Modular GC: A Fully Integrated Micro Gas Chromatography System

Shaunak Manurkar

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Engineering

Leyla Nazhandali, Chair
Masoud Agah
Lynn Abbott

August 18, 2021
Blacksburg, Virginia

Keywords: Micro Gas Chromatography, Embedded Systems, Data Analysis

Copyright 2021, Shaunak Manurkar
Modular GC: A Fully Integrated Micro Gas Chromatography System

Shaunak Manurkar

(ABSTRACT)

Gas Chromatography (GC) is one of the most important and widely used tools in analytical chemistry. However, they are bulky, have a longer measurement cycle, and consume a high amount of power. Micro-Gas Chromatography (µGC) is portable and energy-efficient, which allows onsite, real-time biological, forensic, and environmental analyses. This thesis presents a ready-to-deploy implementation of microfabricated gas chromatography (µGC) system capable of separating complex samples. We describe robust, modular, and scalable hardware and software architecture based on Real-Time Operating System (RTOS) and Python Graphical User Interface (GUI) integrated with various microfabricated devices to realize a fully functional µGC system. A sample heater for headspace injection, microfabricated separation column (µSC), a Photoionization Detector (PI-D), and a flow controller unit are integrated with the modular hardware and software to realize a fully functional Vacuum Outlet µGC system. We have designed a novel auto-calibration method for temperature calibration of the microfabricated devices which does not require changing the electronic circuitry or reprogramming the device. The vacuum outlet µGC setup is tested with various mixture of analytes. For these experiments, an average relative standard deviation (RSD) for retention time repeatability of 2.5% is achieved. Data processing techniques for raw chromatograms, including baseline correction and peak detection, are implemented on a microcontroller board and tested extensively as a part of this work. A novel algorithm for multidimensional analysis for the identification of co-eluting compounds in complex samples is implemented with a prediction accuracy of 94%.
Modular GC: A Fully Integrated Micro Gas Chromatography System

Shaunak Manurkar

(GENERAL AUDIENCE ABSTRACT)

Toxic volatile organic compounds (VOCs) such as benzene and toluene found in gasoline and xylene used in ink, rubber, and leather industries are of concern as they are present at elevated concentrations due to their higher vapor pressure. Sufficient exposure to these toxicants, even at lower concentrations like 100 parts-per-billion-volume (ppbv), may cause adverse health effects. Gas Chromatography (GC) has been the established method for assessing the presence and concentration of VOCs in the environment. Traditional GC systems are bulky, power-hungry, expensive, and require expert supervision for analysis. Recent research in microelectromechanical systems (MEMS) has reduced the size of the GC components, also called micro-GC (µGC), while improving the performance. The majority of the research and development of µGC is aimed at advancing microfabricated components such as preconcentrators, separation columns, and gas detectors. However, the integration of these different components is an important topic that requires more investigation. In this thesis, we present a robust and scalable software and hardware architecture that can be used to develop a portable and modular µGC system. The thesis discusses different experiments to calibrate various microfabricated devices, which are then used to build a fully modular µGC system. We show the separation capacity of the modular µGC system by passing complex compounds like kerosene and diesel. As the chromatogram from the µGC system has noise, the second part of the thesis explores data analysis techniques such as baseline correction, peak detection. These data analysis tools are used to filter the noise, detect relevant peaks in the chromatograms, and identify the compounds in a complex sample.
Dedication

To my family.
Throughout my graduate studies at Virginia Tech, I have received a great deal of support and assistance. Firstly, I would like to express my deepest appreciation to Dr. Leyla Nazhandali. Her invaluable guidance and faith in me were my biggest sources of motivation. I am grateful for her guidance and support throughout the project, which I believe have molded me into a better professional. I extend sincere thanks to Dr. Masoud Agah for allowing me to work in the VT MEMS lab through the course of this collaborative project. I also thank Dr. Leyla Nazhandali, Dr. Masoud Agah, and Dr. Lynn Abbott for agreeing to be part of my Master’s advisory committee. I would like to thank the National Institute for Occupational Safety and Health for supporting this work. I wish to acknowledge Mustahsin Chowdhury for introducing me to the project, his continued help in figure out the design details, and for answering my endless list of questions. I wish to acknowledge the members of VT MEMS lab - with special mention to Nipun Thamatam - for their cheerful companionship. I thank my parents, brother, and everyone else in my family back home in India for their constant support and encouragement throughout my graduate studies. I am indebted to my brother, Shardul Manurkar, whose presence and positivity was an integral part of my education at Virginia Tech. Finally, I express my gratitude to my friends Kunal Joshi, Rohit Murali, Jobin Jose, Rahul Iyer, and Varun Rege for their help and the fun times I spent in Blacksburg.
Contents

List of Figures ix

List of Tables xii

1 Introduction 1

1.1 Motivation ......................................................... 2

1.2 Background ....................................................... 3

1.2.1 Conventional GC System ......................................... 3

1.2.2 μGC System ..................................................... 6

1.3 Contributions ..................................................... 9

1.4 Organization of Thesis ............................................ 10

2 Hardware Architecture 11

2.1 Power Management Board ........................................... 11

2.2 Micro-controller Board ............................................ 13

2.3 Sensor Board ..................................................... 14

2.3.1 Thermal Conductivity Detector ................................ 16

2.3.2 Resistance Temperature Detector .............................. 16

2.4 Valve Board ..................................................... 17
2.5 Modular Design ......................................................... 19

3 Software Architecture .................................................. 22

3.1 Embedded Software Design .......................................... 23

3.1.1 RTOS vs Bare-Metal ............................................... 23

3.1.2 Real-Time Operating System .................................... 24

3.1.3 PID Controller ...................................................... 32

3.2 Graphical User Interface .............................................. 36

4 System Integration and Results ....................................... 40

4.1 μGC Setup .............................................................. 40

4.2 Heater Calibration .................................................... 41

4.3 Temperature Control Methods for μGC .......................... 45

4.3.1 Isothermal Operation ............................................. 45

4.3.2 Temperature Programming Operation ........................ 47

4.4 Vacuum Outlet μGC Experiment ................................... 48

4.4.1 Sample Heating .................................................. 49

4.4.2 Loading State ...................................................... 49

4.4.3 Injection State ..................................................... 50

4.5 Results ................................................................. 51

5 Data Analysis ............................................................. 55
5.1 Baseline Correction and Peak Detection ........................................ 55
  5.1.1 Baseline Correction ....................................................... 56
  5.1.2 Peak Detection ........................................................... 59
  5.1.3 Results ................................................................. 61

5.2 Multidimensional Analysis ....................................................... 63
  5.2.1 Implementation .......................................................... 63
  5.2.2 Results ................................................................. 65

6 Conclusion ........................................................................ 69
  6.1 Future Work ................................................................. 70

Bibliography ......................................................................... 71
List of Figures

1.1 Typical GC System Schematic .............................................. 5
1.2 GC Detector Output .......................................................... 6
1.3 Micro-GC Schematic ........................................................... 7

2.1 $\mu$GC Hardware Architecture ............................................ 12
2.2 Stackable Power Management Board Schematic ....................... 13
2.3 Stackable Microcontroller Board Schematic ............................. 15
2.4 Sensor Board Schematic ..................................................... 17
2.5 Stackable Valve Control Board Schematic ............................... 18
2.6 Stackable $\mu$GC Hardware Design a) Isometric view, b) Side view 20

3.1 Software Architecture for $\mu$GC System ............................... 22
3.2 RTOS vs Bare-Metal Application .......................................... 25
3.3 RTOS Task Priorities .......................................................... 29
3.4 Semaphore API Example ..................................................... 29
3.5 $\mu$GC Firmware Pseudo Code ............................................. 30
3.6 RTOS Task Execution ......................................................... 32
3.7 Typical Control System ...................................................... 33
5.3 a) Raw chromatograph signal from FID detector, b) Baseline estimation c)
Baseline corrected signal d) Peak detection on the baseline corrected signal . 62

5.4 Result of peak detection for Sample 1 with three compounds when passed
through Separation Columns coated with a) BPY b) HOEMIM c) N1888 . . 66

5.5 Result of Multidimensional Analysis with Sample 1 . . . . . . . . . . . . . 67

5.6 Result of Multidimensional Analysis for a Sample with 46 Compounds . . 68
List of Tables

1.1 Commercial Micro GC Systems ........................................ 3

2.1 Voltage Requirements .................................................. 11

5.1 Retention Time for Individual Compound in 3 Stationary Phases .... 64
Chapter 1

Introduction

Among the various analytical methods, gas chromatography (GC) is the most commonly used for separating and analyzing compounds that can be vaporized without decomposition. Gas chromatographs are frequently used with mass spectrometers (GC-MS), which is considered one of the most powerful and accurate tools to separate and identify chemical components. In this technique, samples are first collected from the field through trap-based systems such as sorbent tubes or canisters. Then trained technicians analyze the sample in a laboratory environment. This process takes a considerable amount of time as it requires manual intervention and multiple steps, like sampling, storage, and shipping, before analysis. Thus, it is prone to higher losses and longer measurement cycles. High-performance GCs are developed, but the instruments are bulky and expensive, which seriously hinder their applications [8]. Steered by the ever-increasing demands for on-site and real-time chemical analysis of a wide range of complex mixtures, interest in developing portable analytical instruments has significantly increased in the past few decades. The on-site chemical analysis presents numerous advantages as compared to the laboratory-based measurements. On-site, real-time measurements a) reduce analysis and turnaround time for time-sensitive decisions, b) minimize the contamination in the composition of the sample owing to several processes such as evaporation, adsorption, degradation, and oxidation.
1.1 Motivation

To develop compact, low-power, and field-portable GC instruments, considerable research has been conducted over 40 years. Advances in commercial, field-deployable GC instrumentation have led to significant improvements in performance, reliability, and portability [10]. 

[26], [11] [9], [17], [10] present research on innovative designs for the $\mu$GC system that further reduce cost, size, analysis time and power requirements. A miniaturized version of the gas chromatography (GC) system is a promising technique for the rapid and sensitive analysis of complex chemical mixtures. While a conventional GC is a powerful and versatile tool, they are quite bulky, energy inefficient, and expensive for real-time environmental monitoring applications. Hence, these instruments are normally not field portable. There has been increased marketability of portable GC instruments, and these products are being developed in both research and commercial laboratories. Commercial micro GC Systems are shown in Table 1.1. Albeit these are miniaturized versions of the benchtop GCs for rapid vapor analysis, they suffer severely from deteriorated separation capability or peak capacity due to primarily to the short column length, wide peak width resulting from miniaturization, and requirement for short analysis time. Thus, they usually can separate only a small set of a well-defined class of vapors and often fail when complex analytes mixtures are present. The majority of the research and development of micro-gas chromatography is aimed at advancing microfabricated components such as preconcentrators, separation columns, and gas detectors. However, the integration of these different components is an important topic that requires more investigation.
1.2. Background

The section presents brief information about the design and working principles of the traditional GC systems and a μGC system. Fabrication methods of different microfabricated components such as separation columns, preconcentrators, detectors are also briefly discussed.

1.2.1 Conventional GC System

Gas Chromatography (GC) is a technique in analytical chemistry for separating compounds in a mixture that can be vaporized without decomposing. In order to be fitting for GC analysis, a compound must have sufficient volatility and thermal stability. Generally, a GC can separate components in a mixture if molecules of the compounds are in the vapor phase at around 400° and also do not decompose at these temperatures. A chemical sample with various compounds is injected into a column. Generally, the injector is maintained at a higher temperature (∼ 200°), which vaporizes the volatile sample solutes. The sample traverses
through the column with a carrier gas, also called as mobile phase. The temperature of the column is controlled using an oven. The analytes combined with the mobile phase travel through the column at different rates called the stationary phase. The rate depends on the physical properties of analytes, temperature, composition, and length of the column. The analytes get separated at the stationary phase, thus, exiting the column at a different time. The end of the column is connected to a detector. The detector generates an electric signal as the analytes come out of the column. As the analytes exit the column at different times, we can see peaks of different areas at different times for various analytes in the solution.

The schematic of a basic GC system is shown in Figure 1.1 which consists of an injector system, a column, and a detector. The injector is connected to the column head and provides a mechanism to inject the sample into a continuous flow of carrier gas. For a traditional GC system, split/splitless injectors, programmed temperature vaporized injectors, gas source injectors, etc., are standard types of injector mechanisms. A good separation also depends on the choice of carrier gas. Due to compatibility with various detectors and non-flammable characteristics, helium is a frequent choice for carrier gas. The separation column is the most crucial component in the GC system. The separation of the analytes in the mixture depends on the various properties of the separation column. Conventional GC systems use capillary columns due to their higher separation efficiency. A typical capillary column is a thin, fused silica glass tube coated with a film of polymer stationary phase on the inner wall of the column. The injector injects the sample into the column, and the mobile phase carries the sample across the separation column. The rate at which molecules progress through the column depends on the type of molecule and stationary phase material. Since each type of molecule interacts differently, their rate of progression varies, and the mixture gets separated by the time it reaches the end of the column [6]. Detectors are connected at the end of the line in a GC system. The analytes coming out of the separation column pass through the
detector, generating an electric signal. Then, the electric signal as a function of time is used for qualitative and quantitative analysis of different analytes in the mixture. Traditional GC systems commonly use the Flame Ionization Detector (FID) and the Thermal Conductivity Detector (TCD). TCDs can detect a wide range of compounds, but they have low sensitivity. However, FIDs are usually two or three times more sensitive to the compounds but can primarily detect hydrocarbons. Depending on the applications, there is a wide range of detectors such as Flame Photometric Detectors (FPD), Electron Capture Detector (ECD), Photo-Ionization Detector (PID).

![Figure 1.1: Typical GC System Schematic](image)

Qualitative and/or quantitative analysis is performed on the output captured from the detector. A chromatogram with various technical terms associated with each incident is shown in Figure 1.2. The time of the peak, also called Retention Time, is used to identify the
compound in the mixture. The total height and area of the peak indicate the concentration of the compound in the mixture. An ideal chromatogram is one that has sharp peaks as close as possible without any overlap. The overlapping peaks are called co-eluting peaks, and it reduces the accuracy of qualitative as well as quantitative analysis.

![Figure 1.2: GC Detector Output](image)

1.2.2 $\mu$GC System

As described in the previous section, long separation columns and accurate temperature control through column oven improve analysis accuracy and repeatability of the conventional GC system considerably. However, they are bulky in size, have very high power consumption, have a slower response time, and are expensive. Thus, conventional GC analysis is inhibited in the laboratory. With an increase in demand for rapid chemical analysis combined with advances in microelectromechanical systems (MEMS), research in micro gas chromatography has made considerable advancement. Recent research in MEMS has reduced the size of separation injectors, columns, detectors while increasing the performance. The working
principle of GC is similar to that of conventional GC, except that the different components are miniaturized in μGC to increase portability, decrease power consumption, and increase the speed of analysis [23]. A typical μGC system schematic is shown in the Figure 1.3 which has three main micro-fabricated components: a) Pre-Concentrators (μPCs), b) Separation Columns (μSCs) and c) Detectors. They are explained in the following section thoroughly.

![Figure 1.3: Micro-GC Schematic](image)

**Preconcentrators (μPCs)**

The role of the μPCs is to capture the vapors of low concentration (<1 ppb) from ambient and thermally desorb them to a much smaller volume eluted in a separation column, thereby increasing the effective concentration to facilitate detection as well as providing a sharp injection plug to promote high-speed chromatographic separations. The compound mixture is injected into the μGC system using preconcentrators. Preconcentrators improve the detection limit by enhancing the concentration of the selected analytes. In some cases, PCs can purify a sample stream and reduce interference by selective preconcentration [27]. Preconcentrators are better suited for vapor phase analysis and both, conventional thermally desorbed preconcentrators (TPCs) and μPCs, are used in the current generation of μGC systems. The conventional PCs consist of short-length microbore tubing packed with an ab-
sorbent heated through a wire wrapped around the tube for desorption. Innovative designs such as packed compartments and coated planar surfaces are implemented to manufacture $\mu$PCs. Thus, they are smaller in size, consume lower power, improve detector efficiency. The main disadvantage is that reduction in device size results in narrow surface sites for adsorption and subsequently reduces the sample capacity.

**Separation Column**

Separation columns are responsible for actually separating a sample into different analytes. Thus, they are considered as the most critical component in a GC system. There are fabrication techniques for separation columns, such as packed columns, capillary columns. The packed columns are made of stainless steel, and the stationary phase is directly coated in the column. Most GC applications use capillary columns. The capillary columns have their inner wall of column coated with the stationary phase. Capillary columns provide high efficiency due to the smooth, inert surface of fused silica. [5] presents a development of high-performance, silicon glass $\mu$GC columns equipped with integrated heaters and temperature sensors for temperature programming and integrated pressure sensors for flow control. The small size of microcolumns improves the response time while reducing the power consumption for heating. As compared to conventional columns, they suffer from lower peak capacity, which can be resolved to some extent with novel architectures like semi-packed and multi-capillary columns [13].

**Detectors**

The output from the separation column is passed through the detectors which generates an electronic signal as a function of time. Depending on the application, different types of micro-
detectors are available. $\mu$TCDs are the most popular detector. There exist other detectors such as Surface Acoustic Wave (SAW), Photoionization Detector (PID), Electron Capture Detector (ECD), etc. Sensitivity, selectivity, and power consumption are the essential criteria for selecting the detector for $\mu$GCs [13].

Flow Control

Gas flow in $\mu$GCs is manipulated using conventional mini pumps and valves. The microfabricated counterparts often suffer from lower performance, a complex control scheme, and low reliability. Micro-fabricated pumps and valves are an active area of research as they have a wide range of applications and will play a crucial role in the integration of $\mu$GC systems.

1.3 Contributions

The major contributions of this thesis are listed as follows:

- Developed a robust and scalable software and hardware architecture that can be used to develop a portable $\mu$GC system that is microfabricated device agnostic.

- Designed a methodology for automatic calibration and carried out extensive laboratory experiments for characterization of the microfabricated devices such as preconcentrators and separation columns.

- Implemented a novel algorithm that can identify analytes from a complex sample mixture that is subjected to multidimensional chromatography separation.
1.4 Organization of Thesis

This chapter presented the motivation behind our work and also discussed the background for the GC. The rest of this thesis is organized as follows:

Chapter 2 explains the hardware architecture of the proposed modular µGC system. The chapter describes the functionality of each PCB along with its fabrication process.

Chapter 3 discusses the back-end and front-end software design used to realize a portable µGC system. The details of embedded software, Real-Time Operating System (RTOS), and Graphical User Interface (GUI) are provided in this chapter.

Chapter 4 presents the hardware and software integration of the µGC setup followed by different laboratory experiments to characterize the system. The chapter also discusses the vacuum µGC setup and separation results.

Chapter 5 focuses on implementing data analysis techniques such as baseline correction, peak detection, and multidimensional analysis to detect and identify the compounds in a mixture from a chromatogram.

Chapter 6 concludes the thesis work.
This chapter discusses the electronic design of the proposed modular μGC system. Figure 2.1 shows integration of different electronic modules. The electronic modules are categorized into four parts based on their functionalities. Each module is explained in detail in the following sections. The PCBs for each module is developed with a focus on a modular design, which is explained at the end of this chapter.

2.1 Power Management Board

The power management board is designed to take a higher voltage from a battery as input and give appropriate power to a valve control board, sensor board, and microcontroller board. Table 2.1 shows the voltage requirements for each component.

<table>
<thead>
<tr>
<th>Component</th>
<th>Voltage Rating (V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heaters</td>
<td>18</td>
</tr>
<tr>
<td>Proportional Valves</td>
<td>12</td>
</tr>
<tr>
<td>TCDs</td>
<td>7.5</td>
</tr>
<tr>
<td>On-Off Valves</td>
<td>5</td>
</tr>
<tr>
<td>Microcontroller</td>
<td>3.3</td>
</tr>
</tbody>
</table>

Table 2.1: Voltage Requirements

We are using an 18V battery for input voltage. Input voltage from the battery is connected to multiple DC-DC buck converters, which step down the input voltage to the desired voltage.
level. Based on the current and voltage requirements, Texas Instrument’s LMZM23601 IC was selected. TI’s LMZM23601 is an efficient DC-DC buck converter, and its minimal footprint makes it ideal for our power management board. It has an input voltage range of 4V to 36V. The adjustable output voltage range is from 1.2V to 15V and can deliver up to 1000 mA of output current.

Figure 2.2 shows the schematic of the stackable power management board. There are many design considerations for a power management board as it has high and low power-consuming components on the same board. It contains a microcontroller that consumes 200 mA, whereas the valve control board can draw up to 1000 mA. Such fluctuations in power requirement can cause interference in the circuit leading to unwanted behavior. Therefore, the low power
2.2. Micro-controller Board

The Micro-controller board is the brain of the \( \mu \)GC systems. It takes commands from the user through Graphical User Interface (GUI), controls valves, pumps and heaters, and communicates with the sensor board. Apart from controlling peripherals, it must perform other tasks such as baseline correction and peak detection on the data from the sensor board.

---

Figure 2.2: Stackable Power Management Board Schematic

circuit is kept at a distance from the high current drawing circuit in the PCB layout. Also, to reduce the thermal noise, we have kept spaces between the ICs and proper traces and vias dimension. The dimensions of the traces and vias can be calculated from the Ki-CAD.
Thus, we need a micro-controller that has a sufficient number of GPIOs for controlling different components and, at the same time, must have a CPU capable of performing arduous tasks of peak detection. We have used Texas Instruments’ Tiva C series TM4C1294NCPDT microcontroller. The microcontroller has a high-performance ARM® Cortex™-M-based architecture with a broad set of integration capabilities and a robust ecosystem of software and development tools. This board has 128 pinouts with 256KB single-cycle SRAM and 1024KB flash memory along with 6KB of EEPROM.

Figure 2.3 shows the schematic for the stackable micro-controller board. The micro-controller board receives commands on how to run an experiment from the GUI over UART. Then it segregates the commands based on the components and stores them in the SRAM. For example, the board will differentiate commands according to heaters, pumps, valves and store them in that particular variable. Depending on the commands, the board will control pumps and heater using Pulse Width Modulation (PWM), valves using GPIO ON-OFF, and sensors using SPI. The software aspect of the board will be explained in Chapter 3.

### 2.3 Sensor Board

In this project, we use two types of sensors: The Thermal Conductivity Detector (TCD) and the Resistance Temperature Detector (RTD) for temperature control. RTDs and TCDs give analog output. Thus, we need an Analog-to-Digital Converter (ADC) that converts analog sensor values to digital data such that sensed data can be processed by the microcontroller.

For this purpose, AD7793 from Analog Devices was selected. AD7793 is a 24-bit ADC with low noise programmable gain instrumentation amplifier. AD7793 also provides a digital filter that removes noise from the analog input source. AD7793 communicates with the microcontroller board using Serial Peripheral Interface (SPI). Figure 2.4a and Figure 2.4b
2.3. Sensor Board

Figure 2.3: Stackable Microcontroller Board Schematic
shows schematic for TCD and RTD circuit respectively.

2.3.1 Thermal Conductivity Detector

The thermal conductivity detector (TCD), also known as a katharometer, is a bulk property detector and a chemical-specific detector commonly used in gas chromatography. This detector senses changes in the thermal conductivity of the flowing gas in the central column and compares it to a reference flow of carrier gas. We use a Wheatstone bridge circuit for sensing this resistance change which produces a measurable voltage change. The column effluent flows over one of the resistors while the reference flow is over a second resistor (Figure 2.4a) in the four-resistor circuit. The reference flow across resistor four (Figure 2.4a) of the TCD circuit compensates for drift due to flow or temperature fluctuations. Changes in the thermal conductivity of the column effluent flow across resistor three (Figure 2.4a) will result in a temperature change of the resistor and, therefore, a resistance change that can be measured as a signal.

2.3.2 Resistance Temperature Detector

The resistance temperature detector (RTD) is a sensor used to measure temperature. Many RTD elements consist of fine wire wrapped around a ceramic or glass core, but other constructions are also used. The RTD wire is a pure material, typically platinum, nickel, or copper. The material has an accurate resistance/temperature relationship which is used to provide an indication of temperature. As RTD elements are fragile, they are often housed in protective probes. Resistance temperature detectors are constructed in several forms and offer more stability, accuracy, and repeatability in some cases than thermocouples. They use electrical resistance to generate a voltage and require a power source to operate. The
2.4 Valve Board

Resistance ideally varies nearly linearly with temperature. Lead-wire resistance can be a factor in contaminating results; hence, adopting three- and four-wire, instead of two-wire, connections can eliminate connection-lead resistance effects from measurements. A three-wire connection is sufficient for most purposes and is an almost universal industrial practice. We use a three-wire RTD connection in our application as well.

2.4 Valve Board

Solenoid valves are used for controlling the flow of gas at different stages for the gas chromatography experiment. The flow is controlled by switching the flow between the ports of the solenoid valve. There exist different configurations of solenoid valves, but the type of valve used in our application is a three-port, three-way, two-position Latching Solenoid Valve. The reason for using these valves is that latching solenoid valves only need a pulse of the peak voltage of a specific polarity to move the plunger in a particular direction. When a pulse of different polarity is applied to the valve, the plunger moves in the other direction. The moving of the plunger in a particular direction indicates the flow from the port, which is not blocked by the plunger. Conventional solenoid valves need a continuous supply of voltage to maintain the plunger in a particular direction. Thus, they consume more power than a latching solenoid valve.
We are using LHLA0521111H valve model for this purpose. This valve has three ports named A, B and C. The C port is the common port through which gas or liquid is supplied. This port is situated in the middle. Ports A and B are situated on the lower and upper sides of port C. The state of the valve (meaning the port through which either the gas or liquid flows) is controlled by the polarity applied to the terminals of the valve. The schematic for valve board is shown in Figure 2.5.
2.5 Modular Design

Each modular board described in the previous sections is designed separately in such a way that they all can be stacked on top of each other to form a complete $\mu$GC system. Each board is designed with all its input/output pins arranged in two arrays of stackable header pins on the sides of the board. One of these arrays of stackable header pins is used only to supply power from the power board to the microcontroller, sensor, and valve boards. The other array of stackable header pins is used for communication between all the boards. The reason behind this separation of the stackable pins for power and communication is to provide better isolation of the power lines from the communication buses. If the header pins designated for the power supply are too close to the communication pins, noise might be introduced in the communication channels resulting in malfunctioning/crashing of the $\mu$GC setup. The stackable header pins used for communication consist of

1. GPIO and SPI pins for the chip selection and communication with the ADCs on the sensor board,
2. GPIO and PWM pins for controlling the motor ICs and pumps on the valve board, and
3. GPIO pins on the power management board for controlling the heaters.

Figure 2.6a and 2.6b show the complete stacked $\mu$GC system setup from different angles.

This stackable arrangement of header pins provides a lot of advantages for the general functionality of the $\mu$GC system. Some of them are as follows:

- The modular design helps in the debugging process of each board and in replacing any faulty electronic module. For example, if a power management board being used
by a user starts malfunctioning, the user can simply replace the faulty board with a different functional power management board without modifying the microcontroller, valve and pump, and sensor board circuitry.

- The µGC system also becomes very scalable with this design methodology. If additional module boards have to be added to the system in the future, the board designer needs only to include the two stackable arrays of header pins in their new circuitry without having to tinker with the other circuit boards.

- The need for electrical connections amongst the boards using jumper wires or any such similar connection methods is completely eliminated, thus ensuring that the system will not face functionality issues resulting from physical causes like loose connections, damaged wires, etc. The stacked µGC system thus becomes more organized as well as easier to manage and debug.

- The important objective of making a µGC system that is mobile is also accomplished.
The system can be physically handled and carried with ease by users without worry of components being detached or general structural weaknesses.
Chapter 3

Software Architecture

The chapter focuses on different software technologies used to realize a portable \( \mu \)GC system. The software is designed so that a user without any background knowledge of programming can implement different GC configurations with ease as long as the GC hardware supports it. Figure 3.1 shows an overview of the software architecture. The software architecture is divided into two major parts. Part one discusses firmware on the microcontroller board, real-time operating system (RTOS), scheduling algorithm. The graphical user interface (GUI) and its features are explained in the second part of this chapter.

Figure 3.1: Software Architecture for \( \mu \)GC System
3.1 Embedded Software Design

This section describes in detail the different aspects of the implementation of the embedded software on the microcontroller board. The embedded software runs a Real-Time Operating System (RTOS) written in C and compiled and flashed on the microcontroller board using Code Composer Studio (CCS) provided by Texas Instrument (TI). The embedded application is built using Applications Programming Interfaces (APIs) that simplify the programming by abstracting the underlying implementation and only exposing actions the developer needs [3]. For example, a user can use `getSensorData()` API to read data from detectors in an application, but the API in the background establishes an SPI connection with the sensors, sends appropriate commands, receives data bytes, formats data, and presents it to the user. This allows rapid software development, and the user can focus more on the implementation of the algorithm. In the remaining part of this section, we discuss the comparison between the RTOS and bare-metal implementation, RTOS which runs as the backend of the $\mu$GC system and its features.

3.1.1 RTOS vs Bare-Metal

In a typical embedded environment, when a program starts running a code on a microcontroller, it executes some setup functions. It then performs tasks in a round-robin fashion in an infinite `while` loop. This type of program architecture is known as a Super-Loop. A bare-metal application with a super-loop can be broken down into three crucial fragments

- Initialization: In this step, the application initializes all the required peripherals and software components in `main()`.

- Continuously Running Super-Loop: The super-loop executes the functions of the ap-
application based on different hardware and software interrupts.

- Interrupt Service Routines (ISRs): Different interrupts caused by hardware peripherals such as SPI, I2C, and timers invoke the ISRs.

For simple applications, bare-metal codes are efficient, require less memory footprint, and are easy to understand. However, when implementation is complex which has multiple functionalities, an RTOS excels. Figure 3.2 shows a comparison of a bare-metal application to a minimal RTOS application and then to a more traditional RTOS application. As you can see, the three crucial parts explained previously (init, super-loop, and ISRs) are the same between a bare-metal application and a minimal RTOS application. If an application includes only a couple of tasks with an ISR, super loop implementation will be the best choice. However, more complicated applications with various functionalities can be built very quickly from the minimal RTOS application. This is because a single, centralized super loop can be broken down into multiple individual tasks. Then, these tasks can be implemented and tested individually. RTOS also provides different mechanisms for synchronization and communication between the tasks. This is very helpful when the application involves multiple tasks which should follow deadlines. The following section elaborates on RTOS, its features, and implementation to realize a $\mu$GC system.

### 3.1.2 Real-Time Operating System

A real-time operating system (RTOS) is an operating system (OS) intended to serve real-time application process data as it comes in, typically without buffering delays [4].

Generally, an RTOS is used when the application on the embedded devices needs to perform more than a few simple tasks in a given period of time. An RTOS facilitates an application to be broken down into a fundamental structure that can scale as more features are added.
In our case, bare-metal code is probably unsuitable because of the size of the μGC system. We need something which can scale, and for this reason, we are choosing to use a real-time operating system. Texas Instruments (TI) provides its version of RTOS called TI-RTOS, which is scalable for all TI devices. It allows users to focus on developing an application by providing a real-time kernel for creating threads, synchronization methods, and device drivers. We are using an RTOS just as a way to pretend to have multiple threads of execution on an embedded target, even if each of those threads is virtualized. One of our objectives in this project is to design our algorithms in such a way that they can run concurrently. In line with how our algorithms must stream their computations as opposed to buffering the entire signal at once, threads provide an easy way to achieve concurrent streams. The main features of the TI-RTOS which will accelerate the development process are mention below.

- **Threading:** TI-RTOS provides a robust library for concurrent programming on an embedded system.

- **Synchronization:** This includes mutexes, semaphores, and condition variables. In order
to synchronize task behavior between multiple tasks and prevent data races, we will need to use locking devices.

- Deterministic scheduler: Kernel system calls complete operation in a predictable time.

- ”Zero-latency Interrupts” mode: This allows the kernel to be used in hard real-time systems.

- Highly customizable kernel: It allows unnecessary functions can be omitted to lower memory footprint to meet MCU memory constraints by allowing static configurations

- Consistent APIs across device families: It facilitates code portability for device drivers.

As we are familiarized with the basics of RTOS, the remaining part of this section explains implementation of synchronization methods, tasks, and scheduling algorithm in detail.

**Threads and Synchronization Techniques**

A μGC system has to execute several seemingly different functions simultaneously. The functions include storing data from the sensor whenever available or controlling peripheral devices based on a timer event. In such cases, the time of execution is essential. For example, in order to get a sharp injection, the valve must be switched as fast as possible. In the RTOS terminology, these functions are called *threads*. The threads can be defined as any independent lines of instructions carried out by the processor. The main advantage of the TI-RTOS is that it allows the application to be structured as a collection of threads, each of which performs an unrelated and modularized function. Multithreaded programs run on a single processor by allowing higher-priority threads to preempt lower-priority threads and allowing various types of interaction between threads, including blocking, communication,
and synchronization [16]. Based on the execution and preemption characteristics, the TI RTOS kernel provides different threads with different priorities. The thread types are:

- **Hardware Interrupt Threads (HWI):** These are the highest priority threads that respond to external asynchronous events, such as UART interrupts. They can be preempted only by the other HWI threads; otherwise, they must run to completion. All the HWIs share the same stack.

- **Software Interrupt Threads (SWI):** SWIs are similar to that of HWIs, but they can be triggered by the program by calling certain APIs. They have higher priority than task threads but lower priority than HWIs. Like HWIs, they also must run to completion and share the same stack as HWI.

- **Task Threads:** Tasks are common OS threads with lower priority than idle threads and higher priority than the SWIs. As each task has its stack, it can be blocked if necessary resources are not available. TI-RTOS kernel provides various synchronization and inter-task communication mechanisms. Thus, tasks are the most popular type of threads for developing any real-time application.

- **Idle Loop Threads:** It is the lowest priority task in the TI-RTOS kernel, which runs background tasks such as CPU load determination, stack checking, etc.

We used the threading module to break down the entire \( \mu \)GC application into independent threads according to their functionality with different priorities. These threads are:

- **Global Event Clock:** A global software clock with the highest priority with 1 Hz as frequency. At every tick, the clock checks if other tasks are completed. If all the tasks are completed, then post the semaphore for the UART task.
• UART Task: UART task receives and transmits data over UART. The UART task has the highest priority as the task is used to receive commands and send chromatograph. The UART packet protocol for transmission and reception is explained in the later sections.

• Valve Task: Valve task operates valves based on a timer associated with it. It has a higher priority than the pump task but a lower priority than a UART task.

• Pump Task: Pump task controls pumps based on a clock associated with it. It has higher priority than the heater/sensor task but lower priority than a valve task.

• Sensor Task: The sensor task gathers data from five detectors and five RTDs in a round-robin fashion and sends the data over UART. It communicates with the sensor over an SPI bus. It also has a clock associated with it which runs at 10 Hz. Based on the values from the RTDs, the sensor task calculates a PID value and controls PWM for the heaters. It has the lowest priority in the \( \mu \text{GC} \) application.

Tasks according to their priorities are shown in Figure 3.3. The task creation and priority assignment are done dynamically. Thus in the future, not only can we change the priorities of the tasks during the runtime of the application, but we can add or remove tasks if needed.

These task threads are synchronized using semaphores. Semaphore is a mechanism used to coordinate access to a common resource by multiple processes and thus, preventing race conditions. Tasks use semaphores to block till an event occurs or a resource becomes available. TI RTOS provides \texttt{semaphore\_pend()} API to block a task. \texttt{semaphore\_post()} API is used to notify that the resource is available. Figure 3.4 shows a basic use case of the semaphore APIs.

Thus, now combining the concepts of threads and semaphores, Figure 3.5 shows the pseudo-code for the \( \mu \text{GC} \) system.
3.1. Embedded Software Design

Figure 3.3: RTOS Task Priorities

```c
void Task_1() {
    // get data from sensor
    Semaphore_post(handle);
    // Task Complete
}

void Task_2() {
    while(condition) {
        Semaphore_pend(handle);
        // Process sensor data
    }
}
```

Figure 3.4: Semaphore API Example
void Event_clockHandler()
{
    if (currentState < totalState)
    {
        if (all process complete)
        {
            Semaphore_post(valve_semaphore);
        }
        else
        {
            Semaphore_post(uart_semaphore);
        }
        globalTime++;
    }
}

void UART_Task()
{
    while(1)
    {
        Semaphore_pend(uart_semaphore);
        getExperimentData();
        startUARTInterrupt();
        Semaphore_post(valve_semaphore);
    }
}

void UART_Interrupt()
{
    // Interrupt when UART receives STOP
    // signal from GUI
    Software_Reset();
}

void Valve_Task()
{
    while(1)
    {
        Semaphore_pend(valve_semaphore);
        switchValves();
        startValveTimer();
        Semaphore_post(pump_semaphore);
    }
}

void ValveTimerHWI()
{
    if (ValveTime)
    {
        switchValve();
        valveProcessCompleted = true;
    }
}

void Pump_Task()
{
    while(1)
    {
        Semaphore_pend(pump_semaphore);
        startPump();
        startPumpTimer();
        Semaphore_post(Sensor_semaphore);
    }
}

void PumpClockSWI()
{
    if (heaterReachedSetpoint)
    {
        stopHeaters();
        heaterProcessCompleted = true;
    }
}

void Sensor_Task()
{
    while(1)
    {
        Semaphore_pend(sensor_semaphore);
        startSensorTimer();
        getDetectorData();
        if (heaters)
        {
            getRTDData();
            filterRTDData();
            calculatePID();
            controlHeaterPWM();
        }
    }
}

void PumpClockSWI()
{
    if (heaterReachedSetpoint)
    {
        stopHeaters();
        heaterProcessCompleted = true;
    }
}

Figure 3.5: µGC Firmware Pseudo Code
Scheduling Algorithm

A scheduler is at the core of any RTOS. The execution of various threads is managed by the scheduler. TI-RTOS has *Preemptive Priority* based scheduler. In a priority scheduling algorithm, each thread is assigned a priority which is an integer value. In our implementation, a larger integer value corresponds to a higher priority. The priorities are assigned to the threads based on the type of work, the importance of the work, etc. For example, in the $\mu$GC system, the UART task has higher priority than the valve, pump, and sensor tasks, since the communication with the PC host is carried out by the UART task. It is also imperative that if a lower priority task is running on CPU and if a higher priority task asks for CPU, the lower priority task should preempt. For example, even if the sensor task is collecting data, the valve task must be able to preempt the sensor task as it has a higher priority. Figure 3.6 shows the execution of different tasks for $\mu$GC RTOS application. In a preemptive scheduler, a thread continues to run unless

- It completes its execution

- A higher priority thread becomes ready for execution, i.e., a higher priority thread will preempt the running thread with low priority.

- The thread yields the processor, i.e., the task calls `sleep()`.

This covers the implementation of different tasks using semaphores and preemptive scheduler in the TI-RTOS that forms as backend of the scalable $\mu$GC system. In the next section, we discuss the implementation of PID controller task as it is the most important thread that maintains the temperature of the devices in the $\mu$GC system.
3.1.3 PID Controller

Figure 3.7 shows a general control system. Proportional, Integrator, and Derivative (PID) controllers control uncertain systems using a feedback loop. The system tracks a setpoint or reference by letting the controller adjust the signal going into the system. The controller uses the setpoint and feedback from the sensor to generate and control the signal. In our case, heaters are used to heat $\mu$PCs and $\mu$GCs from room temperature to a specific setpoint. Resistance Temperature Detectors (RTDs) are the sensor that provides the measurement of the current temperature. The setpoint is provided by the user, whereas continuous temperature monitoring is done through the RTD sensor. The controller combines the two inputs and produces a control signal. If the measured temperature is more than the setpoint, the heater is turned off. However, if the measured temperature is less than the setpoint, the heater is turned on. Systems we encounter in day-to-day life are generally in continuous time. Nevertheless, we are implementing the PID controller discretely in software as the digital system are far more robust, cheap, and easy to reprogram.
Continuous time domain representation of the PID controller is shown in the Figure 3.8. The PID controller continuously calculates the error term $e(t)$ by subtracting setpoint $r(t)$ from the measured value $y(t)$ as shown in the Eqn. 3.1

$$e(t) = r(t) - y(t) \quad (3.1)$$

The error signal is then fed to the controller, which generates a control signal $u(t)$. The error signals pass through proportional, integral, and derivative gains to generate a control signal. The controller’s objective is to reduce the error between setpoint and measured value over time by adjusting the control output, such as the power to the heater in our case.
proportional, integral, and derivative gains impact the control output in the following way:

- Proportional gain $k_p$ is proportional to the error value $e(t)$.
- Integral gain $k_i$ accumulates the past error values and integrates them over time.
- Derivative gain $k_d$ is a prediction of the error term, based on the current rate of change of the error.

Therefore, by combining these terms, the overall equation for the PID controller output is

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d(\tau) + K_d \frac{de(t)}{d(t)} \quad (3.2)$$

Now, depending on the system, the gains ($K_p, K_i, K_d$) must be tuned in order to get the optimal control output. Depending on the application, one can choose only one or two control terms, such as P, PI, PD, to get the appropriate control output. One of the main advantages of using the PID controller is that it can be tuned using only the output of the measured process variable, and thus, the plant model is not required.

In order to implement the PID controller on a microcontroller, we have to convert the continuous-time PID into the discrete-time PID controller. So, if we consider $\delta t$ as sampling time, the proportional term can be calculated as

$$p[t_k] = e[t_k] \quad (3.3)$$

The integral term can be discretized via trapezoidal approximation

$$\int_0^t e(t)d(\tau) = \sum_{i=1}^t e(t_i)\Delta t = i[t_k] \quad (3.4)$$
Similarly, the derivative term can be approximated as

\[
\frac{de(t_k)}{dt} = \frac{e(t_k) - e(t_{k-1})}{\Delta t} = d[t_k]
\]  

By combining Eqn. 3.3, 3.4 and 3.5, the discrete controller output can be calculated as

\[
u[t_k] = p[t_k] + i[t_k] + d[t_k]
\]  

Eqn. 3.6 can be implemented in the software easily and pseudocode is shown in the algorithm 3.1.

```
Listing 3.1: Discrete PID Controller

1. previous_error = 0
2. integral = 0
3. def PIDController_Update(setpoint, measurement):
   4.   error = setpoint - measurement
   5.   proportional = error
   6.   integral = integral + error * dt
   7.   derivative = (error - previous_error) / dt
   8.   control_signal = Kp * proportional + Ki * integral + Kd * derivative
   9.   previous_error = error
10.  return control_signal
11.}
```

Initial values for error and integral are kept as zero. The values for the constants $K_p$, $K_i$ and $K_d$ are calculated empirically. This concludes the implementation of the PID controller on the microcontroller board and the applications of the PID controller are explained in the Chapter 4.
3.2 Graphical User Interface

In order to simplify the use of GC hardware for any user, a Graphical User Interface (GUI) was developed in Python using PyQt libraries as shown in Figure 3.9. The GUI is divided into two parts according to its functionality. Figure 3.9a shows the control panel tab which allows user to

- Select the number of valves, pumps and heaters.
- Program the valves and pumps with timer delays.
- Create Temperature Programming profile for the heaters.

With the help of GUI data visualization tab shown in Figure 3.9b, user can

- Visualize data from RTDs (for heaters) and detectors in Real-Time.
- Store RTD and detector data for further analysis.

The GUI allows users from non-programming backgrounds to focus more on designing GC experiments and visualizing experiment data without requiring to code the microcontroller board.

As mentioned above, GUI was designed using PyQt libraries in Python. Python is a simple yet powerful object-orientated language. Its simplicity makes it easy to learn, but its power means that large and complex applications can be created. Its interpreted nature means that Python programmers are very productive because there is no edit/compile/link/run development cycle. PyQt is a set of Python bindings for a cross-platform Qt application framework [1]. Thus, PyQt is not a programming language of its own, but it is a framework written in Python which provides Application Programming Interfaces (APIs) to create GUI
for all the platforms such as desktops, mobiles, and embedded. It also provides features such
as multi-threading, signals, and slots for communication [2].

The flow of execution for the GUI is shown in Figure 3.10. When a user starts the GUI, the
following three threads are spawned in the background.

- **UART Transmission Thread**: Takes experiment data from the user using a thread-safe
  queue mechanism and transmits it over UART to the GC system.

- **UART Receive Thread**: Receives data from GC system, categorizes the data according
to packets. If it is a sensor data packet, then it stores the data in a variable. If it is
  an information packet, then it displays on the GUI.

- **Update Graph**: Updates the graph after every one second after UART starts receiving
  the data.

These threads form the back-end of the GUI and since these threads are running parallel to
the front-end, the GUI does not freeze even if UART communication is blocking in nature.

After initializing the threads, the user can input different parameters such as the number
of valves, pumps, and heaters. Users can program valves and pumps states along with their
timings. Heaters can be programmed to perform an isothermal experiment or a temperature
programming experiment. Chapter 4 shows the laboratory characterization and experiments
with the μGC system for different samples.
Figure 3.9: Python based GUI Application
3.2. Graphical User Interface

Start

Wait for Experiment Setup (Number of Valves, Pumps, Heaters)

Wait for Experiment Parameters (Valve, Pump and Heater Position)

Encode Experiment Parameters into UART Packets

Create UART Receive Thread
Run Thread

Create UART Transmission Thread
Run Thread

Create Timer Thread for Updating Graph
Run Thread

Sensor Data

Experiment Parameters

Queue

PyQt Signal

Wait for the Completion of Experiment

Run Experiment

Terminate Experiment

Figure 3.10: Graphical User Interface Flowchart
Chapter 4

System Integration and Results

A field-ready prototype of the µGC model was assembled by integrating stackable boards along with the microfabricated components. This chapter starts with the physical setup of the µGC system. We describe different experiments such as heater calibration, temperature programming, vacuum µGC performed with the µGC setup. The results of these experiments are discussed at the end of the chapter in detail.

4.1 µGC Setup

The µGC setup consists of a six-port valve to inject the sample, a separation column, a heater to heat the sample, a 3-port latching valve, a commercial Photo-Ionization Detector (PID), and fused-silica capillaries for flow control. All these components were connected to the electronics as shown in Figure 4.1. A small helium cylinder is used for carrier gas to make it portable. The µGC system is connected to the host laptop using a USB cable. The host system runs the GUI to control the peripheral components and receives detector data to plot a real-time graph. The following sections in this chapter provide details about the different experiments we carried out with the µGC system.
4.2 Heater Calibration

While putting the $\mu$GC setup together, we realized that RTD and heater resistances vary considerably between different devices, mainly due to different device requirements, sizes, and chip-to-chip variation during fabrication. For example, a preconcentrator has a heater and RTD resistance of 70 $\Omega$ and 1800 $\Omega$ respectively, whereas the resistance of heater and RTD for a separation column is 23 $\Omega$ and 205 $\Omega$ respectively. Therefore, if the temperature control is required, each RTD must be calibrated before adding in the $\mu$GC system. [19] suggests a manual RTD calibration method in which the authors connect a thermocouple to
the outside of the sorbent cavity (backside of the RTD/heater), held in place by the thermal paste, and record the temperature every 1 second using a multimeter. Albeit being a simple and accurate method, it is not scalable and requires constant manual intervention.

Thus, we designed a new setup for automated calibration of any device with a heater and RTD. In this design, a calibrated reference RTD is connected to the back of the device and glued in place by thermal paste or tape. The characteristics such as temperature coefficient of resistance ($\alpha$) material are known for the reference RTD. The idea is that whenever the device heater is heated to a setpoint, the resistance of the reference RTD will also change according to the device’s RTD as it is directly proportional to the device’s heater/RTD. This can be seen from Figure 4.2. Temperature can be then calculated from the reference RTD from the equation

$$R = R_\alpha(1 + \alpha(T - T_\alpha))$$  \hspace{1cm} (4.1)

![Device's RTD Resistance Plot](image1)

(a) Device’s RTD Resistance Plot

![Reference Calibrated RTD Temperature Plot](image2)

(b) Reference Calibrated RTD Temperature Plot

Figure 4.2: Device Heater/RTD Calibration with a Calibrated Reference RTD

Listing 4.1 shows the algorithm for the auto-calibration the microfabricated devices. PT 1000 RTD [20] is used in our calibration experiments which has resistance of 1000 $\Omega$ at 0°C
and α of 0.00385 Ω/Ω/°C. By substituting these values, the eqn. 4.1 can be rewritten as

\[ R = 1000(1 - 0.00385T) \]  \hspace{1cm} (4.2)

Listing 4.1: Automatic Device Calibration

```python
setpoints = [50, 70, 90, 110]
temp_res = {}  # Dictionary
for sp in setpoints:
    Heat device heater to the SP
    Measure device resistance through RTD
    Measure temperature through calibrated RTD
    Wait for calibrated RTD to reach a steady state
    Calculate the average value of temperature at steady state
    Update temp_res dictionary with key as temperature and value as resistance

Calculate relationship between temperature and resistance through curve fit
```

This experiment is then repeated for different setpoints to collect the device’s RTD resistance data and corresponding temperature data from the reference RTD. Then, curve fitting optimization is used to find a mathematical function that best fits these observations. The relationship between temperature and resistance for an RTD is linear [20]. So, a mapping function can be defined as:

\[ \text{Resistance} = a \times \text{Temperature} + b \]  \hspace{1cm} (4.3)

Using curve fitting optimization libraries in Python, parameters \( a \) and \( b \) are calculated as 0.32 and 200.21, respectively. The plot 4.3 shows the linear relationship between resistance and temperature.
CHAPTER 4. SYSTEM INTEGRATION AND RESULTS

Figure 4.3: Heater Calibration Curve Fitting

With the calibration, the chip was heated from room temperature to 100°C (Figure 4.4) and maintained the temperature for around six minutes using the PID algorithm 3.1.3. The Root Mean Squared Error (RMSE) at constant temperature calculated from the results is 0.92°C. Thus, the heater calibration with PID can be used with the μGC system.

In order to automate the device calibration, it can be cycled through multiple setpoints. For example, users can input different setpoints and hold time through GUI, as shown in Figure

(a) SC Temperature Setpoint  
(b) SC Temperature Setpoint (Zoomed In)

Figure 4.4: Temperature Accuracy with Heater Calibration and PID
4.3 Temperature Control Methods for $\mu$GC

Temperature control plays an essential part in getting a desirable separation in GC. The temperature of the column can either be maintained at a setpoint throughout the experiment (Isothermal), or it can be increased at a predetermined ramp rate (Temperature Programming). Separation can be achieved faster at a higher temperature but with poor resolution. On the other hand, lower temperatures provide better resolution but increase the run time of the experiment. In this section, we discuss the temperature control modes and their implementation in our $\mu$GC.

4.3.1 Isothermal Operation

The column temperature is maintained at a constant temperature throughout the GC run throughout the isothermal temperature condition. Analytes with similar retention times are generally subjected to the isothermal temperature operation. Calculating the temperature setpoint for the isothermal operation is very important. At low column temperature, the initial peaks will be well-spaced. However, the compounds that stay longer on the column will show the dispersion of the peaks, also known as band-broadening, and thus resulting in longer analysis runs. Higher temperature may reduce the band-broadening as the analytes...
(a) Multiple Setpoints through GUI

(b) Temperature Profile

Figure 4.5: Automated Device Calibration with Multiple Setpoints
will elute faster but will result in co-elution making separation and identification difficult. Thus typically, the temperature is set to around the midpoint of the boiling point range of the analytes in the mixture. Figure 4.6 shows the $\mu$SC temperature profile for isothermal experiment with a setpoint of 100°C.

![Temperature Profile](image)

Figure 4.6: $\mu$SC Isothermal Experiment with Setpoint 100°C

### 4.3.2 Temperature Programming Operation

The drawbacks of the isothermal mode are mitigated by temperature programming mode. The temperature program refers to the ability to maintain the lower column temperature for a short period of time and then heat the column to a setpoint with a predetermined ramp rate during the GC run. Thus, lower temperature at the start of the run allows better resolution for analytes with lower retention time. Increasing the column temperature with a ramp rate
forces the heavier compound to exit the column faster. Temperature programming reduces the band-broadening resulting in sharper peaks for the heavy compounds in the mixture and run-time of the GC experiments. Figure 4.7 shows the $\mu$SC temperature profile for a temperature programming experiment with ramps rates $10^{\circ}C/min$, $15^{\circ}C/min$ and $50^{\circ}C/min$ with a final setpoint $80^{\circ}C$.

4.4 Vacuum Outlet $\mu$GC Experiment

After completing the characterization experiments, we started working on building a Vacuum Outlet $\mu$GC using the components mentioned earlier. As a vacuum GC can be created without adding any other instruments, its implementation is easy and has vast applications. The vacuum outlet GCs have numerous advantages such as greater sensitivity, shorter run
times, faster elution of less volatile analytes, and reduced thermal degradation. The vacuum µGC has the following states: 1) Sample Heating, 2) Loading, and 3) Injection. The analysis is done on the host side as the µGC sends detector output to the host in real-time. Different stages of operation mentioned above are explained in detail in the following sections.

4.4.1 Sample Heating

We are using Static Headspace Sampling to introduce the gas phase of the sample into the µGC column. The liquid sample to be analyzed is kept in a sealed vial (Sample Holder) on a heater. The temperature of the heater is controller by the µGC system. As the temperature of the sample holder increases, volatile compounds begin to move into the gas phase above the sample until a state of equilibrium is reached \[25\]. Depending on the mixture, the temperature of the sample holder and hold time can be programmed to get optimum results. In this state, the flow is blocked by switching 3-port and 6-port valves.

4.4.2 Loading State

First, the separation column is heated to an initial setpoint that is closer to the room temperature (\(\sim 30^\circ C\)). The 6-port valve is switched to a position shown in the Figure 4.8, and the 3-port valve is then automatically switched to vent to allow the flow of analytes through the sample loop. Typically, the sample is loaded into the sample loop for around 30 seconds. The vital thing to note is that the sample holder maintains its temperature throughout this stage.
4.4.3 Injection State

After the sample is loaded into the sample loop, the 6-port valve is switched such that the helium will flow through the sample loop and push the analytes through the sample loop. The temperature of the separation column is increased from the 30°C to 80°C at a ramp rate of 10°C. At the start of this state, the heater for the sample holder is turned off, and the 3-port valve blocks the flow as shown in Figure 4.9.
4.5 Results

The $\mu$GC was tested by loading a mixture of five compounds. The mixture contains an equal amount (100$\mu$L) of pyridine, benzene, toluene, chlorobenzene, and C11. The sample is injected into the system using the static headspace sampling technique mentioned earlier. The sample holder was heated for 2 mins at 120°C to reach the headspace equilibrium and loaded into the sampling loop for 30 seconds using the 6-port valve. The experiment was performed four times. Average relative standard deviation (RSD) for peak areas and peak heights for all analytes were less than 2%, whereas retention time repeatability for all the peaks in samples was around 2.5%. Figure 4.10 shows the chromatograph from the TCD detector. However, due to the less sensitive TCD detector, we could see only one peak on the chromatograph. The amplitude of the peak, which indicates the concentration of the analyte, was very low. Other peaks could not be distinguished from the noise. Therefore, the
TCD detector was not useful for the μGC system. Nevertheless, we could replace the TCD sensor with a PID sensor by changing the ADC gain due to robust hardware and software design. Figure 4.11 shows the chromatograph from the PID detector connected at the end of the system. Pyridine, benzene, toluene, chlorobenzene were identified correctly, but C11 was not identified because the PID sensor had low resolution. Thus, it was co-eluting with other compounds. The μGC system separated and detected the analytes in the sample within 120 seconds. We used complex samples such as lemon essential oil, lavender, diesel, and kerosene for the next experiments. The separation results are shown in Figure 4.12. The μGC system can successfully separate low boiling volatile organic compounds (VOCs) present in the mixture. These results show that the modular μGC system can detect and separate the mixture with a much shorter sampling time and low sample volume than the conventional system to complete one entire analysis cycle.
Figure 4.11: PID Output With Temperature Programming

- Benzene
- Toluene
- Pyridine
- Chlorobenzene

Sample Holder Temperature: 130°C
μGC Temperature Programming: 30°C to 80°C
Ramp Rate: 15°C/min
Run Pressure: 26psi
Stationary Phase: [PG6614]/[NT2]
Figure 4.12: a) Lemon b) Lavender c) Diesel d) Kerosene sample separation under following conditions: Sample holder temperature: 120°C; µSC temperature programming: 30°C to 80°C; Ramp rate: 15°C/min; Run pressure: 20psi; Stationary phase: [P6614][NTf2]
Chapter 5

Data Analysis

The chromatograph data from the $\mu$GC consists of various environmental noises, shifted baseline. This severely impairs the ability to accurately identify analytes in a sample. Thus, post-processing techniques like signal smoothing and baseline correction have been studied in depth. In this chapter, we first explain in detail the baseline correction algorithm and its implementation. Then, we discuss peak detection and multi-dimensional analysis to identify the compounds in the sample.

5.1 Baseline Correction and Peak Detection

A typical $\mu$GC experiment run generates a large amount of data. The experiment data contains essential information about the compounds, such as volume, concentration, etc. However, computing the characteristics of analytes in the mixture is very difficult as the data has environmental and system noise. Therefore, it is necessary to smooth and correct the baseline of the chromatograph. Then, peaks are extracted from the baseline-corrected chromatograph. The flow of data analysis for a chromatograph is shown in Figure 5.1. The subsequent subsections discuss baseline correction, peak detection followed by the results.
5.1.1 Baseline Correction

Any chromatograph measured by the GC system generally has a wandering baseline due to varying experimental conditions such as temperature, residuals in the columns, especially if an experiment is running for hours. Baseline also varies from sample to sample, albeit the other conditions are exact \[7\]. When this drifted chromatograph is used for quantitative and qualitative analysis such as peak detection and identification, the accuracy of the GC system is impaired. Baseline Correction is essentially a preprocessing technique of taking an input signal which is mostly smooth with the exception of a few peaks and downshifting the signal proportionally so that significant peaks can meaningfully be identified through a separate algorithm. This section aims to document the process in baseline detection, which we have currently made to implement a baseline correction algorithm. One assumption that we made is that the peaks in the raw signal are positive. Based on that assumption, existing baseline correction algorithms can be categorized as follows.

- Differencing and Filtering \[24\]
- The Maximum Entropy Approach \[22\]
- Multivariate Baseline Modelling \[21\]
- Using Asymmetric Cost Function \[18\]

During our experiments, the first thing we noticed while analyzing the raw signal was that
the signal trends upwards over time (Figure 5.3a). As a natural side effect of the column heating the gas over time, the raw detector signal we get back needs to be corrected the x-axis downwards. We must remove this upward bias through baseline correction. After going through different baseline correction methods mentioned above, *Asymmetric Least Squares Smoothing (AsLS)* [12] was implemented. Implementation details and results are as follows.

**Implementation**

Let $y$ be a raw signal of length $N$, which is assumed to be sampled at equal intervals. Let $z$ be another signal with the following properties

- Must follow the trend of $y$.
- Must be smooth.

Assuming $y$ and $z$ are column vectors, $z$ can be calculated by minimizing the following regularized least square function.

$$S(z) = (y - z)^T(y - z) + \lambda z^TD^TDz,$$  \hspace{1cm} (5.1)

where $D$ is the difference matrix. Considering $D$ as a second order difference matrix, $D$ can be written as

$$D = \begin{bmatrix}
1 & -2 & 1 & 0 & \cdots & 0 & 0 \\
0 & 1 & -2 & 1 & \cdots & 0 & 0 \\
& & & & & & \\
& & & & & & \\
0 & 0 & 0 & \cdots & 1 & -2 & 1
\end{bmatrix}$$ \hspace{1cm} (5.2)

The first term in eqn 5.1 calculates the fitness to the data, whereas the second term puts penalty on non-smooth behaviour of $z$. The tuning parameter $\lambda$ maintains the balance
between the two terms. A weight vector \( w \) is introduced to correct the baseline. Let \( W \) be a diagonal matrix with \( w \) on its diagonal such that

\[
W = \text{diag}(w)
\]

By putting \( W \) in eqn 5.1, penalized least square function is obtained.

\[
S(z) = (y - z)^T W(y - z) + \lambda z^T D^T D z.
\] (5.3)

In order to minimize eqn 5.3, we will calculate partial derivative with respect to \( z^T \) and set it to zero, i.e. \( \delta S/\delta z^T = 0 \).

\[
\frac{\delta S}{\delta z^T} = -2W(y - z) + 2\lambda D^T D z = 0.
\] (5.4)

By rearranging eqn 5.4,

\[
z = (W + \lambda D^T D)^{-1} W y.
\] (5.5)

A brief pseudo-code overview of how it works looks like this

```plaintext
Listing 5.1: Baseline Correction using Asymmetric Least Squares
1  y = raw chromatograph signal
2
3  lambda = smoothness constant
4  p = symmetry constant
5
6  m = length(y)
7  D[m][m] = Unit matrix of size (m x m)
8
9  // Calculate difference between successive rows of D like
```
5.1. Baseline Correction and Peak Detection

Essentially, by applying this algorithm, we can take an input signal and offset it by its rising baseline. The next step in data processing is peak detection which is explained in the following section.

5.1.2 Peak Detection

After we successfully correct the baseline of the raw signal, we need to detect relevant peaks in the signal. Notice that, since the peaks inside of the signal can potentially erratically be polluted with some signal noise, we need a slightly more sophisticated algorithm to find these peaks. In essence, we need to take the output of the baseline correction algorithm and find peaks within the corrected signal.
As we traverse through the data set, we make decisions by using the state machine (Figure 5.2) at each data point in our set.

We then use a lookahead buffer to determine whether the point we have is a maximum (peak) or minimum. This buffer essentially checks the next N elements from the current element and checks for the following:

- To find Minima:
  - If we previously found a maximum (or are just starting the traversal), we need to find a minimum.
  - A minimum has been found when the next N elements are all LARGER than the current element - in other words, a minimum is always followed by a significant amount of points that are significantly larger than the current minimum.
To find Maxima:

- If we previously found a minimum (or are just starting the traversal), we need to find a maximum.

- A maximum has been found when the next N elements are all SMALLER than the current element - in other words, a maximum is always followed by a significant amount of points that are significantly smaller than the current minimum.

After applying baseline correction and peak detection algorithms, we discuss the results in the next section.

### 5.1.3 Results

For this experiment, we created a mixture of 22 compounds. Figure 5.3 shows output at different stages, starting from collecting raw chromatograph from the FID detector to the peak detection. Figure 5.3a shows the drifting baseline in the raw signal. The estimated baseline is then calculated using the 5.1 and the output is shown in Figure 5.3b. By subtracting the estimated baseline from the raw signal, the baseline-corrected signal is calculated as shown in Figure 5.3c. The last step is peak detection, which takes the baseline-corrected signal as input and calculates peaks in the signal. Figure 5.3d shows the peaks overlaid on the baseline-corrected signal.

After baseline correction, the peak detection algorithm was able to detect all 22 compounds. We repeated the experiments with multiple samples with a different number of analytes. The peak detection algorithm was able to detect all the peaks in the baseline-corrected signal. The relative standard deviation (RSD) for the difference between retention time in raw signal and the baseline-corrected signal was <5%. Thus, we can conclude that baseline
Figure 5.3: a) Raw chromatograph signal from FID detector, b) Baseline estimation c) Baseline corrected signal d) Peak detection on the baseline corrected signal
correction and peak detection algorithm provides satisfactory results and can be used for the applications such as multidimensional analysis which is discussed in the next section.

5.2 Multidimensional Analysis

In a typical GC system setup, a single separation column and detector are used to separate and analyze a mixture. With the advancements in the \( \mu \)GC, the column length is reduced to enable faster analysis and reduce power consumption. This, however, reduced the separation ability of the \( \mu \)GC system. One solution to this problem is to use two or more columns. \cite{15} defines multidimensional separation when the separation of targeted analytes is repeated in two or more separation columns with different polarities, which are coupled in series or parallel to the column by which the first separation occurred. The parallel \( \mu \)GC technique using different stationary phases has become a new avenue for research. As the length of the columns becomes shorter for either fast GC analysis or portable GCs, the use of multiple columns becomes more critical, especially when the sample is more heterogeneous in terms of compound polarity. A single nonpolar stationary phase cannot adequately separate all compounds \cite{14}.

5.2.1 Implementation

At the start of the experiment, the retention time for an individual compound in each stationary phase is calculated. Table 5.1 shows the retention time for the five compounds which will be used for this experiment. A mixture, \texttt{mix1}, with different combinations of compounds from the Table 5.1 are created. While making the mixtures, it is made sure that some of the compounds in these mixtures co-elute with other compounds in the same
mixture in the same stationary phase. For example, \textit{mix1} can contain benzene, heptane, and pyridine, so that benzene and heptane might co-elute in the stationary phase with ionic liquid \textit{BPY}. Now, with this setup, the objective is to identify all the compounds, co-eluting or otherwise, present in \textit{mix1}, \textit{mix2} and \textit{mix3} accurately. The remaining section explains the algorithm for multidimensional analysis in detail.

\begin{table}[h]
\centering
\begin{tabular}{|l|c|c|c|}
\hline
\textbf{Compound} & \textbf{BPY Ret Time ($t_r$)} (min) & \textbf{HOEMIM Ret Time ($t_r$)} (min) & \textbf{N1888 Ret Time ($t_r$)} (min) \\
\hline
Benzene & 0.251 & 0.0103 & 0.205 \\
Butylbenzene & 1.634 & 1.678 & 1.825 \\
Heptane & 0.105 & 0.103 & 0.154 \\
Pyridine & 1.553 & 2.367 & 1.367 \\
Ethylebenzene & 0.812 & 0.855 & 0.777 \\
\hline
\end{tabular}
\caption{Retention Time for Individual Compound in 3 Stationary Phases}
\end{table}

First, we create a master lookup table with the name and retention time for each compound. Then, the program reads the master lookup table and creates a class for each compound. The class stores a \texttt{count} of occurrences of a particular compound across all the stationary phases for a particular mixture. The \texttt{count} is initialized to zero. Now, each chromatograph for a mixture is run through the peak detection algorithm. The peak detection algorithm returns a list of peaks and the respective retention time. Now, retention times from each chromatograph can be compared with the reference retention times from the master lookup table, and error is calculated. The compound name associated with the retention time with the least error is nominated as the actual component in the mixture, and the \texttt{count} for that compound is increased by one. Higher the \texttt{count} for a compound, greater is the confidence that compound exists in the mixture.

We implemented this algorithm in Python and experimented with multiple mixtures with co-eluting compounds, and the results obtained from those experiments are discussed in the
5.2.2 Results

A mixture (sample 1) was prepared with compounds butylebenzene, heptane, and ethylbenzene. Butylebenzene and ethylbenzene co-elute in the separation column with the stationary phase of ionic liquid BPY and HOEMIM. This can be seen from the chromatograms 5.4a and 5.4b. These chromatograms show only one peak, albeit three compounds are present in the sample. But, the chromatogram 5.4c from the separation column coated with ionic liquid N1888 shows peaks for all three compounds present in the sample. Figure 5.5 shows the output from the multidimensional analysis. If a compound is present in the mixture, it is indicated as ‘Y’; otherwise, ‘-’ indicates that the compound is not found in the mixture. As we can see, the separation columns with BPY and HOEMIM were able to detect and identify only the heptane. In contrast, the separation column with N1888 as stationary phase was able to detect and identify butylebenzene, heptane, and ethylbenzene. The algorithm was able to identify all the compounds in sample 1 accurately. We also tried different mixtures with different analytes from Table 5.1, and the algorithm was able to detect co-eluting compounds accurately.

As we gained confidence in the system, we applied the algorithm on the data from the paper [14]. The author prepared a sample of 46 compounds. The details about the sample and the co-eluting peaks in the mixture can be found in the paper [14]. The multidimensional analysis described in the paper was carried out manually by the author. Thus, we used the multidimensional analysis algorithm to automatically detect and identify the compounds in the sample. Figure 5.6 shows the output of the multidimensional analysis. Separation columns with HOEMIM and N1888 as stationary phases were able to detect only 36 peaks,
Figure 5.4: Result of peak detection for Sample 1 with three compounds when passed through Separation Columns coated with a) BPY b) HOEMIM c) N1888
5.2. Multidimensional Analysis

whereas the separation column with the stationary phase BPY detected 39 peaks. Then, the peak list and the calculated retention times from these chromatograms were passed through the multidimensional analysis. The algorithm was able to identify 43 compounds out of a total of 46 compounds in the mixture. We ran this algorithm on ten different mixtures with co-eluting compounds, and it predicted the compounds in the sample with an accuracy of \( \sim 94\% \). These results show that the multidimensional analysis algorithm works well with the GC system, yielding a result that surpasses the output from the traditional GC system.

Figure 5.5: Result of Multidimensional Analysis with Sample 1
Figure 5.6: Result of Multidimensional Analysis for a Sample with 46 Compounds
Chapter 6

Conclusion

In this thesis, we demonstrated a prototype implementation of a truly modular μGC system suitable for the separation of complex compounds like diesel and kerosene. The modular architecture of the μGC system allows faster and accurate analysis over other conventional systems. We show that the robust software architecture allows replacing the microfabricated components. For example, for the vacuum μGC experiments, the μTCDs were replaced with the μPI-Ds without changing the GUI or the embedded code.

An automated way to calibrate the devices for the μGC system with a high level of accuracy was presented as a part of the thesis. We successfully implemented a vacuum outlet μGC system using the novel GUI software. A detailed result of the μGC system is reported for different samples in a simulated field environment, highlighting the advantages of such a generalized and modular μGC system. Successful implementation of the different data analysis techniques such as baseline correction and peak detection on the μGC system was demonstrated in this thesis. An algorithm for multidimensional analysis was developed, and results highlight the advantages of multidimensional analysis over conventional analysis techniques.
6.1 Future Work

The reported prototype implementation of the $\mu$GC system can be further improved by making the following design changes.

- PID algorithm can be auto-tuned by using the Ziegler-Nichols method that does auto-tuning for the PID controller.

- The current implementation of the baseline correction and peak detection algorithm requires parameter tuning to get an accurate result. One improvement is to implement the adaptive baseline correction and peak detection algorithm, which automatically adapts to the change in the chromatogram data and thus, does not require parameter tuning.

- Data analysis tab can be added to the GUI application for visualization of baseline correction, peak detection, and multidimensional analysis.

- Data analysis algorithms can be implemented in the embedded system along with the existing RTOS implementation to realize a fully independent $\mu$GC system.

The system can be further evaluated by comparing its accuracy and robustness through extended field tests in indoor and outdoor environments against conventional systems.
Bibliography


