement. For four-point measurement, the characteristics can be divided into static and dynamic characteristics. Static characteristics include the measurement resolution of the resistance, the measurable resistance range, and the measurement accuracy. Dynamic characteristics includes the response speed of the measurement system. For force measurement and position measurement, since they are not urgently needed in this work. Only the force measurement is briefly evaluated in terms of measuring range and amplification circuit. The evaluation of the position sensor will be done in the future. Table 5.1 provides an overview of all measurement characteristics of the setup.

Table 5.1: Overview of measurement characteristics of the stretching test setup

Four-point measurement	Resolution	Maximum 0.133 kΩ
	Measurement Range	0 – 2868 kΩ
	Accuracy	≥ 95% (1 kΩ – 2.7 GΩ)
	Dynamic response	≤ 5 ms delay for step response
Force measurement (Not calibrated)	Measurement Range	-3.3 / Gain ≤ V_{in} ≤ 3.3 / Gain
	Gain	Channel 0: 367.2 Channel 1: 206.3 Channel 2: 111.6
Position measurement (Not calibrated)	Measurement Range	0 - 3.3 V

5.2 Evaluation of four-point measurement

5.2.1 Resolution

The resolution of the ADC of the PSoC is 12 bit and its reference voltage $Vref$ is configured as 3.3 V, so the least Significant Bit LSB of the ADC can be calculated by the following formula:

$$LSB = \frac{2Vref}{2^N} = \frac{6.6V}{2^{12}} = 0.0016113\ V \qquad \text{formula 5-1}$$

According to the description in Table 3.3, the maximum output of the current source is 10 μA (12.12 μA in fact). The minimum resistance change is

$$dR = \frac{LSB}{I_{SET}} = \frac{0.00161V}{12.12\ \mu A} = 0.133\ k\Omega \qquad \text{formula 5-2}$$

So, the maximum resolution of the four-point measurement is 0.133 kΩ.

5.2.2 Measurement Range

The minimum measurable resistance for four-point measurement is 0 kΩ and the maximum measurable resistance depends on the full scale range (FSR) of the ADC. With a 3.3 V reference voltage and in the case of differential input, the ADC has a FSR of -3.29842 V to 3.29842 V. So, the maximum resistance that can be measured by the ADC is

$$R_{max} = \frac{V_{max}}{I_{SET}} = \frac{3.29842\ V}{1.15\ \mu A} = 2868\ k\Omega \qquad \text{formula 5-3}$$

Thus, the effective measurement range of the four-point measurement is from 0 to 2868 kΩ.

Figure 5.1 shows the relationship between the true resistance value of the resistor and the measured value. When the resistance is within the measurement range, four-point measurement can measure the resistance of the resistor. When the measured resistance is greater than 2998 kΩ, the measurement result is constant and invalid.

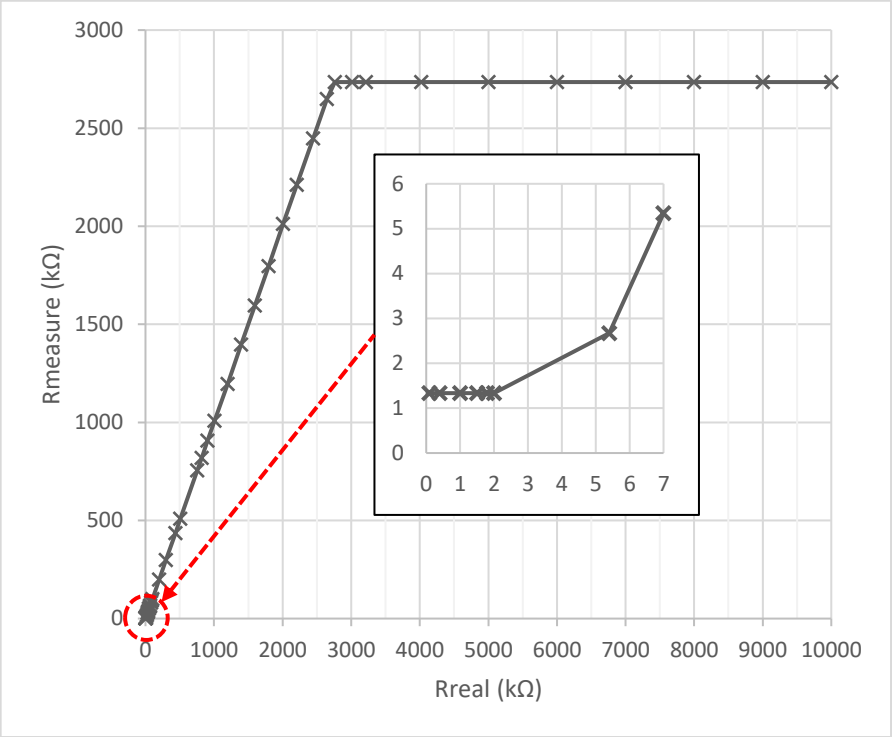


Figure 5.1: Characteristics of the measurement result in relation to the real resistance

5.2.3 Accuracy

Figure 5.2 shows the relative measurement error with respect to the resistance in logarithmic scale. In the case of that the output current of the current source is 1 µA, the average error is high when the resistance is between 0 and 5 kΩ, with a constant measurement result of 1.34 kΩ. when the resistance is in the range of 7 kΩ to 100 kΩ, the accuracy rises gradually to +1%. When the resistance is in the range of 100 kΩ to 2700kΩ, the accuracy is very high, approximately from -0.8% to 0.3%. When the resistance exceeds 2700 kΩ, the error increases, because the resistance reaches the maximum measurement resistance of the system. Therefore, the optimal measurement range for four-point measurement is between 100 kΩ and 2700 kΩ with a static accuracy of more than 99%. In the case of that the output current of the current source is 10 µA, the accuracy is much higher than the current of 1 µA in the range of 1 kΩ to 100 kΩ, which is over 95%.

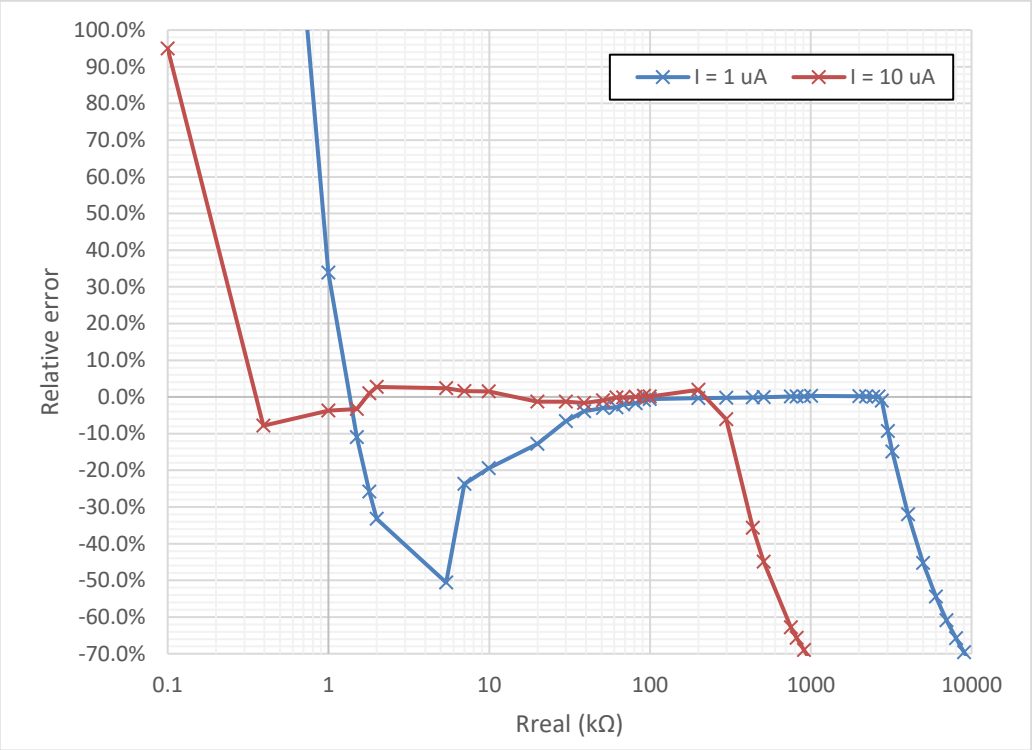


Figure 5.2: Relative error of the resistance measurement result in relation to the sample resistance in logarithmic scale

5.2.4 Dynamic response

Figure 5.3 shows the step response of the four-point measurement. The resistance changes abruptly from 0 to 1 MΩ, which is achieved through a decade resistor box. The measurement is taken at a sampling frequency of 200 Hz. The measured resistance changes from 0 to 1 MΩ in one sampling period, i.e., 5 ms, after the resistance is changed. With a higher sampling frequency, the response time could be even shorter. For stretching test, the interval between each step of the stepper motor is typically above 10 ms at normal stretch speed. Therefore, a response time of at least 5 ms for a four-point measurement completely meet the required dynamic performance.

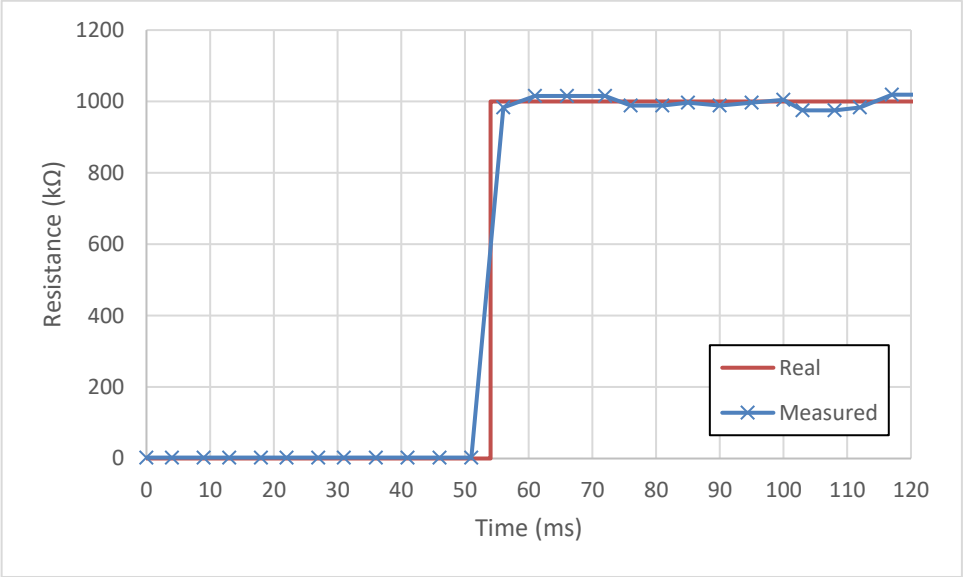


Figure 5.3: Step response of the measurement system

5.3 Evaluation of force measurement

Since the force measurement is not urgently needed in this work. Therefore, the evaluation and calibration of the force measurement will be carried out in the future. For now, only the input range and the gain of the force measurement circuit is characterized.

5.3.1 Measurement range

For the pre-amplifier circuit in the force measurement, the input signal cannot be amplified infinitely. When the input voltage reaches a certain value, it cannot be amplified by the gain. Because the output voltage of the operational amplifier is limited. According to the data sheet, the output range of the two operational amplifiers is approximately 0.2 V to Vdda - 0.2 V. The actual range can be 0-Vdda. In addition, the amplified voltage must be in the effective measurement range of the ADC's differential input, i.e. - 3.3 to 3.3 V.

Taking $Vdda = 3.3\,V$, $R1 = R2 = R$, referring to formula 3-5, the input voltage must satisfy the following conditions:

$$\begin{cases} 0 \le V_+ + \dfrac{V_+ - V_-}{Rg} R \le 3.3\,V \\ 0 \le V_- - \dfrac{V_+ - V_-}{Rg} R \le 3.3\,V \\ -3.3\,V \le (V_+ - V_-) \cdot Gain \le 3.3\,V \end{cases}$$ formula 5-4

where Gain is the gain of the pre-amplifier circuit. $Gain = 2R/Rg + 1$ (see formula 3-6) The range of input voltage can be solved by linear programming.

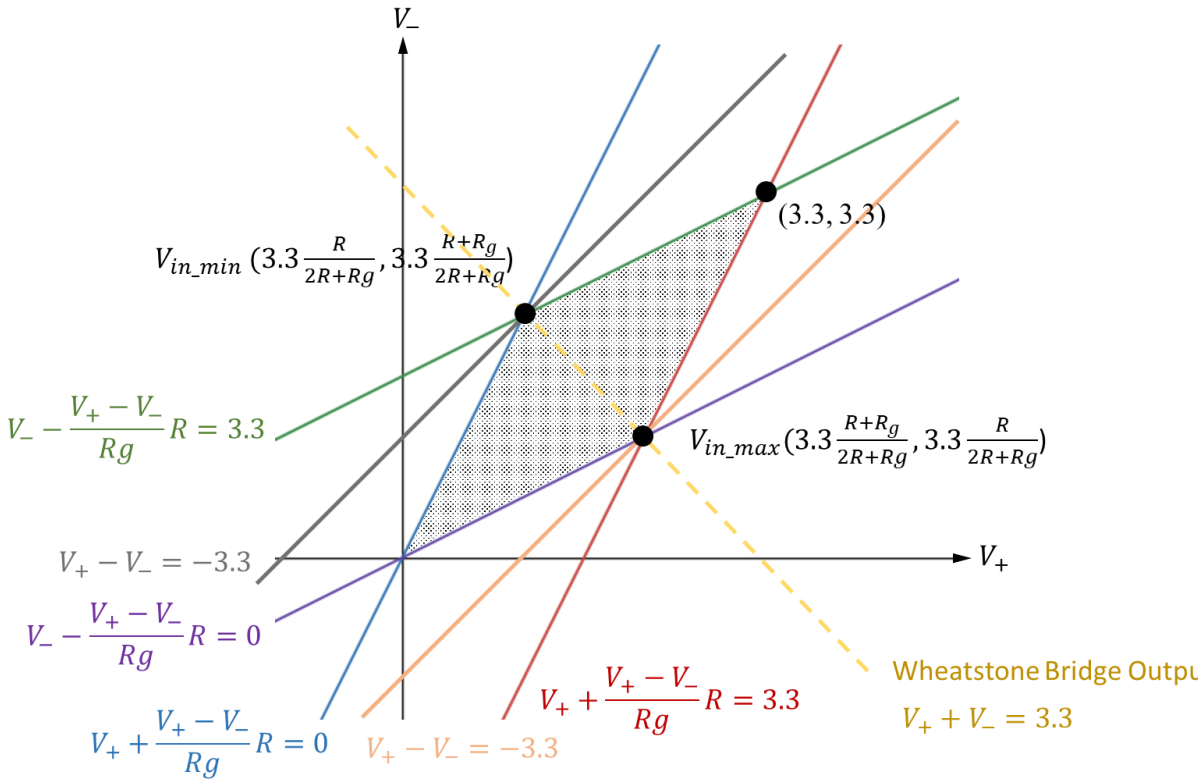


Figure 5.4: Feasible domain of the input voltage of the pre-amplifier in amplification mode for force measurement

As shown in Figure 5.4, the shaded area indicates the feasible domain of the input voltage. The input voltage at the point V_{in_min} is taken as the minimum and at the point V_{in_max} as the maximum. So, the range of input voltage $V_{in} = V_+ - V_-$ is

$$-\frac{3.3R_g}{2R + R_g} \leq V_{in} \leq \frac{3.3R_g}{2R + R_g} \qquad \text{formula 5-5}$$

When the gain is large, the above range approximates to

$$-\frac{3.3}{Gain} \leq V_{in} \leq \frac{3.3}{Gain} \qquad \text{formula 5-6}$$

The dashed line in Figure 5.4 shows the output voltage of the Wheatstone bridge of the force sensor. The output voltage of the Wheatstone bridge lies in the feasible domain from the start of the equilibrium state until the maximum input voltage is reached. So, the pre-amplifier is perfectly suited for the amplification of force sensor signals. However, for the amplification of

other signals, which is not from the Wheatstone bridge, unexpected amplification results may occur, since the input voltage V_+ and V_- are likely to be outside the feasible domain.

5.3.2 Gain

In the previous section, it is mentioned that the actual gain may be slightly different from the theoretical calculation due to the PSoC internal route and switch resistance. In this section, the actual gain of the pre-amplifier circuit is evaluated.

The actual gain is measured experimentally. A Wheatstone bridge circuit consisting of a potentiometer and three 1 MΩ standard resistors is built. The actual gain is calculated by the averaging of the gain in the case of different potentiometer resistance.

Table 5.2 shows the evaluation results of the pre-amplifier circuit. The last two columns are the theoretical gain and the actual gain of different reference channels.

Table 5.2: Evaluation results of the gain of the pre-amplifier circuit in amplification mode

Channel/Gain	R1	R2	Rg	Rg value	Gain (theoretical)	Gain (Experiment)
0	220 kΩ	220 kΩ	Rg_0	1.1 kΩ	400	367.2
1			Rg_1	2.2 kΩ	200	206.3
2			Rg_2	4.4 kΩ	100	111.6

By comparing the theoretical gain and the actual gain, it can be found that the error of gain is larger when the gain is larger. In practice, a moderate deviation of gain is irrelevant. Because the influence of the error can be eliminated by experimentally calibrating the force to the measured voltage.

Figure 5.5 shows the relationship between the input voltage and the amplified voltage in the case of different gains. It can be found that the pre-amplifier circuit has good linear amplification capability. When the input voltage exceeds a certain value, the amplification no longer remains linear, and the output voltage saturates because the input voltage is out of the feasible domain.

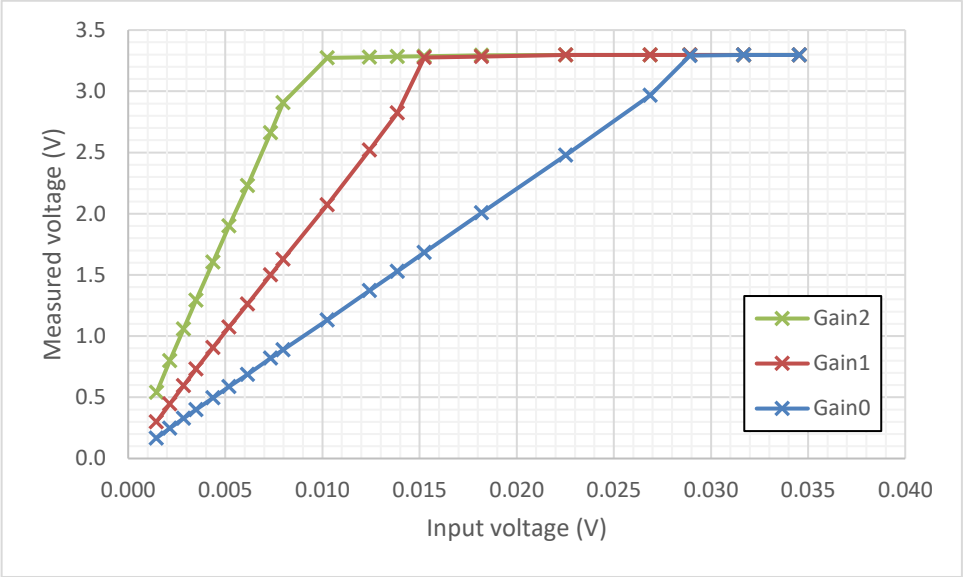


Figure 5.5: Input-output characteristics of the pre-amplifier in amplification mode

6 Measurement characteristics of bending test setup

6.1 Overview

The measurement characteristic of the stretching test setup only includes the characteristics of the resistance measurement. The characteristics can be divided into static and dynamic characteristics. Static characteristics include the measurement resolution of the resistance, the measurable resistance range, and the measurement accuracy. Dynamic characteristics include the response speed of the measurement system and the coupling of different measurement channels. Table 6.1 provides an overview of all measurement characteristics of the bending test setup.

Table 6.1: Overview of measurement characteristics of the bending test setup

Resistance measurement	Resolution	Approximate 0.16% of the reference resistance
	Measurement Range	0-2.47 GΩ (for current reference configuration)
	Accuracy	≥ 99% (R_longitudinal: R_lateral = 1:1) ≥ 95% (R_longitudinal: R_lateral = 1:9 to 9:1) < 95% (other)
	Dynamic response	≤ 5 ms delay for step response
	Dynamic coupling	No coupling problem

6.2 Resolution

The resolution of the resistance measurement is not a constant but is related to the current resistance value of the longitudinal and lateral resistor and the reference resistance. The smaller the reference resistance, the higher the resolution of the measurement. Typically, the resolution can be up to 0.16% of the reference resistance. For example, at a reference resistance of 10 kΩ, the resolution can be approximately 16 Ω.

6.3 Measurement range

The minimum measurable resistance is 0 Ω. The maximum measurable resistance depends on the maximum reference resistance. Assume that the maximum reference is R_{ref_max}. When the maximum measurable resistance is reached, the voltage of the reference should be 0.00161 V, which is the minimum voltage recognizable by the ADC. Since the voltage of the excitation source is 3.3 V, the voltage of the measured resistance should be 3.29842 V, which is the maximum voltage recognizable by the ADC.

So, the maximum measurable resistance R_{max} is

$$R_{max} = \frac{3.29842\ V}{0.0016113\ V} R_{ref\_max} = 2047 R_{ref\_max} \qquad \text{formula 6-1}$$

In summary, the measurement range of the bending test setup is 0 to 2047Rref_max. In this work, R_{ref_max} is 1 MΩ, So, the measurement range is 0 to 2.47 GΩ.

6.4 Accuracy

Since the setup measures the resistance of longitudinal and lateral resistors parallel, the total resistance of the two resistors and the difference between them are both factors that affect the accuracy. So, the measurement accuracy is measured from the above two aspects.

First, to evaluate the effect of total resistance on the accuracy, the following evaluation method is used: keeping the resistances of the longitudinal and lateral resistors, which are simulated by standard resistors, the same and changing their total resistance, their individual resistances is measured, and the relative errors are calculated to measure the accuracy.

Figure 6.1 shows the evaluation results, indicating the relative error of each measured resistance in relation to their resistance in logarithmic scale. It can be found that when the total resistance is smaller than 30 kΩ, the measurement error is relative larger, which is above 1%. When the total resistance is greater than 30 kΩ, the relative error stays within $\pm 1\%$.

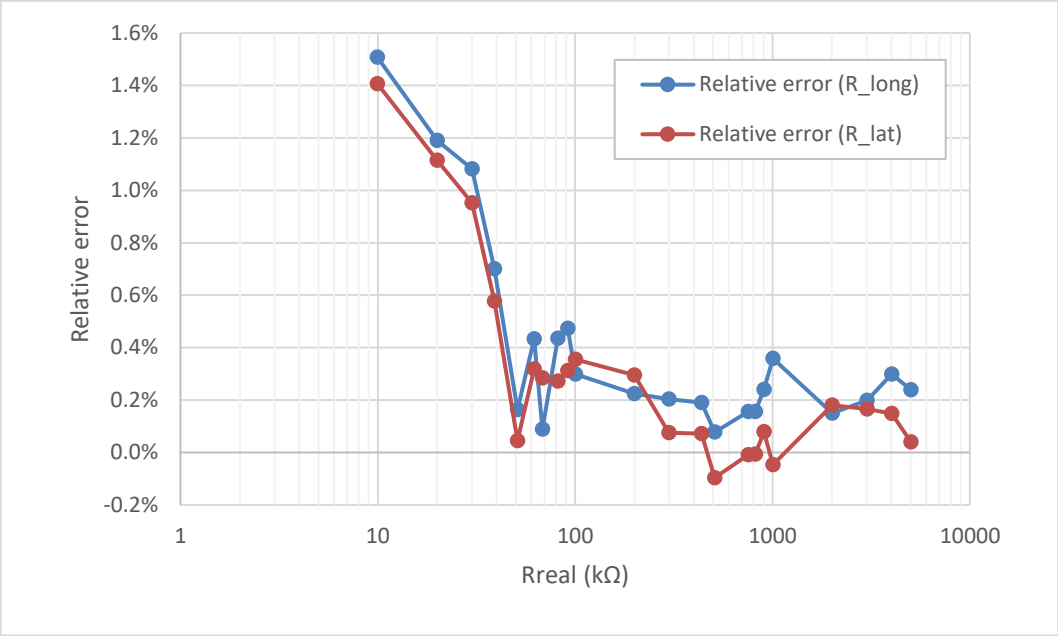


Figure 6.1: Relative error of the resistance measurement result in relation to the sample resistance in logarithmic scale (R_long : R_lat = 1 : 1)

Then, to evaluate the influence of the ratio between the longitudinal and lateral resistors on the measurement accuracy, the following accuracy evaluation method is used: keeping the total resistances in both directions constant, changing the ratio of the longitudinal and lateral resistance, which is simulated by a potentiometer, the relative error is calculated to measure the accuracy.

Figure 6.2 shows the relationship between the ratio of the resistances in two directions and the relative error when the total resistance is kept at 1909 kΩ. The ratio is expressed as the ratio of the longitudinal resistance to the total resistance.

It can be found that for the resistor with a larger percentage its measurement accuracy is higher. And for the resistor with a smaller percentage the accuracy is lower. When the percentage of the longitudinal resistance is from 0.25 to 0.75 i.e., the resistance ratio is from 1/3 to 3, the measurement results of both directions have high accuracy (Both relative errors are within 2%).

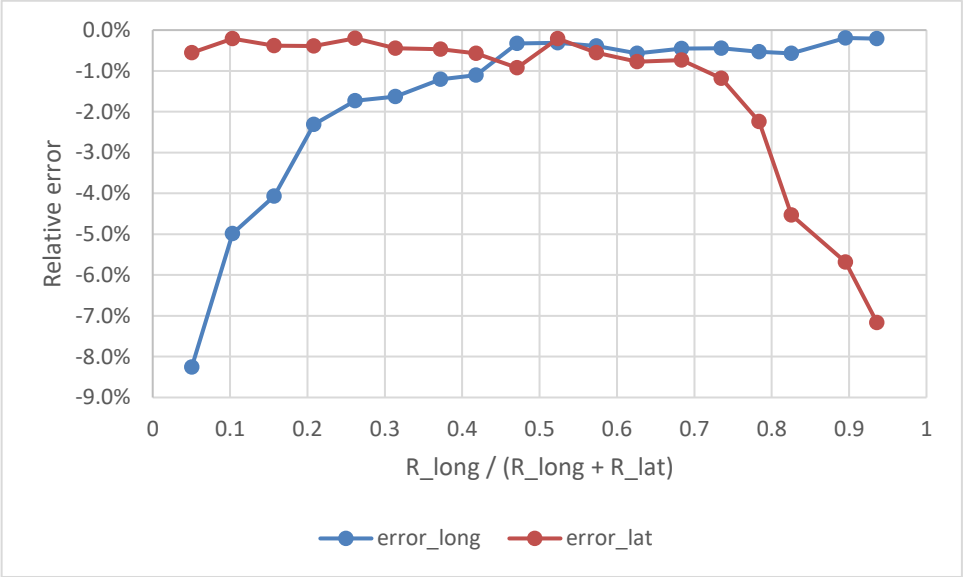
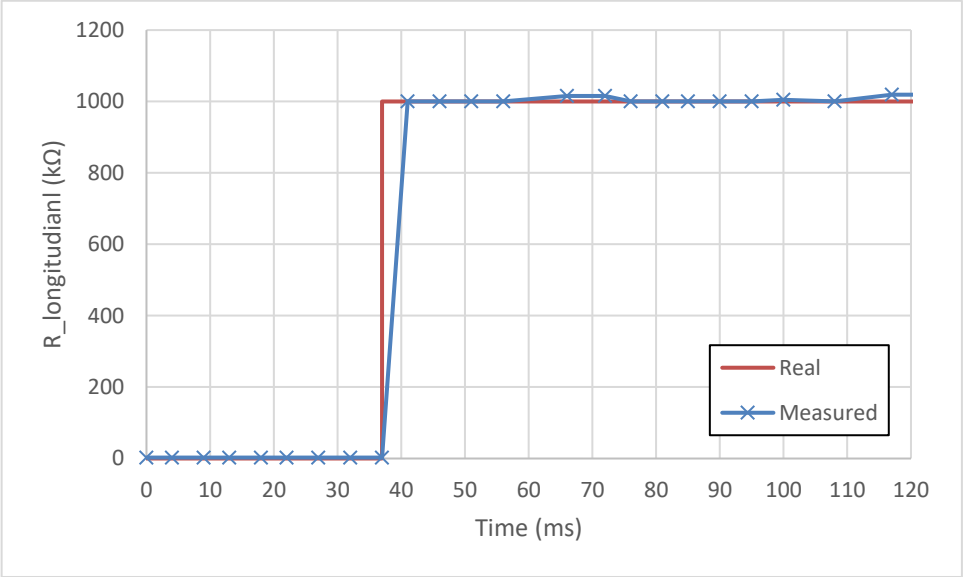


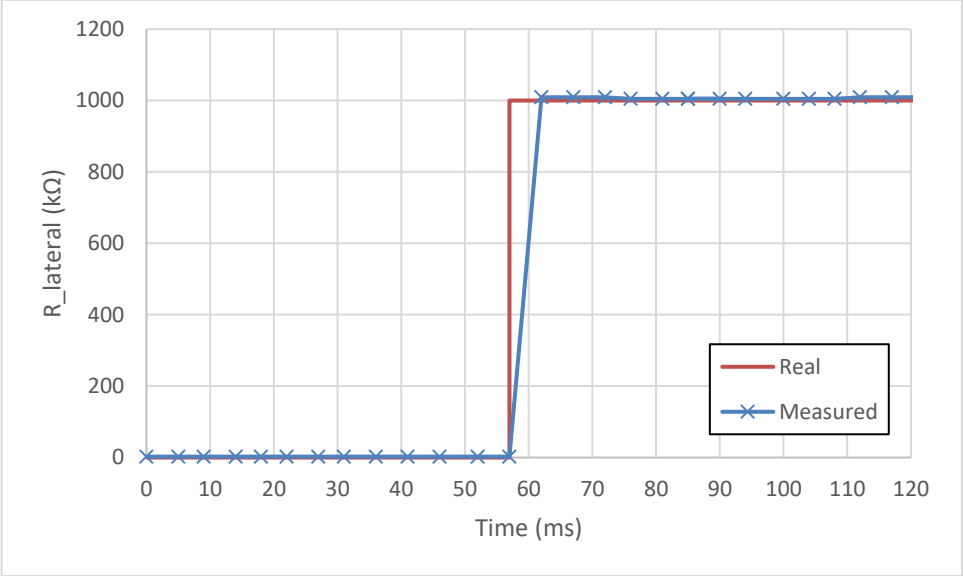
Figure 6.2: Relative error of the resistance measurement result in relation to the ratio of the longitudinal and lateral resistance

6.5 Dynamic response

Figure 6.3 shows the step response of the longitudinal and lateral measurement channels. The resistance changes abruptly from 0 to 1 MΩ, which is achieved through a decade resistor box. When one of the channels is evaluated, the other one is shorted with a wire. The measurement is taken at a sampling rate of 200 Hz. For both measurement channels, the measured resistance changes from 0 to 1 MΩ in one sampling period, i.e., 5 ms, after the resistance is changed. With a higher sampling frequency, the response time could be even shorter.



(a)



(b)

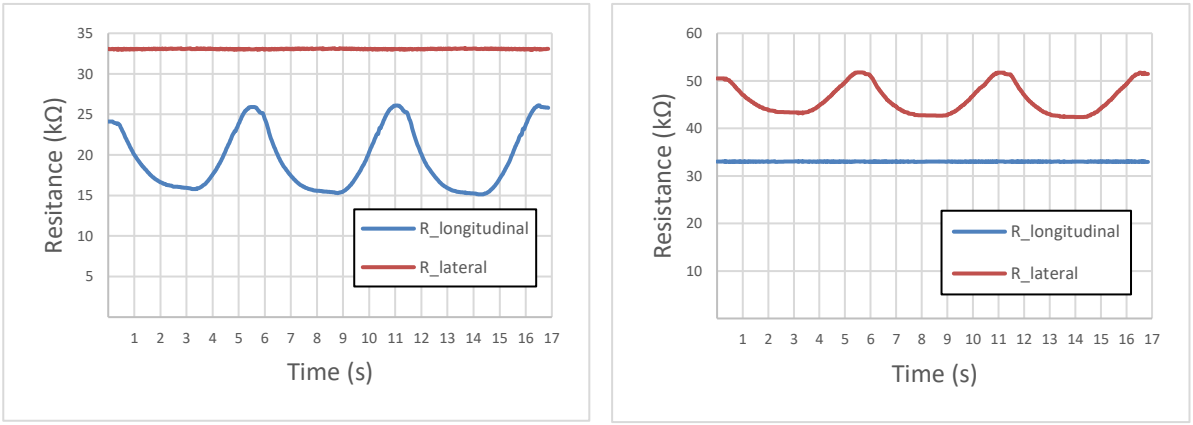
Figure 6.3: Step response of the longitudinal (a) and lateral (b) resistance of the measurement system

6.6 Dynamic coupling

Since the resistances in both directions are measured alternately using multiplexers, it is possible that the measurement of two resistances coupled to each other, which means that a change of one resistor affects the measured value of the other. Therefore, it is necessary to experimentally determine whether the measurements of the two resistances are independent of each other.

The following method is used for the evaluation: keeping one of the two resistance constant and changing the other one, which is simulated by a standard resistor and a potentiometer, measurements are made on both resistances. If the measurement of the standard resistor does not change, it is not coupled to the other resistor.

A 33 kΩ standard resistance is used as the unchanged resistance. Figure 6.4 shows the evaluation results. Figure 6.4(a) shows the case that R_longitudinal changes and R_lateral is fixed. Figure 6.4(b) shows the opposite case to Figure 6.4(a). The results show that while one of the resistances in both directions is changing, the measurement of the other one remains constant. Therefore, the measurements of longitudinal and lateral resistance are independent of each other.



(a) (b)

Figure 6.4: Evaluation result of the dynamic coupling of different measurement channels. (a): Longitudinal resistance changes and lateral resistance is constant. (b): Lateral resistance changes and lateral longitudinal is constant

7 Sample test and evaluation

In this chapter the electrical characteristics of the samples is evaluated using finished stretching and bending test setup. In Roman Vulikh's work "Charakterisierung des Siebdruckprozesses zur Herstellung dehnbarer Sensoren", a sensor mask with flexible sensor structure and varies samples is designs and manufactured for screen printing (see Figure 7.1). With the sensor mask, number of test samples are manufactured using screen printing. These samples vary in multiple dimensions. In terms of process, the variable is blade type and bake-out process. In terms of geometric parameters, the variables are line width and line form. In terms of material the variable is Carbon paste and Silber paste.

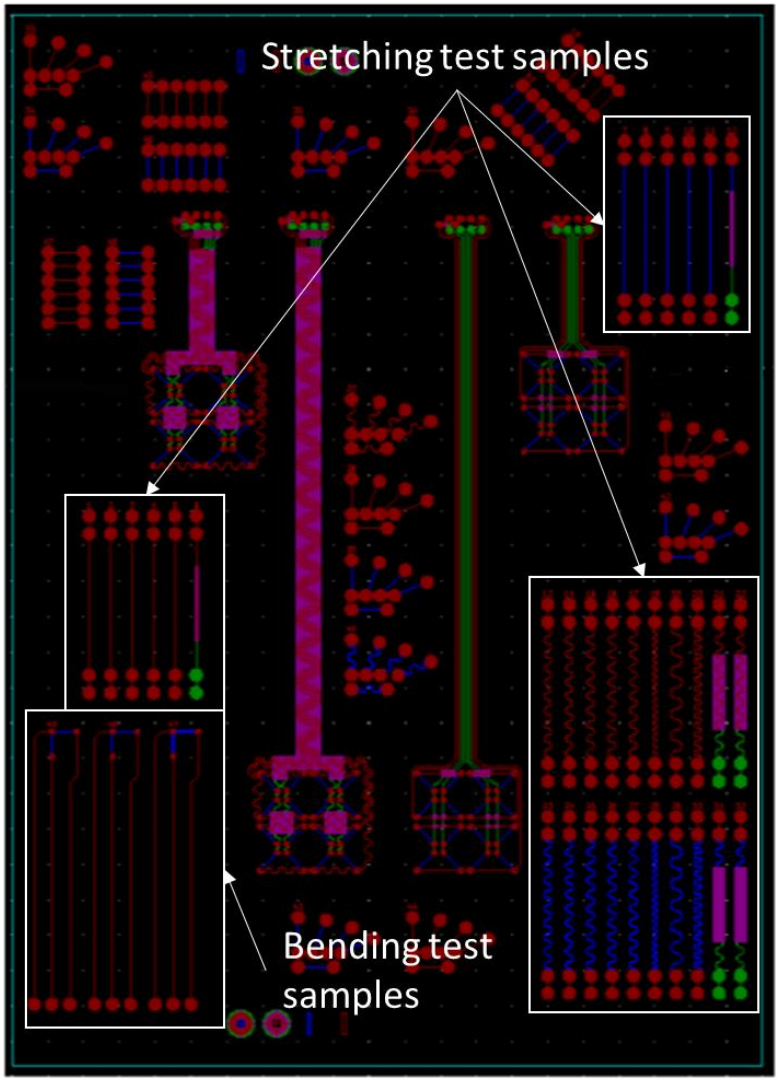


Figure 7.1: CAD sketch of the sensor mask for screen printing with stretching and bending test samples [2]

7.1 Stretching test

7.1.1 Sample

In the stretching test, the sample, whose code is RTI4O-30 is selected as the test sample. Table 7.1 shows its detailed parameters. This sample is structure #30 on the sensor mask. The manufacturing code is RTI4O, which means blade Triflex 2 and bake-out process is used for screen printing. The tested object is made of Carbon paste and the substrate is made of Polyurethan. The form of the tested object is a straight line with the linewidth of 500um. The initial resistance of the resistor is 400 kΩ.

Table 7.1: Specification of the sample for stretching test

Sample code	Manufacturing parameters		Material		Geometry		Initial Resistance
RTI4O-30	Blade	Bake-out	Tested material	Substrate	Form	Line width	400 kΩ
	Triflex 2	Yes	Carbon paste	Polyurethan	Straight Line	500 μm	

7.1.2 Test parameter

Table 7.2 shows the test parameters used for the stretching tests. The original length of the sample is 40 mm. The maximum strain to which the sample is stretched is 10%. The stretching speed is 0.2 mm/s. One cycle is performed

Table 7.2: Test parameters of the stretching test

Test number	Sample length	Maximum strain	Equivalent stretching length	Stretching Speed	Cycles
1	40 mm	10%	4 mm	0.2 mm/s	1

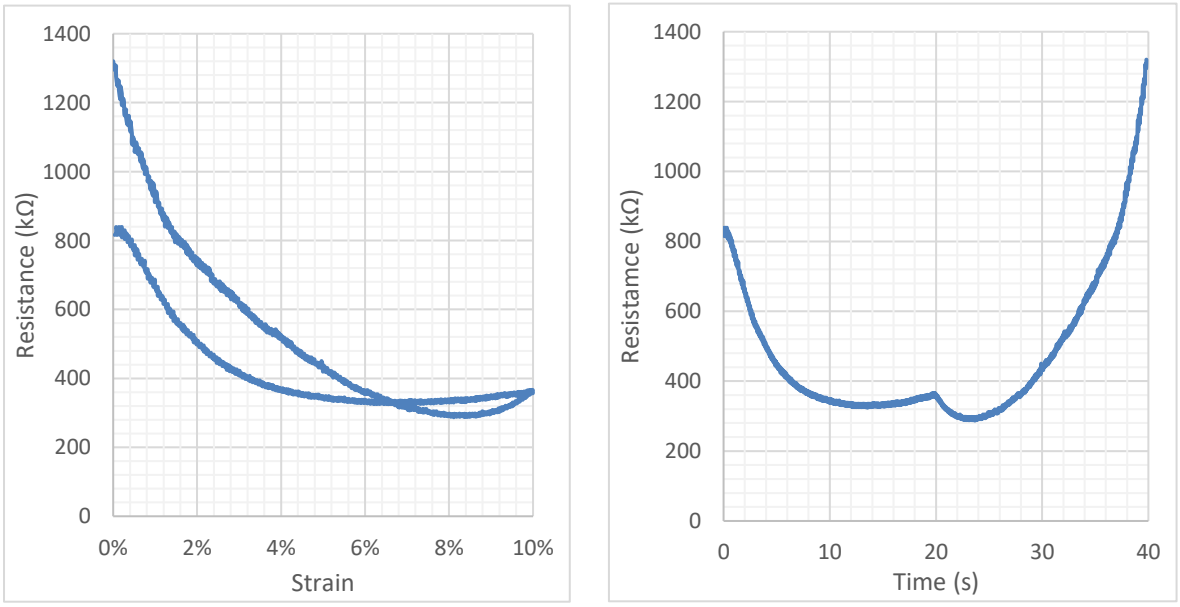
7.1.3 Test Result

Ideally, the desired test results should have the following characteristics: as the sample is stretched to be longer, the resistance of the sample increase linearly with the length of the

stretch. As the sample is stretched from the maximum strain to the initial length, the resistance should return along the original path.

Figure 7.2 shows the test results. In Figure 7.2(a) is the resistance-strain characteristics of the sample with the strain of the sample in the horizontal coordinate and the resistance of the sample in the vertical coordinate. In Figure 7.2(b) is the resistance-time characteristics of the sample. The horizontal coordinate is the time since the start of the test, and the vertical coordinate is the resistance of the sample.

The result shows that the sample resistance is negatively correlated with strain. The resistance of the sample decreases as the sample is stretched from the natural state to the maximum strain and the resistance of the sample increases when the sample returns to the natural state from the maximum strain state. The path of resistance change is not the same for the process of elongation and return. When the length is fully restored to the original length, the resistance does not return to the initial state but is slightly larger than the original resistance.



(a) (b)

Figure 7.2: Test result of the stretching test. (a) Resistance-strain characteristics of the sample (b) Resistance-time characteristics of the sample

Obviously, the actual test results deviate significantly from the expected results. The resistance-deformation characteristics of the sample are exactly opposite to the theoretical characteristics. The sample also shows a significant hysteresis.

To ensure the reliability of the test results, the test is performed again and a RIGOL MSO1104 oscilloscope is used to measure the sample instead of the setup's measurement system. Figure 7.3 shows the test result. The horizontal axis is the time, and the vertical axis is the voltage of the resistance. The voltage is linear with the sample resistance since the excitation current is constant, so the figure can reflect the resistance change of the sample with the time during the test.

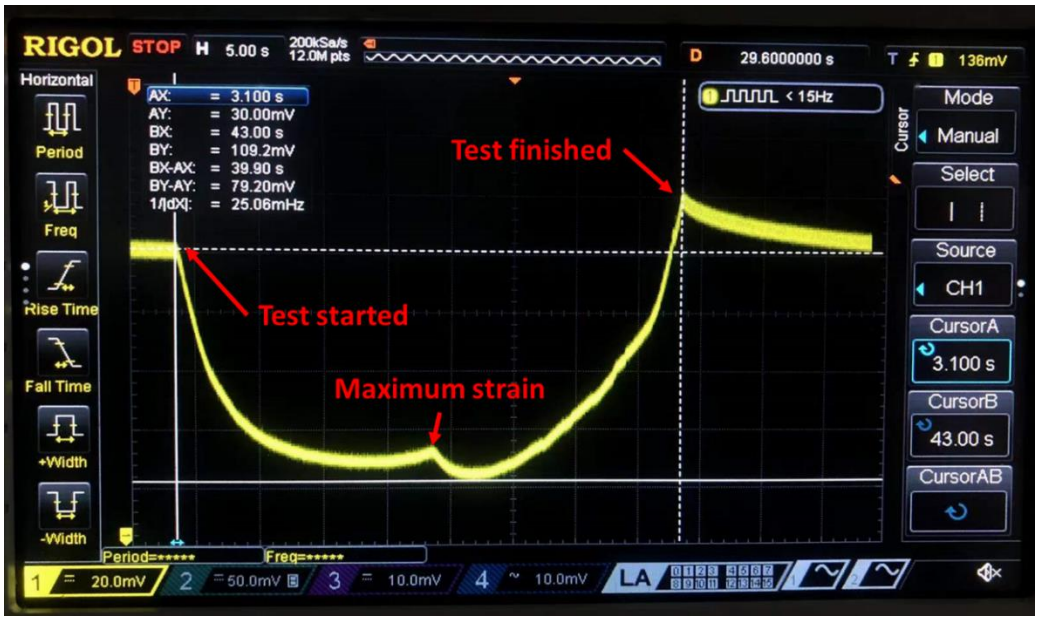


Figure 7.3: Resistance-time characteristics of the sample measured by RIGOL MSO1104 oscilloscope

By comparing the results obtained by the oscilloscope and the setup, it can be found that the resistance-time characteristics is essentially the same in both tests. Therefore, measurement error due to the measurement itself can be excluded from the causes of the unexpected test result.

The test of other samples and further research on the causes of the unexpected characteristics will be carried out in the future, which is outside the scope of this work.

7.2 Bending test

7.2.1 Sample

In the stretching test, the sample, whose code is RTI4O-53 is selected as the test sample. Table 7.3 shows its detailed parameters. This sample is structure #53 on the sensor mask. The

manufacturing code is RTI4O, which means blade Triflex 2 and bake-out process is used for screen printing. The resistor is made of Carbon paste, the conductive tracks are made of Silber paste and the substrate is made of polyurethan. The initial resistance of the resistor is 12.79 kΩ for the longitudinal resistor and 47.36 kΩ for the lateral resistor.

Table 7.3: Specification of the sample for bending test

Sample code	Manufacturing parameters		Material			Geometry	Initial Resistance
RTI4O-53	Blade	Bake-out	Resistor	Track	Substrate	Line width	12.79 kΩ (longitudinal)
	Triflex 2	Yes	Carbon paste	Silber paste	Polyurethan	800 μm	47.36 kΩ (lateral)

7.2.2 Test parameter

Table 7.4 shows the test parameters used for the bending tests. Three tests are performed. The directions of the three tests are positive bending, negative bending and alternate double bending respectively. In all of three rests, the bending speed are 5 r/min for the stepper motor and the number of cycles is 10.

Table 7.4: Test parameters of the bending test

Test Number	Bending direction	Bending speed	Cycles
1	Positive	5 r/min	10
2	Negative		
3	Alternate double		

7.2.3 Test Result

Ideally, the test result is expected to have the following characteristics:

- For the longitudinal resistor, in positive bending, its resistance decreases with the bending radius due to compression and reaches the minimum value at half of the bending cycle, when the bending radius is at its maximum. In negative bending, the

resistance increases with the bending radius due to stretching and reaches the maximum value at half of the bending cycle.

- For the lateral, since it is parallel to the bending axis, its length will not change with bending. So, its resistance remains unchanged.

The following part is the actual measurement results.

Test 1: Positive bending

Figure 7.4 shows the results of the first four cycles of Test 1. The horizontal coordinate is the number of steps of the stepper motor. Every 3200 steps, one bending cycle is completed exactly. The vertical coordinate is the percentage of the current resistance relative to the initial resistance before test. Taking the first bending cycle as an example, it can be found that the longitudinal resistance decreases from 0 to 1600 steps, during which the bending radius increases. And it reaches a minimum at 1600 steps, at which the bending radius is maximum. In the process from 1800 to 3600 steps, the resistance does not fully recover to the initial value but a larger value. For the lateral resistance, it has similar characteristics to the longitudinal resistance, but the rate of change of the resistance is much smaller.

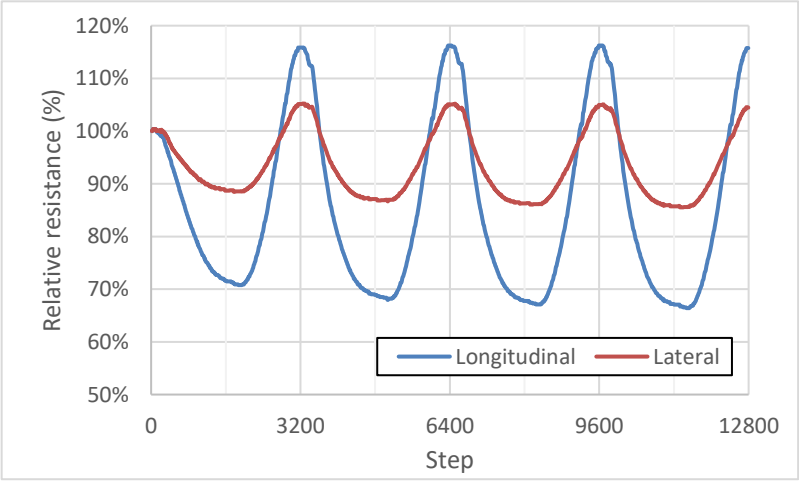


Figure 7.4: Test result of the positive bending: resistance-step characteristics of the sample

The actual test results did not match the expected results in two points: First, after the first bending, the resistance of the natural length has an offset to the initial state, which indicates the hysteresis of the sample. Second, the lateral resistance did not remain the same, but experienced a similar resistance change as did the longitudinal resistance.

Test 2: Negative bending

Figure 7.5 shows the results of the first four cycles of Test 2. The horizontal coordinate is the number of steps of the stepper motor. Every 3200 steps, one bending cycle is completed exactly. Taking the first cycle as an example, in the first half cycle the longitudinal resistance decreases with increasing bending. Then, unlike test 1, the resistance does not recover from 1600 steps due to the decrease of bending radius, but the trend slows down and then continues to decrease. It does not start to recover until about 2200 steps. For the lateral resistance, it has similar characteristics to the longitudinal resistance, but the rate of change of the resistance is much smaller.

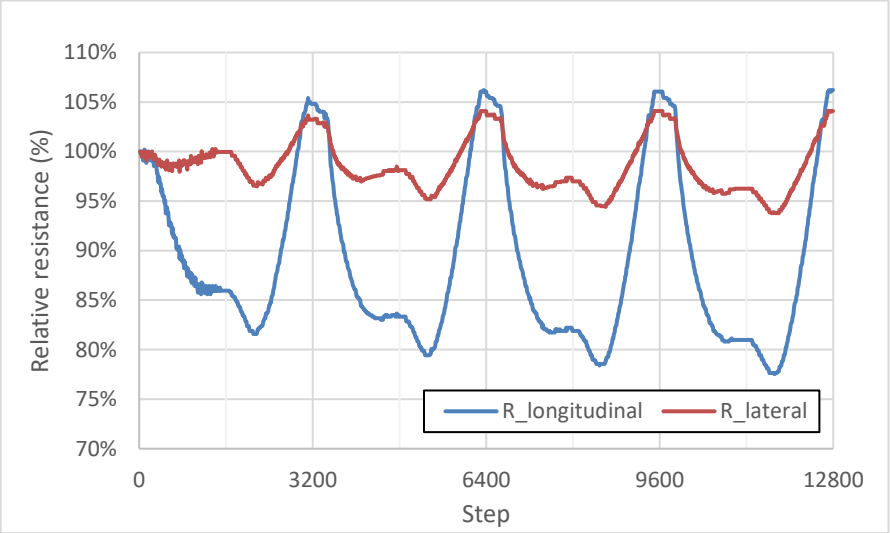


Figure 7.5: Test result of the negative bending: resistance-step characteristics of the sample

The result shows three points that doesn't meet the expectation. First, theoretically, when the sample is bent negatively, the longitudinal resistance should become larger with increasing bending radius since the resistor material is stretched. But the experimental results are completely opposite to the expectation. Second, the lateral resistance doesn't remain the same, but experienced a similar resistance change as did the longitudinal resistance. Third, the resistance of the natural length has an offset to the initial state and the resistance characteristics of the first half cycle and the second half cycle is different, which indicates the hysteresis of the sample.

Test 3: Alternate double bending

Figure 7.6 shows the results of the first six cycles of Test 3. The horizontal coordinate is the number of steps of the stepper motor. Every 3200 steps, one bending cycle is completed exactly.

Odd number of cycles is positive bending and even number of cycles is positive bending. Taking the first and second cycle as an example, the first cycle is positive bending, and the second cycle is negative bending. In both positive and negative bending, the longitudinal resistance decreases with increasing bending. But the minimum resistance in positive bending is smaller than negative bending. In addition, in the second half of the cycle, with the decrease of bending radius, the resistance in positive bending can recover to larger resistance than negative bending. For the lateral resistance, it has similar characteristics to the longitudinal resistance, but the rate of change of the resistance is much smaller.

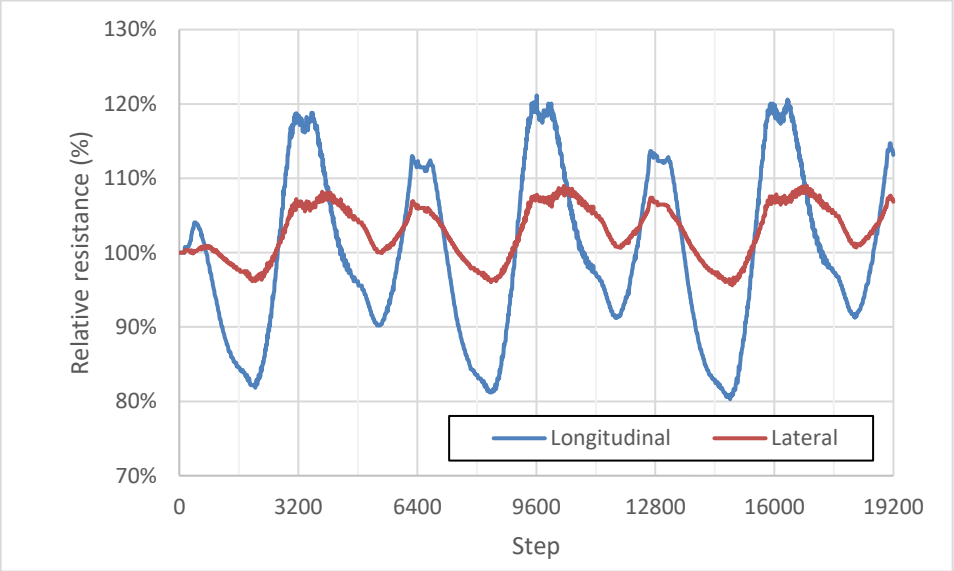


Figure 7.6: Test result of the alternate double bending: resistance-step characteristics of the sample

There are three points of the experiment result that doesn't not meet the expectation. First, in even cycles, i.e., negative bending cycles, the resistance should increase with the bending radius, which means the resistance should increases in the first half cycle and then decreases in the second half cycle. But the actual result is contrary to the expectation. Second, the lateral resistance doesn't remain the same, but experienced a similar resistance change as did the longitudinal resistance. Third, the resistance of the natural length has a direction-dependent offset to the initial state, which indicates the hysteresis of the sample.

In summary, the electrical characteristics of the used samples in bending test are not satisfying in all three tests. The test of other samples and further research on the causes of the erroneous results and related improvement measures will be carried out in the future, which is outside the scope of this work.

8 Conclusion and outlook

In this work, a test environment for the characterization of a new type of flexible sensor system is built. This test environment enables the measurement of deformation-resistance characteristics of the flexible sensor elements under bending and stretching conditions for the evaluation of materials, structures and processes of the new sensor. The test environment is based on the existing stretching and bending test setup. In terms of hardware, retaining the original mechanical structure and electrical motion system, the upgrade of the electrical measurement system is focused on. Instead of the original Arduino UNO microcontroller board, a Cypress PSoC 6 BLE Pioneer board is used, which contains a highly integrated and programmable SoC on it, as the development platform of the new measurement system for both setups to build a flexibly dynamically changeable measurement circuit. In terms of software, new programs are written for microcontrollers of the motion and measurement system, and a GUI software is designed and implemented on the host computer, making it convenient and intuitive to perform tests and read test data. After the building of the new test environment, the measurement systems of the stretching and bending setup are evaluated and the results showed that the measurement systems have good static accuracy and dynamic response speed within the demand range. Finally, the deformation-resistance characteristic of the real samples is briefly tested and evaluated in the new test environment and the experimental result deviates from the expected electrical characteristics. Further studies related to it will be carried out in the future with the help of the new test environment.

Although the test environment meets the basic requirements for testing, including bending and stretching operation on samples and resistance measurements of the samples, the environment still has the following issues that need to be addressed in the future:

First, for the stretching test setup, there is still unignorable noise in the measurement of the sample resistance, which is based on the four-point measurement method, caused by the excitation current noise. The problem can be avoided in the future by using other resistance measurement methods, such as the reference resistance-based measurement method.

Second, in the stretching test setup, the force and position sensors of the measurement system are not calibrated. currently, only the measurement of their original sensor signals is implemented. The calibration will be carried out in the future. In addition, the sensitivity of the force sensor is too low for the existing sample and will be replaced with a more sensitive model in the future.

9 References

[1] E. Koch, A. Dietzel, Skin attachable flexible sensor array for respiratory monitoring (2016).

[2] Roman Vulikh, Charakterisierung des Siebdruckprozesses zur Herstellung dehnbarer Sensoren, Studienarbeit, 2021.

[3] Arjowiggins, Screen printing, https://arjowigginscreativepapers.com/en/printing/printing-and-finishing-advice/screen-printing/.

[4] C.W. Foster, R.O. Kadara, C.E. Banks, Screen-printing electrochemical architectures, Springer, Cham, 2016.

[5] Vincent Gottwald, Aufbau eines Dehnungsprüfstands für Leiterbahnelemente mit integrierter Widerstandsmessung, Bachelorarbeit, 2018.

[6] Jimmy Retzlaff, Philip Hoop, Entwicklung eines automatisierten Biegemessstandes für ein flexibles Sensorpatch, Projektarbeit, 2016.

[7] Wikipedia, Cypress PSoC, [February 21, 2022], https://en.wikipedia.org/wiki/Cypress_PSoC.

[8] Cypress Semiconductor, PSOC® PROGRAMMABLE SYSTEM-ON-CHIP: Higher Integration, Faster Time-To-Market, Greater Embedded Design Flexibility.

[9] Cypress Semiconductor, CY8CKIT-062-BLE: PSoC 6 BLE Pioneer Kit Guide.

[10] Honxvay, CY8C6347BZI-BLD53 BGA116, http://honxvay.com/aspcms/product/2019-12-12/209.html.

[11] Cypress Semiconductor, CY8C6347BZI-BLD53, [February 21, 2022], https://www.infineon.com/cms/en/product/microcontroller/32-bit-psoc-arm-cortex-microcontroller/psoc-6-32-bit-arm-cortex-m4-mcu/cy8c6347bzi-bld53/?utm_source=cypress&utm_medium=referral&utm_campaign=202110_globe_en_all_integration-product_types.

[12] Cypress Semiconductor, PSoC Creator Help Topics.

[13] Cypress Semiconductor, PSoC® 6 MCU: PSoC 62 Architecture Technical Reference Manual (TRM): PSoC 62 MCU.

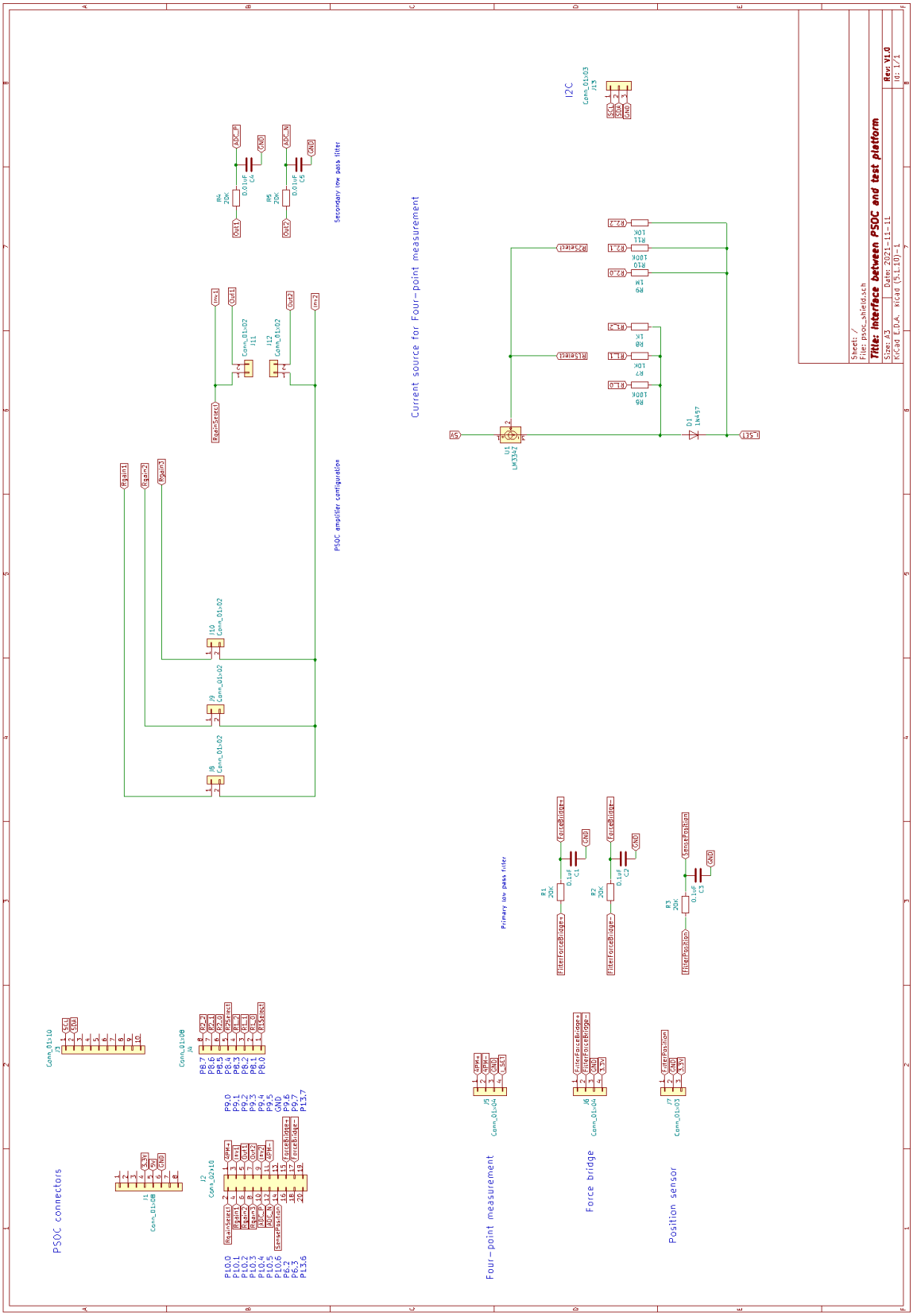
[14] Meenakshi Sundaram R, Vaisakh K V, PSoC 6 MCU Hardware Design Considerations.

[15] Texas Instruments, LM134/LM234/LM334 3-Terminal Adjustable Current Sources datasheet (Rev. E).

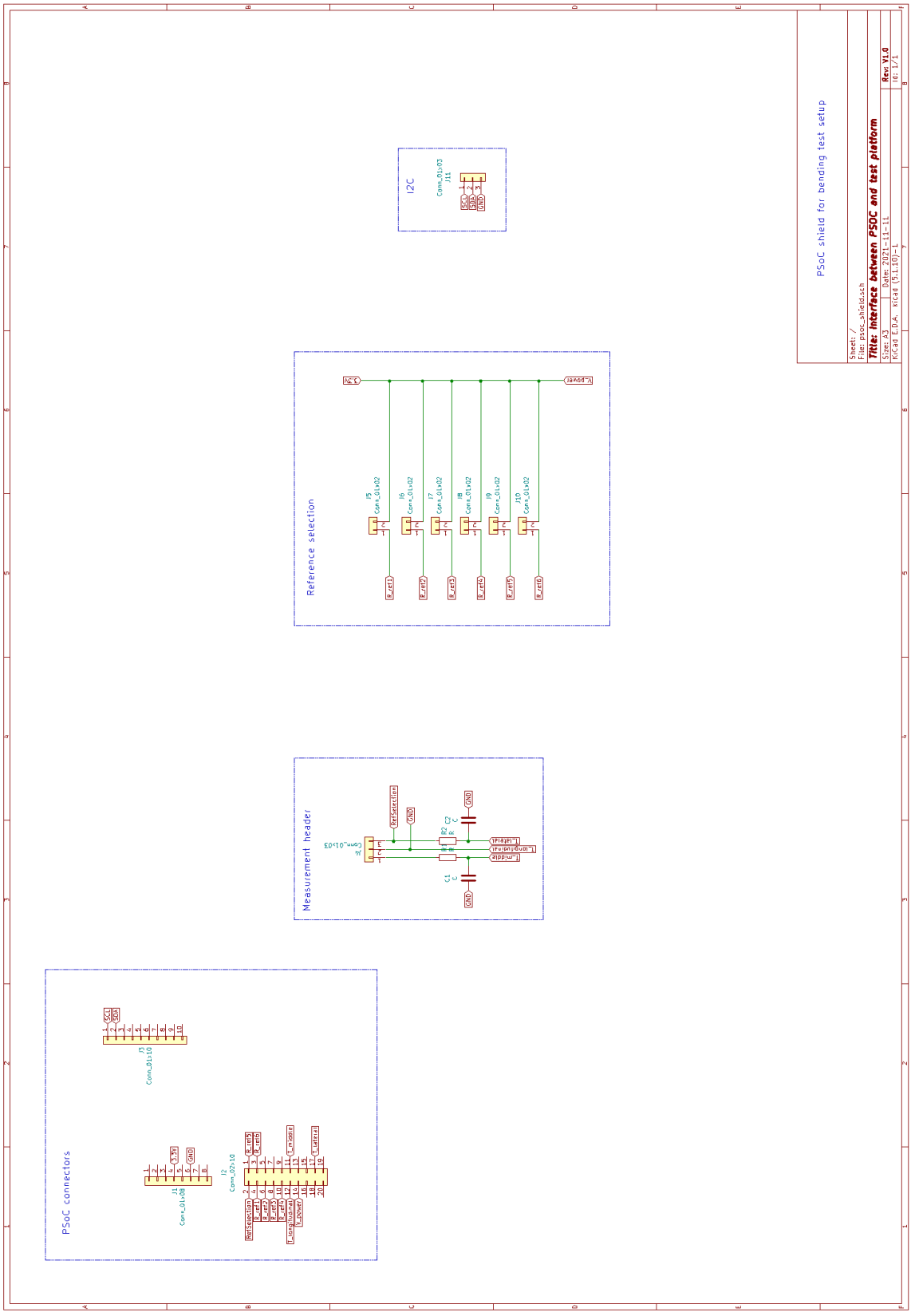
[16] LOADCELL TECHNOLOGY, LAS-A S-type load cells.

[17] TT Electronics, Slide Potentiometer Model PS100.

Annex A: Schematic of PSoC shield for stretching test setup



Annex B: Schematic of PSoC shield for bending test setup



Annex C: CD