A DESIGN OF WIRELESS LOW POWER STRAIN MEASUREMENT BASED ON IOT TECHNOLOGY

Ming Liu
University of Rhode Island, ming_liu@uri.edu

Follow this and additional works at: https://digitalcommons.uri.edu/theses

Recommended Citation
Liu, Ming, "A DESIGN OF WIRELESS LOW POWER STRAIN MEASUREMENT BASED ON IOT TECHNOLOGY" (2020). Open Access Master's Theses. Paper 1876.
https://digitalcommons.uri.edu/theses/1876

This Thesis is brought to you for free and open access by DigitalCommons@URI. It has been accepted for inclusion in Open Access Master's Theses by an authorized administrator of DigitalCommons@URI. For more information, please contact digitalcommons@etal.uri.edu.
A DESIGN OF WIRELESS LOW POWER STRAIN
MEASUREMENT BASED ON IOT TECHNOLOGY

BY

MING LIU

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

ELECTRICAL ENGINEERING

UNIVERSITY OF RHODE ISLAND

2020
MASTER OF SCIENCE

OF

MING LIU

APPROVED:

Thesis Committee:

Major Professor    Tao Wei

Otto Gregory

Godi Fisher

Nasser H. Zawia
DEAN OF THE GRADUATE SCHOOL

UNIVERSITY OF RHODE ISLAND
2020
ABSTRACT

Strain measurement is widely implemented in the civilization, mechanical, electrical and material industrial and research field. The traditional strain measurement devices are tremendous in dimension and power consumption. Most of them connect with computers with wire communication such as RS232, USB, etc. With the rapid development of the Internet of Things, many devices have been connected in wireless communication which enables long-range and low-power measurement in the industry.

This thesis is focused on the design and development of a low-power strain measurement system based on LoRa IoT technology, which requires accuracy strain gauge measurement, low-power consumption, and long-range wireless communication. Special contributions to the presented system include high-accuracy strain sensing circuit, low-power design, and long-range wireless communication. Additionally, a Graphic User Interface software is designed and developed on the computer to receive, plot the real-time strain data transfer from devices, and save to files. To verify the accuracy and effectiveness of the system, a comparative test is performed with the manufactured strain measurement device. Finally, the optimization of the system in the future is suggested. The hardware schematic, PCB files, firmware, and software are also listed in the appendix.
ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Tao Wei, who support me to do my research and finish my thesis from I started studying in his NEXT LAB. His enthusiasm for the work and broad knowledge give me great help and lots of guidance. Without his tutoring, I would not finish my master’s program. I also would like to be grateful to Prof. Otto Gregory who leads the wonderful project of the strain measurement research. Additionally, I would thank my dear lab mates Noah Burke, Clark Yao, Alfred Xu, Thomas Mauldin for their kind help and advice during my study and research.

Finally, I would like to thank my whole family for their love, support, and encouragement while I pursued my degree.
# TABLE OF CONTENTS

ABSTRACT ................................................................................................................................. ii

ACKNOWLEDGMENTS ............................................................................................................. iii

TABLE OF CONTENTS ............................................................................................................. iv

LIST OF TABLES ..................................................................................................................... vi

LIST OF FIGURES .................................................................................................................. vii

CHAPTER 1 INTRODUCTION ................................................................................................. 1

1.1 The need for IoT design ................................................................................................. 1
1.2 The need for low power design .................................................................................. 3
1.3 The design of project .................................................................................................... 4

CHAPTER 2 HIGH ACCURATE STRAIN MEASUREMENT ..................................................... 6

2.1 Strain gauge measurement .......................................................................................... 6
2.2 Optical Strain measurement ....................................................................................... 7
2.3 Strain measurement in this design ............................................................................... 9

CHAPTER 3 SYSTEM DESIGN .............................................................................................. 11

3.1 Overview of the design ............................................................................................... 11
3.2 Hardware design ......................................................................................................... 12
3.3 Firmware design ......................................................................................................... 27
3.4 Software design ......................................................................................................... 48

CHAPTER 4 EVALUATION ..................................................................................................... 51

4.1 Strain gauge measurement evaluation ....................................................................... 51
4.2 System performance evaluation ................................................................................ 57

CHAPTER 5 CONCLUSION AND OPTIMIZATION .............................................................. 63
5.1 Conclusion .................................................................................................................. 63
5.2 Optimization for the system design in the future ................................................. 65

APPENDICES ................................................................................................................. 68

A.1 Hardware design...................................................................................................... 68
A.2 Firmware design..................................................................................................... 84
A.3 Software design ..................................................................................................... 103

BIBLIOGRAPHY ............................................................................................................. 106
# LIST OF TABLES

| TABLE | PAGE |
|-------|------|
| Table 1.1 A list of IoT technologies. | 2 |
| Table 3.1 ADG707 Truth Table | 11 |
| Table 3.2 Pin map between ADG707 and STM32 | 30 |
| Table 3.3 Timing Characteristics of AD7799 | 31 |
| Table 3.4 On-chip registers of AD7799 | 33 |
| Table 3.5 Configuration of SPI2 port on STM32 | 34 |
| Table 3.6 Default values of some registers of AD7799 | 35 |
| Table 3.7 Initialized configuration of AD7799 | 36 |
| Table 3.8 Low-power mode of STM32L073 | 38 |
| Table 3.9 Configuration of SPI1 port on STM32L073 | 40 |
| Table 3.10 LoRa modem performances example | 41 |
| Table 3.11 Range of spreading factors | 41 |
| Table 3.12 Cyclic coding overhead | 42 |
| Table 3.13 LoRa bandwidth with different parameter | 43 |
| Table 3.14 LoRa operation mode functionality | 44 |
| Table 3.15 DIO mapping of LoRa Mode | 47 |
| Table 4.1 Main specifications of Model P3 | 55 |
| Table 4.2 Measure configuration of 5244D | 58 |
# LIST OF FIGURES

| FIGURE | PAGE |
|--------|------|
| Figure 1.1 Strain measurement wireless network | 3 |
| Figure 2.1 Typical strain gauge | 7 |
| Figure 2.2 Schematic of Fiber Bragg Gating sensors | 8 |
| Figure 2.3 Typical digital image correlation acquisition system | 9 |
| Figure 3.1 Schematic of the entire system | 12 |
| Figure 3.2 Typical application of LT3042 | 14 |
| Figure 3.3 Topology of power system | 14 |
| Figure 3.4 PCB layout of LT3042 | 15 |
| Figure 3.5 Typical strain gauge installation | 16 |
| Figure 3.6 Wheatstone bridge circuit diagram | 17 |
| Figure 3.7 Function block diagram of ADG707 | 18 |
| Figure 3.8 Schematic of ADG707 | 19 |
| Figure 3.9 Functional block of AD7799 | 20 |
| Figure 3.10 Pin connection of AD7799 | 21 |
| Figure 3.11 Diagram of STM32L073 and the peripherals | 23 |
| Figure 3.12 Clock configuration of STM32L073 | 24 |
| Figure 3.13 Schematic of the CH340G circuit | 25 |
| Figure 3.14 SX1272 block diagram | 26 |
| Figure 3.15 Wireless communication circuit | 27 |
| Figure 3.16 Infrastructure of Keil MDK development environment | 28 |
Figure 3.17 STM32CubeMX diagram ................................................................. 28
Figure 3.18 AD7799 read and write cycle timing diagrams ......................... 31
Figure 3.19 SPI connection between STM32 and AD7799 .......................... 34
Figure 3.20 Single conversion timing diagram ............................................. 37
Figure 3.21 Function block of data sampling .............................................. 37
Figure 3.22 Pin connection between STM32 and SX1272 ............................ 40
Figure 3.23 LoRa packet structure ............................................................... 44
Figure 3.24 LoRa data buffer ..................................................................... 45
Figure 3.25 SPI timing diagram (single access) ........................................... 46
Figure 3.26 LoRa modulation transmission sequence ..................................... 47
Figure 3.27 LoRa receive sequence .............................................................. 48
Figure 3.28 Graphical interface of Windows software .................................... 49
Figure 3.29 Flow chart of the software ......................................................... 50
Figure 4.1 Strain drift of strain gauge by temperature ................................... 52
Figure 4.2 Strain drift of strain gauge after temperature compensation .......... 53
Figure 4.3 Overview of P3 strain indicator and recorder ............................... 54
Figure 4.4 Testing platform of the strain gauge measurement system ............ 56
Figure 4.5 Force sequence of the test platform .......................................... 57
Figure 4.6 Compare testing with P3 and our designed system ....................... 57
Figure 4.7 Frequency spectrum of the output of the power systems ............... 59
Figure 4.8 Comparison of the current of the full-function design ................. 60
Figure 4.9 Comparison of the current of the compact design ....................... 61
Figure 4.10 Long-time testing for temperature compensation ....................... 61
Figure 5.1 Three generations of design................................................................. 63

Figure 5.2 Architecture of LoRaWAN ................................................................. 66
CHAPTER 1

INTRODUCTION

Strain measurement is a key element to detect the quantitative deformation of the object material. The material of a certain component or object can be elongated or compressed, and the strain of the object will change due to the factor of external force, thermal influence, and internal force.

The Strain measurement has been widely accepted and implemented in the civilization, mechanical, electrical, and material industrial and research field. To get the result of strain measurement, various measurement principles and devices have been achieved in many industrial areas.

This design is to develop an innovative high-resolution strain measurement with the Internet of Things technology and low-power consumption to meet the increasing need for in-situ strain measurement.

1.1 The need for the IoT design

The Internet-of-Things (henceforth: IoT), which is a new technology paradigm envisioned as a global network of devices and machines interact with each other, has been growing tremendously in the last decades with the internet expanded extensively and fast-developing. Many of the most prestigious management-consulting companies such as Gartner, McKinsey give a very optimistic estimate about the future of the IoT. Gartner says 5.8 billion enterprise and automotive IoT Endpoint will be used in 2020, a 21% increase from 2019. Statista Research Department predicts 75.44 billion
devices will be connected with the IoT worldwide by 2025. The internet of things devices such as machines and sensors are expected to generate 79.4 Zettabytes of data in 2025 which is predicted by IDC (International Data Corporation).

Among all the IoT domains, the industrial Internet of Things has been great advances with the technologies of the industrial wireless network (IWNs), big data, and cloud computing. These emerging technologies have brought great opportunities for promoting industry upgrades and allowed the revolution of the next industrial generation namely Industry 4.0. With electrical engineering development, there are many kinds of IoT technologies that have been invented as shown in Table 1.1.

Table 1.1 A list of IoT technologies

| Technology       | Frequency                  | Data Rate         | Range            | Power Usage | Cost  |
|------------------|----------------------------|-------------------|------------------|-------------|-------|
| 2G/3G            | Cellular Bands            | 10 Mbps           | Several Miles    | High        | High  |
| Bluetooth/BLE    | 2.4Ghz                    | 1, 2, 3 Mbps      | ~300 feet        | Low         | Low   |
| 802.15.4         | subGhz, 2.4GHz            | 40, 250 kbps      | > 100 square miles | Low        | Low   |
| LoRa             | subGhz                    | < 50 kbps         | 1-3 miles        | Low         | Medium |
| LTE Cat 0/1      | Cellular Bands            | 1-10 Mbps         | Several Miles    | Medium      | High  |
| NB-IoT           | Cellular Bands            | 0.1-1 Mbps        | Several Miles    | Medium      | High  |
| SigFox           | subGhz                    | < 1 kbps          | Several Miles    | Low         | Medium |
| Weightless       | subGhz                    | 0.1-24 Mbps       | Several Miles    | Low         | Low   |
| Wi-Fi            | subGhz, 2.4Ghz, 5Ghz      | 0.1-54 Mbps       | < 300 feet       | Medium      | Low   |
| Wireless HART    | 2.4Ghz                    | 250 kbps          | ~300 feet        | Medium      | Medium |
| ZigBee           | 2.4Ghz                    | 250 kbps          | ~300 feet        | Low         | Medium |
| Z-Wave           | subGhz                    | 40 kbps           | ~100 feet        | Low         | Medium |
Traditional strain measurement devices such as the P3 strain indicator manufactured by VISHAY and MM120 strain gauge tester by Micro-Measurement are all standalone devices. The data measured can be transferred to the computer with serial or USB cable. The devices bring high prices and low flexibility in the real industry field and will get worse when the number of test points becomes higher.

Combining the strain measurement and the IoT technology can bring up a wireless strain measurement network. The real-time strain data can be accessed by the local network or cloud remotely. All the data can be calculated or processed on computers or the cloud. This brings huge convenience for data monitoring and analysis to the users. The strain measurement wireless network diagram is shown in Fig 1.1.

![Strain measurement wireless network diagram](image)

**Fig 1.1 Strain measurement wireless network**

1.2 The need for the low power design

With the development of electronics, the chips and components of the electrical device have been smaller and lower power consumption design, especially in the wireless field. All the IoT technologies focus on the low-power technique which
enables the devices to have long-time performance supplied by the battery. Low power
design also enables the strain measurement device to be free of power cable and
expand the working range in various applications.

1.3 The design of the project

The motivation of this project is to design a wireless low power strain
measurement based on IoT technology which can be powered. The work aims to
increase the measurement accuracy, lower the power consumption, and expand the
wireless range.

Chapter 2 investigates the state-of-art ways to measure the strain in the industry
and the lab which includes strain gauge, optical fiber sensor, and digital image
correction. In the end, it gives the choice of this design and reason.

Chapter 3 shows the detailed design of the hardware, firmware, and software of
the system. It includes power system design, strain gauge sensing circuit, analog to
digital conversion, microcontroller, and wireless communication. Some of the system
diagrams, schematics, and PCB layouts are shown in this chapter. Firmware and
software of the system are also shown in this chapter. The configuration and parameter
of the peripherals such as ADC, multiplexer, wireless chip are given and the flow
chart of the software is illustrated in detail.

Chapter 4 concludes the evaluation and verification of the design. It shows the
power performance, temperature compensation, and performance comparison with the
manufactured device.

Chapter 5 gives several optimization options for design in the future.
Finally, in the appendix, the schematics, PCB layouts, and the code of the system are illustrated in detail.
CHAPTER 2

HIGH ACCURATE STRAIN MEASUREMENT

There are several kinds of technologies to measure the strain of the object: electrical and optical. The most commonly used instruments to measure strain are electrical strain gauge. There also optical methods to measure the strain which are mainly Fiber Bragg Gating sensors and digital image correlation.

2.1 Strain gauge measurement

The strain gauge is invented by Edward E. Simmons and Arthur C. Ruge in 1938, the most common structure of strain gauge consists of an insulating flexible backing which supports a metallic foil pattern. The gauge can be stuck to the object to be measured by a specific adhesive.

In the elastic deformation range of materials, the methods of calculating the material stresses from the measured strains are based on the Hooke’s Law of which the simple form is:

\[ \sigma = \varepsilon \cdot E \]

(1.1)

where \( \sigma \) is material stress for which the unit is [N/mm\(^2\)], \( \varepsilon \) is the strain for which the unit is [m/m] and \( E \) is the modulus of elasticity, also called Young’s modulus, for which the unit is [N/mm\(^2\)].
A typical strain gauge contains a thin, long conductive strip in a zig-zag pattern of parallel lines as shown in Fig 2.1. The relationship between the resistance change of strain gauge and strain of the measured material object can be expressed as:

\[ K = \frac{\Delta R}{\Delta \varepsilon} \frac{R}{\varepsilon} \]  

(1.2)

where \( K \) is defined as the gauge factor of the strain gauge, \( \Delta R \) is the resistance change due to strain and \( R \) is the initial resistance, and \( \varepsilon \) is the strain to which the measurement is subjected.

The advantages of the strain gauge are that it is sensitive, inexpensive, with the strong output signal (high GF), and high-pressure range. The disadvantage of strain gauge is highly sensitive to temperature variation.

2.2 Optical strain measurement

2.2.1 Optical fiber strain sensor measurement
The optical fiber strain sensors, also called Fiber Bragg Gating sensors, were first available for commercial use in 1995 from 3M and Photonics. (Strain Measurement with Fiber Bragg Gating Sensors). Since 2000 there are more than 20 companies offer Fiber Bragg Gating sensors.

To create the actual strain sensor, the optical fiber during production with a named Fiber Bragg Gating which is a pattern of material interference. When the light travels in the fiber and hits the Fiber Bragg Gating pattern, parts of the light are reflected, while others pass through as shown in Fig 2.2. The reflected light shows the minor difference of the fiber from the rest caused by the strain or temperature. (From HBM website)

![Fig 2.2 Schematic of Fiber Bragg Gating sensors](image)

Compared with a traditional electrical strain gauge, the Fiber Bragg Gating sensors match well with the new composite materials such as glass or carbon fiber-reinforced composites. They can be integrated into the object or adherence on the surface. The measuring range is very high(<10,000μm/m) and well suitable for highly stressed composite construction. There are also some weaknesses for FBG sensors of
which the most disadvantage is high price and complex construction for the measurement.

2.2.2 Digital image correlation strain measurement

From the twenty-first century, two-dimensional digital image correlation (2D DIC) has been widely accepted and implemented in the mechanical experiments. It directly provides full-field strains and displacement by comparing the digital images of the object before and after deformation as shown in Fig 2.3. (Two-dimensional digital image correlation for in-plane displacement and strain measurement: a review)

![Diagram of digital image correlation system]

Fig 2.3 Typical digital image correlation acquisition system

Digital image correlation strain measurement also has some advantages and disadvantages. The great advantage of digital image correlation strain measurement is simple measurement setup and specimen preparation which only a fixed CCD camera is needed for recording. But the disadvantages are also obvious that the measurement heavily depends on the quality of the CCD imaging system.

2.3 Strain measurement in the design
This design is to develop a wireless strain measurement network in low power and low price. To minimize the device size and the price, this device uses the strain gauge to measure the strain of the system.
CHAPTER 3

SYSTEM DESIGN

This chapter presents the detailed design of the system which contains hardware, firmware, and software. The first section discusses the overview of the design. The second section shows the hardware design which includes power design, sensing design, analog conversion design, microcontroller design, and communication design. The next one states the firmware design for all the hardware functions which includes analog to digital conversion, low power design, wireless communication design. Finally, the design of Graphic User Interface software which receives, plots and saves data from MCU is discussed in the software design section.

3.1 Overview of the design

This design is a system that contains the whole architecture of the wireless data sampling system in the industrial field and the schematic is showed in Fig 3.1.

The system has eight-channel signal input which contains an 8 to 1 channel multiplexer. Because the signal is generated by the strain gauge sensor and the measurement is to detect the change of the resistance of the sensor. The change of resistance is so small that a Wheatstone bridge is implemented to enlarge the sensitivity and change the resistance signal to the voltage signal for the next sampling.

To acquire the voltage signal, an ADC is used for converting the voltage signal into the digital signal. The digital signal is transferred to the MCU by the SPI port and processed inside the MCU.
The MCU is the core part of the hardware system and in charge of data sampling and calculating. After calculation, the MCU transfers the strain data to the desktop or laptop through wireless modules.

Same as other digital systems, the power system is also critical to the whole design. LDO power chips are used to supply all the parts' power.

3.2 Hardware design

3.2.1 Power design

The power supply module is one of the important parts of the embedded system. Except for the power supply function, the quality of the power supply also has a great influence on the performance of the system. Inappropriate power supply design will
bring lots of noise to the whole system and downgrade the performance of the system, especially of the analog signal data acquisition subsystem.

In embedded system power design, there are two kinds of power supplies one is the switch-mode power supply and the other is the low-dropout linear regulator. Switch-mode power supplies (SMPS) are the most popular power supply because of their high efficiency. The defining feather of SMPSs is that they store energy in a capacitor or inductor, and repetitively switches its transistor on and off. SMPSs’ main benefit is the extremely high efficiency and low heat level and the efficiency level typically is above 80%. While the main disadvantages of SMPS are complexity, cost, and the high levels of noise and ripple which dramatically decrease the accuracy of the output voltage.

LDOs are simply regulators compared with SMPSs and don’t have the inductor, capacitor, or switcher. LDOs drop excess input voltage across a transistor which operates in the active region and creates a power supply with very simple regulation. The advantage of LDO is very little noise and requires no inductor for operation. They create highly accurate and low noise output voltage which are often used in the low-power application.

Base on the advantage and disadvantage between LDOs and SMPSs, the design choose the LT3042 made by Analog Device Inc. to be the power regulator for the system. LT3042 is a high-performance low dropout linear regulator feathering ultralow noise and ultrahigh PSRR for powering noise-sensitive RF applications. The schematic of the typical application for LT3042 is shown in Fig3.2.
Fig 3.2 Typical application of LT3042

LT3042 supplies 200mA at a typical 350mV dropout voltage, $2\mu V/\sqrt{Hz}$ spot noise from 10kHz to 10MHz, and $0.8\mu V_{\text{RMS}}$ in a 10Hz to 100kHz bandwidth.

The wireless low-power data sampling system is essentially a mixed-signal design that contains the precise analog to digital conversion and high-frequency RF communication. To isolate the mutual influence of the analog signal and RF signal, two LT3042s are used to supply the analog circuit and digital circuit separately. The topology of the whole power system is shown in Fig 3.3.

Fig 3.3 Topology of power system
To minimize the signal influence of analog circuits and digital circuits, the layout of analog ground and digital ground are separately and connected to the input power ground at one point.

![PCB layout of LT3042](image)

**Fig 3.4 PCB layout of LT3042**

The PCB layout for LT3042 is shown in Fig 3.4. To minimize the effects of PCB trace and solder inductance by Kelvin, connect OUTS and SET pin capacitor GND(C24) directly to the output capacitor(C22) terminals using split capacitor techniques.

Transients are short duration spikes in voltage or current that could damage the circuit in many ways. In this design, to suppress the transients, The Transient Voltage Suppressor(TVS) is used to suppress transients. The TVS is ACPDQC3V3T-HF and manufactured by Comchip. The working peak reverse voltage is 3.3V and the typical breakdown voltage is 4.1V.
3.2.2 Sensing design

Sensing design is the most important part of the analog signal design which contains strain gauge selection and analog signal detection circuit design.

The various strain measurements are discussed in detail in chapter 2. To meet the requirement the design of and lower the cost, the strain gauge is chosen to be the sensor to measure the strain. For the signal of strain gauge is very small, Wheatstone bridge is used to detect the strain gauge signal which is suitable for weak signal detection.

3.2.2.1 Strain gauge

In this design, precision the strain gauge SGD-3/350-LY11 manufactured by OMEGA Engineering Inc. is chosen to be the sensor. The rugged construction and flexibility of the OMEGA strain gauge make them suitable for highly accurate static and dynamic measurements. The tolerance of the gauge is ±0.30% and the resistance is 350 ohms. The typical installation of the strain gauge installation is shown in Fig 3.5.

![Fig 3.5 Typical strain gauge installation](image)

3.2.2.2 Strain detection circuit
The Wheatstone bridge is an electrical circuit used to measure an unknown electrical resistance by balancing two legs of a bridge circuit, one leg of which includes the unknown component. The primary benefit of the circuit is its ability to provide extremely accurate measurements in contrast with something like a simple voltage divider. Fig 3.6 shows the Wheatstone bridge circuit diagram.

![Wheatstone bridge circuit diagram](image)

To analyze the circuit based on Kirchhoff’s first and second laws, $V_g$, the output voltage of galvanometer can be calculated by the equation below:

$$V_g = \left(\frac{R_2}{R_1+R_2} - \frac{R_X}{R_X+R_3}\right)V_S$$  \hspace{1cm} (2.1)

where $V_S$ is the voltage from P1 to P2, $V_S$ is the voltage of the supply.

So the unknown $R_x$, resistance to being measured can be calculated by the equation below:

$$R_X = \frac{R_2V_S-(R_1+R_2)V_g}{R_1V_S+(R_1+R_2)V_g}R_3$$  \hspace{1cm} (2.2)
To simplify the calculate, the design chooses $R_4 = R_5 = R_6 = 360$ ohm. Because the resistance of strain gauge is 350 ohms, the output voltage $V_o$ can be positive which keeps in the best linear range of the ADC.

Resistors model ERA-3AEB361V manufactured by Panasonic Electronic Components are used in the bridge and their ±0.1% tolerance can minimize the error of the measurement.

3.2.2.3 Multiplexer

To acquire the multi-channel signal, the multiplexer is designed to connect 8 channels signals from a sensing bridge and sample the signal in the loop. ADG707 which contains low-voltage, CMOS analog multiplexer comprising eight differential channels is suitable for the design. The low-power consumption and operating supply make ADG707 ideal for battery-powered and low-power instruments.

![Fig 3.7 Function block diagram of ADG707](image)

The ADG707 switches one of eight differential inputs to a common differential by the 3-bit binary address lines A0, A1, and A2. An EN input on the chip is used to
enable or disenable the device. Fig 3.8 shows the schematic of the ADG707 in the design.

![Fig 3.8 Schematic of ADG707]

3.2.3 Data sampling design
ADC is the core part of the data sampling circuit. There many kinds of ADC available for data conversion. Due to the properties of the signal generated by the Wheatstone bridge, which frequency is not high, but amplitude is low, the sigma-delta ADC is the best choice for strain gauge signal conversion.

AD7799 made by Analog Device Inc. is chosen as the conversion chip in which the functional block diagram is shown in Fig 3.10. The AD7799 contains a low noise, 24-bit Σ-Δ ADC with three differential analog inputs. The on-chip, low-noise instrumentation amplifier enables the ADC to directly interface the ultra-small signal.
The pin connection of AD7799 is shown in Fig 3.10. As the demonstration in Fig 3.10, the analog input channel 1 is connected to the output of the Wheatstone bridge which samples the strain gauge signal. The analog channel 2 is connected to the dividing resistor which is used to monitor the analog voltage of the system.

The Wheatstone bridge always consumes power whenever the sampling takes place. So, another advantage of AD7799 is there is a low-side power switch inside the chip which can save power when the sampling system is idling. In this design, the cold side of the Wheatstone bridge is connected in series to the low-side power switch pin (PSW). In normal operation, the switch is closed to measure the output of the bridge, while in idling or standby mode the switch can be opened to significantly reduce the power consumed.

The digital port of AD7799 to the microcontroller is SPI port and connected to the microcontroller.

AVDD and DVDD are connected to analog power and digital power separately.
3.2.4 Microcontroller design

The microcontroller is the core part of the embedded system and controls the data sample, calculation, transfer through the system. In a low-power design, a powerful microcontroller is needed but the low-power performance is critical.

In this design, the STM32L073 manufactured by ST Microelectronics is selected to be the microcontroller of the system. The STM32L073 is based on the ARM Cortex-M0+ core and provide an ultra-low-power platform. The current is down to 0.29μA in Standby mode and 0.43μA in Stop mode. In Run mode, the running current could be down to 93μA/MHz. The Flash memory of STM32L073 is up to 192KB and SRAM up to 20KB which provides sufficient memory space for firmware code storage and running. In STM32L073, up to two SPIs can communicate at up to 16Mbits/s in slave and master modes in full-duplex and half-duplex communication mode. In this design, two SPI ports can communicate with ADC and wireless chip separately. There are also USART and serial wire debug port for serial communication and debugging.

The diagram of STM32L073 and the peripherals are shown in Fig 3.11. Except for the ADC and wireless chip, STM32 also controls the multiplexer, LED and button by GPIO pins.
STM32L073 also are offered in 6 different footprint packages from 48 to 100 pins which are suitable for various kinds of instruments. In this design, there are two versions of design which use different package STM32L073s. The first version contains the STM32L073RZTx which footprint is LQFP64 and chip size is 10×10mm, while the second version has an STM32L073CZT6 on the board of which footprint is LQFP48 and size is only 7×7mm. The smaller footprint contributes to the smaller board size which offers more flexibility in PCB design.

Clock management is a very important part of microcontroller design which has an impact on performance and power consumption. Three different clock sources can be used to drive the master clock SYSCLK. To enable the ultra-low-power clock source, a 32.768kHz low-speed external crystal serves as the master clock source. The clock configuration of the STM32L073 in this design is shown in Fig 3.12.
3.2.5 Communication design

To perform flexible and robust communication with a laptop, the system has wireless and UART communication at the same time.

3.2.5.1 UART communication design

UART communication is very common in embedded system design which is a basic function and easy to implement. In STM32L073 architecture, there are 5 UART ports and 1 USB port available for serial communication. The STM32 serial microcontroller offers st-link to enable serial communication between the microcontroller and the laptop via USB port.

To convert the voltage between USB and UART port, a USB bus adaptor, CH340G, is implemented in the design. The schematic of the CH340G circuit is shown in Fig 3.13.
3.2.5.1 LoRa wireless communication design

After considering all the characteristics of IoT technologies includes data rate, transmission range, power usage, and cost shown in Table1.1, LoRa technology has been selected as the wireless communication in the design.

The LoRa (abbreviated for Long Range) is a spread spectrum modulation technique derived from chirp spread spectrum(CSS) technology. LoRa device and wireless radio frequency technology is a long-range, low power wireless platform that has become the de-facto technology for Internet of Things networks worldwide. LoRa devices and the open LoRaWAN protocol enable smart IoT applications that solve the biggest challenges in the home life and manufacturing field.

In this design, the SX1272 manufactured by Semtech inc. is the main part of the LoRa wireless communication design. The SX1272 can achieve a sensitivity of over -137dBm using a low-cost crystal and bill of materials with Semtech’s patent. The high sensitivity combined with the integrated +20dBm power amplifier yields industrial
optimal range and robustness. The low RX current of 10mA and 100nA register retention enables the low power consumption ability. Except for the LoRa module, SX1272 also supports high-performance FSK modes for the system including WMBus, IEEE802.15.4g. The block diagram of SX1272 is shown in Fig 3.14.

Fig 3.14 SX1272 block diagram

The entire circuit of wireless communication based on SX1272 is shown in Fig 3.15 with the switch, crystal, filters, antenna, etc.
Fig 3.15 Wireless communication circuit

There are two main RF parts of the SX1272 circuit, one is the RF-output circuit and the other is the RF-input circuit. To optimize the design, the RF-input path and RF-output path share the same antenna with an RF switcher to lower the cost of design.

3.3 Firmware design

Firmware design is an important part of the embedded system. STM32L073 is based on ARM Cortex-M0 architecture and 32-bit RISC core. ARM Compiler provides the most complete and most accurate support for the latest architectural features and extensions of the ARM architecture.

Keil MDK is the complete software development environment for most ARM Cortex-M based microcontrollers including all the STM32 microcontrollers. MDK includes the uVersion IDE and debugger, ARM C/C++ compiler, and essential middleware components. In this design, Keil MDK is the main software development environment for firmware design. Fig 3.16 shows the infrastructure of the Keil MDK environment.
Fig 3.16 Infrastructure of Keil MDK development environment

With the ARM architecture development, the configuration of STM32 is becoming more and more complicated. To simplify the configuration, ST Inc. offers the STM32CubeMX which is a graphical tool that allows a very easy configuration STM32 microcontrollers through a step-by-step process. Fig 3.17 shows the STM32CubeMX diagram.

Fig 3.17 STM32CubeMX diagram
The configuration of STM32 is very critical in hardware coding. Be careful to configure the GPIO port, especially the Debug port. Multiple ports configuration may cause malfunction of the STM32.

3.3.1 Multi-channel Sampling

In this design, it is important to sample 8 channel strain gauge signals to the sensing circuit in a loop. As mentioned before, a multiplexer ADG707 which contains 8 differential channels is designed to sample different channels. The truth table of ADG707 is shown in Table 3.1.

Table 3.1 ADG707 Truth Table

| A2 | A1 | A0 | EN | ON Switch Pair |
|----|----|----|----|----------------|
| X  | X  | X  | 0  | NONE           |
| 0  | 0  | 0  | 1  | 1              |
| 0  | 0  | 1  | 1  | 2              |
| 0  | 1  | 0  | 1  | 3              |
| 0  | 1  | 1  | 1  | 4              |
| 1  | 0  | 0  | 1  | 5              |
| 1  | 0  | 1  | 1  | 6              |
| 1  | 1  | 0  | 1  | 7              |
| 1  | 1  | 1  | 1  | 8              |

The pins of ADG707 are connected to the GPIO port of STM32L073 and the pin map is shown in Table 3.2.
Table 3.2 Pin map between ADG707 and STM32

| No. | ADG707 | STM32 |
|-----|--------|-------|
| 1   | EN     | PA8   |
| 2   | A0     | PA9   |
| 3   | A1     | PA10  |
| 4   | A2     | PA11  |

To control the ADG707, the STM32 is configured the PA8, PA9, PA10, PA11 pins as push-pull output GPIO pins. The operation frequency of the PA port is configured to the low frequency which is 2.097 MHz.

At the beginning of the firmware, there is an initialization method that includes GPIO initialization. The relative GPIO ports are configured as push-pull output function and the operation frequency is configured as low frequency. The main function of the firmware is considered as an infinite loop that contains the data sampling, calculation, and transfer to the laptop. At the beginning of the data sampling, change the channel connected to strain gauges, and start analog to digital conversion. The firmware code is shown in the appendix.

3.3.2 Analog to Digital Data Conversion

Analog to digital data conversion is based on the AD7799 and communication between AD7799 and STM32L073. AD7799 communicates with STM32 in SPI protocol which is very popular in embedded system peripheral communication.

The diagram of the AD7799 read and write cycle timing diagram is shown in Figure 3.18.
As Fig 3.18 shows, when data transfer between AD7799 and STM32, the MSB is transferred in the first and the data will be latched on the rising edge of the SCLK signal. To enable multiple ADC operation, the CS signal must pull down during the transfer. The timing of the write and read operation must follow the timing characteristics shown in Table 3.3, otherwise, the operation of AD7799 may be failed.

Table 3.3 Timing Characteristics of AD7799
| Parameter | Limit at Tmin, Tmax | Unit   | Conditions/Commons                        |
|-----------|---------------------|--------|------------------------------------------|
| \( \tau_3 \) | 100 ns min          |        | SCLK high pulse width                    |
| \( \tau_4 \) | 100 ns min          |        | SCLK low pulse width                     |

**Read Operation**

| \( \tau_1 \) | 0 ns min |        | CS falling edge to DOUT active time      |
|             | 60 ns max |        | DVDD=4.75V to 5.25V                      |
|             | 80 ns max |        | DVDD=2.7V to 3.6V                       |
| \( \tau_2 \) | 0 ns min |        | SCLK active edge to data valid delay     |
|             | 60 ns max |        | DVDD=4.75V to 5.25V                      |
|             | 80 ns max |        | DVDD=2.7V to 3.6V                       |
| \( \tau_5 \) | 10 ns min |        | Bus relinquish time after cs inactive edge|
|             | 80 ns max |        |                                          |
| \( \tau_6 \) | 0 ns min |        | SCLK inactive edge to CS inactive edge   |

**Write Operation**

| \( \tau_7 \) | 10 ns min |        | SCLK inactive edge to DOUT high          |
| \( \tau_8 \) | 0 ns min  |        | CS falling edge to SCLK active edge setup time|
| \( \tau_9 \) | 30 ns min |        | Data valid to SCLK edge setup time       |
| \( t_{10} \) | 25 ns min |        | Data valid to SCLK edge hold time        |
| \( t_{11} \) | 0 ns min  |        | CS rising edge to SCLK edge hold time    |
STM32 controls the AD7799 via several on-chip registers which are listed in Table 3.4.

Table 3.4 On-chip registers of AD7799

| Read Address | Register Name        | Function                                                                 |
|--------------|----------------------|-------------------------------------------------------------------------|
| 0x00         | Communication Register | Indicate the next operation is whether write or read and on which register. |
| 0x00         | Status Register      | Indicate the status of AD7799.                                           |
| 0x08         | Mode Register        | Used to select the operating mode, update rate, and low-side power switch |
| 0x10         | Configuration Register | Configure the AD7799 for polar mode, buffer and burnout current         |
| 0x18         | Data Register        | Store the conversion result from AD7799                                 |
| 0x20         | ID Register          | Store identification number for AD7799                                  |
| 0x28         | IO Register          | Select the function of AIN3 pins                                         |
| 0x30         | Offset Register      | Used to store the offset data in channels                                |
| 0x38         | Full-scale Register  | Hold the full-scale calibration coefficient for AD7799                   |

To operate the registers on AD7799, all communication must start with a write operation to the communication register. The data written to the communication register determines whether the next operation is to write or read operation and to
which register to be accessed. Then, the data will be read from or written to the register chosen by the operation.

To operate the registers on AD7799, the SPI port of STM32L073 is designed to communicate in the design. There are two SPI ports on STM32L073 and SPI2 is configure as communication with AD7799 as Fig 3.19 shows.

![Fig 3.19 SPI connection between STM32 and AD7799](image)

To match the SPI communication protocol of AD7799, the configuration of the SPI2 port on STM32 is shown in Table 3.5. At the beginning of the firmware beginning, SPI2 port will be initialized as the configuration shown in Table 3.5.

**Table 3.5 Configuration of SPI2 port on STM32**

| Parameters             | Configuration |
|------------------------|---------------|
| Frame Format           | Motorola      |
| Data Size              | 8 Bits        |
| First Bit              | MSB First     |
| Prescaler              | 16            |
| Baud Rate              | 1000kBits/s   |
| Clock Polarity(CPOL)   | Low           |
| Clock Phase(CPHA)      | 1 Edge        |
When the STM32 boots up, all of the parts in the STM32 are initialized including SPI2 port. After initializing the SPI2 port following the configuration in Table 3.5, the AD7799 will be initialized to be ready for the data sample.

AD7799 initialization function includes two parts, one is a self-checking method and the other one is the registers' initialization method. To check if the SPI communication is correct, the self-checking method is to read some of the registers after the SPI initialization method and their default values are shown in Table 3.6.

Table 3.6 Default values of some registers of AD7799

| Reg Address | Register Name       | Default Value |
|-------------|---------------------|---------------|
| 0x08        | Mode Register       | 0x000A        |
| 0x10        | Configuration Register | 0x0710      |
| 0x20        | ID Register         | 0xX9          |

In the self-checking method, STM32 will read the value in the registers of AD7799 and compare it with the default values, if the values match the default ones, the firmware will run to the next step. While the values don’t match the default ones, the error exception will be thrown out and firmware will reset.

When the initialization is method complete, the AD7799 will be configured as Table 3.7 below.

Table 3.7 Initialized configuration of AD7799
| Reg Address | Register Name    | Function          | Configuration       |
|-------------|------------------|-------------------|---------------------|
| 0x08        | Mode Register    | Operating Mode    | Single-Conversion   |
|             |                  |                   | Mode                |
|             |                  | Update Rate       | 16.7Hz              |
|             |                  | Settle Time       | 120ms               |
|             |                  | Rejection@50/60Hz | 65dB                |
| 0x10        | Configuration Register | Polar Type | Bipolar            |
|             |                  | Internal Buffer   | Enable              |
|             |                  | Gain              | 32                  |
|             |                  | Channel Select    | AIN1                |

After configuration for the AD7799, the data can be sampled in an infinite loop between all the 8 channel signals which connected to different strain gauges. Because the system will sample all the 8 channel data by the multiplexer to AD7799, the data conversion mode should be configured as the single conversion and, the mode register will be configured before each sample. The timing diagram of the single conversion is shown in Fig3.20.
Fig 3.20 Single conversion timing diagram

The function block diagram of the AD7799 data sample function is shown in Fig 3.21.

Fig 3.21 Function block of data sampling
3.3.3 Low Power Design

Low power design is an important part of the system design which enables the system to work under low power consumption and for a long time. There are three main parts of low power design which include ADC, STM32 and LoRa chip parts.

Fig 3.9 shows that there is a low-side power switch in AD7799 which is connected to the cold side of the Wheatstone bridge. It can significantly reduce the power consumption when the switch opens when the system is idling.

In the Mode register, MR12 bit is Power Switch Control Bit. MR12 can be cleared to open the power switch to save power consumption. There also MR15 to MR13 bits to set the AD7799 operation mode. When the data sampling is idling, MR15 and MR13 could set 010 to configure AD7799 in idle mode.

There are seven low-power modes in STM32L073 provided to achieve the best compromise between low power consumption, short start time, and system performance. The low-power modes of STM32L073 are listed in Table 3.8.

Table 3.8 Low-power mode of STM32L073

| Mode                  | Mode Function                                                                 |
|-----------------------|-------------------------------------------------------------------------------|
| Sleep mode            | Only the CPU is stopped, all peripheral continues to operate and can wake up the CPU when an event occurs. |
| Low-power run mode    | The clock frequency and the number of enabled peripherals are both limited.   |
| Low-power sleep mode  | Both the clock frequency and the number of enabled peripherals are limited; a typical example would be to |
have a timer running at 32 kHz

| Mode                           | Description                                                                 |
|-------------------------------|------------------------------------------------------------------------------|
| Stop mode with RTC            | Achieves the lowest power consumption while retaining the RAM and register contents and real-time clock. |
| Stop mode without RTC         | Achieves the lowest power consumption while retaining the RAM and register contents. All clocks are stopped. |
| Standby mode with RTC         | Achieve the lowest power consumption and real-time clock.                   |
| Standby mode without RTC      | Achieve the lowest power consumption.                                        |

The system is designed to work as a low-power run mode because the sampling and transferring data processes are always running.

When the low-power run mode is running, the frequency of STM32L073 will be limited in 131kHz.

3.3.4 Wireless Communication Design

The main part of LoRa communication is to control the SX1272 LoRa chip through SPI1 port and GPIO pins of STM32L073. The pin connection between STM32L073 and SX1272 is shown in Fig 3.22.
STM32L073 configure SPI1 port as the master mode to communicate with SX1272 and the PA port as GPIO to operate the DIO pins of SX1272. The configuration of the SPI1 port is shown in Table 3.9.

Table 3.9 Configuration of SPI1 port on STM32L073

| Parameters                | Configuration    |
|---------------------------|------------------|
| Frame Format              | Motorola         |
| Data Size                 | 8 Bits           |
| First Bit                 | MSB First        |
| Prescaler                 | 8                |
| Baud Rate                 | 2000kBits/s      |
| Clock Polarity (CPOL)     | Low              |
| Clock Phase (CPHA)        | 1 Edge           |
| CRC Calculation           | Disable          |
| NSS Signal Type           | Software         |
As mention before, LoRa wireless communication uses a proprietary spread spectrum modulate technique. The spectrum modulation and forward error correction techniques enable the LoRa modem to increase the range and robustness of radio communication links compare to traditional FSK based modulation. In table 3.10, LoRa modem performance based on a coding 4/5 is shown.

Table 3.10 LoRa modem performances example

| Bandwidth (kHz) | Spread Factor | Nominal Rb (bps) | Sensitivity (dBm) |
|----------------|---------------|-----------------|------------------|
| 125            | 6             | 9380            | -122             |
| 125            | 12            | 293             | -137             |
| 250            | 6             | 18750           | -119             |
| 250            | 12            | 586             | -134             |
| 500            | 6             | 37500           | -116             |
| 500            | 12            | 1172            | -131             |

The LoRa modulation is performed by representing each bit of payload information by multiple chips of information. The rate at which the spread information is sent is referred to as the symbol rate (Rs), the ratio between the nominal symbol rate and chip rate is the spreading factor and represents the number of symbols sent per bit of information. The ranges of values accessible with the LoRa modem are shown in Table 3.11.

Table 3.11 Range of spreading factors

| Spreading Factor | Spreading Factor | LoRa Demodulator |
|------------------|------------------|------------------|
|                  |                  |                  |
To further improve the robustness of the link the LoRa modem employs cyclic error coding to perform forward error detection and correction. The error coding brings up a transmission overhead and the resultant additional data overhead per transmission is shown in Table 3.12.

| Coding Rate | Cyclic coding rate | Overhead Ratio |
|-------------|--------------------|----------------|
| 1           | 4/5                | 1.25           |
| 2           | 4/6                | 1.5            |
| 3           | 4/7                | 1.75           |
| 4           | 4/8                | 2              |

An increase in signal bandwidth permits the use of a higher effective data rate, thus reducing transmission time at the expense of reduced sensitivity improvement. The range of bandwidth relate to most situations is given in the table below.
Table 3.13 LoRa bandwidth with different parameter

| Bandwidth (kHz) | Spreading Factor | Coding Rate | Nominal Rb (bps) | Sensitivity (dBm) |
|-----------------|------------------|-------------|------------------|-------------------|
| 125             | 12               | 4/5         | 293              | -136              |
| 250             | 12               | 4/5         | 586              | -133              |
| 500             | 12               | 4/5         | 1172             | -130              |

The LoRa symbol rate can be determined by the equation below:

\[ R_S = \frac{BW}{2^SF} \]  

(3.1)

where BW is the programmed bandwidth and SF is the spreading factor. The transmitted signal is a constant envelope signal.

The LoRa modem employs two types of the packet format, explicit and implicit. The explicit includes a short header that contains information about the number of bytes, coding rate, and whether a CRC is used in the packet. The packet format is shown in Fig 3.23.

Fig 3.23 LoRa packet structure

To operate the LoRa modem, the MCU can access three types of digital interface LoRa, static configuration registers, status registers, and a FIFO data buffer through
SPI port. Configure registers are accessed by the SPI interface and readable in all device modes. However, they should be written only in Sleep and Standby modes.

Table 3.14 shows the operation mode and functionality of LoRa.

Table 3.14 LoRa operation mode functionality

| Operation Mode | Description |
|----------------|-------------|
| SLEEP          | Low-power mode. In this mode only SPI and configure registers are accessible. LoRa FIFO is not accessible. |
| STANDBY        | Both crystal oscillator and LoRa baseband blocks are turned on. RF front-end and PLLs are disable |
| FSTX           | This is a frequency synthesis mode for the transmission. The PLL selected for transmission is locked and active at the transmit frequency. The RF front-end is off. |
| FSRX           | This is a frequency synthesis mode for the reception. The PLL selected for the reception is locked and active at the receive frequency. The RF front-end is off. |
| TX             | When activated the SX1272 powers all remaining blocks required for transmitting, ramps the PA transmits the packet, and returns to Standby mode. |
| RXCONTINUOUS   | When activated the SX1272 powers all remaining blocks required for reception, processing all received data until a new user request is made to change the operating mode |
| RXSINGLE       | When activated the SX1272/73 powers all remaining blocks required for reception, remains in this state until a valid |
packet has been received and then returns to Standby mode.

| CAD | When in CAD mode, the device will check a given channel to detect LoRa preamble signal |

The SX1272 is equipped with a 256 byte RAM data buffer that is uniquely in LoRa mode. All accesses to LoRa FIFO data buffer are done via the SPI interface and the diagram of the user-defined memory mapping of the FIFO data buffer is shown in Fig 3.24.

![LoRa data buffer diagram](image)

Fig 3.24 LoRa data buffer

The STM32 can access the configuration register of the LoRa modem through the SPI interface via a synchronous full-duplex protocol corresponding as shown in Table 3.9. There are three modes to access the registers are provided:

- **SINGLE** access: an address byte followed by a data byte is sent for write access whereas an address byte is sent and a read byte is received for the read access. The NSS pin goes low at the beginning of the frame and goes high after the data byte.
BURST access: the address byte is followed by several data bytes. The address is automatically increased internally between each data byte. This mode is available for both read and write accesses. The NSS pin goes low at the beginning of the frame and stays low between each byte. It goes high only after the last byte transfer.

FIFO access: if the address byte corresponds to the address of the FIFO, then succeeding data bytes will address the FIFO. The address is not automatically incremented but is memorized and does not need to be sent between each data byte. The NSS pin goes low at the beginning of the frame and stays low between each byte. It goes high only after the last byte transfer.

Fig 3.25 shows a typical SPI single access to a register.

Fig 3.25 SPI timing diagram (single access)

In SX1272 communication, six of SX1272 general purpose IO pins are available used in LoRa mode. Their mapping is shown below and depends on the configuration of registers RegDioMapping1 and RegDioMapping2.

Table 3.15 DIO mapping of LoRa Mode

| DIOx Mapping | DIO5    | DIO4    | DIO3    | DIO2    | DIO1    | DIO0    |
|--------------|---------|---------|---------|---------|---------|---------|
| 00           | ModeReady | CadDetected | CadDone | FhssChangeChannel | RxTimeout | RxDone  |
| 01           | ClkOut  | PLLock  | ValidHeader | FhssChangeChannel | FhssChangeChannel | TxDone  |
| 10           | ClkOut  | PLLock  | PayloadCrcError | FhssChangeChannel | CadDetected | CadDone  |
To operate the wireless communication, the LoRa modem performs state machine to transmit and receive data in sequence. In transmit mode power consumption is optimized by enabling RF, PLL and PA block only when the packet data need to be transmitted. Fig 3.26 shows the typical LoRa transmit sequence.

![LoRa modulation transmission sequence](image)

Fig 3.26 LoRa modulation transmission sequence

As shown in Fig 3.26, data transmission is initiated by sending the TX mode request. When completion, the TxDone interrupt is issued and the radio returns to Standby mode.

The LoRa modem can work in two distinct reception modes which include single receive mode and continuous receive mode. Fig 3.27 shows the LoRa receive sequences for both single and continuous modes of operation.
3.4 Software design

This data sampling system is a whole system that contains Windows software on the laptop or desktop. The Windows software contains a Graphic User Interface that can display the data transfer from the LoRa modem and save data to the laptop.

The graphical interface of Windows software is shown in Fig 3.28.
The main graphical interface contains two main parts, one is the chart display window and another is the configuration window. There eight waveforms show in the chart display window and each waveform represents one channel strain gauge data.

The software is programmed in Python which is more popular nowadays and implemented widely in the data analysis field. Tkinter has long been an integral part of Python and provides a robust and platform independent window toolkit.

The flow of the software is shown in Fig 3.30
Because the chart display function will need more resources of the system, there are two other threads to design to receive and save strain gauge data transfer from the MCU. The detailed code is shown in the appendix.
CHAPTER 4

EVALUATION

This chapter presents the evaluation of the system. The evaluation includes two main parts, one is the strain gauge measurement performance evaluation and the system performance evaluation.

In strain gauge measurement performance evaluation, the temperature compensation will be discussed because of the influence of the environment temperature change. In the next, a serious comparison testing will be implemented to verify the system performance of our system.

We will discuss system performance evaluation. The noise performance of the power system influences the system noise dramatically. The power noise performance will be presented. At last, low power performance will be discussed and presented.

4.1 Strain gauge measurement evaluation

4.1.1 Temperature compensation of measurement

In the strain gauge measurement system, a quarter bridge is designed to convert the resistant signal to the voltage signal. There are two kinds of error, one is temperature drift of the strain gauge sensors and the other is the error caused by the lead resistances.

Stain gauge manufacturers attempt to minimize sensitivity to temperature by processing the gauge material to compensate for the thermal expansion. The temperature characteristics of SGD-3/350-LY11 are to 11ppm/°C on the steel surface
which brings up tremendous temperature drift in the testing. Fig 4.1 shows one empty strain gauge sensor change in one day and the error is up to 160 micros at most.

Fig 4.1 Strain drift of strain gauge by temperature

To remove the error caused by the temperature drift, one strain gauge of the system is used to be the temperature compensation channel. This strain gauge will not be adherence to the surface of the test object but near it. This strain gauge we call it the compensation sensor.

To compensate for the temperature drift, the resistant change of the compensation sensor will be calculated and present the temperature coefficient on the strain gauge sensors. Then the coefficient will be considered in the calculation of other channels which shows below:

\[ \Delta \varepsilon = \frac{R_x - R_{xo}}{R_{xo}} \times \frac{R_{ro}}{R_x - R_{ro}} \]  

(4.1)

where \( R_x \) is the current resistance of the strain gauge sensor of measurement channel, \( R_{xo} \) is the initial resistance of the strain gauge sensor of measurement channel, \( R_r \) is the current resistance of the strain gauge sensor of compensation channel, \( R_{ro} \) is the
initial resistance of the strain gauge sensor of the compensation channel. The compensated strain measured in one day is shown in Fig 4.2 and the fluctuating range is limited in 20 micros mostly.

![Strain drift after temperature compensation](image)

**Fig 4.2** Strain drift of strain gauge after temperature compensation

4.1.2 Comparison with commercial instrumentation

To evaluate the performance of the system, the P3 strain indicator, and recorder which is manufactured by VISHAY Micro-Measurements. The overview of the P3 strain indicator and recorder is shown in Fig4.3.
The model P3 strain indicator and recorder is a portable, battery-powered precision instrument for resistive strain gauges and strain-gauge-based transducers. P3 accepts full- half- and quarter-bridge inputs and all required bridge completion components for 120- 350- and 1000-ohm bridges are supplied. The Model P3 utilizes modern digital signal processing technology to provide excellent noise rejection and stability. Proprietary scaling and linearization algorithms provide unsurpassed
measurement accuracy for strain gauge bridge measurements. The main specifications of Model P3 are shown in Table 4.1.

Table 4.1 Main specifications of Model P3

| Specification                          | Performance                                      |
|---------------------------------------|-------------------------------------------------|
| **Bridge configuration**              | Types Quarter-, half-, and full-bridges         |
| Bridge Impedance                      | 60 to 2000 Ω                                    |
| Quarter Bridge                        | 120Ω, 350Ω, and 1000Ω ± 0.01%                   |
| Half Bridge                           | 1000Ω ± 0.01%                                   |
| **Data conversion**                   | A/D converter Delta-sigma with a programmable    |
|                                       | gain instrumentation amplifier                   |
| Resolution                            | 24bits, Noise-free resolution:18bits             |
| Normal mode                           | 50 or 60 sampling/sec                            |
| Filter                                | Integrated linear phase FIR $sinc^5$ filter.     |
|                                       | Software selectable output rate provides >120dB rejection of |
|                                       | 50 or 60Hz and higher level harmonics.          |
| **Measurement Range/Resolution**      | Strain range ± 31,000με at GF=2.000              |
| Resolution                            | ±1 με at GF=2.000                                |
| **Measurement Accuracy**              | Accuracy ±0.1% of reading ±3 counts              |
| Gauge Factor                          | Range 0.500 to 9.900                             |
The testing platform is a model 5966 universal testing system manufactured by INSTRON which capability is up to 10kN. The testing platform is shown in Fig 4.4.

Because the strain gauge on the object surface cannot be connected to two measurement systems simultaneously, performance testing will implement the same force sequence on the object separately for two measurement systems. The profile of the force sequence of the test platform is shown in Fig 4.5. The force on the object is from 0kN to 7kN and the step is 1kN. On each different force, the testing system will keep the force for 60 seconds.
To compare P3 and the designed measurement system, the test is implemented on the two systems respectively. The result of compare testing is shown in Fig 4.6. The correlation between the two measurement systems is 0.9982.

4.2 System performance evaluation

4.2.1 Power noise performance
To evaluate the performance of power noise, the Picoscope 5244D made by pico technology is used to measure the frequency spectrum of the output of the power system. The configuration of 5244D is shown in Table 4.2. The frequency spectrum of the output of the power systems is shown in Fig 4.7.

Table 4.2 Measure configuration of 5244D

| Parameter            | Configuration |
|----------------------|---------------|
| Resolution           | 14bit         |
| Spectrum range       | 16MHz         |
| Coupling mode        | DC            |

The spectrum of digital power output
4.2.2 Low-power performance

To verify the low-power design of the system, the power supply current of the system is measured between the system in idle mode and sample mode. A multimeter was serial in the power line of the system to measure the current of the system. Fig 4.8 shows the comparison of the current of the system under idle mode and sample mode of full-function design. As the figure shows, the current of the system is about 38mA in sampling mode while only 22mA in idle mode.
a) The current of the system in sample mode

b) The current of the system in idle mode

Fig 4.8 Comparison of the current of the full-function design
To further minimize the power consumption, a smaller package of STM32L073 is used as the MCU of the system. The current of the compact design in running mode decreases to about 27mA while the current in idle mode is still about 22mA, which shows in Fig 4.9.

a) The current of the system in sample mode

b) The current of the system in idle mode

Fig 4.9 Comparison of the current of the compact design
4.2.3 Longtime testing with temperature compensation

To test the temperature compensation of the system in actual strain measurement, long-time testing with a regular 3kN force waveform is implemented. The result of the long-time strain measurement within one day is shown in Fig 4.10. The drift caused by temperature is suppressed significantly.

![Fig 4.10 Long-time testing for temperature compensation](image-url)
CHAPTER 5

CONCLUSION AND OPTIMIZATION

5.1 Conclusion

5.1.1 Evolution of the system design

To make sure the strain measurement system improved, the design job has three generations of design which are prototype design, full-function design, and compact design. The hardware of the three generations of design is shown in Fig 5.1.

![Three generations of design](image)

Fig 5.1 Three generations of design
(from left to right: prototype design, full-function design, compact design)

The first generation is the prototype design which ensures that the whole concept of design can fulfill the requirement of the proposal. In this step, an evaluation board based on STM32 and LoRa technology is implemented to make sure the digital part of
the design works properly. A daughterboard, which contains the multiplexer, sensing circuit, and the analog-digital converter, was designed to sample the strain gauge signal and convert to the digital signal. The power of the daughterboard is supplied by the evaluation board and the signal transferred to the evaluation board through the IPC connector. In this step, the number of input signal channels is four and there are three boards, which are sampling daughterboard, microcontroller board, and LoRa wireless communication board, in the system. The size of the prototype design is ??*?? and three layers.

Improvements were made to the prototype design. These changes happened because tests showed that the original design is big, expensive, and low stability. The second generation of design is the full function design which contains 8 channel signal input, wireless communication, UART communication, and individual power for analog and digital circuit design. In this generation, all the analog and digital circuits are designed in a single PCB and the size is only 46×103mm (1.81×4.34 inches). The current of full function design is about 38mA in sampling mode while only 22mA in idle mode.

To minimize the system size and suitability for battery power supply, changes were made in the third-generation design. The third generation of design is the compact function which remains 8 channel signal inputs, wireless communication, and a single power module design while removing the UART communication. In a compact design, the package of STM32L073 changes from LQFP64 to LQFP48, and the chip size shrinks from 10×10mm to 7×7mm. With these efforts, the whole size of the board is limited in 38×64mm (1.49×2.5 inches). The current of the compact
function design decreased to 27mA in sampling mode compare with 38mA for full function design, which means the power consumption drops on a large scale.

5.1.2 Conclusions

To improve the traditional strain measurement, a low-power strain measurement design based on IoT technology is introduced in this thesis. The use of strain measurement is presented and the shortage of the traditional strain measurement device is introduced at first. After analysis of the majority of strain measurement methods, including strain gauge and optical measurement, the strain gauge measurement is the best choice in this design. Many IoT technologies are listed in Table 1.1. After comparing the characteristics including power usage, transfer range, and cost, etc, the LoRa technology should be the best way for this design because of its lower power consumption and longer transfer range.

In the design, there are three generations includes prototype design, full-function design, and compact design which development from prototype to realistic. The hardware, firmware, and software of the design have been introduced in details and the technical information such as schematics, PCB layouts, program code, and bill of materials are listed.

To test the performance of the system, a variety of experiments have been implemented. The comparative testing with P3, which manufactured by VISHAY, is shown that the performance of the design is the same as the commercial devices. It presents that this design is successful and can be implemented in the industry.

5.2 Optimization for the system design in the future

5.2.1 Minimize the power and dimension
In the next step, the goal of the design is to lower the power consumption and decrease the dimension of the system. With the development of the electrical engineer, the power consumption and size of the microcontroller and wireless chip has become smaller than before. In recent years, some manufacturers have delivered the microcontroller integrated with the wireless chip which decreases power consumption and size largely. For example, Microchip launched the SAM R34/R35 which includes an ultra-low-power, high-performance 32-bit microcontroller (MCU), LoRa transceiver and software stack. It offers the industry’s lowest power LoRa System-in-Package for long-range, low power design.

In the next step, the microcontroller integrated wireless function may be selected to be the MCU of the system. This will lower the power consumption and decrease the dimension of the system largely.

5.2.2 Upload data to the cloud to analyze

![Fig 5.2 Architecture of LoRaWAN](image)
In the current design, we realized the communication in the local wireless network in LoRa technology. LoRa technology also offers LoRaWAN which defines the communication protocol and system architecture for the network while the LoRa physical layer enables the long-range communication link. The LoRaWAN has the most influence in determining the battery life, network capacity, quality of service, security, and variety of applications. The architecture of LoRaWAN is shown in Fig 5.2.

There are more optimizations, such as sensors, analog circuit, digital circuit, RF circuit, for the future development. For example, higher precious strain gauge sensor can be implemented in the design for higher accuracy, half-bridge or full-bridge Wheatstone bridge can be used to remove the influence of temperature, high-integrated MCU and wireless chip will decrease the power consumption dramatically. More optimizations will make the design improved in a huge improvement in the future.
APPENDICES

A.1 Hardware design

A.1.1 Prototype design

A.1.1.1 Schematic
Page 2/2 Power circuit

A.1.1.2 PCB layouts
Overview of PCB layout

Fig 2/5 Top Layer
Fig 3/5 Second Layer

Fig 4/5 Third layer
A.1.2 Full-function design

A.1.2.1 Schematics

Page 1/4 STM32 and peripheral circuit
A.1.2.2 PCB layouts

Fig1/5 Overview of PCB Layout

Fig2/5 Top Layer
Fig3/5 Second Layer

Fig4/5 Third Layer
### A.1.2.3 Bill of material

| Qty | Value  | Package | Parts                      |
|-----|--------|---------|----------------------------|
| 4   |        | C0402   | C210, C214, C219, C220     |
| 1   |        | C0402   | C226                       |
| 3   |        | C0805   | D1, D2, D3                 |
| 2   |        | R0402   | R208, R209                 |
| 2   |        | R0805   | L1, L05                    |
| 1   | 0R     | R0603   | R203                       |
| 1   | 0.1R   | R0805   | R17                        |
| 3   | 0.1uF  | C0603   | C10, C11, C13              |
| 5   | 0.1uF  | C0402   | C203, C204, C205, C221,    |
|     |        |         | C227                       |
| 6   | 0.1uF  | C0603   | C15, C30, C123, C124,     |
|     |        |         | C127, C128                 |
| 2   | 0.47uF | C0805   | C3, C24                    |
| 1   | 0R     | INDC1005X45 | L206                 |
| 1   | 0R     | R0603   | U204                       |
| 2   | 0R     | R0603   | R8, R206                   |
| 1   | 1.2pF  | C0402   | C212                       |
| 1   | 1.8pF  | C0603   | C213                       |
| 2   | 100R   | R0402   | R129, R214                 |
|   |   |   |   |
|---|---|---|---|
| 1 | 100R | R0402 | R207 |
| 4 | 100nF | C0603 | C6, C14, C115, C130 |
| 1 | 10118193-0001LF | FCI_10118193-0001LF | J3 |
| 1 | 10K | R0603 | R30 |
| 2 | 10nF | C0603 | C8, C16 |
| 1 | 10nF | C0402 | C207 |
| 2 | 10uF | A/3216-18R | C9, C12 |
| 2 | 15pF | C0402 | C201, C202 |
| 2 | 1nF | C0402 | C217, C222 |
| 1 | 1uF | C0603 | C29 |
| 2 | 1uF | C0603 | C224, C225 |
| 1 | 1uF | C0603 | C206 |
| 2 | 200k | R0603 | R6, R21 |
| 8 | 22-23-2031 | 22-23-2031 | PN1, PN2, PN3, PN4, PN5, PN6, PN7, PN8 |
| 2 | 22-27-2021-02 | 6410-02 | J4, J5 |
| 1 | 22-27-2051-05 | 6410-05 | J2 |
| 2 | 22pF | C0402 | C17, C18 |
| 3 | 280 | R0603 | R4, R5, R9 |
| 1 | 3.3pF | C0402 | C215 |
| 2 | 33.2K | R0805 | R1, R23 |
| 1 | 33pF | C0402 | C209 |
| 5 | 360R | R0603 | R10, R11, R12, R14, R16 |
|   | Value  | Part Number | Count |
|---|--------|-------------|-------|
| 1 | 4.7k   | R0603       | 4.7k  |
| 1 | 4.7pF  | C0402       | C211  |
| 2 | 4.7pF  | C0603       | C31, C32 |
| 5 | 4.7uF  | C1206       | C1, C2, C5, C7, C21 |
| 1 | 4.7uF  | C0805       | C223  |
| 2 | 453K   | R0805       | R2, R25 |
| 1 | 47pF   | C0402       | C208  |
| 2 | 47pF   | C0402       | C216, C218 |
| 2 | 49.9K  | R0805       | R3, R26 |
| 2 | 51R    | R0603       | R7, R27 |
| 1 | ABS25-32.768KHZ-6-T | XTAL_ABS25-32.768KHZ-6-T | Y2 |
| 1 | AD7799BRUZ | RU_16 | U2 |
| 1 | ADG707BRUZ | RU_28 | U3 |
| 1 | ADG849YKSZ-500RL7 | KS_6 | U202 |
| 1 | C22 | R0603 | R15 |
| 1 | CH340G | SOIC16 | U11 |
| 1 | DSX321G | DSX321G_FOOTPRINT | OSC4 |
| 2 | EXB-N8V101JX | EXBN8V | R204, R213 |
| 2 | FSM2JSMAATR | SWITCH_FSM2JSMAA | U6, U7 |
| 2 | LQG15HS10NJ02D | INDC1005X45 | L203, L205 |
| 1 | LQG15HS6N2S02D | INDC1005X45 | L204 |
|   | Part Number   | Description       | Quantity | Notes |
|---|---------------|-------------------|----------|-------|
| 1 | LQW15AN33NJ00D | INDC1005X45       |          | L201  |
| 2 | LT3042IMSEPBF  | MSOP-10_MSE       |          | U5, U8|
| 1 | NC7SZ04M5X     | SOT95P280X145-5N  |          | U203  |
| 1 | NX2016SA       | NX2016SA_PCB      |          | U$2   |
| 1 | SMA-J-P-H-ST-EM1 | SMA-J-P-H-ST-EM1 |          | J1    |
| 2 | SPLIT_CAP      | SPLIT_CAP_1206    |          | C4, C22|
| 1 | STM32L0X3RL    | LQFP64-10X10MM    |          | U1    |
| 1 | SX1272IMLTRT   | QFN65P600X600X100-29N |          | U4    |

A.1.3 Compact design

A.1.3.1 Schematics
Fig 1/5 Peripheral circuit of STM32

Fig 2/5 Sensing and ADC circuit
Fig 3/5 LoRa communication
A.1.3.2 PCB layouts
Fig 1/5 Overview of PCB Layout

Fig 2/5 Top Layer
Fig 3/5 Second Layer

Fig 4/5 Third Layer

84
A.2 Firmware code

A.2.1 Main function

int main(void)
{
    bool isMaster = true;
    uint8_t i;
    uint32_t
    sum_check = 0;
    HAL_Init();
    SystemClock_Config();
    DBG_Init();
    HW_Init();
    
    /*Disbale Stand-by mode*/
    LPM_SetOffMode(LPM_APPLI_Id, LPM_Disable);
    
    /* Led Timers*/
    TimerInit(&timerLed, OnledEvent);
    TimerSetValue(&timerLed, LED_PERIOD_MS);
    TimerStart(&timerLed);
// Radio initialization
RadioEvents.TxDone = OnTxDone;
RadioEvents.RxDone = OnRxDone;
RadioEvents.TxTimeout = OnTxTimeout;
RadioEvents.RxTimeout = OnRxTimeout;
RadioEvents.RxError = OnRxError;

Radio.Init(&RadioEvents);
Radio.SetChannel(RF_FREQUENCY);

#if defined( USE_MODEM_LORA )
Radio.SetTxConfig(MODEM_LORA, TX_OUTPUT_POWER, 0, LORA_BANDWIDTH,
LORA_SPREADING_FACTOR, LORA_CODINGRATE,
LORA_PREAMBLE_LENGTH, LORA_FIX_LENGTH_PAYLOAD_ON,
true, 0, 0, LORA_IQ_INVERSION_ON, 3000);

Radio.SetRxConfig(MODEM_LORA, LORA_BANDWIDTH, LORA_SPREADING_FACTOR,
LORA_CODINGRATE, 0, LORA_PREAMBLE_LENGTH,
LORA_SYMBOL_TIMEOUT, LORA_FIX_LENGTH_PAYLOAD_ON,
0, true, 0, 0, LORA_IQ_INVERSION_ON, true);
#elif defined( USE_MODEM_FSK )
Radio.SetTxConfig(MODEM_FSK, TX_OUTPUT_POWER, FSK_FDEV, 0,
FSK_DATARATE, 0,
FSK_PREAMBLE_LENGTH, FSK_FIX_LENGTH_PAYLOAD_ON,
true, 0, 0, 0, 3000);

Radio.SetRxConfig(MODEM_FSK, FSK_BANDWIDTH, FSK_DATARATE,
0, FSK_AFC_BANDWIDTH, FSK_PREAMBLE_LENGTH,
0, FSK_FIX_LENGTH_PAYLOAD_ON, 0, true,
0, 0, false, true);
#else
#error "Please define a frequency band in the compiler options."
#endif

Radio.Rx(RX_TIMEOUT_VALUE);
while (1)
{
    if (res_chn0_diff > 0.001)
        res_chn0_diff = 0.0004;
    else
        res_chn0_diff += 0.0001;
res_chn1_diff = 0.0006;
res_chn2_diff = 0.0008;
res_chn3_diff = 0.0010;
res_chn4_diff = 0.0012;
res_chn5_diff = 0.0014;
res_chn6_diff = 0.0016;
res_chn7_diff = 0.0018;
*(float*)(bytes0) = res_chn0_diff;
*(float*)(bytes1) = res_chn1_diff;
*(float*)(bytes2) = res_chn2_diff;
*(float*)(bytes3) = res_chn3_diff;
*(float*)(bytes4) = res_chn4_diff;
*(float*)(bytes5) = res_chn5_diff;
*(float*)(bytes6) = res_chn6_diff;
*(float*)(bytes7) = res_chn7_diff;

for(i = 0; i < 4; i++)
{
    Buffer[i] = bytes0[i];
}
for(i = 0; i < 4; i++)
{
    Buffer[i + 4] = bytes1[i];
}
for(i = 0; i < 4; i++)
{
    Buffer[i + 8] = bytes2[i];
}
for(i = 0; i < 4; i++)
{
    Buffer[i + 12] = bytes3[i];
}
for(i = 0; i < 4; i++)
{
    Buffer[i + 16] = bytes4[i];
}
for(i = 0; i < 4; i++)
{
    Buffer[i + 20] = bytes5[i];
}
for(i = 0; i < 4; i++)
{
    Buffer[i + 24] = bytes6[i];
}
for(i = 0; i < 4; i++)
{
    Buffer[i + 28] = bytes7[i];
}

PRINTF("%s\n", &PingMsg);
PRINTF("%f\n", res_chn0_diff);
for(i = 0; i < 32; i++)
{
    sum_check += Buffer[i];
}
for(i = 0; i < 32; i++)
{
    PRINTF("%c", Buffer[i]);
}

Buffer[32] = sum_check;
Radio.Send(Buffer, 32);
DelayMs(500);

A.2.2 Initial function of AD7799

/*
**AD7799 Initial Program
*/
uint8_t AD7799_Init(void)
{
    uint16_t AD7799_ID;
    uint8_t AD7799_Data[2] = { 0xff, 0xff };
    uint16_t AD7799_Val, AD7799_Temp;

    AD7799_Reset();
    AD7799_ID = AD7799_ReadReg(ADC_REG_ID);
    if ((AD7799_ID & 0xff00) != 0x4900)
        return 1;
    AD7799_ID = AD7799_ReadReg(ADC_REG_CONFIG);
    AD7799_ID = 0x071f;
    AD7799_WriteReg(ADC_REG_CONFIG, AD7799_ID);
    AD7799_ID = 0x0000;
    AD7799_ID = AD7799_ReadReg(ADC_REG_CONFIG);
    AD7799_Reset();
    AD7799_ID = AD7799_ReadReg(ADC_REG_CONFIG);

    AD7799_Calibration(chn1, Gain32);
    AD7799_Val = 0x0610;          //Bipolar, gain=64, REF_DET DISABLE, BUF Enable, AIN1 Channel
    AD7799_WriteReg(ADC_REG_CONFIG, AD7799_Val);
    AD7799_Temp = AD7799_ReadReg(ADC_REG_CONFIG);

    AD7799_Val = 0x300f;          //single convert, PSW enable frequency of adc is 4.17Hz
    AD7799_WriteReg(ADC_REG_MODE, AD7799_Val);
    AD7799_Temp = AD7799_ReadReg(ADC_REG_MODE);
    return 0;
}
A.2.3 Reset function of AD7799

```c
void AD7799_Reset(void)
{
    uint8_t  AD7799_Com[4] = { 0xFF,0xFF,0xFF,0xFF };  
    HAL_GPIO_WritePin(AD_CS_PORT, SPI2_NSS_Pin, GPIO_PIN_RESET); 
    HAL_SPI_Transmit(&hspi2, AD7799_Com, 4, 50);  
    HAL_GPIO_WritePin(AD_CS_PORT, SPI2_NSS_Pin, GPIO_PIN_SET); 
}
```

A.2.4 Read register function of AD7799

A.2.5 Write register function of AD7799

```c
uint16_t        AD7799_ReadReg(uint8_t reg)
{
    uint8_t       AD7799_Command;  
    uint8_t       AD7799_DumCom[2] = {0xff, 0xff};  
    uint8_t       AD7799_Data2[2] = {0xff, 0xff};  
    uint16_t      AD7799_Val;  
    AD7799_Command = reg|0x40;  
    HAL_GPIO_WritePin(AD_CS_PORT, SPI2_NSS_Pin, GPIO_PIN_RESET);  
    HAL_SPI_Transmit(&hspi2, &AD7799_Command, 1, 50); //write register  
    HAL_SPI_Receive(&hspi2, AD7799_Data2, 2, 50);  
    HAL_GPIO_WritePin(AD_CS_PORT, SPI2_NSS.Pin, GPIO_PIN_SET);  

    AD7799_Val = (uint16_t)AD7799_Data2[0];  
    AD7799_Val = AD7799.Val << 8;  
    AD7799.Val += (uint16_t)AD7799_Data2[1];  
    return        AD7799.Val;  
}
```

A.2.6 Calibration function of AD7799

```c
uint8_t        AD7799_Calibration(uint8_t Channel, uint8_t Gain)
{
    uint8_t       AD7799_Data[2] = { 0xff, 0xff };  
    uint8_t       ret;  
    uint16_t      AD7799.Val, AD7799.Temp;  
    AD7799_Data[0] = 0x00 | Gain;  
    AD7799_Data[1] = 0x10 | Channel;  
    AD7799.Val = (uint16_t)AD7799_Data[0];  
    AD7799.Val = AD7799.Val << 8;  

    return        ret;  
}
```
AD7799_Val += (uint16_t)AD7799_Data[1];

AD7799_WriteReg(ADC_REG_CONFIG, AD7799_Val);
AD7799_Temp = AD7799_ReadReg(ADC_REG_CONFIG);
AD7799_Data[0] = 0x80;
AD7799_Data[1] = 0x0F;
AD7799_Val = (uint16_t)AD7799_Data[0];
AD7799_Val = AD7799_Val << 8;
AD7799_Val += (uint16_t)AD7799_Data[1];
AD7799_WriteReg(ADC_REG_MODE, AD7799_Val);

ret |= AD7799_WaitBusy();
AD7799_Temp = AD7799_ReadReg(ADC_REG_MODE);

AD7799_Data[0] = 0xA0;
AD7799_Data[1] = 0x0F;
AD7799_Val = (uint16_t)AD7799_Data[0];
AD7799_Val = AD7799_Val << 8;
AD7799_Val += (uint16_t)AD7799_Data[1];
AD7799_WriteReg(ADC_REG_MODE, AD7799_Val);

ret |= AD7799_WaitBusy();
AD7799_Temp = AD7799_ReadReg(ADC_REG_MODE);

return ret;
}

A.2.7 Read data function of AD7799

uint32_t AD7799_GetData(void)
{
    uint8_t AD7799_Command = 0;
    uint8_t AD7799_DumCom[3] = { 0xff, 0xff, 0xff };  
    uint8_t Conv_data[3];
    uint8_t AD7799_Data1[2];
    uint16_t AD7799_Data;
    uint16_t command1 = 0;
    AD7799_Data1[0] = 0x20;
    AD7799_Data1[1] = 0x0A;
    HAL_GPIO_WritePin(AD_CS_PORT, SPI2_NSS_Pin, GPIO_PIN_RESET);

    AD7799_Command = ADC_REG_MODE;
    HAL_SPI_Transmit(&hspi2, &AD7799_Command, 1, 50);
    HAL_SPI_Transmit(&hspi2, AD7799_Data1, 2, 50);

    HAL_GPIO_WritePin(AD_CS_PORT, SPI2_NSS_Pin, GPIO_PIN_RESET);
    HAL_SPI_Transmit(&hspi2, &AD7799_Command, 1, 50);
    HAL_SPI_Transmit(&hspi2, AD7799_Data1, 2, 50);

    return ret;
}
AD7799_Command = 0x18 | 0x40;
HAL_SPI_Transmit(&hspi2, &AD7799_Command, 1, 50);
HAL_Delay(200);

HAL_SPI_TransmitReceive(&hspi2, AD7799_DumCom, Conv_data, 3, 100);
HAL_GPIO_WritePin(AD_CS_PORT, SPI2_NSS_Pin, GPIO_PIN_RESET);
AD7799_Data = ((uint32_t)Conv_data[0]) << 16;
AD7799_Data += ((uint32_t)Conv_data[1]) << 8;
AD7799_Data += (uint32_t)Conv_data[2];
return AD7799_Data;

A.2.8 Select channel function

void select_channel(uint8_t channel_number)
{
    switch(channel_number)
    {
        case 0x04:
            HAL_GPIO_WritePin(A0_PORT, A0_Pin, GPIO_PIN_RESET); //A0 reset
            HAL_GPIO_WritePin(A1_PORT, A1_Pin, GPIO_PIN_RESET); //A1 reset
            HAL_GPIO_WritePin(A2_PORT, A2_Pin, GPIO_PIN_RESET); //A2 reset
            HAL_GPIO_WritePin(EN_PORT, EN_Pin, GPIO_PIN_SET); //EN set
            break;
        case 0x05:
            HAL_GPIO_WritePin(A0_PORT, A0_Pin, GPIO_PIN_RESET); //A0 reset
            HAL_GPIO_WritePin(A1_PORT, A1_Pin, GPIO_PIN_RESET); //A1 reset
            HAL_GPIO_WritePin(A2_PORT, A2_Pin, GPIO_PIN_RESET); //A2 reset
            HAL_GPIO_WritePin(EN_PORT, EN_Pin, GPIO_PIN_SET); //EN set
            break;
        case 0x06:
            HAL_GPIO_WritePin(A0_PORT, A0_Pin, GPIO_PIN_RESET); //A0 reset
            HAL_GPIO_WritePin(A1_PORT, A1_Pin, GPIO_PIN_SET); //A1 set
            HAL_GPIO_WritePin(A2_PORT, A2_Pin, GPIO_PIN_RESET); //A2 reset
            HAL_GPIO_WritePin(EN_PORT, EN_Pin, GPIO_PIN_SET); //EN set
            break;
        case 0x07:
            HAL_GPIO_WritePin(A0_PORT, A0_Pin, GPIO_PIN_RESET); //A0 reset
            HAL_GPIO_WritePin(A1_PORT, A1_Pin, GPIO_PIN_SET); //A1 set
            HAL_GPIO_WritePin(A2_PORT, A2_Pin, GPIO_PIN_RESET); //A2 reset
            HAL_GPIO_WritePin(EN_PORT, EN_Pin, GPIO_PIN_SET); //EN set
            break;
        case 0x03:
            HAL_GPIO_WritePin(A0_PORT, A0_Pin, GPIO_PIN_RESET); //A0 reset
            HAL_GPIO_WritePin(A1_PORT, A1_Pin, GPIO_PIN_RESET); //A1 reset
            break;
    }
A.2.9 AD data convert to resistant data

```c
float convert_to_res(uint32_t AD_data)
{
    uint32_t full_range = 0x00ffffff;
    float zero_data = 0x00800000;
    float temp, temp1, res;
    temp1 = (float)AD_data;
    temp = (float)((zero_data-temp1)/zero_data);

    res = 360*(1/((temp/32)+0.5)-1);
    res -= 7;
    return res;
}
```

A.2.10 SX1272 Initialization
uint32_t SX1272Init( RadioEvents_t *events )
{
    uint8_t i;
    RadioEvents = events;
    // Initialize driver timeout timers
    TimerInit( &TxTimeoutTimer, SX1272OnTimeoutIrq );
    TimerInit( &RxTimeoutTimer, SX1272OnTimeoutIrq );
    TimerInit( &RxTimeoutSyncWord, SX1272OnTimeoutIrq );

    LoRaBoardCallbacks->SX1272BoardSetXO( SET );

    SX1272Reset();
    SX1272SetOpMode( RF_OPMODE_SLEEP );
    LoRaBoardCallbacks->SX1272BoardIoIrqInit( DioIrq );

    for( i = 0; i < sizeof( RadioRegsInit ) / sizeof( RadioRegisters_t ); i++ )
    {
        SX1272SetModem( RadioRegsInit[i].Modem );
        SX1272Write( RadioRegsInit[i].Addr, RadioRegsInit[i].Value );
    }

    SX1272SetModem( MODEM_FSK );
    SX1272.Settings.State = RF_IDLE;
    return ( uint32_t )LoRaBoardCallbacks->SX1272BoardGetWakeTime( ) +
            RADIO_WAKEUP_TIME; // BOARD_WAKEUP_TIME;
}

A.2.11 SX1272 Set Channel

void SX1272SetChannel( uint32_t freq )
{
    uint32_t channel;
    SX1272.Settings.Channel = freq;
    SX_FREQ_TO_CHANNEL( channel, freq );
    SX1272Write( REG_FRFMSB, ( uint8_t )( ( channel >> 16 ) & 0xFF ) );
    SX1272Write( REG_FRFMID, ( uint8_t )( ( channel >> 8 ) & 0xFF ) );
    SX1272Write( REG_FRFLSB, ( uint8_t )( channel & 0xFF ) );
}

A.2.12 SX1272 Sending data

void SX1272Send( uint8_t *buffer, uint8_t size )
{
    uint32_t txTimeout = 0;
    switch( SX1272.Settings.Modem )
    {

case MODEM_FSK:
{
    SX1272.Settings.FskPacketHandler.NbBytes = 0;
    SX1272.Settings.FskPacketHandler.Size = size;

    if( SX1272.Settings.Fsk.FixLen == false )
    {
        SX1272WriteFifo( ( uint8_t* )&size, 1 );
    } else
    {
        SX1272Write( REG_PAYLOADLENGTH, size );
    }

    if( ( size > 0 ) && ( size <= 64 ) )
    {
        SX1272.Settings.FskPacketHandler.ChunkSize = size;
    } else
    {
        memcpy1( RxTxBuffer, buffer, size );
        SX1272.Settings.FskPacketHandler.ChunkSize = 32;
    }

    // Write payload buffer
    SX1272WriteFifo( buffer, SX1272.Settings.FskPacketHandler.ChunkSize );
    SX1272.Settings.FskPacketHandler.NbBytes +=
    SX1272.Settings.FskPacketHandler.ChunkSize;
    txTimeout = SX1272.Settings.Fsk.TxTimeout;
}
break;

case MODEM_LORA:
{
    if( SX1272.Settings.LoRa.IqInverted == true )
    {
        SX1272Write( REG_LR_INVERTIQ, ( ( SX1272Read( REG_LR_INVERTIQ ) &
        RFLR_INVERTIQ_TX_MASK & RFLR_INVERTIQ_RX_MASK ) | RFLR_INVERTIQ_RX_OFF |
        RFLR_INVERTIQ_TX_ON ) );
        SX1272Write( REG_LR_INVERTIQ2, RFLR_INVERTIQ2_ON );
    } else
    {
        SX1272Write( REG_LR_INVERTIQ, ( ( SX1272Read( REG_LR_INVERTIQ ) &
        RFLR_INVERTIQ_TX_MASK & RFLR_INVERTIQ_RX_MASK ) | RFLR_INVERTIQ_RX_OFF |
        RFLR_INVERTIQ_TX_OFF ) );
        SX1272Write( REG_LR_INVERTIQ2, RFLR_INVERTIQ2_OFF );
    }
    SX1272.Settings.LoRaPacketHandler.Size = size;
// Initializes the payload size
SX1272Write( REG_LR_PAYLOADLENGTH, size );

// Full buffer used for Tx
SX1272Write( REG_LR_FIFOTXBASEADDR, 0 );
SX1272Write( REG_LR_FIFOADDRPTR, 0 );

// FIFO operations can not take place in Sleep mode
if( ( SX1272Read( REG_OPMODE ) & ~RF_OPMODE_MASK ) == RF_OPMODE_SLEEP )
{
    SX1272SetStby( );
    DelayMs( 1 );
}
// Write payload buffer
SX1272WriteFifo( buffer, size );
txTimeout = SX1272.Settings.LoRa.TxTimeout;
}
break;
}
SX1272SetTx( txTimeout );

A.2.13 SX1272 Set receive configuration

void SX1272SetRx( uint32_t timeout )
{
    bool rxContinuous = false;
    TimerStop( &TxTimeoutTimer );
    switch( SX1272.Settings.Modem )
    {
    case MODEM_FSK:
    {
        rxContinuous = SX1272.Settings.Fsk.RxContinuous;

        // DIO0=PayloadReady
        // DIO1=FifoLevel
        // DIO2=SyncAddr
        // DIO3=FifoEmpty
    }
// DIO4=Preamble
// DIO5=ModeReady
SX1272Write( REG_DIOMAPPING1, ( SX1272Read( REG_DIOMAPPING1 ) &
RF_DIOMAPPING1_DIO0_MASK &
RF_DIOMAPPING1_DIO1_MASK &
RF_DIOMAPPING1_DIO2_MASK ) |
RF_DIOMAPPING1_DIO0_00 |
RF_DIOMAPPING1_DIO1_00 |
RF_DIOMAPPING1_DIO2_11 );

SX1272Write( REG_DIOMAPPING2, ( SX1272Read( REG_DIOMAPPING2 ) &
RF_DIOMAPPING2_DIO4_MASK &
RF_DIOMAPPING2_MAP_MASK ) |
RF_DIOMAPPING2_DIO4_11 |
RF_DIOMAPPING2_MAP_PREAMBLEDETECT );

SX1272.Settings.FskPacketHandler.FifoThresh = SX1272Read( REG_FIFOThRESH ) & 0x3F;

SX1272Write( REG_RXCONFIG, RF_RXCONFIG_AFCAUTO_ON |
RF_RXCONFIG_AGCAUTO_ON | RF_RXCONFIG_RXTRIGER_PREAMBLEDETECT );

SX1272.Settings.FskPacketHandler.PreambleDetected = false;
SX1272.Settings.FskPacketHandler.SyncWordDetected = false;
SX1272.Settings.FskPacketHandler.NbBytes = 0;
SX1272.Settings.FskPacketHandler.Size = 0;
}
break;
case MODEM_LORA:
{
if( SX1272.Settings.LoRa.IqInverted == true )

96
{ 
    SX1272Write( REG_LR_INVERTIQ, ( ( SX1272Read( REG_LR_INVERTIQ ) & RFLR_INVERTIQ_TX_MASK & RFLR_INVERTIQ_RX_MASK ) | RFLR_INVERTIQ_RX_ON | RFLR_INVERTIQ_TX_OFF ) );
    SX1272Write( REG_LR_INVERTIQ2, RFLR_INVERTIQ2_ON );
}
else
{
    SX1272Write( REG_LR_INVERTIQ, ( ( SX1272Read( REG_LR_INVERTIQ ) & RFLR_INVERTIQ_TX_MASK & RFLR_INVERTIQ_RX_MASK ) | RFLR_INVERTIQ_RX_OFF | RFLR_INVERTIQ_TX_OFF ) );
    SX1272Write( REG_LR_INVERTIQ2, RFLR_INVERTIQ2_OFF );
}

rxContinuous = SX1272.Settings.LoRa.RxContinuous;
if( SX1272.Settings.LoRa.FreqHopOn == true )
{
    SX1272Write( REG_LR_IRQFLAGSMASK, //RFLR_IRQFLAGS_RXTIMEOUT |
                  //RFLR_IRQFLAGS_RXDONE |
                  //RFLR_IRQFLAGS_PAYLOADCRCERROR |
                  RFLR_IRQFLAGS_VALIDHEADER |
                  RFLR_IRQFLAGS_TXDONE |
                  RFLR_IRQFLAGS_CADDONE |
                  //RFLR_IRQFLAGS_FHSSCHANGEDCHANNEL |
                  RFLR_IRQFLAGS_CADDETECTED );

    // DIO0=RxDone, DIO2=FhssChangeChannel
    SX1272Write( REG_DIAGMAPPING1, ( SX1272Read( REG_DIAGMAPPING1 ) & RFLR_DIAGMAPPING1_DIO0_MASK & RFLR_DIAGMAPPING1_DIO2_MASK ) | RFLR_DIAGMAPPING1_DIO0_00 | RFLR_DIAGMAPPING1_DIO2_00 );
}
else
{
SX1272Write( REG_LR_IRQFLAGSMASK, //RFLR_IRQFLAGS_RXTIMEOUT |
   //RFLR_IRQFLAGS_RXDONE |
   //RFLR_IRQFLAGS_PAYLOADCRCERROR |
   RFLR_IRQFLAGS_VALIDHEADER |
   RFLR_IRQFLAGS_TXDONE |
   RFLR_IRQFLAGS_CADDONE |
   RFLR_IRQFLAGS_FHSSCHANGEDCHANNEL |
   RFLR_IRQFLAGS_CADDETECTED );

   // DIO0=RxDone
   SX1272Write( REG_DIOMAPPING1, ( SX1272Read( REG_DIOMAPPING1 ) &
   RFLR_DIOMAPPING1_DIO0_MASK ) | RFLR_DIOMAPPING1_DIO0_00 );
   }
   SX1272Write( REG_LR_FIFORXBASEADDR, 0 );
   SX1272Write( REG_LR_FIFOADDRPTR, 0 );
   }
   break;
}
memset( RxTxBuffer, 0, ( size_t )RX_BUFFER_SIZE );
SX1272.Settings.State = RF_RX_RUNNING;
if( timeout != 0 )
{
   TimerSetValue( &RxTimeoutTimer, timeout );
   TimerStart( &RxTimeoutTimer );
}
if( SX1272.Settings.Modem == MODEM_FSK )
{
   SX1272SetOpMode( RF_OPMODE_RECEIVER );
   TimerSetValue( &RxTimeoutSyncWord, SX1272.Settings.Fsk.RxSingleTimeout );
   TimerStart( &RxTimeoutSyncWord );
}
else
{
    if( rxContinuous == true )
    {
        SX1272SetOpMode( RFLR_OPMODE_RECEIVER );
    }
}
else
{
    SX1272SetOpMode( RFLR_OPMODE_RECEIVER_SINGLE );
}
}

A.2.14 SX1272 Set transmit configuration
void SX1272SetTx( uint32_t timeout )
{
    TimerStop( &RxTimeoutTimer );

    TimerSetValue( &TxTimeoutTimer, timeout );

    switch( SX1272.Settings.Modem )
    {
    case MODEM_FSK:
    {
        // DIO0=PacketSent
        // DIO1=FifoEmpty
        // DIO2=FifoFull
        // DIO3=FifoEmpty
        // DIO4=LowBat
// DIO5=ModeReady
SX1272Write( REG_DIOMAPPING1, (SX1272Read( REG_DIOMAPPING1 ) &
RF_DIOMAPPING1_DIO0_MASK &
RF_DIOMAPPING1_DIO1_MASK &
RF_DIOMAPPING1_DIO2_MASK ) |
RF_DIOMAPPING1_DIO1_01 );

SX1272Write( REG_DIOMAPPING2, (SX1272Read( REG_DIOMAPPING2 ) &
RF_DIOMAPPING2_DIO4_MASK &
RF_DIOMAPPING2_MAP_MASK ) );
SX1272.Settings.FskPacketHandler.FifoThresh = SX1272Read( REG_FIFOTHRESH ) &
0x3F;
}
break;
case MODEM_LORA:
{
if( SX1272.Settings.LoRa.FreqHopOn == true )
{
SX1272Write( REG_LR_IRQFLAGSMASK, RFLR_IRQFLAGS_RXTIMEOUT |
RFLR_IRQFLAGS_RXDONE |
RFLR_IRQFLAGS_PAYLOADCRCERROR |
RFLR_IRQFLAGS_VALIDHEADER |
//RFLR_IRQFLAGS_TXDONE |
RFLR_IRQFLAGS_CADDONE |
//RFLR_IRQFLAGS_FHSSCHANGEDCHANNEL |
RFLR_IRQFLAGS_CADDETECTED );

// DIO0=TxDone, DIO2=FhssChangeChannel
SX1272Write( REG_DIOMAPPING1, (SX1272Read( REG_DIOMAPPING1 ) &
RFLR_DIOMAPPING1_DIO0_MASK & RFLR_DIOMAPPING1_DIO1_MASK & RFLR_DIOMAPPING1_DIO2_MASK ) |
RFLR_DIOMAPPING1_DIO0_01 | RFLR_DIOMAPPING1_DIO2_00 );
else
{
    SX1272Write( REG_LR_IRQFLAGSMASK, RFLR_IRQFLAGS_RXTIMEOUT | RFLR_IRQFLAGS_RXDONE | RFLR_IRQFLAGS_PAYLOADCRCERROR | RFLR_IRQFLAGS_VALIDHEADER | //RFLR_IRQFLAGS_TXDONE | RFLR_IRQFLAGS_CADDONE | RFLR_IRQFLAGS_FHSSCHANGEDCHANNEL | RFLR_IRQFLAGS_CADDETECTED );

    // DIO0=TxDone
    SX1272Write( REG_DIOMAPPING1, ( SX1272Read( REG_DIOMAPPING1 ) & RFLR_DIOMAPPING1_DIO0_MASK ) | RFLR_DIOMAPPING1_DIO0_01 );
}

break;
}

SX1272.WriteBuffer( uint16_t addr, uint8_t *buffer, uint8_t size )
{
    uint8_t i;

    //NSS = 0;
HW_GPIO_Write( RADIO_NSS_PORT, RADIO_NSS_PIN, 0 );

HW_SPI_InOut( addr | 0x80 );

for( i = 0; i < size; i++ )
{
    HW_SPI_InOut( buffer[i] );
}

//NSS = 1;
HW_GPIO_Write( RADIO_NSS_PORT, RADIO_NSS_PIN, 1 );
}

A2.16 SX1272 read buffer

void SX1272ReadBuffer( uint16_t addr, uint8_t *buffer, uint8_t size )
{
    uint8_t i;

    //NSS = 0;
    HW_GPIO_Write( RADIO_NSS_PORT, RADIO_NSS_PIN, 0 );
    HW_SPI_InOut( addr & 0x7F );
    for( i = 0; i < size; i++ )
    {
        buffer[i] = HW_SPI_InOut( 0 );
    }

    //NSS = 1;
    HW_GPIO_Write( RADIO_NSS_PORT, RADIO_NSS_PIN, 1 );
}
A.3 Software code (in Python)

A.3.1 Serial port detection

```python
def serial_detection():
    port_list = list(serial.tools.list_ports.comports())
    return port_list

print("Detect Com")
self.PortSelectlabel.setText("Serial Port showed here")
self.SerialPortcomboBox.clear()
port_list = list(serial.tools.list_ports.comports())
if len(port_list) <= 0:
    print("The serial port can't find!")
    self.PortSelectlabel.setText("The serial port can't find!"))
else:
    for i in range(0, len(port_list)):
        print("port", i, port_list[i])
        port_name0 = port_list[i].description
        self.SerialPortcomboBox.addItem(port_name0)
```

A.3.2 Get serial port data

```python
def getSerialData(self, frame, lines, lineValueText, lineLabel, filewriter):
    # currentTimer = time.clock()
    currentTimer = time.process_time()
    self.plotTimer = int((currentTimer -
    self.previousTimer) * 1000) # the first reading will be erroneous
    self.previousTimer = currentTimer
    # timeText.set_text('Plot Interval = ' + str(self.plotTimer) + ' ms')
    privateData = copy.deepcopy(
        self.rawData[:]) # so that the 8 values in our plots will be synchronized to the same sample time
    value1 = []
    # filewriter.writerow([privateData])
    for i in range(self.numPlots):
```
data = privateData[(i * self.dataNumBytes):(self.dataNumBytes + i * self.dataNumBytes)]
value, = struct.unpack(self.dataType, data)
self.data[i].append(value)  # we get the latest data point and append it to our array
# self.savedata[i].append(value)
lines[i].set_data(range(self.plotMaxLength), self.data[i])
lineValueText[i].set_text('[' + lineLabel[i] + ']' = ' + str(value))
value1.append(value)
filewriter.writerow(value1)

A.3.3 Background threads

def backgroundThread(self):  # retrieve data
time.sleep(1.0)  # give some buffer time for retrieving data
self.serialConnection.reset_input_buffer()
while (self.isRun):
    self.serialConnection.readinto(self.rawData)
self.isReceiving = True
print(self.rawData)
print(self.port)

A.3.4 Plot and save data

# TK used to be here
lineLabel = ['ch1', 'ch2', 'ch3', 'ch4', 'ch5', 'ch6', 'ch7', 'ch8']
style = ['r', 'tab:cyana', 'tab:blue', 'tab:gray', 'tab:brown', 'tab:orange', 'tab:green', 'tab:purple']  # linestyles for the different plots
# timeText = ax.text(0.70, 0.95, '', transform=ax.transAxes)
lines = []
lineValueText = []

# starts background thread
# put our plot onto Tkinter's GUI
root = Tk.Tk()
app = Window(fig, root, s)
if len(port_list) <= 0:
    print("The serial port can't find!")
self.PortSelectlabel.setText("The serial port can't find!")
else:
    for i in range(0, len(port_list)):
        print("port", i, port_list[i])
        portName = port_list[i].device
    app.SerialPortcomboBox['values'] = port_list
    app.SerialPortcomboBox.bind("<<ComboboxSelected>>", app.showCurrentPort)
    dt = datetime.now()
    nowtime_str = dt.strftime('%y_%m_%d_%H_%M_%S_')
    filename = nowtime_str + 'data' + '.csv'
    with open(filename, 'w', newline='') as csvfile:
        filewriter = csv.writer(csvfile)  # delimiter=',', quotechar='\', quoting=csv.QUOTE_MINIMAL
        filewriter.writerow(['ch1', 'ch2', 'ch3', 'ch4', 'ch5', 'ch6', 'ch7', 'ch8'])
        for i in range(numPlots):
            lines.append(ax.plot([], [], style[i], label=lineLabel[i])[0])
            lineValueText.append(ax.text(0.70, 0.90 - i * 0.05, '', transform=ax.transAxes))
            anim = animation.FuncAnimation(fig, s.getSerialData, fargs=(lines, lineValueText, lineLabel, filewriter), interval=pltInterval)  # fargs has to be a tuple
            plt.legend(loc="upper left")
    root.mainloop()  # use this instead of plt.show() since we are encapsulating everything in Tkinter
BIBLIOGRAPHY

P.R.Couchman, W.A.Jesser, D.Kuhlmann-Wilsdorf, J.P.Hirth, “On the concepts of surface stress and surface strain”, Volume 33, Issue 3, December 1972, Pages 429-436.

A. D. Kersey, T. A. Berkoff, and W. W. Morey, "Multiplexed fiber Bragg grating strain-sensor system with a fiber Fabry–Perot wavelength filter," Opt. Lett. 18, 1370-1372 (1993).

Serge M. Melle, Kexing Liu, and Raymond M. Measures, "Practical fiber-optic Bragg grating strain gauge system," Appl. Opt. 32, 3601-3609 (1993).

Komurlu, E., Cihangir, F., Kesimal, A. et al. Effect of Adhesive Type on the Measurement of Modulus of Elasticity Using Electrical Resistance Strain Gauges. Arab J Sci Eng 41, 433–441 (2016).

J. Kosmann, O. Völkerink, M.J. Schollerer, D. Holzhüter, C. Hühne, “Digital image correlation strain measurement of thick adherend shear test specimen joined with an epoxy film adhesive,” International Journal of Adhesion and Adhesives, Volume 90, 2019, Pages 32-37, ISSN 0143-7496.

Stefan Harenberg  Matthias Pahn  Viktória Malárics-Pfaff  Frank Dehn  Antonio Caggiano  Diego S. Schicchi  Sha Yang  Eddie Koenders. “Digital image correlation strain measurement of ultra-high-performance concrete-prisms under static and cyclic bending-tensile stress,” Volume20, Issue4, Special Issue: Special theme “Fatigue”, August 2019, Pages 1220-1230.
In Lee, Kyoochun Lee, “The Internet of Things (IoT): Applications, investments, and challenges for enterprises,” Business Horizons, Volume 58, Issue 4, 2015, Pages 431-440, ISSN 0007-6813.

M Hung, “Leading the IoT, Gartner Insights on how to lead in a connected world,” Gartner Research, 2017.

A. Zhamanov, Z. Sakhiyeva, R. Suliyev, and Z. Kaldykulova, "IoT smart campus review and implementation of IoT applications into education process of the university," 2017 13th International Conference on Electronics, Computer and Computation (ICECCO), Abuja, 2017, pp. 1-4, doi: 10.1109/ICECCO.2017.8333334.

Rashmi Sharan Sinha, Yiqiao Wei, Seung-Hoon Hwang, “A survey on LPWA technology: LoRa and NB-IoT,” ICT Express, Volume 3, Issue 1, 2017, Pages 14-21.

Vangelista L., Zanella A., Zorzi M. (2015) Long-Range IoT Technologies: The Dawn of LoRa™. In: Atanasovski V., Leon-Garcia A. (eds) Future Access Enablers for Ubiquitous and Intelligent Infrastructures. FABULOUS 2015. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 159. Springer, Cham

R.L. Sakaguchi, C.T. Sasik, M.A. Bunczak, W.H. Douglas, Strain gauge method for measuring polymerization contraction of composite restoratives, Journal of Dentistry, Volume 19, Issue 5, 1991, Pages 312-316, ISSN 0300-5712.
Analog Devices, “LT3042: 20V, 200mA, Ultralow Noise, Ultrahigh PSRR RF Linear Regulator Data Sheet,” [online] Analog Devices. Available at: <https://www.analog.com/media/en/technical-documentation/data-sheets/3042fb.pdf>.

Analog Devices, “AD7798/AD7799: 3-Channels, Low Noise, Low Power, 16-/24-Bits, Sigma-Delta ADC with On-Chip In-Amp Data Sheet,” [online] Analog Devices. Available at: <https://www.analog.com/media/en/technical-documentation/data-sheets/AD7798_7799.pdf>.

Analog Devices, “ADG706/ADG707: +1.8V to +5.5V/±2.5V, 2.5Ω Low-Voltage, 8-/16-Channel Multiplexers Data Sheet,” [online] Analog Devices. Available at: <https://www.analog.com/media/en/technical-documentation/data-sheets/ADG706_707.pdf>.

OMEGA, “Precision Strain Gage, Precision Linear Pattern For Static And Dynamic Applications,” [online] OMEGA Inc. Available at:<https://www.omega.co.uk/techref/pdf/StrainGage_Measurement.pdf>.

K. Hoffmann, “Applying the Wheatstone Bridge Circuit,” 1974.

STMicroelectronics, “STM32L073RZ: Ultra-low-power Arm Cortex-M0+ MCU with 192 Kbytes Flash, 32 MHz CPU, USB, LCD Data Sheet,” [online] STMicroelectronics. Available at: <https://www.st.com/resource/en/datasheet/stm32l073rz.pdf>.
SEMTECH, “SX1272 Development Kit User Guide,” [online] SEMTECH. Available at:

<https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/2R000000HURU/c0BOzOaZp53hID5pzpu_scWIXl2.X5Oo2iYzCSmYWS8>

INSTRON, “5960 Series Dual Column Table Frames,” [online] INSTRON, Available at <https://www.instron.us/-/media/literature-library/manuals/5960-dual-column-table-frames-system-support.pdf?la=en-US>.

Micro-Measurements, “P3 Strain Indicator and Recorder,” [online] Micro-Measurements, Available at <http://www.vishaypg.com/docs/11102/p3.pdf>.