Steam Sterilizer for Healthcare and Industry Applications

Submitted by

MD. JAMSHEDUL ALAM	EEE2201025005
MOHAMMAD SHAKIL AHMED	EEE2201025010
SHIPRA SARKAR	EEE2201025023
ABDULLAH AL MAMUN	EEE2201025063
SAIFUL ISLAM	EEE2202026016

.

Supervisor

Md. Ashikur Rahman
Lecturer and Coordinator, Department of EEE
Sonargaon University

Department of Electrical & Electronic Engineering
Sonargaon University (SU)
147/I, Panthapath,
Dhaka-1215,

Date of Submission: September, 2025

DECLARATION

We, the undersigned students of the Department of Electrical and Electronic Engineering, Sonargaon University, hereby declare that the project report entitled **"Steam Sterilizer for Healthcare and Industry Applications"** is the outcome of our own work, carried out under the supervision of Md. Ashikur Rahman, Lecturer, Department of EEE, Sonargaon University.

We affirm that the work presented in this report is based on our original effort, investigations, and practical implementation. It has not been submitted, either in whole or in part, to any other institution or authority for the award of any academic degree or professional qualification.

Through this project, we aimed to design and develop a reliable, low-cost steam sterilizer by integrating mechanical, electrical, and software systems. The knowledge, analysis, and results presented here reflect our dedication to applying engineering skills in solving real-world problems.

We take full responsibility for the accuracy and authenticity of the content of this report.

Md. Jamshedul Alam EEE2201025005	Mohammad Shakil Ahmed EEE2201025010	Shipra Sarkar EEE2201025023

Abdullah Al Mamun EEE2201025063	Saiful Islam EEE2202026016

CERTIFICATEION

This is to certify that the project report entitled **"Steam Sterilizer Steam Sterilizer for Healthcare and Industry Applications"** is a bonafide record of the work carried out by –

Md. Jamshedul Alam	EEE2201025005
Mohammad Shakil Ahmed	EEE2201025010
Shipra Sarkar	EEE2201025023
Abdullah Al Mamun	EEE2201025063
Saiful Islam	EEE2202026016

in partial fulfilment of the requirements for the degree of Bachelor of Science in Electrical and Electronic Engineering (EEE) at Sonargaon University.

The project has been completed under my supervision and guidance, and it is a genuine record of the successful work done by them. I wish the students every success in their future activities.

Supervisor

……

Md. Ashikur Rahman
Lecturer and Coordinator, Department of EEE
Sonargaon University

ACKNOWLEDGEMENT

We sincerely express our gratitude to the Almighty for His blessings, which enabled us to complete this project successfully.

We are profoundly grateful to Md. Ashikur Rahman, Lecturer, Department of Electrical and Electronic Engineering, Sonargaon University, for his invaluable guidance, scholarly advice, and constant encouragement throughout the course of this project. His constructive feedback, patience, and motivation played a pivotal role in the successful completion of this work.

We also extend our sincere thanks to the Head of the Department of Electrical and Electronic Engineering, Sonargaon University, for their support and facilitation during this project.

Our heartfelt appreciation goes to our course mates for their cooperation and assistance in discussions and practical work related to this project.

Finally, we would like to acknowledge all the respected faculty members of the Department of Electrical and Electronic Engineering for their guidance and support, which significantly contributed to the successful completion of this project.

ABSTRACT

This project presents the design and development of a Steam Sterilizer for Healthcare and Industry Applications that integrates mechanical fabrication, electronic control, and software automation into a unified system. The sterilizer was constructed with a stainless-steel pressure chamber, heating elements, and safety valves, combined with sensors for real-time monitoring of temperature and pressure. A microcontroller-based control board with PID algorithms was implemented to maintain stable heating at the required set-point, ensuring accurate sterilization conditions. Additionally, a SCADA system was developed to provide a user-friendly interface for monitoring, control, database logging, and batch-wise report generation.

During testing, the system achieved a stable sterilization temperature of 121°C within 30 minutes, with pressure levels accurately tracked by both sensors and SCADA. The PID controller effectively minimized overshoot and maintained consistent thermal regulation. Validation with a Bowie-Dick test pack confirmed successful steam penetration and contamination-free results, ensuring process reliability.

The project demonstrates a low-cost, reliable, and efficient sterilization solution suitable for hospitals, laboratories, and small industries, offering an affordable alternative to costly imported systems.

CONTENTS

DECLARATION 2
CERTIFICATEION 3
ACKNOWLEDGEMENT 4
ABSTRACT 5
Chapter 1: Introduction 12
1.1 Project Goal 12
1.2 Motivation 12
1.3 Objectives 13
1.4 Features of Steam Sterilizer 13
1.5 Social Impact 14
1.6 Problem Statement 15
Chapter 2: LITERATURE REVIEW 16
2.1 Existing System 16
2.2 Literature Review 16
Chapter 3: Function of the Steam Sterilizer 17
3.1. Block Diagram of Working Principle of Steam Sterilizer 18
3.2 Working Flow of Steam Sterilizer. 18
Chapter 4: Construction of the Main Body 19
4.1 Introduction 19
4.2 Components Used for Main Body 19
4.3 Fabrication of Main Body 26
4.3.1 Designing 26
4.3.2 Materials Collection 27
4.3.3 Workshop / Fabrication 27
4.3.4 Testing 27
4.3.5 Finishing 28
Chapter 5: Electrical Design 29
5.1 Motherboard Design & Manufacturing 29
5.1.1 Designing the electrical circuit 29
5.1.2 Preparing the schematic diagram 30

5.1.3 Converting the schematic into a PCB layout 30
5.1.4 Sending the PCB design for production. 31
5.1.5 Collecting Components for PCB Assembly 32
5.1.6 Assembling the components on the PCB 37
5.1.7 Testing the PCB 37
5.2 Control Panel 38
5.2.1 Designing the layout 38
5.2.2 Collecting components 38
5.2.3 Building the panel in-house. 41
5.2.4 Testing the control panel. 41
Chapter 6: Software Desing 42
6.1 Microcontroller Programming 42
6.1.1 Controller Code with Explanation 42
6.1.2 Overall Working Principle 46
6.1.3 Purpose in Project 46
6.1.4 Program Flowchart 46
6.2 SCADA Programming and Database Integration 47
6.2.1 Designing SCADA & Database Integration with Explanation 47
6.2.1.1 Page 01 (Loading.cs) 47
6.2.1.2 Page 02 (Login.cs) 48
5.2.1.3 Page 03 (Register.cs) 55
5.2.1.4 Page 04 (ChangePass.cs) 56
6.2.1.5 Page 05 (Home.cs) 58
6.2.1.6 Page 06 (CreateRecipe.cs) 60
6.2.1.7 Page 07 (SelectRecipe.cs) 61
6.2.1.8 Page 08 (Start.cs) 65
6.2.1.9 Page-09 (Crystal Report) 76
Chapter 7: Implementation & Testing 89
7.1 Implementation of the Project: 89
7.2 Calculation 89
7.3 Testing 91
7.4 Result 91

Chapter 8: Discussion & Conclusion 96
8.1 Conclusion 96
8.2 Limitation 96
8.3 Future Scope 97
References 98
Notes: 99

List of Tables

Table 3. 1 Standard Temperature, Pressure, and Time for Steam Sterilization Applications. 17

Table 5. 1 BOM sheet of PCB 31

Table 7. 1 Price of Existing Sterilizer 95

List of Figures

Figure 3. 1 Block Diagram of Working Principle of Steam Sterilizer 18
Figure 3. 2 Working Flow of Steam Sterilizer. 18

Figure 4.1 Stainless Steel Cylinder 19
Figure 4.2 Pressure gauge 20
Figure 4.3 K-Type Thermocouple 20
Figure 4.4 K TYPE TEMPERATURE SENSOR (MAX6675) 21
Figure 4.5 MAX6675 connection Diagram 21
Figure 4.6 Pressure Transducer Sensor 21
Figure 4.7 Water Heater 22
Figure 4.8 Electric Solenoid Valve 22
Figure 4.9 Pressure Relief Safety Valve (PRV) 23
Figure 4.10 Steam Exhaust Valve 23
Figure 4.11 Tri Clamp (TC) 23
Figure 4.12 Water Label Sensor 24
Figure 4.13 Socket 24
Figure 4.14 Syphon pipe 25
Figure 4.15 Temperature Gauge 25
Figure 4.16 3D Drawing of the Machine 26
Figure 4.17 Fabrication of Steam Sterilizer Main Body 27
Figure 4.18 Top view After Finishing 28
Figure 4.19 Front View After Complete Fabrication & Finishing. 28

Figure 5. 1 Electrical Circuit Diagram 29
Figure 5. 2 Schematic circuit diagram 30
Figure 5. 3 PCB Layout (Front & Back) 31
Figure 5. 4 3D view of PCB (Front & Back) 31
Figure 5. 5 PCB after Fabrication (Front & Back) 32
Figure 5. 6 ATmega328P 32
Figure 5. 7 ULN2003AN 33
Figure 5. 8 PS2501-4 33
Figure 5. 9 Relay 34
Figure 5. 10 Crystal Oscillator 34
Figure 5. 11 Resistor 34
Figure 5. 12 Capacitors 35
Figure 5. 13 Push Button 35
Figure 5. 14 LED 35
Figure 5. 15 USB Port 36

Figure 5. 16 screw terminal 36
Figure 5. 17 Assembling the components on the PCB 37
Figure 5. 18 Testing PCB 37
Figure 5. 19 Layout of control Panel 38
Figure 5. 20 SMPS (Switch Mode Power Supply) 38
Figure 5. 21 Motherboard 39
Figure 5. 22 Earth Leakage Circuit Breakers (ELCB) 39
Figure 5. 23 Solid State Relay (SSR) 40
Figure 5. 24 Indicators 40
Figure 5. 25 Connectors 41

Figure 6. 1 Program Flowchart 46
Figure 6. 2 Loading Page 47
Figure 6. 3 Login page 49
Figure 6. 4 Load Username 51
Figure 6. 5 Hide Password 51
Figure 6. 6 Show Password 51
Figure 6. 7 Limit login attempts 52
Figure 6. 8 Disable Login button 53
Figure 6. 9 Register page 55
Figure 6. 10 Change Password page 56
Figure 6. 11 Home Page 58
Figure 6. 12 Gauge 59
Figure 6. 13 Create Recipe 60
Figure 6. 14 Select Recipe 61
Figure 6. 15 Load Recipe 62
Figure 6. 16 Load Recipe Details 63
Figure 6. 17 Start page 65
Figure 6. 18 Recipe Info 66
Figure 6. 19 Tracker 72
Figure 6. 20 Select Batch 76
Figure 6. 21 Download Crystal Report 81

Figure 7. 1 Logic for PID 90
Figure 7. 2 PID controlling Output 90
Figure 7. 3 Batch Report 92
Figure 7. 4 Example of Bowie-Dick Test Pack showing proper and improper colour change results. 93
Figure 7. 5 Actual Bowie-Dick Test Pack result showing complete uniform colour change. 94

Chapter 1: Introduction

1.1 Project Goal

The main goal of this project is to design and develop a low-cost, reliable **steam sterilizer** system that integrates mechanical, electrical, and software components into a single functional unit. Through this project, we aim to demonstrate our technical ability in PCB design, control logic development, electrical diagram preparation, and microcontroller-based programming, while also showcasing our creativity in engineering design and automation.

This project serves as a practical example of how theoretical knowledge can be applied to solve real-world problems. By completing the full process—from concept design, component selection, and fabrication, to programming, testing, and validation—we not only build a functional sterilizer but also prove our competence in handling complete engineering tasks.

Ultimately, the goal is twofold:

1. To provide an affordable and efficient sterilization solution for hospitals, laboratories, and industries.
2. To demonstrate our readiness and capability to design, build, and control complex machines by combining hardware and software in a professional way.

1.2 Motivation

Our motivation for this project arises from a pressing real-world need: sterilization is essential in hospitals, laboratories, pharmaceuticals, and food industries, where contamination-free equipment is critical for safety and quality. In particular, surgical instruments must be sterilized before every operation to protect patients from infection.

However, most existing sterilization systems are imported, expensive, and complex, which makes them inaccessible to many rural clinics, small hospitals, and community laboratories. This gap in affordability and accessibility motivated us to design a low-cost yet effective steam sterilizer that can be manufactured and maintained locally.

At the same time, we were motivated by the opportunity to strengthen our technical expertise. This project challenges us to combine PCB design, electrical circuits, microcontroller programming, and automation into a unified system. By successfully completing this work, we prove that we can apply our engineering knowledge to create a machine that meets both industrial requirements and societal needs.

1.3 Objectives

- Design and Development:
  To design and develop a reliable, low-cost steam sterilizer that fulfils the requirements of healthcare, laboratory, and industrial applications.
- System Integration:
  To integrate mechanical, electrical, and software systems into a cohesive and fully functional unit.
- Practical Application of Knowledge:
  To apply theoretical engineering knowledge in PCB design, circuit development, and programming to real-world implementation.
- Demonstration of Engineering Competency:
  To showcase the team's ability to manage the complete engineering process—from concept design and fabrication to programming, testing, and validation.

1.4 Features of Steam Sterilizer

The steam sterilizer developed in this project incorporates several key features, which can be grouped into the following categories:

A. Performance Features

1. Effective Sterilization (*Destroys bacteria, viruses, fungi, and spores through high-pressure steam).*
2. Reliable Performance (*Provides consistent sterilization results, maintaining high hygiene standards).*
3. Energy Efficient (*Optimized to minimize power consumption while ensuring maximum sterilization output).*

B. Safety and Hygiene Features

4. Safe Reuse of Instruments (*Ensures that medical, laboratory, and industrial tools are safe for repeated use).*
5. Essential for Healthcare and Surgery (*Guarantees that all surgical and medical instruments are sterilized before use).*
6. Eco-Friendly Technology (*Uses steam instead of harmful chemicals, ensuring both user safety and environmental protection).*

C. Usability Features

7. User-Friendly Operation *(Designed with simple and intuitive controls, requiring minimal training).*
8. Compact and Portable (*Easy to install, move, and operate in both large and small organizations).*

D. Economic Features

9. Low-Cost Solution (*An affordable design that makes advanced sterilization technology accessible to small-scale institutions).*
10. Wide Application (*Versatile use across hospitals, laboratories, pharmaceutical industries, and food processing plants).*

1.5 Social Impact

The steam sterilizer project is not only an engineering achievement but also a contribution to society in multiple ways. Its impact can be summarized as follows:

1. Improved Public Health
   - Prevents the spread of infections and diseases by sterilizing surgical and medical instruments.
   - Reduces hospital-acquired infections (HAIs), ensuring safer patient care.

2. Affordable Access for Small Institutions
   - Low-cost design makes it suitable for rural clinics, small hospitals, and local laboratories.
   - Provides underprivileged communities with access to advanced sterilization methods.

3. Enhanced Food Safety
   - Prevents contamination in food processing plants, ensuring safe and hygienic food for consumers.
   - Helps reduce foodborne illnesses and supports public well-being.

4. Support for Research & Pharmaceuticals
   - Ensures accuracy in laboratory experiments and pharmaceutical production through reliable sterilization.
   - Maintains high standards in scientific research and medicine development.

5. Job Creation & Economic Growth
   - Local manufacturing, assembly, and maintenance of sterilizers can create new employment opportunities.
   - Encourages small and medium enterprises (SMEs) to engage in technological innovation.

6. Promotion of Hygiene Awareness
   - Encourages healthcare workers, industrial professionals, and the general public to adopt hygienic practices.
   - Spreads awareness about the critical role of sterilization in building a healthier society.

7. Contribution to Sustainable Development Goals (SDGs)[i]
   - SDG 3 – Good Health & Well-being: Promotes safer medical practices and reduces infection risks.
   - SDG 8 – Decent Work & Economic Growth: Supports job creation and local entrepreneurship.
   - SDG 9 – Industry, Innovation & Infrastructure: Advances affordable technological innovation and industrial development.

1.6 Problem Statement

Sterilization is a critical process in hospitals, laboratories, pharmaceutical industries, and food processing plants to ensure that instruments, equipment, and materials are safe, hygienic, and contamination-free. However, most available sterilization systems are expensive, highly sophisticated, and often imported, making them inaccessible for small-scale institutions such as rural clinics, local hospitals, and community laboratories. This creates a significant gap in ensuring safe medical practices and public health protection.

The lack of affordable and reliable sterilization equipment can lead to unsafe reuse of instruments, increased hospital-acquired infections (HAIs), compromised laboratory results, and contamination in food and pharmaceutical production. These risks directly threaten patient safety, hinder healthcare delivery, and negatively impact community well-being.

Therefore, there is a pressing need to design and develop a low-cost, reliable, and efficient steam sterilizer that can be locally manufactured and maintained. Such a system would not only reduce dependency on costly imports but also make sterilization technology accessible to underprivileged healthcare facilities and small industries. By addressing this gap, the project contributes to improving public health, ensuring hygiene, and supporting sustainable technological development.

Chapter 2: LITERATURE REVIEW

2.1 Existing System

In hospitals and healthcare settings, sterilization is primarily carried out using physical, chemical, or radiation-based methods. Ethylene Oxide (EtO) and Gamma Radiation are commonly used in the medical device manufacturing industry because they can sterilize complex and sensitive equipment. However, in hospital settings, the Steam Autoclave is the most widely used and practical method due to its cost-effectiveness, accessibility, faster sterilization cycles, and reliable temperature control. The autoclave operates with high-pressure saturated steam at temperatures between 121–134°C, ensuring complete destruction of all microorganisms, including spores [1].

Surgical instruments are routinely processed in washer–disinfector machines, which include pre-wash, detergent wash, thermal rinse with lubricant, and drying steps. Instruments then undergo pre-vacuum steam sterilization, considered the gold standard for ensuring sterility. Studies have shown that MRSA and VRE are completely eliminated during this process, but resilient spores such as *Clostridium difficile* may survive about 33% of the washer–disinfector cycle, emphasizing the critical importance of subsequent steam sterilization [2]. According to current Chinese standards, surgical instruments must follow strict protocols in pressure steam sterilizers. Risk points are identified at every stage, and seven critical workflows have been mapped and standardized. Data collection, potential failure analysis, and preventive measures have significantly improved sterilization quality and reduced high-risk situations [3].

2.2 Literature Review

Literature emphasizes that relying solely on pressure gauge readings in steam sterilizers is insufficient. A 1937 study showed that air-steam mixtures in the chamber often result in lower actual temperatures than indicated, and increasing pressure requires more time for the entire load to reach the target temperature. Solutions proposed include the gravity discharge system to remove air and direct internal thermometer control to monitor temperature accurately [7].

Recent studies have focused on improving sterilization reliability through early warning systems, risk checklists, and computational models. Comparing conventional and validation groups, the use of these tools enhanced sterilization quality, reduced equipment downtime, and increased process safety [6]. In practice, surgical instruments sometimes fall on the floor or additional instruments may be unexpectedly needed during procedures. In such cases, Immediate-Use Steam Sterilization (IUSS) allows rapid sterilization and safe reuse [4][5].

In resource-limited or off-grid settings, conventional autoclaves may not be feasible due to electricity or fuel constraints. Emerging passive solar steam generation technologies offer potential solutions for generating high-temperature steam in remote areas, though efficiency at >100°C remains a challenge [8]. Overall, the literature demonstrates that steam sterilization, workflow optimization, and advanced technologies are critical for reducing healthcare-associated infections and ensuring patient safety [1][2][3][4][5][6][7][8].

Chapter 3: Function of the Steam Sterilizer

A steam sterilizer, commonly known as an autoclave, is a device that uses pressurized steam to sterilize equipment, tools, and materials. The main purpose of the steam sterilizer is to destroy microorganisms such as bacteria, viruses, fungi, and spores, ensuring that medical, laboratory, or industrial instruments are safe for reuse.

The working principle of a steam sterilizer is based on the fact that moist heat under pressure is highly effective in eliminating harmful microorganisms. In this process, water is heated to generate steam, which is then pressurized inside a closed chamber. The high-temperature steam penetrates [ii] the objects and kills germs by denaturing [iii] their proteins.

Steam sterilizers are widely used in hospitals, laboratories, pharmaceutical industries, and food processing plants. In industrial applications, they are also used to sterilize machine parts and materials that require contamination-free operation.

This process will be completed if an object is kept at a certain temperature for a certain period of time.

Table 3. 1 Standard Temperature, Pressure, and Time for Steam Sterilization Applications.

Temperature (°C)	Pressure (psi)	Time (minutes)	Application
121°C	15 psi	15–30 mins	Standard load (surgical tools, glassware)
126°C	20 psi	10–15 mins	Faster sterilization of similar loads.
134°C	30 psi	3–5 mins	Flash sterilization (emergency use).
115°C	10 psi	30–40 mins	Heat-sensitive materials.
132°C	27 psi	4 mins	Prion sterilization (e.g., for brain surgery tools).

Gay-Lussac's Law Statement:

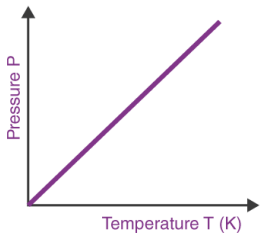
At constant volume, the pressure of a gas is directly proportional to its temperature.

Formula:

$$\frac{p}{T} = Constant$$

At atmospheric pressure, steam temperature is 100°C.

This law explains why increasing the steam temperature inside a fixed chamber directly raises the pressure, which is essential for sterilization.



3.1. Block Diagram of Working Principle of Steam Sterilizer

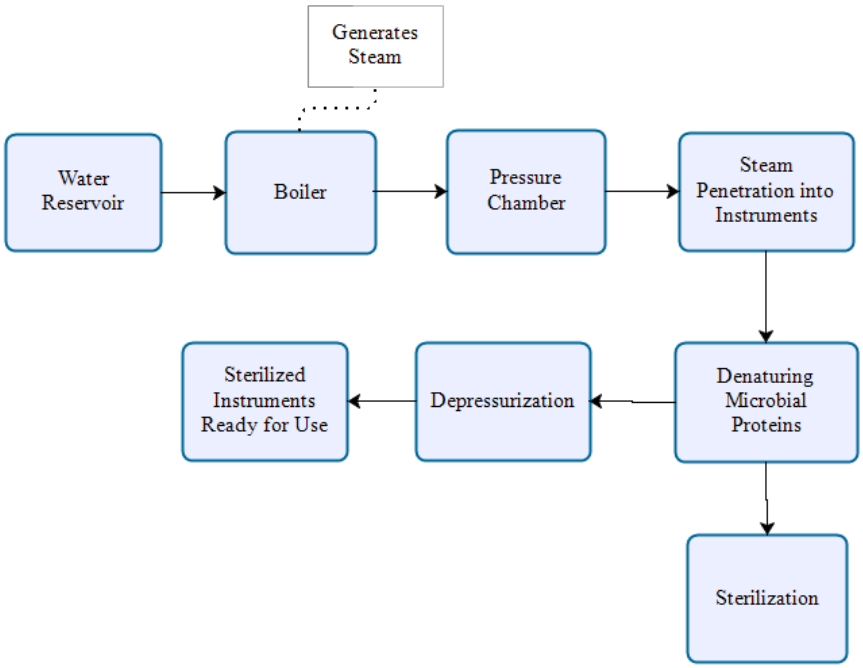


Figure 3. 1 Block Diagram of Working Principle of Steam Sterilizer

3.2 Working Flow of Steam Sterilizer.

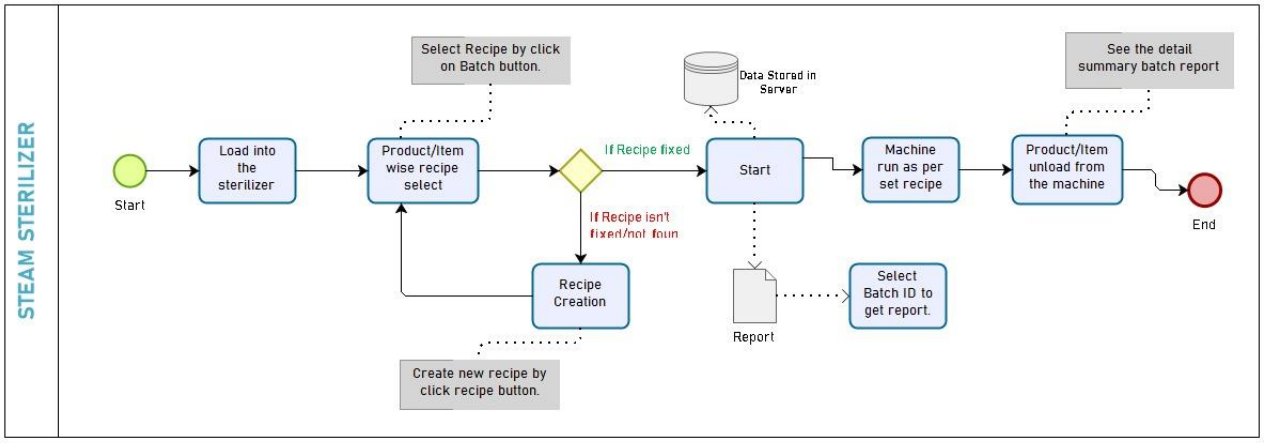


Figure 3. 2 Working Flow of Steam Sterilizer.

For this project, the steam sterilizer serves as a practical example of applying engineering design and automation skills. It demonstrates how mechanical design, electrical circuits, and programming can work together to build a system that controls temperature, pressure, and timing automatically. The device highlights the integration of PCB design, control panel design, microcontroller programming, and SCADA interface for both operation and monitoring.

Chapter 4: Construction of the Main Body

4.1 Introduction

The main body of a steam sterilizer is the most critical part of the system, as it functions as the pressure chamber where sterilization actually takes place. It must be designed to withstand high temperature and pressure without leakage or deformation. A strong and well-constructed main body ensures durability, provides safety during operation, and maintains the required conditions for effective sterilization performance.

4.2 Components Used for Main Body

The following main components were used in the construction of the steam sterilizer body:

- Stainless Steel Cylinder
- Pressure Gauge
- K-Type Thermocouple
- K Type Temperature Sensor (Max6675)
- Pressure Transducer Sensor
- Water Heater 1000 Watt
- Electric Solenoid Valve (2w-200-20)
- Pressure Relief Safety Valve (Prv)
- Steam Exhaust Valve
- Tri-Clamp (Tc)
- Water Level Sensor
- Socket Fitting
- Syphon Pipe

4.2.1 Stainless Steel Cylinder



Figure 4.1 Stainless Steel Cylinder

The pressure chamber is the main body of the steam sterilizer, constructed from high-quality stainless steel. It forms the cylindrical container where the sterilization process takes place, capable of withstanding high temperature and pressure without deformation or corrosion. Its smooth surface ensures hygiene and makes cleaning and maintenance easy.

This component is used in our project because it provides durability, safety, and reliable performance. By using stainless steel, the chamber can handle repeated sterilization cycles and

maintain consistent pressure, which is essential for effective sterilization of instruments and materials.

4.2.2 Pressure Gauge



Figure 4.2 Pressure gauge

The pressure gauge is a device used to measure the internal pressure of the steam sterilizer, ensuring the system operates within safe limits. In our project, we use a gauge with psi/Bar scaling to provide accurate and easy-to-read pressure readings. It helps monitor the sterilization process and prevents overpressure conditions. The gauge is essential for safety and maintaining consistent sterilization performance. Its simple design allows quick verification without affecting the operation of the chamber.

4.2.3 K-Type Thermocouple



Figure 4.3 K-Type Thermocouple

A K-type thermocouple is a widely used temperature sensor made from two dissimilar metal wires, Chromel (nickel-chromium) and Alumel (nickel-aluminium). It is valued for its wide temperature range (typically –200°C to 1260°C), good stability, and strong resistance to oxidation, making it reliable in harsh environments. With a sensitivity of approximately 41 µV/°C, it produces a measurable voltage change corresponding to temperature variations. Due to its accuracy, durability, and cost-effectiveness, the K-type thermocouple is commonly used in industrial processes, food processing, semiconductor manufacturing, laboratory testing, and other applications requiring precise temperature monitoring.

4.2.4 K Type Temperature Sensor (Max6675)

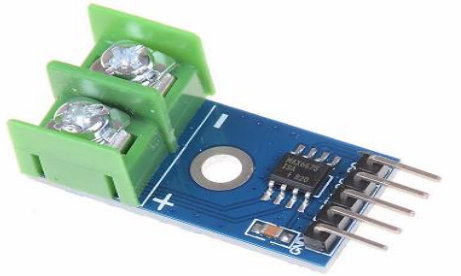


Figure 4.4 K TYPE TEMPERATURE SENSOR (MAX6675)

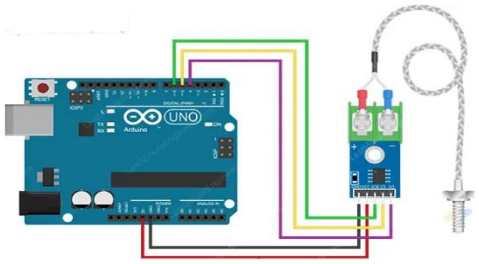


Figure 4.5 MAX6675 connection Diagram

This temperature sensor and module makes use of the Maxim MAX6675 K-Thermocouple to digital converter IC to provide a microcontroller compatible digital serial interface (SPI compatible) to provide accurate temperature compensated measurement of the supplied K-Type thermocouple sensor. It has a 12bit resolution providing temperature readings from 0°C to 1024°C (max temperature of supplied sensor is 450°C) with a resolution of 0.25°C. Screw terminals allow for connection to the thermocouples spade connectors and a 5pin standard 0.1″ header provides an interface to a microcontroller such as an Arduino development board.

4.2.5 Pressure Transducer Sensor



Figure 4.6 Pressure Transducer Sensor

A Pressure Transducer Sensor is a compact device used to measure the pressure of oil, fuel, gas, water, and air. It operates on a 5 V DC supply and provides an analog output signal, making it easy to integrate with controllers and monitoring systems. It is suitable for a wide range of applications in industrial, automotive, and fluid monitoring systems. The sensor is designed with a metal and ABS housing, offering durability and stable performance, while requiring a very low operating current (≤10 mA). Due to its affordability, versatility, and ease of use, it is commonly applied in projects involving automation, pressure monitoring, and fluid control.

4.2.6 Water Heater 1000 Watt



Figure 4.7 Water Heater

A 1000-watt water heater is an electric device designed to heat water using 1000 watts of power consumption. It is generally lightweight and portable, making it suitable for everyday purposes such as heating water for domestic use, preparing tea or coffee, or warming water in buckets and tubs. Its compact design ensures ease of handling, while the high wattage rating allows for fast and efficient heating performance. Due to its convenience, affordability, and versatility, the 1000-watt water heater is commonly used in households, dormitories, and small-scale applications where quick hot water availability is required.

4.2.7 Electric Solenoid Valve (2w-200-20)



Figure 4.8 Electric Solenoid Valve

A 2W-200-20 solenoid valve is an electro-mechanical device used to control the flow of fluids such as water, air, gas, or oil. It is a 2-way, 2-position, normally closed brass valve, meaning it has two ports (inlet and outlet) and remains closed until energized. Operating on 110V AC, it supports pressures up to 10 kg/cm² (0.8 MPa) and temperatures from –5°C to +80°C. With its durable, corrosion-resistant design and stable performance, this valve is widely used in water treatment, industrial automation, and circulation systems.

4.2.8 Pressure Relief Safety Valve (PRV)



Figure 4.9 Pressure Relief Safety Valve (PRV)

A Pressure Relief Valve (PRV) is a safety device that automatically releases excess pressure to protect systems from failure, explosion, or damage. Made commonly of brass with threaded connections, it uses a spring-operated mechanism to maintain safe operating limits. PRVs are widely used in water, oil, air, and heating systems, ensuring reliability, safety, and compliance with international standards.

4.2.9 Steam Exhaust Valve



Figure 4.10 Steam Exhaust Valve

A steam exhaust valve allows the safe and controlled release of exhaust steam, gases, or condensate, preventing pressure buildup and ensuring efficient operation. Typically made of bronze or brass in sizes from 1/4" to 3", these valves feature high pressure ratings, fire-tested sealing, and asbestos-free gland packings. They are widely used in turbines, boilers, engines, and steam pipelines for reliable performance and safety.

4.2.10 Tri-Clamp (TC)



Figure 4.11 Tri Clamp (TC)

A Tri-Clamp is a hygienic, leak-proof piping system that joins two ferrule ends with a gasket and clamp. Made of stainless steel, it is durable, easy to assemble, and ideal for food, beverage, and pharmaceutical applications where sanitation and quick maintenance are critical.

4.2.11 WATER LEVEL SENSOR



Figure 4.12 Water Label Sensor

water level sensors are durable, corrosion-resistant devices used to monitor and control water levels, preventing overflows, ensuring proper supply, and protecting equipment such as pumps. Suitable for high-temperature, corrosive, and hygienic environments, they are widely applied in food processing, chemical industries, sewage treatment plants, reservoirs, and dams.

4.2.12 Socket Fitting



Figure 4.13 Socket

An SS socket is a type of fitting that is used to connect two pipes or tubes. The socket is designed to fit over the end of a pipe and is held in place with a weld or a screw. SS sockets are made using stainless steel, a type of alloy known for its excellent corrosion-resistant properties.

4.2.13 Syphon Pipe



Figure 4.14 Syphon pipe

A syphon pipe is a device that moves liquids from one place to another using gravity and pressure differences. It works by drawing liquid up through a U-shaped tube, after which gravity causes it to flow down to a lower level.

4.2.14 Temperature Gauge



*Figure 4.**Error! No text of specified style in document.**15 Temperature Gauge*

A temperature gauge is an instrument used to measure and display the temperature of a system, substance, or environment in real time. It works by sensing heat through a probe or sensor and converting it into a readable value, typically shown on a dial, digital screen, or indicator. Temperature gauges are commonly used in industries, laboratories, vehicles, and household appliances to monitor processes, ensure safety, and maintain efficiency. They come in various types, such as analog, digital, and infrared, each designed for specific applications and accuracy requirements.

4.3 Fabrication of Main Body

This section describes the step-by-step process of building the main body of the machine, from design and materials collection to assembly, testing, and finishing. It shows how the mechanical structure is carefully fabricated and prepared for integration with electrical and control systems.

4.3.1 Designing

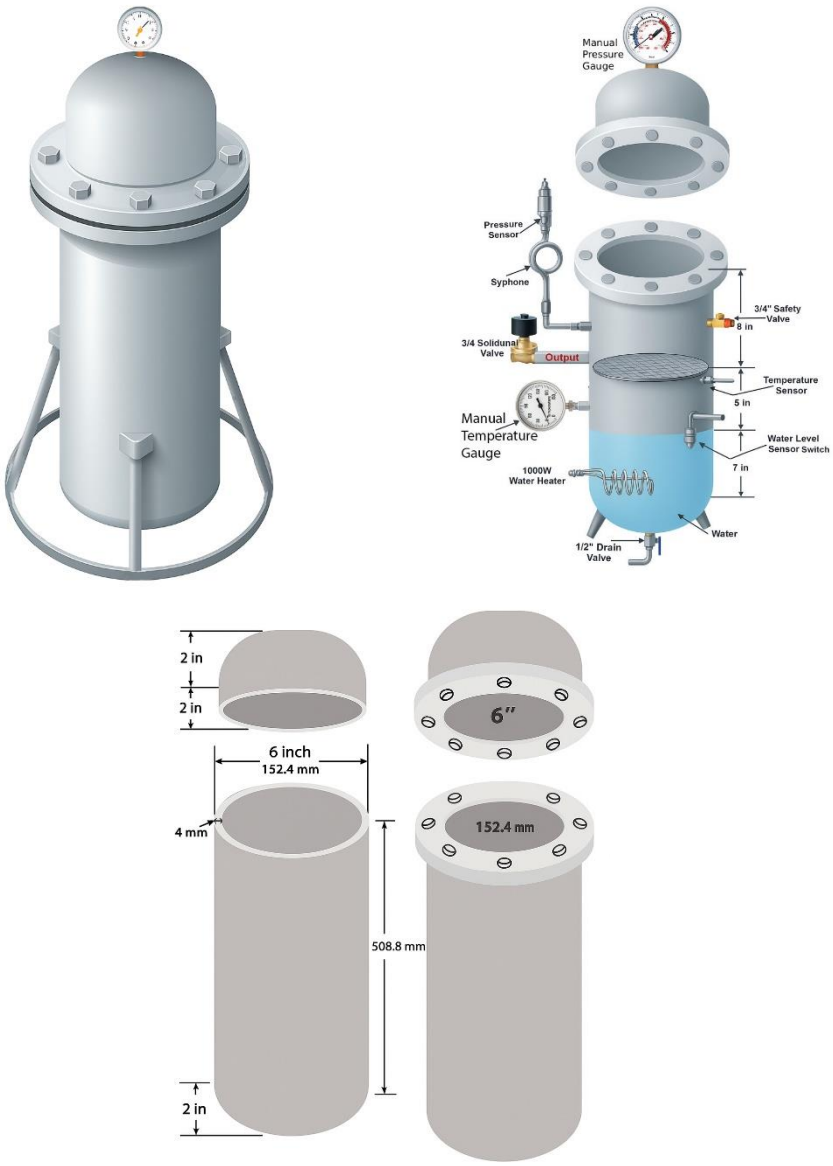


Figure 4.16 3D Drawing of the Machine

A detailed 3D model of the Steam Sterilizer's main body is created, showing the mechanical structure, dimensions, and placement of all components. Adobe Illustrator is used to develop precise drawings, allowing visualization of the assembly before fabrication. This phase ensures all parts fit accurately, identifies potential issues early, and guides the workshop fabrication process.

4.3.2 Materials Collection

All required materials, including Stainless Steel Cylinder, Pressure Gauge, Thermocouple Heater, Tri-Clamp (Tc), Socket Fitting, valves, sensors, and other mechanical components, are collected according to design specifications. Each material is checked for quality, durability, and suitability to withstand high pressure and temperature conditions. Proper material selection ensures safety, reliability, and long-term performance of the sterilizer.

4.3.3 Workshop / Fabrication



Figure 4.17 Fabrication of Steam Sterilizer Main Body

The main body is assembled in the workshop using cutting, shaping, and welding techniques. Precision is maintained to ensure proper alignment of all components, forming the structural framework of the pressure chamber. Special attention is given to joint quality, surface smoothness, and dimensional accuracy to ensure safety and operational efficiency.

4.3.4 Testing

Once fabricated, the main body undergoes rigorous testing. This includes leak inspections, pressure testing (LIC test), and dimensional checks to confirm safety and compliance with design standards. Testing ensures that the pressure chamber can handle high-pressure steam without deformation or leakage.

4.3.5 Finishing

Finally, the body is polished, cleaned, and any sharp edges are removed. After finishing, the main body is ready for integration with electrical components, sensors, and control systems, completing the mechanical assembly of the Steam Sterilizer.



Figure 4.18 Top view After Finishing



Figure 4.19 Front View After Complete Fabrication & Finishing.

Chapter 5: Electrical Design

This chapter explains the electrical system of the Steam Sterilizer, covering the design, fabrication, assembly, and testing of both the motherboard (PCB[iv]) and the control panel. The chapter highlights how electrical components are integrated with the mechanical system to control temperature, pressure, and timing automatically.

5.1 Motherboard Design & Manufacturing

The motherboard is the heart of the Steam Sterilizer's electrical system. It manages sensor inputs, controls actuators[v], interfaces with the control panel, and communicates with the SCADA[vi] system, allowing the user to monitor and control the system through a user-friendly graphical interface (GUI[vii]). The following steps were followed to develop it:

5.1.1 Designing the electrical circuit

The electrical circuit is designed to connect all sensors, relays, heaters, and valves efficiently. The design ensures proper voltage levels, current handling, and safety features. Careful planning prevents short circuits and ensures smooth communication between components.

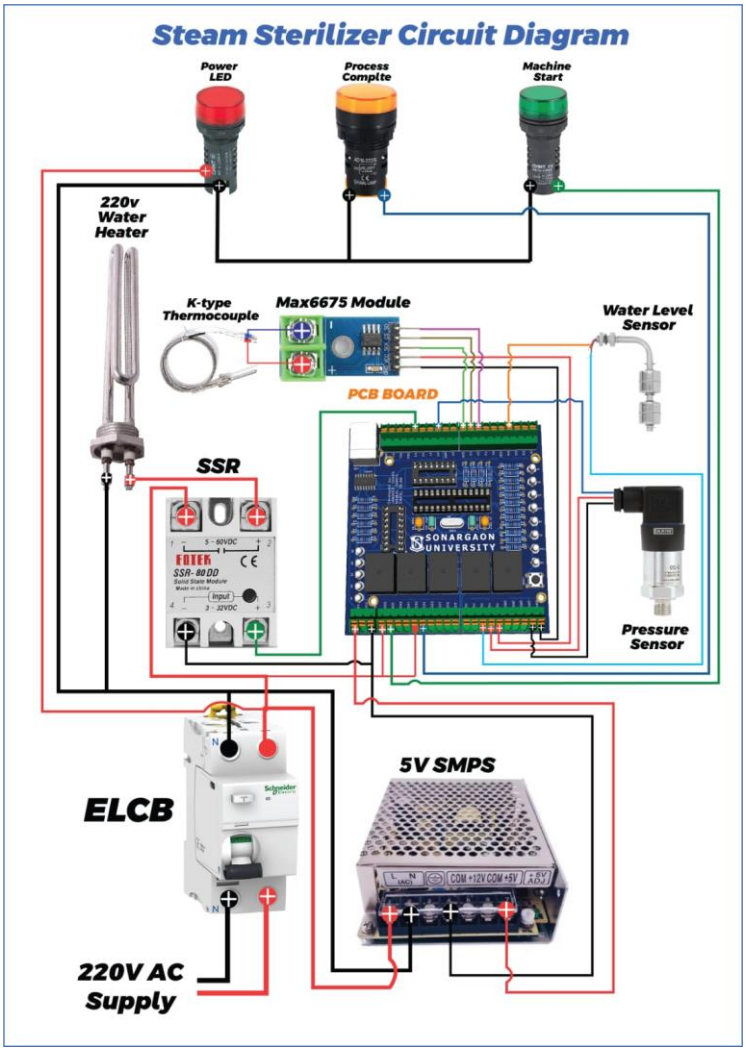


Figure 5. 1 Electrical Circuit Diagram

5.1.2 Preparing the schematic diagram

The electrical circuit design was converted into a detailed schematic diagram using EasyEDA, an online PCB design platform. This schematic shows the interconnections between sensors, actuators, relays, and the microcontroller, ensuring that all signals and power lines are correctly mapped before PCB layout.

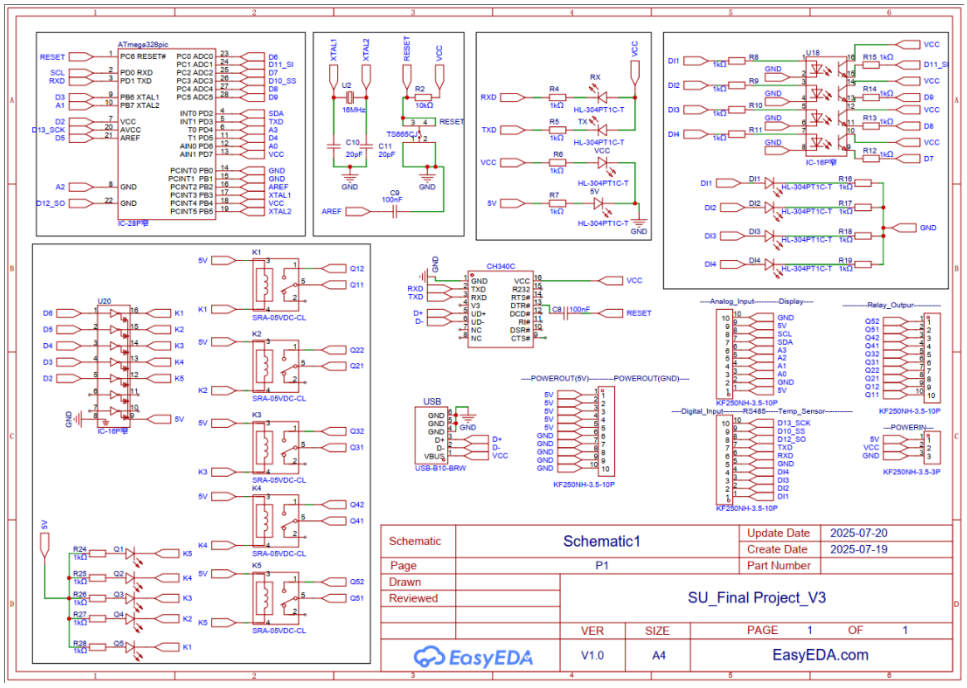
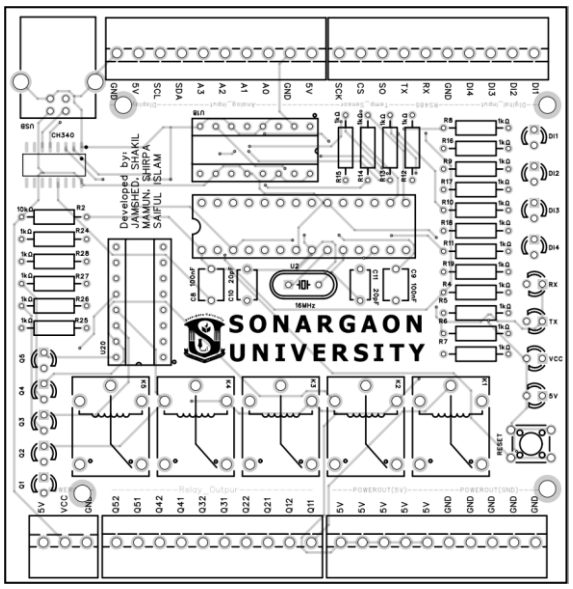


Figure 5. 2 Schematic circuit diagram

5.1.3 Converting the schematic into a PCB layout

Once the schematic was finalized, it was converted into a PCB layout using EasyEDA's PCB editor. Components were placed optimally, and copper traces were routed to minimize noise, interference, and heat buildup. Ground planes and power distribution were carefully managed to ensure stable performance.



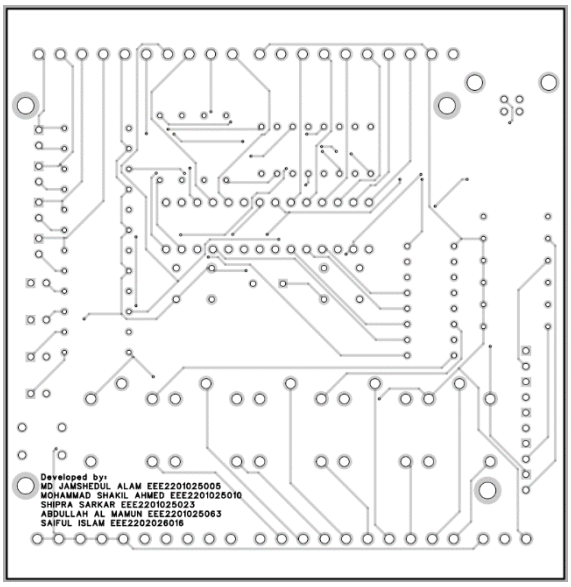


Figure 5. 3 PCB Layout (Front & Back)

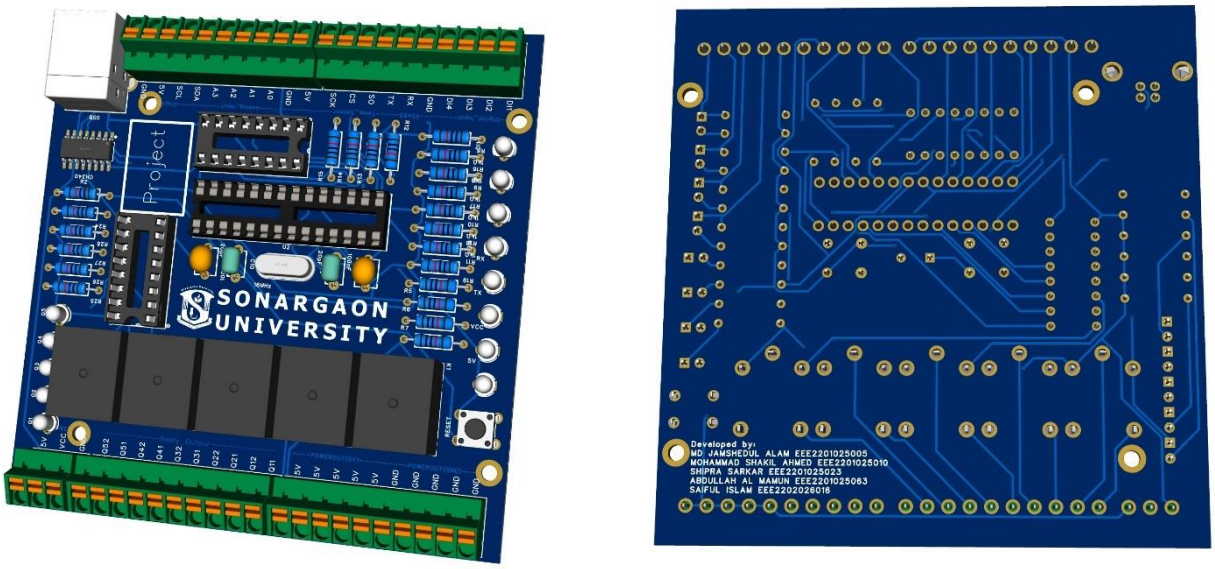


Figure 5. 4 3D view of PCB (Front & Back)

5.1.4 Sending the PCB design for production.

The completed PCB design files (Gerber files) were exported from EasyEDA and sent to JLCPCB, a professional PCB manufacturing company. The boards were fabricated according to project specifications, with high-quality copper layers, accurate drill holes, and proper solder masks to ensure durability and long-term use.

Table 5. 1 BOM sheet of PCB

SL	Quantity	Comment	Designator	Footprint	Value	Manufacturer Part	Manufacturer	Supplier Part
1	13	HL-304PT1C-T	5V,DI1,DI2,DI3,DI4,Q1,Q2,Q3,Q4,Q5,RX,TX,VCC	LED-TH_BD3.9-P2.54-L-FD		HL-304PT1C-T	HONGLITRONIC	C568940
2	4	KF250NH-3.5-10P	----------Relay_Outpur---------	CONN-TH_KF250NH-3.5-10P		KF250NH-3.5-10P	KEFA	C976564
3	1	KF250NH-3.5-3P	---POWERIN---	CONN-TH_KF250NH-3.5-3P		KF250NH-3.5-3P	KEFA	C976557
4	1	IC-28P窄	ATmega328pic	DIP-28_L36.0-W10.2-P2.54-LS10.2-BL		IC-28P窄	BOOMELE	C16653
5	2	100nF	C8,C9	CAP-TH_L5.0-W4.0-P5.00-D1.0	100nF	CT4G-0805-2X1-63V-0.1μF-K	TORCH	C400087
6	2	20pF	C10,C11	CAP-TH_L4.8-W3.2-P5.08-D0.5	20pF	CC4-0805-CG-50V-20pF-J	TORCH	C123152
7	1	CH340C	CH340	SOP-16_L10.0-W3.9-P1.27-LS6.0-BL		CH340C	WCH	C84681
8	5	SRA-05VDC-CL	K1,K2,K3,K4,K5	RELAY-TH_SRA-X-CX		SRA-05VDC-CL		C9900014999
9	1	10kΩ	R2	RES-TH_BD2.3-L6.5-P10.50-D0.5	10kΩ	MF1/4W-10KΩ±1% T	CCO	C119347
10	21	1kΩ	R4,R5,R6,R7,R8,R9,R10,R11,R12,R13,R14,R15,R16,R17,R18,R19,R24,R25,R26,R27,R28	RES-TH_BD2.3-L6.5-P10.50-D0.5	1kΩ	MF1/4W-1KΩ±1% T	CCO	C119324
11	1	TS665CJ	RESET	SW-TH_4P-L6.0-W6.0-P4.50-LS6.5		TS665CJ	SHOU HAN	C393938
12	1	16MHz	U2	HC-49US_L11.5-W4.5-P4.88	16MHz	XIHCELNANF-16MHZ	TAITIEN	C295075
13	2	IC-16P窄	U18,U20	DIP-16_L20.3-W10.2-P2.54-LS7.6-BL-2		IC-16P窄	BOOMELE	C2326
14	1	USB-B10-BRW	USB	USB-B-TH_USB-B10-BRW		USB-B10-BRW	XUNPU	C720549

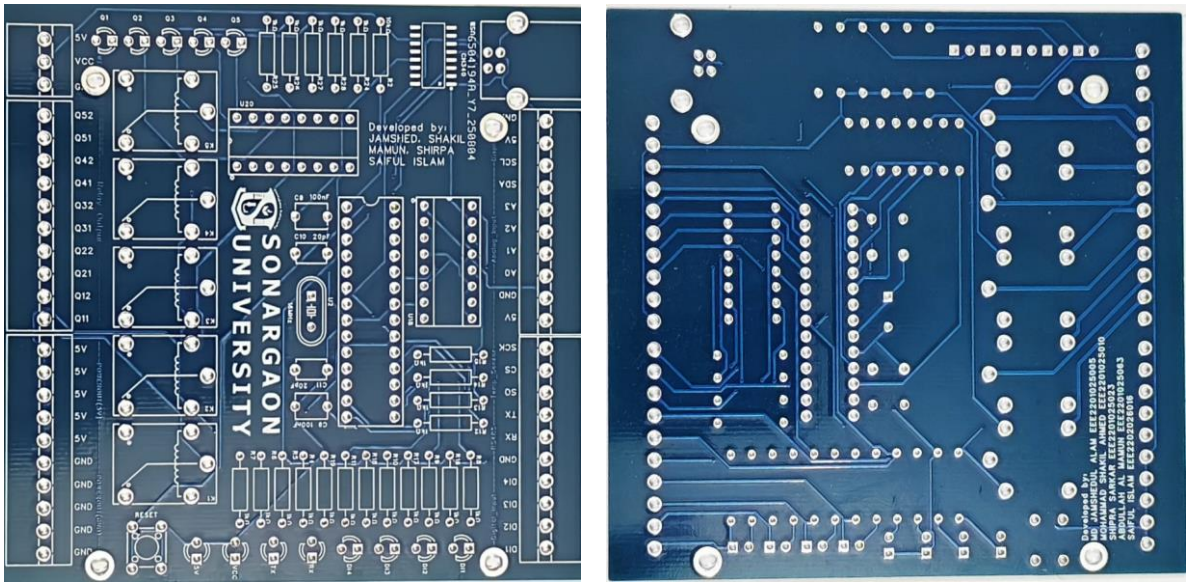


Figure 5. 5 PCB after Fabrication (Front & Back)

5.1.5 Collecting Components for PCB Assembly

After receiving the fabricated PCB from the manufacturer, all required electronic components were collected and prepared. The components include:

- ICs (Integrated Circuits)
- Relays
- Crystal Oscillator
- Resistors
- Capacitors
- Push Buttons
- LEDs
- USB Port
- Screw Terminal

5.1.5.1 ICs (Integrated Circuits)

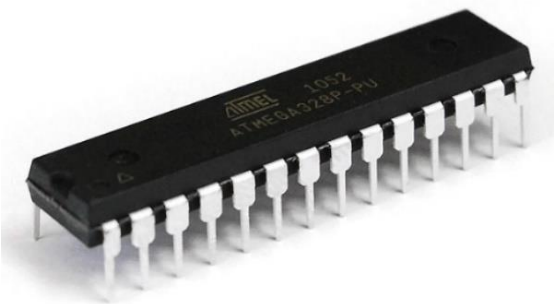


Figure 5. 6 ATmega328P

The ATmega328P is an 8-bit microcontroller from Atmel's AVR family, commonly used in Arduino boards. It provides digital I/O pins, timers, ADC, UART, SPI, and I²C interfaces, making it highly versatile for embedded systems. In this project, it acts as the main controller, processing sensor data, running the PID algorithm, and coordinating the overall operation of the steam sterilizer.

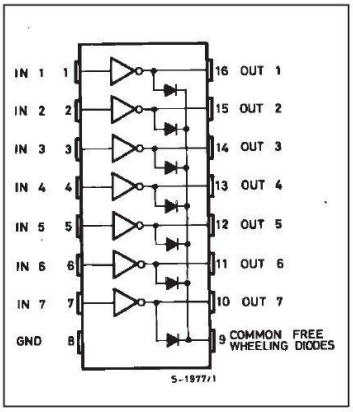


Figure 5. 7 ULN2003AN

The ULN2003AN is a Darlington transistor array IC that can drive high-current loads such as relays, solenoids, or motors directly from logic-level signals. Each channel can handle up to 500 mA, and built-in diodes protect against back EMF from inductive loads. In this project, it is used to drive relays that control the heater, valves, and other actuators, enabling the microcontroller to handle higher power devices safely.

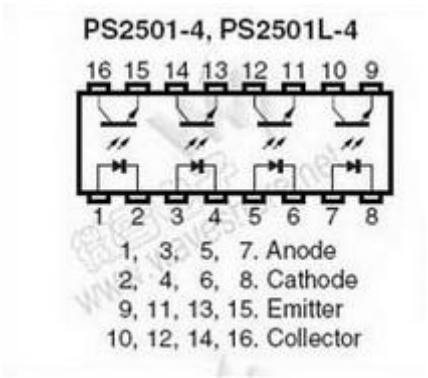


Figure 5. 8 PS2501-4

The PS2501-4 is a quad-channel optocoupler designed to provide electrical isolation between high-voltage circuits and the low-voltage microcontroller system. It transfers signals using an LED and phototransistor pair, protecting the control board from surges or noise. In the sterilizer circuit, it ensures safe and reliable communication between the control signals and the power section.

5.1.5.2 Relays



Figure 5. 9 Relay

Relay is an electrically operated switch that uses a small control signal to control a larger electrical load. It works on the principle of electromagnetism—when a low voltage (such as 5V DC) is applied to its coil, the coil generates a magnetic field that pulls a switch contact, thereby connecting or disconnecting the load circuit.

5.1.5.3 Crystal Oscillator



Figure 5. 10 Crystal Oscillator

Crystal oscillator is an electronic component that generates a stable clock signal using the resonance of a quartz crystal, ensuring precise timing in circuits. In a microcontroller, it acts as the main clock source or "heartbeat," synchronizing all internal operations, instruction execution, and communication processes to maintain accuracy and proper functionality.

5.1.5.4 Resistors



Figure 5. 11 Resistor

Resistor is an electronic component that opposes the flow of electric current and controls the voltage and current in a circuit. It is used to limit current, divide voltage, protect components, and provide biasing in electronic circuits.

5.1.5.5 Capacitors

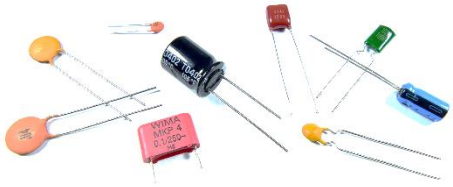


Figure 5. 12 Capacitors

Capacitor is an electronic component that stores electrical energy in the form of an electric field. It is used for filtering, smoothing voltage, coupling and decoupling signals, energy storage, and timing applications in electronic circuits.

5.1.5.6 Push Buttons



Figure 5. 13 Push Button

Push Button is a simple switch that allows current to flow only when it is pressed. It is used in circuits to give manual input, such as starting or stopping a process, resetting a system, or sending signals to a controller.

5.1.5.7 LEDs



Figure 5. 14 LED

LEDs (Light Emitting Diodes) are electronic components that emit light when current flows through them. They are used in circuits as indicators to show the status of power, signals, or operations in a controller-based system.

5.1.5.8 USB Port



Figure 5. 15 USB Port

USB Port (Universal Serial Bus Port) is a standard interface used to connect electronic devices for communication and power supply. It is commonly used for programming, data transfer, and providing 5V DC power to the circuit.

5.1.5.9 Screw Terminal



Figure 5. 16 screw terminal

Screw terminal is a type of electrical connector that secures a wire by tightening it under a screw. It provides a reliable and strong connection between the wire and a device or circuit. Screw terminals are commonly used in electrical panels, control systems, and industrial equipment.

5.1.6 Assembling the components on the PCB

After collecting all electronic components—such as resistors, capacitors, ICs, LED, crystal oscillator, relays, and connectors—are carefully mounted on the board. The soldering process is carried out manually to ensure accuracy and strong connections. Each joint is inspected to confirm that the components are placed in their correct positions and securely soldered, ensuring the PCB functions as intended.

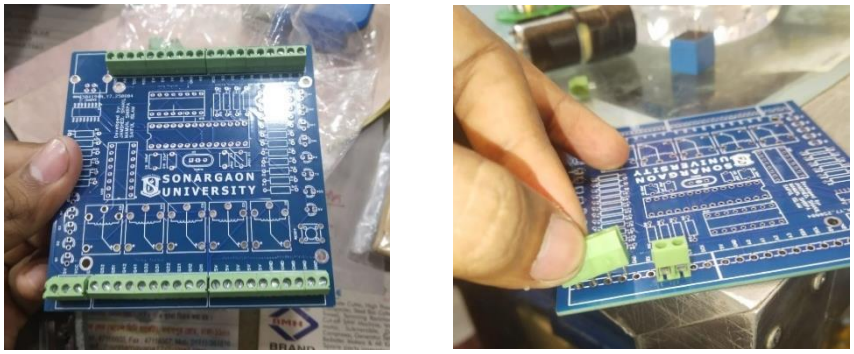


Figure 5. 17 Assembling the components on the PCB

5.1.7 Testing the PCB

Once all components are soldered onto the PCB, the board is carefully tested to verify its functionality. First, a continuity test is performed using a multi-meter to ensure that there are no short circuits or open connections. After confirming the wiring integrity, the board is powered on to check the voltage supply and stability. Input and output signals of the ICs, microcontroller, and crystal oscillator are monitored to confirm proper operation. Finally, the actuators and sensors are connected to validate that the PCB can correctly send and receive signals as per the design requirements.

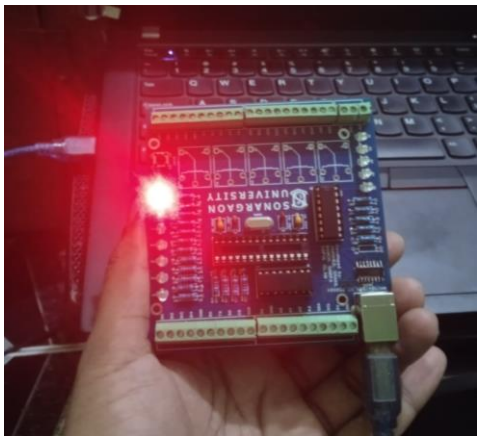


Figure 5. 18 Testing PCB

5.2 Control Panel

5.2.1 Designing the layout

The control panel layout was designed using AutoCAD Electrical to ensure proper arrangement and wiring of all components. The design emphasized safety, functionality, and ease of operation. Components such as SMPS, Motherboard (PCB), Circuit Breakers (CB), Solid State Relays (SSR), Indicators, Connectors, and Ports were placed in a logical manner for efficient wiring and convenient user access. The AutoCAD design also helped in minimizing wiring complexity.

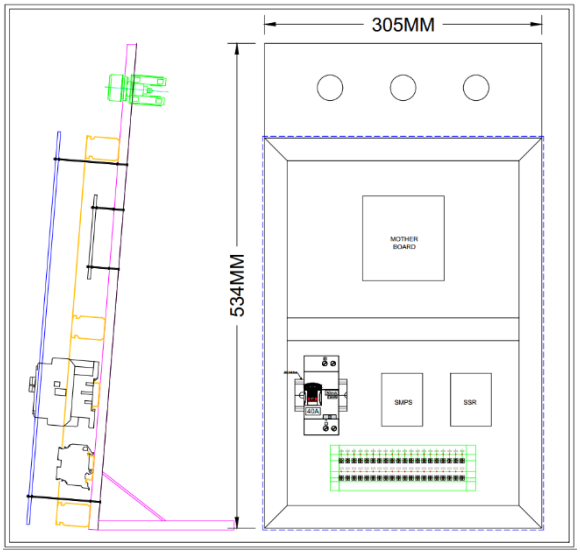


Figure 5. 19 Layout of control Panel

5.2.2 Collecting components

After completing the layout design, the required components were collected. These included the SMPS for stable power supply, the motherboard (PCB) for system control, circuit breakers for protection, solid-state relays for switching loads, indicators for status display, and necessary connectors and ports for external interfacing.

- SMPS
- Motherboard
- Circuit Breakers (CB)
- SSR
- Indicators
- connectors

5.2.2.1 SMPS



Figure 5. 20 SMPS (Switch Mode Power Supply)

Switch Mode Power Supply (SMPS) is an electronic power supply that converts electrical power efficiently using high-frequency switching and energy storage components like inductors and capacitors. Unlike traditional linear power supplies, SMPS regulates voltage by rapidly switching the input on and off and controlling the energy delivered to the output. It is widely used in computers, industrial machines, and consumer electronics because it is compact, lightweight, generates less heat, and provides stable output voltages for sensitive electronic components.

5.2.2.2 Motherboard



Figure 5. 21 Motherboard

Motherboard is the main printed circuit board (PCB) of a electronic device that connects and allows communication between all components.

5.2.2.3 ELCB (Earth Leakage Circuit Breaker)



Figure 5. 22 Earth Leakage Circuit Breakers (ELCB)

An Earth Leakage Circuit Breaker (ELCB) is a safety device that disconnects the power supply when it detects leakage current flowing to the ground, which could indicate insulation failure or electric shock risk. In this project, the ELCB is used to protect both the operator and the equipment by ensuring safe operation of the steam sterilizer against electrical faults.

5.2.2.4 SSR



Figure 5. 23 Solid State Relay (SSR)

Solid State Relay (SSR) is an electronic switching device that switches electrical loads on or off without any moving parts, using semiconductor components like thyristors, triacs, or transistors. SSRs provide fast, silent, and reliable switching compared to mechanical relays and are commonly used in industrial automation, heating systems, and machinery where precise control and long operational life are required.

5.2.2.5 Indicators



Figure 5. 24 Indicators

Indicators are devices, usually lights or LEDs, used to visually show the status of a machine, circuit, or system. They inform the user about conditions such as power ON/OFF, faults, operation modes, or process status. Indicators are widely used in industrial machines, control panels, and electronic devices to provide quick, clear, and immediate feedback, helping operators monitor and respond to system conditions efficiently and safely.

5.2.2.6 Connectors



Figure 5. 25 Connectors

Connectors are components used to join electrical wires, cables, or electronic circuits together, allowing the flow of electricity or signals between them. They provide a secure and often removable connection, making it easy to assemble, maintain, or replace parts of a system without soldering.

5.2.3 Building the panel in-house.

The assembly of the control panel was carried out in-house. The components were mounted inside a metallic enclosure, following industrial wiring standards. Neat and structured wiring ensured both safety and troubleshooting convenience. Proper grounding and shielding were maintained to minimize electrical noise and improve system reliability.

5.2.4 Testing the control panel.

After assembly, the panel was tested thoroughly. The SMPS output was checked for stable voltage supply, the PCB was powered up, and the SSR operation was verified under load. Indicators, circuit breakers, connectors, and communication ports were also tested individually. Finally, the entire system was integrated and tested for proper coordination between the control panel, motherboard, and external actuators.

Chapter 6: Software Desing

6.1 Microcontroller Programming

The controller code has been developed using the Arduino IDE, which provides a simple and efficient platform for programming microcontrollers. The code is responsible for controlling the connected devices, executing predefined logic, and ensuring smooth communication between the hardware components and the user interface. Additionally, the program enables data exchange and communication with the SCADA system, allowing real-time monitoring and control of the machine through the integrated platform.

6.1.1 Controller Code with Explanation

6.1.1.1 Variable Declarations

```
unsigned long startMillis;
unsigned long currentMillis;
const unsigned long period = 1000;
```

These variables are used for tracking time using millis(). startMillis stores the last recorded time, currentMillis stores the present time, and period defines the measurement interval (1000 ms = 1 second).

6.1.1.2 Temperature Sensor Setup (MAX6675)

```
#include "max6675.h"

int thermoDO  = 12;
int thermoCS  = 10;
int thermoCLK = 13;

float upper, lower, old, current;

MAX6675 thermocouple(thermoCLK, thermoCS, thermoDO);
```

The MAX6675 module is used with a K-type thermocouple to measure temperature. Pins are defined for Data Out, Chip Select, and Clock. The variables upper and lower define tolerance levels (+/- 0.5 °C) to avoid unnecessary repeated serial printing.

6.1.1.3 Process Parameters

```
float Temp_Set_Point = 90;
float Time_Set_Point = 2;
bool SterilizetionComplited;
```

Temp_Set_Point stores the target sterilization temperature (default 90 °C). Time_Set_Point defines required sterilization duration in minutes (default 2 minutes). SterilizetionComplited is a flag that indicates sterilization completion.

6.1.1.4 PID Control Setup

```
#include <PID_v1.h>
#include "flprogUtilites.h"
double _PID_1_Out=0.0;
double _PID_1_SP=0.0;
double _PID_1_In=0.0;PID _PID_1(&_PID_1_In, &_PID_1_Out, &_PID_1_SP, 10, 2,
6000, DIRECT);
```

The PID_v1 library is used to control heater output. Inputs are the current temperature, setpoint temperature, and output power. The PID parameters are Kp=10, Ki=2, Kd=6000. Output limits are set to 0–1000, and mode is DIRECT.

6.1.1.5 Timer Variables

```
bool timer1I = 0;
bool timer1O = 0;
unsigned long timer1P = 0UL;
```

Timer variables manage sterilization duration. timer1I starts timing when the setpoint temperature is reached. timer1O indicates completion, and timer1P stores the start time.

6.1.1.6 Setup Function

```
void setup() {
  Serial.begin(9600);
  Serial.setTimeout(10);

  pinMode(11, OUTPUT);
  pinMode(9, OUTPUT);

  digitalWrite(11, LOW);
  digitalWrite(9, LOW);

  _PID_1.SetOutputLimits(0, 1000);
  _PID_1.SetMode(AUTOMATIC);
  _PID_1.SetSampleTime(10);

  startMillis = millis();
}
```

Initializes serial communication at 9600 baud. Configures pin 11 as indicator LED and pin 9 as heater control. PID output range and mode are set. Initializes millis()-based timing.

6.1.1.7 Loop Function

```
void loop() {
  if (Serial.available() > 0) {
    firstchar = Serial.read();
    RXdata = Serial.readString();
    RXdata.trim();
```

```
  }

  switch (firstchar) {
    case 'A':
      if (RXdata == "P") { ... }
      if (RXdata == "_Stop") { ... }
      break;
    case 'B':
      Temp_Set_Point = RXdata.toFloat();
      break;
    case 'C':
      Time_Set_Point = RXdata.toFloat();
      break;
  }

  Temp_Section();
  PID_Section();
  Timer1_Section();

  RXdata = "";
  firstchar = '\0';
}
```

Main program cycle: reads serial commands, processes them, and calls three sections: Temp_Section() for reading temperature, PID_Section() for heater control, and Timer1_Section() for timing sterilization duration.

6.1.1.8 Temperature Section

```
void Temp_Section(void) {
  currentMillis = millis();

  if (currentMillis - startMillis >= period) {
    current = thermocouple.readCelsius();
    upper   = current + 0.5;
    lower   = current - 0.5;

    if (current != old) {
      if ((old <= lower) || (old >= upper)) {
        Serial.print("T");
        Serial.println(current);
        old = current;
      }
    }

    startMillis = currentMillis;
  }
}
```

Reads thermocouple temperature every 1 second. Prints new temperature only if change is greater than +/- 0.5 °C to reduce serial communication load.

6.1.1.9 PID Control Section

```
void PID_Section(void) {
  _PID_1_In = current;
  _PID_1_SP = Temp_Set_Point;
  _PID_1.Compute();
  _tempVariable_unsignedlong = millis();

  if (_tempVariable_unsignedlong - _PID_1_StartTime > 100) {
    _PID_1_StartTime += 100;
  }

  if (_PID_1_Out > _tempVariable_unsignedlong - _PID_1_StartTime) {
    _PID_1_RelOut = 1;
  } else {
    _PID_1_RelOut = 0;
  }

  if (digitalRead(11) == HIGH) {
    digitalWrite(9, _PID_1_RelOut);
  } else {
    digitalWrite(9, LOW);
  }
}
```

PID computes new heater output. Uses time-proportional control: compares PID output with elapsed time in each cycle to decide ON/OFF state. Heater only operates when sterilization program is active.

6.1.1.10 Timer Section

```
void Timer1_Section(void) {
  if (Temp_Set_Point <= current) {
    if (timer1I) {
      if (flprog::isTimer(timer1P, (Time_Set_Point * 60000))) {
        timer1O = 1;
      }
    } else {
      timer1I = 1;
      timer1P = millis();
    }
  } else {
    timer1O = 0;
    timer1I = 0;
  }

  SterilizetionComplited = timer1O;

  if (SterilizetionComplited == 1 && digitalRead(11) == HIGH) {
    // Serial.println("bSterilizetionComplited");
  }
}
```

Ensures sterilization time is completed only after reaching set temperature. Uses flprog::isTimer() to count required minutes. Once complete, SterilizetionComplited flag is set.

6.1.2 Overall Working Principle

1. **Initialization:** System waits for command via serial input.
2. **Start Command**: User sends 'AP' to enable heater and indicator.
3. **Temperature Regulation:** MAX6675 measures temperature and PID controls heater.
4. **Sterilization Timer:** Once temperature ≥ setpoint, timer starts. After duration, sterilization completes.
5. **Stop Command:** User sends 'A_Stop' to turn heater and indicator off.

6.1.3 Purpose in Project

This project implements a microcontroller-based sterilization system. It measures temperature using a thermocouple, regulates heater output via PID control, ensures sterilization is performed for a set safe duration, and allows user control via serial communication.

6.1.4 Program Flowchart

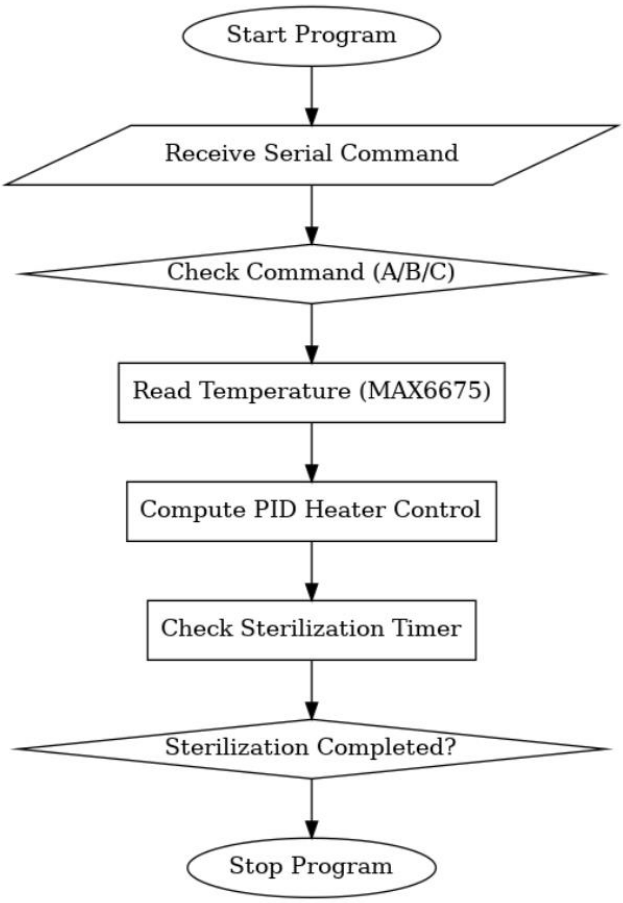


Figure 6. 1 Program Flowchart

6.2 SCADA Programming and Database Integration

The SCADA system for this project was developed using Microsoft Visual Studio Community 2022, which provides a robust [viii] environment for designing both the interface and the control logic. The SCADA interface allows operators to monitor the machine in real-time, send control commands to the microcontroller, and track operational parameters such as temperature, timing, and system status. Through this system, users can fully control the machine, start/stop processes, and ensure smooth execution of the sterilization operation, making it interactive and user-friendly. Additionally, operators can create and save custom recipes for different sterilization processes, and generate one-click batch reports for every production cycle, ensuring efficiency, traceability, and ease of operation.

For data management, SQL Server Management Studio is used to store and retrieve all process-related information, including batch reports, historical logs, and sensor data. The SCADA system communicates with the SQL database to record every event and process parameter, enabling traceability and reporting. This integration ensures that operators can generate detailed batch reports, analyse machine performance over time, and maintain a reliable record of production data for quality control and auditing purposes.

6.2.1 Designing SCADA & Database Integration with Explanation

6.2.1.1 Page 01 (Loading.cs)

Loading Function



Figure 6. 2 Loading Page

```
private void loading_Load(object sender, EventArgs e)
{
    progressBar1.Left = (this.ClientSize.Width - progressBar1.Width) / 2;
    timer1.Start();
}

private void timer1_Tick(object sender, EventArgs e)
{
    if (progressBar1.Value < 100)
    {
        progressBar1.Value += 1; // increase by 1%
```

```
        // Show progress %
        labelprogress.Text = $"{progressBar1.Value}%";
    }
    else
    {
        timer1.Stop();
        this.Hide();
        login loginForm = new login();
        loginForm.Show();
    }
}
```

This code displays a centred loading progress bar that increments until 100%, then stops the timer, hides the loading form, and opens the login form.

6.2.1.2 Page 02 (Login.cs)

How to add Database from visual studio

In **Solution Explorer** → Right-**click** your project → **Add** → **New Item** → Choose: **Service-based Database** (Local Database)
Name it: Database1.mdf (or your preferred name) → Click **Add**

Create user table for login

```
CREATE TABLE [dbo].[Users] (
    [Id]       INT        IDENTITY (1, 1) NOT NULL,
    [UserName] NCHAR (10) NULL,
    [Password] NCHAR (10) NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC)
);
```

Move Connection String to App.config (for universal database connection)

```
<connectionStrings>
<add name="SterilizerDB"
   connectionString="Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=D:\GUI\Steam_Sterilizer\Stea
m_Sterilizer\DB_Steam_Sterilizer.mdf;Integrated Security=True"
   providerName="System.Data.SqlClient"/>
</connectionStrings>
```

DatabaseHelper.cs

For universal Database connection.

```
using System;
using System.Data.SqlClient;
using System.Configuration;

namespace STEAM_STAIRELAZER
{
    public static class DatabaseHelper
```

```
    {
        // Static field inside static class is OK
        private static readonly string connStr =
ConfigurationManager.ConnectionStrings["SterilizerDB"].ConnectionString;
//if got error in configurationManger than Right click on References from
solution explorer ➞ Add References ➞ Assemblies ➞ Framework ➞ find
system configuration ➞ check the box ➞ ok

        public static SqlConnection GetConnection()
        {
            return new SqlConnection(connStr);
        }
    }
}
```

Login Page Desing & Functions.



Figure 6. 3 Login page

Loging button Click Event

```
private void Login_Button_Click(object sender, EventArgs e)
{
    string username = UserName_comboBox.Text.Trim();
    string password = Password_Textbox.Text.Trim();

    if (string.IsNullOrEmpty(username) || string.IsNullOrEmpty(password))
    {
        labelError.Text =  "Username & Password must be filled..";
        return;
    }

    using (SqlConnection conn = DatabaseHelper.GetConnection())
    {
        string query = "SELECT Username FROM Users WHERE Username =
@Username AND Password = @Password";
        SqlCommand cmd = new SqlCommand(query, conn);
        cmd.Parameters.AddWithValue("@Username", username);
        cmd.Parameters.AddWithValue("@Password", password);
```

```
        try
        {
            conn.Open();
            SqlDataReader reader = cmd.ExecuteReader();

            if (reader.Read())
            {
                LoggedInUser.Username = reader["Username"].ToString();
                loginAttempts = 0; // reset on success
                this.Hide();
                Home homeForm = new Home();
                homeForm.Show();
            }
            else
            {
                loginAttempts++; // increment attempts
                int remaining = 3 - loginAttempts;
                label_Remaining.Visible = true;
                label_Remaining.Text = "you can try " +
remaining.ToString() + " more time";
                labelError.Text = "Invalid username or password.";
                if (loginAttempts >= 3)
                {

                    Login_Button.Enabled = false;
                    countdownRemaining = lockoutSeconds;
                    label_Remaining.Text = $"Try again in
{countdownRemaining} seconds.";
                    label_Remaining.Visible = true;
                    lockoutTimer.Start();
                }
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Login error: " + ex.Message);
        }
    }
}
```

This code handles the login button click by validating input, checking credentials against the Users table in SQL Server, resetting attempts on success, and enforcing a 3-attempt lockout with a countdown timer on repeated failures.

Holds the currently logged-in username

```
public static class LoggedInUser
{
    public static string Username { get; set; }
}
```

Load usernames Function.

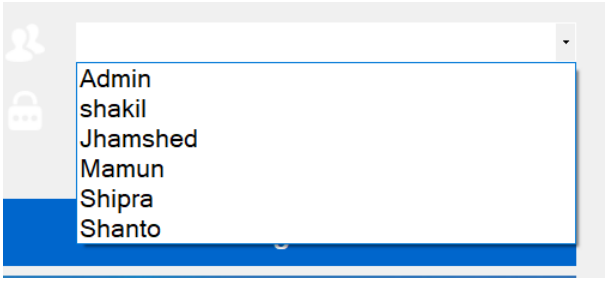


Figure 6. 4 Load Username

```
private void LoadUsernames()
{
    using (SqlConnection conn = DatabaseHelper.GetConnection())
    {
        string query = "SELECT Username FROM Users";
        SqlCommand cmd = new SqlCommand(query, conn);
        try
        {
            conn.Open();
            SqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                UserName_comboBox.Items.Add(reader["Username"].ToString());
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Error loading usernames: " + ex.Message);
        }
    }
}
```

This function retrieves all usernames from the Users table in the database and populates the UserName_comboBox dropdown for convenient selection.



Figure 6. 5 Hide Password

Show / Hide Password

// Password show/hide toggle using eye icon
//add eye_open.png in Resources (Double click ➞ Add new resources)

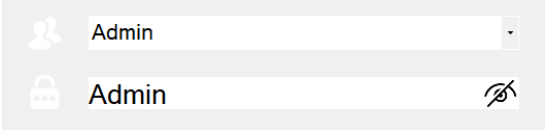


Figure 6. 6 Show Password

```
bool isPasswordVisible = false;
```

```
private void eyeToggleIcon_Click_1(object sender, EventArgs e)
{
    if (isPasswordVisible)
    {
        Password_Textbox.UseSystemPasswordChar = true;
        eyeToggleIcon.Image = Properties.Resources.eye_open;
        isPasswordVisible = false;
    }
    else
    {
        Password_Textbox.UseSystemPasswordChar = false;
        eyeToggleIcon.Image = Properties.Resources.eye_closed;
        isPasswordVisible = true;
    }
}
```

Caps Lock Warning

```
private void Password_Textbox_Enter(object sender, EventArgs e)
{
    labelCapsWarning.Text = Control.IsKeyLocked(Keys.CapsLock) ? "Caps Lock
is ON" : "";
}
```

Enable ENTER key for login (Enter key Event)

```
private void Password_Textbox_KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Enter)
    {
        Login_Button.PerformClick();
    }
}
```

AutoFocus Function

```
private void login_Load(object sender, EventArgs e)
{
     UserName_comboBox.Focus();
}
```

Limit Login Attempts Function

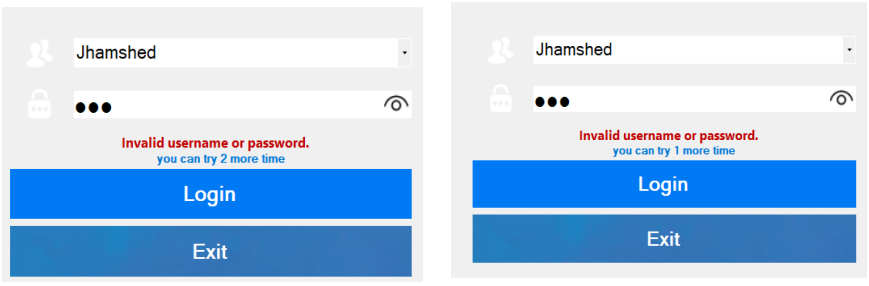


Figure 6. 7 Limit login attempts

```
int loginAttempts = 0;
int lockoutSeconds = 30;
int countdownRemaining;
Timer lockoutTimer = new Timer();
//Inside Form_Load event (or constructor)
lockoutTimer.Interval = 1000; // 1 second
lockoutTimer.Tick += LockoutTimer_Tick;

//Inside login button click
if (reader.Read())
{
    LoggedInUser.Username = reader["Username"].ToString();
    loginAttempts = 0; // reset on success
    this.Hide();
    Home homeForm = new Home();
    homeForm.Show();
}
else
{
    loginAttempts++; // increment attempts
    int remaining = 3 - loginAttempts;
    label_Remaining.Visible = true;
    label_Remaining.Text = "you can try " + remaining.ToString() + " more time";
    labelError.Text = "Invalid username or password.";
    if (loginAttempts >= 3)
    {

        Login_Button.Enabled = false;
        countdownRemaining = lockoutSeconds;
        label_Remaining.Text = $"Try again in {countdownRemaining} seconds.";
        label_Remaining.Visible = true;
        lockoutTimer.Start();
    }
}
```

This logic limits login attempts to three; on failure, it shows remaining tries, and after three unsuccessful attempts, it disables the login button and starts a 30-second countdown before allowing another attempt.

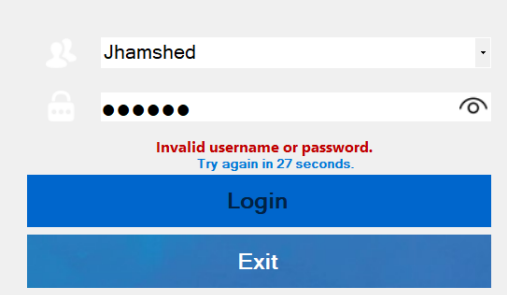


Figure 6. 8 Disable Login button

```
private void timerLockOut_Tick(object sender, EventArgs e)
 {
     countdownRemaining--;
     label_Remaining.Text = $"Try again in {countdownRemaining} seconds.";
```

```
    if (countdownRemaining <= 0)
    {
        lockoutTimer.Stop();
        Login_Button.Enabled = true;
        label_Remaining.Visible = false;
        loginAttempts = 0;
    }
}
```

This function updates the countdown during a lockout period, re-enables the login button, hides the message, and resets the attempt count once the countdown reaches zero.

Remove Screen Flickering / Screen Blinking

//Just add this code in your page

```
protected override CreateParams CreateParams
{
    get
    {
        var cp = base.CreateParams;
        // Enable WS_EX_COMPOSITED to reduce flicker
        cp.ExStyle |= 0x02000000;
        return cp;
    }
}
```

Password Hashing

What is Password Hashing?

Hashing is the process of **converting plain text (like your password) into a fixed-length, irreversible string** using a hashing algorithm (e.g., SHA-256).
Hashing is **one-way**. You can hash a password, but you can't "unhash" it to get the original back.

Why Use Hashing for Passwords?

If someone gains access to your database, they could see **plain passwords** unless they're **hashed**.
Hashing protects your users by storing this instead:

Original: MySecurePass123
SHA-256 Hash: bfd1cf81df749aeb10f0d97d6adf71a8f51aa9b218cb0b2a6cd527f5a67d996b

Even if someone steals the database, they only see the hash — not the password.

Benefits of SHA-256 Hashing

Feature	Benefit
One-way function	Can't reverse it to original
Fixed length	Output is always 64 characters
Collision-resistant	Very unlikely two inputs give same hash
Fast and reliable	SHA-256 is fast and widely supported

OUTPUT (Example)

Action	Output Stored in DB
User input	mypassword123
Stored	ef92b778bafe771e89245b89ecbc08a44a4e166c...

SecurityHelper.cs

```
using System.Security.Cryptography;
using System.Text;

public static class SecurityHelper
{
    public static string HashPassword(string password)
    {
        using (SHA256 sha256 = SHA256.Create())
        {
            byte[] bytes = Encoding.UTF8.GetBytes(password);
            byte[] hashBytes = sha256.ComputeHash(bytes);
            StringBuilder sb = new StringBuilder();
            foreach (byte b in hashBytes)
                sb.Append(b.ToString("x2"));

            return sb.ToString(); // Final hashed password
        }
    }
}
```

Inside button Loging before password

This static class provides a method to **securely hash a password** using SHA-256, converting it into a fixed-length hexadecimal string for safe storage or verification.

Convert password to SHA-256 hash

```
string hashedPassword = SecurityHelper.HashPassword(password);

//Replace all password with hashedPassword inside of button login
```

Add user info, password insert with encryption

5.2.1.3 Page 03 (Register.cs)

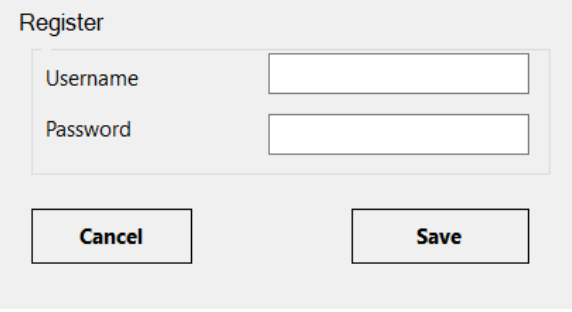


Figure 6. 9 Register page

```
private void btn_save_Click(object sender, EventArgs e)
{
    string username = txtUsername.Text.Trim();
    string rawPassword = txtPassword.Text;

    if (username == "" || rawPassword == "")
    {
        MessageBox.Show("Username and Password are required!");
        return;
    }

    string hashedPassword = SecurityHelper.HashPassword(rawPassword);

    string query = "INSERT INTO Users (Username, Password) VALUES (@username, @password)";
    SqlConnection conn = new
SqlConnection(ConfigurationManager.ConnectionStrings["SterilizerDB"].ConnectionString);
    SqlCommand cmd = new SqlCommand(query, conn);
    cmd.Parameters.AddWithValue("@username", username);
    cmd.Parameters.AddWithValue("@password", hashedPassword);

    conn.Open();
    cmd.ExecuteNonQuery();
    conn.Close();

    MessageBox.Show("User added successfully!");
    txtUsername.Text = "";
    txtPassword.Text = "";


}
```

This function saves a new user by taking the entered username and password, hashing the password using SHA-256, inserting the data into the Users table in the database, and then clearing the input fields after successful addition.

5.2.1.4 Page 04 (ChangePass.cs)

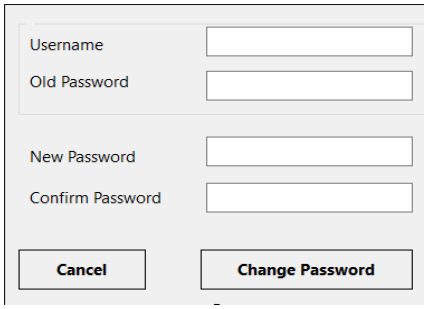


Figure 6. 10 Change Password page

```
private void btn_changePass_Click(object sender, EventArgs e)
 {
     string username = txtUsername.Text.Trim();
     string oldPasswordHash = SecurityHelper.HashPassword(txtOldPassword.Text);
     string newPassword = txtNewPassword.Text;
     string confirmPassword = txtConfirmPassword.Text;
```

```
        if (newPassword != confirmPassword)
        {
            MessageBox.Show("New passwords do not match.");
            return;
        }

        SqlConnection conn = new
SqlConnection(ConfigurationManager.ConnectionStrings["SterilizerDB"].ConnectionString);

        // Verify old password
        string verifyQuery = "SELECT COUNT(*) FROM Users WHERE Username = @username AND Password
= @password";
        SqlCommand verifyCmd = new SqlCommand(verifyQuery, conn);
        verifyCmd.Parameters.AddWithValue("@username", username);
        verifyCmd.Parameters.AddWithValue("@password", oldPasswordHash);

        conn.Open();
        int match = (int)verifyCmd.ExecuteScalar();

        if (match == 1)
        {
            // Update new password
            string newHash = SecurityHelper.HashPassword(newPassword);
            string updateQuery = "UPDATE Users SET Password = @newPassword WHERE Username =
@username";
            SqlCommand updateCmd = new SqlCommand(updateQuery, conn);
            updateCmd.Parameters.AddWithValue("@newPassword", newHash);
            updateCmd.Parameters.AddWithValue("@username", username);

            updateCmd.ExecuteNonQuery();
            MessageBox.Show("Password changed successfully.");
            txtUsername.Text = "";
            txtOldPassword.Text = "";
            txtNewPassword.Text = "";
            txtConfirmPassword.Text = "";
        }
        else
        {
            MessageBox.Show("Old password is incorrect.");
        }

        conn.Close();

    }
```

User to change their password by verifying the old password, ensuring the new password and confirmation match, hashing the new password, updating it in the database, and clearing the input fields upon success.

6.2.1.5 Page 05 (Home.cs)

Create New Home Page UI Layout

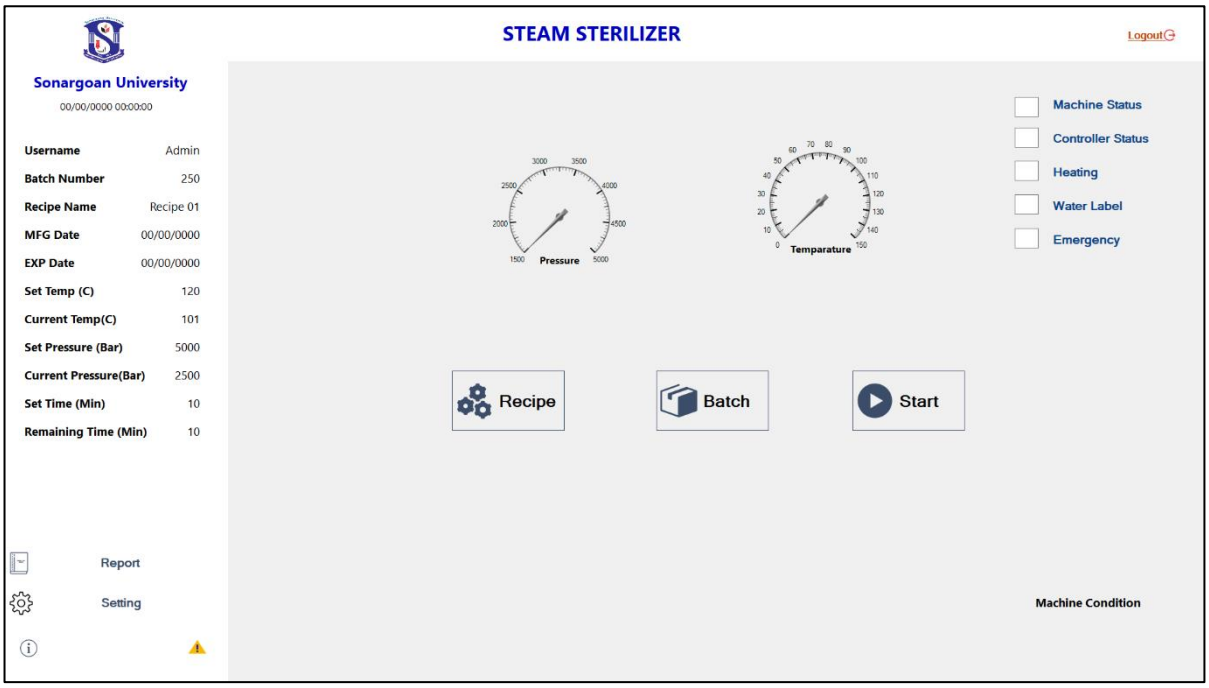


Figure 6. 11 Home Page

Desing Top Header Panel

Add a **Panel** named panelTop

- Dock: Top
- Height: 80px
- BackColor: White

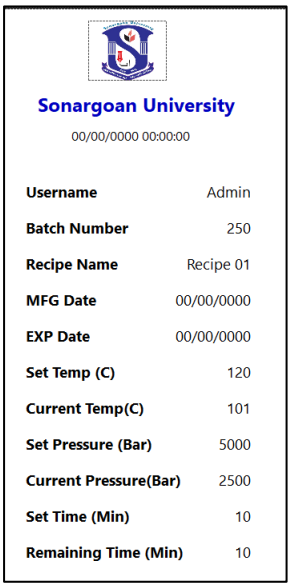
Inside panelTop, add:

- Label lblTitle:
  - Text: STEAM STERILIZER
  - Font: Bold, 20pt, White
  - Alignment: Centered
- PictureBox logo:
  - Left side (add company/university logo)
- LinkLabel linkLogout (Top-right)
  - Text: Logout
  - Name: linkLogout
  - Font: Regular
  - Cursor: Hand
  - Event: linkLogout_LinkClicked





Desing Left Sidebar Panel



- Add Panel named panelLeft
  - Dock: Left
  - Width: 280px
  - BackColor: White
- Inside panelLeft:
  - Label lblUserInfo: Username: Admin
  - Label lblDateTime: auto-updating via Timer
  - Label lblBatchNumber
    - Placeholder: “Batch Number”
  - label lblRecipeName
    - Placeholder: “Recipe Name”
  - label lblMFG, lblEXP
    - Labels: MFG Date / EXP Date
  - Add paired labels
    - Set Temp / Current Temp
    - Set Time / Remaining Time
    - Set Pressure / Current Pressure

Desing Main Area

- Add two AGauge controls (if you don’t have AGauge, I’ll guide you on how to install it).
  - aGaugeTemp, aGaugePressure
  - Style: 0–150 range
  - Labels under each: "Temperature", "Pressure"

Desing AGauge / Speedo meter / Analog input show in Analog meter UI #VisualStudio



Figure 6. 12 Gauge

Github link : `https://github.com/APradyut/CSGaugeMeterForArduino/blob/master/AGauge.cs`

step 1 : Download AGauge.cs & AGauge.Designer.cs from GitHub
Step 2 : Right click of your project name form solution explorer
Step 3 : Add Existing Item > select downloaded AGauge.cs & AGauge.Designer.cs
Step 4 : Add AGauge form toolbox (If not found - Reload the project / Run & close)

Set from properties
Min Value 1000
Max Value 4000
ScaleLinesMajorStepValue → set to 10

Code for Gauge

```
aGauge1= temparature value
```

Desing Right Side Panel

- Add 4 CheckBoxes or custom panels for indicators:
  - chkHeating, chkEmergency, chkWaterLevel, chkMachineStatus
  - Custom styling (red/green indicators optional)

Desing Bottom Buttons

- Add three large Buttons:
  - btnRecipe, btnBatch, btnStart
  - Flat style, color-coded if needed

Desing Right Vertical Panel (Small Icons)

- Add vertical Panel docked to right.
- Add 4–5 small Buttons or PictureBoxes:
  - btnOperator, btnNext, btnReport, btnWarning, btnInfo, btnStop

6.2.1.6 Page 06 (CreateRecipe.cs)

Desing Create Recipe Page

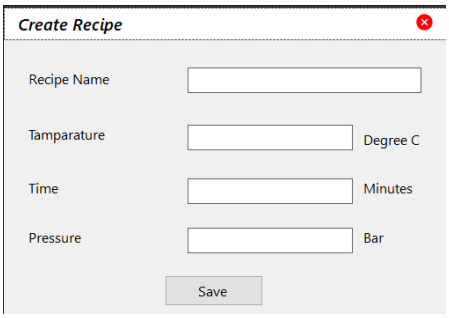


Figure 6. 13 Create Recipe

Create Table for Recipe

```
CREATE TABLE [dbo].[RecipeDB_Table] (
    [Id]           INT            IDENTITY (1, 1) NOT NULL,
    [DateAndTime] DATETIME2 (7) NULL,
    [Recipe_Name] TEXT           NULL,
    [Temp]         REAL           NULL,
    [Time]         REAL           NULL,
    [Pressure]        REAL                        NULL,
    CONSTRAINT [PK_RecipeDB_Table] PRIMARY KEY CLUSTERED ([Id] ASC)
);
```

Button click Event for Save Recipe

```
private void Save_Button_Click(object sender, EventArgs e)
 {
     DateTime DateAndTime = (DateTime.Now);
     string Recipe_Name = Recipe_Name_comboBox.Text;
     double Temp = Convert.ToDouble(Temp_textBox.Text);
     double Time = Convert.ToDouble(Time_textBox.Text);
     double Pressure = Convert.ToDouble(Pressure_textBox.Text);

     using (SqlConnection conn = DatabaseHelper.GetConnection())
     {
         try
         {
```

```
            string query = "INSERT INTO RecipeDB_Table VALUES
(@DateAndTime,@Recipe_Name,@Temp,@Time,@Pressure)";
            SqlCommand cmd = new SqlCommand(query, conn);
            conn.Open();
            cmd = new SqlCommand(query, conn);
            cmd.Parameters.AddWithValue("@DateAndTime", DateAndTime);
            cmd.Parameters.AddWithValue("@Recipe_Name", Recipe_Name);
            cmd.Parameters.AddWithValue("@Temp", Temp);
            cmd.Parameters.AddWithValue("@Time", Time);
            cmd.Parameters.AddWithValue("@Pressure", Pressure);

            int records = cmd.ExecuteNonQuery();
            if (records > 0)
            {
                MessageBox.Show("Recipe Created successfully!", "Success",
MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
        }
        catch (Exception ex)
        { MessageBox.Show(ex.Message)  }

        finally
        {
            conn.Close();
        }
        this.Hide();
    }
 }
```

6.2.1.7 Page 07 (SelectRecipe.cs)

Desing for Select Batch Data

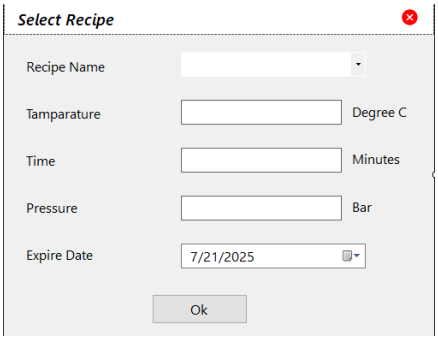


Figure 6. 14 Select Recipe

Create Batch table

```
CREATE TABLE [dbo].[BatchDB_Table] (    [Id]          INT            IDENTITY (1, 1) NOT NULL,
    [Recipe_Name] NVARCHAR (MAX) NULL,
    [Operator]    NVARCHAR (255) NULL,
    [Temp]        REAL           NULL,
    [Time]        REAL           NULL,
    [Pressure]    REAL           NULL,
    [MFG]         DATE           NULL,
    [EXP]         DATE           NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC)
);
```

Functions for Recipe Page

Load Recipe Name

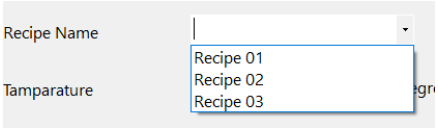


Figure 6. 15 Load Recipe

```
private void LoadRecipes()
 {
     using (SqlConnection conn = DatabaseHelper.GetConnection())
     {
         string query = "SELECT DISTINCT CAST(Recipe_Name AS VARCHAR(MAX)) AS
Recipe_Name\r\nFROM RecipeDB_Table\r\n";

         try
         {
             conn.Open();

             using (SqlCommand cmd = new SqlCommand(query, conn))
             {
                 using (SqlDataReader reader = cmd.ExecuteReader())
                 {
                     HashSet<string> uniqueRecipes = new HashSet<string>();

                     while (reader.Read())
                     {
                         string recipeName = reader["Recipe_Name"].ToString();

                         if (uniqueRecipes.Add(recipeName))
                         {
                             Recipe_Name_comboBox.Items.Add(recipeName);
                         }
                     }
                 }
             }
         }
         catch (Exception ex)
         {
             MessageBox.Show("Error loading Recipe Names: " + ex.Message, "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
         }
         finally
         {
             conn.Close();
         }
     }
 }
```

Load Recipe Details

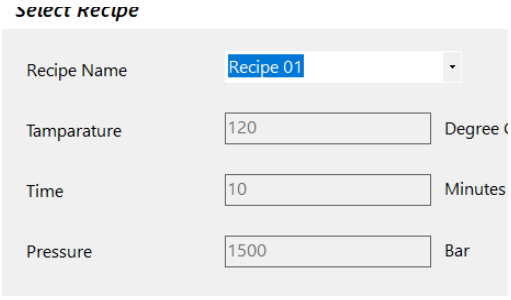


Figure 6. 16 Load Recipe Details

```
private void LoadRecipeDetails(string selectedRecipe)
{
    if (string.IsNullOrEmpty(selectedRecipe))
        return;

    using (SqlConnection conn = DatabaseHelper.GetConnection())
    {
        string query = @"
    SELECT TOP 1 Temp, Time, Pressure
    FROM RecipeDB_Table
    WHERE CAST(Recipe_Name AS VARCHAR(MAX)) = @RecipeName";

        try
        {
            conn.Open();

            using (SqlCommand cmd = new SqlCommand(query, conn))
            {
                cmd.Parameters.AddWithValue("@RecipeName", selectedRecipe);

                using (SqlDataReader reader = cmd.ExecuteReader())
                {
                    if (reader.Read())
                    {
                        Temp_textBox.Text = reader["Temp"].ToString();
                        Time_textBox.Text = reader["Time"].ToString();
                        Pressure_textBox.Text = reader["Pressure"].ToString();
                    }
                    else
                    {
                        // Clear if not found
                        Temp_textBox.Clear();
                        Time_textBox.Clear();
                        Pressure_textBox.Clear();
                        MessageBox.Show("No data found for the selected recipe.", "Info",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                    }
                }
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Error loading recipe details: " + ex.Message, "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
```

```
        finally
        {
            conn.Close();
        }
    }
}
```

Recipe Update Button Click Event

```
private void Recipe_update_button_Click(object sender, EventArgs e)
 {
     string Recipe_Name = Recipe_Name_comboBox.Text;
     string Operator = LoggedInUser.Username;
     double Temp = Convert.ToDouble(Temp_textBox.Text);
     double Time = Convert.ToDouble(Time_textBox.Text);
     double Pressure = Convert.ToDouble(Pressure_textBox.Text);
     DateTime MFG = (DateTime.Now);
     DateTime EXP = Convert.ToDateTime(EXP_dateTimePicker.Text);

     if (string.IsNullOrEmpty(Recipe_Name_comboBox.Text))
     {
         MessageBox.Show("Recipe Name cannot be empty.", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
         return;
     }
     using (SqlConnection conn = DatabaseHelper.GetConnection())
     {
         try
         {
             string query = "UPDATE BatchDB_Table SET Recipe_Name = @Recipe_Name, Operator =
@Operator,Temp = @Temp, Time =@Time, Pressure = @Pressure, MFG = @MFG, EXP = @EXP WHERE
BatchDB_Table.Id = 1";

             conn.Open();
             SqlCommand cmd = new SqlCommand(query, conn);
             cmd = new SqlCommand(query, conn);
             cmd.Parameters.AddWithValue("@Recipe_Name", Recipe_Name);
             cmd.Parameters.AddWithValue("@Operator", Operator);
             cmd.Parameters.AddWithValue("@Temp", Temp);
             cmd.Parameters.AddWithValue("@Time", Time);
             cmd.Parameters.AddWithValue("@Pressure", Pressure);
             cmd.Parameters.AddWithValue("@MFG", MFG);
             cmd.Parameters.AddWithValue("@EXP", EXP);

             int records = cmd.ExecuteNonQuery();
         }
         catch (Exception ex)
         {
             MessageBox.Show(ex.Message);
         }
         finally
         {
             conn.Close();
         }

         this.Hide();}}
```

6.2.1.8 Page 08 (Start.cs)

Desing Start Page

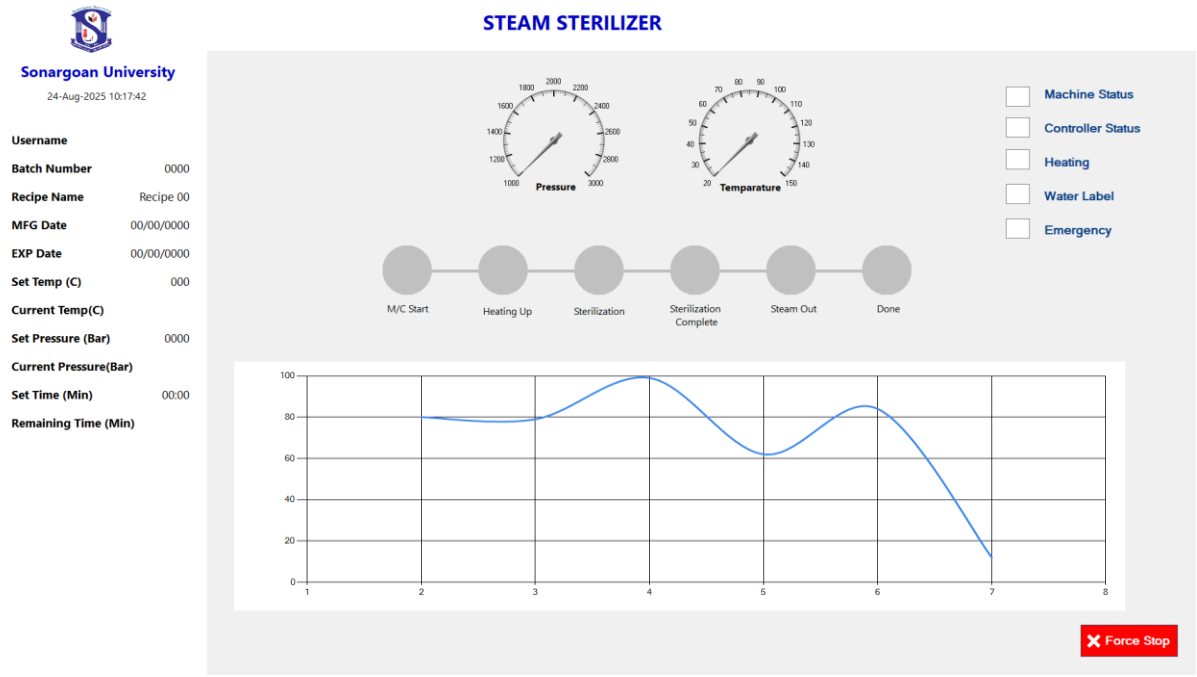


Figure 6. 17 Start page

Create Start Database table

```
CREATE TABLE [dbo].[StartDB_Table] (
    [Id]           INT            IDENTITY (1, 1) NOT NULL,
    [Batch_ID]     NVARCHAR (50)  NULL,
    [Recipe_Name] TEXT           NULL,
    [DateAndTime] DATETIME       NULL,
    [Temperature] FLOAT (53)     NULL,
    [Status]       TEXT           NULL,
    [Pressure]     FLOAT (53)     NULL,
    [Operator]     TEXT           NULL,
    [MFG]          DATE           NULL,
    [EXP]          DATE           NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC)
);
```

Functions for Start Page

Load Recipe info

Recipe Name	Recipe 03
MFG Date	2025-07-21
EXP Date	2025-07-21
Set Temp (C)	130
Current Temp(C)	000
Set Pressure (Bar)	3000
Current Pressure(Bar)	0000
Set Time (Min)	5
Remaining Time (Min)	00:00

Figure 6. 18 Recipe Info

```
private void LoadRecipeinfo()
{
    using (SqlConnection conn = DatabaseHelper.GetConnection())
    {
        string query = @" SELECT TOP 1 Temp, Time, Pressure, Recipe_Name, MFG, EXP FROM
BatchDB_Table ";

        try
        {
            conn.Open();

            using (SqlCommand cmd = new SqlCommand(query, conn))
            {


                using (SqlDataReader reader = cmd.ExecuteReader())
                {
                    if (reader.Read())
                    {
                        lblRecipeName.Text = reader["Recipe_Name"].ToString();
                        lblSetTemp.Text = reader["Temp"].ToString();
                        lblSetTime.Text = reader["Time"].ToString();
                        lblSetPressure.Text = reader["Pressure"].ToString();
                        lblMFG.Text = Convert.ToDateTime(reader["MFG"]).ToString("yyyy-MM-
dd");
                        lblEXP.Text = Convert.ToDateTime(reader["EXP"]).ToString("yyyy-MM-
dd");
                    }
                    else
                    {
                        // Clear if not found
                        //  Temp_textBox.Clear();

                        MessageBox.Show("No data found for the selected recipe.", "Info",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                    }
                }
            }
        }
```

```
        catch (Exception ex)
        {
            MessageBox.Show("Error loading recipe details: " + ex.Message, "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        finally
        {
            conn.Close();
        }
    }

}
```

Load Batch ID

```
private void LoadBatchID()
{
    using (SqlConnection conn = DatabaseHelper.GetConnection())
    {
        try
        {
            string query = "SELECT TOP 1 [Batch_ID] FROM StartDB_Table ORDER BY [Id] DESC";
            conn.Open();
            SqlCommand cmd = new SqlCommand(query, conn);
            cmd = new SqlCommand(query, conn);
            SqlDataReader reader1;
            reader1 = cmd.ExecuteReader();
            if (reader1.Read())
            {
                String space = reader1["Batch_ID"].ToString();
                lblBatchNumber.Text = space;          // Display Batch ID
            }

        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);     // Show error message
        }
        finally
        {
            conn.Close();   // Close connection
        }
    }

}
```

Connect & Receive Date from controller

```
char firstchar;
string Rtemp;                               // Variable for Recived Tempareture
String RTime;                               // Variable for Recived Time
string RXData;                              // Incoming serial data
public delegate void d1(string RXData);     // Delegate for cross-thread serial data updates
public static String Tem_value1;
```

```
void Connect()
 {
     try
     {
         SerialPort1.PortName = "COM15";          // Change to your COM port
         SerialPort1.BaudRate = 9600;
         SerialPort1.DataReceived += SerialPort1_DataReceived_1; // Attach event handler
         SerialPort1.Open();                              // Open serial port
     }
     catch (Exception)
     {

     }
 }

private void SerialPort1_DataReceived_1(object sender,
System.IO.Ports.SerialDataReceivedEventArgs e)
{
    if (SerialPort1.IsOpen)
    {
        string RXData = SerialPort1.ReadLine();         // Read line from serial
        d1 writeit = new d1(SerialDataReceived2Form);   // Delegate to update UI
        Invoke(writeit, RXData);                        // Invoke on UI thread
    }
}

// Handling the Received Data in UI
private void SerialDataReceived2Form(string RXData)
{
    // Prepend the received data with current time and add to the textbox
    textBoxReceive1.Text = (DateTime.Now.ToLongTimeString() + "  " + RXData + "\r\n") +
Environment.NewLine + textBoxReceive1.Text;

    firstchar = RXData[0];  // Extract first character to determine the data type
    using (SqlConnection conn = DatabaseHelper.GetConnection())
    {
        switch (firstchar)
        {
            case 'a':
                // Extract and convert the numeric part from the second character onward
                Single numdata = Convert.ToSingle(RXData.Substring(1));
                int numdata2 = Convert.ToInt16(numdata);

                DateTime DateAndTime = (DateTime.Now);
                try
                {
                    // Insert analog value into Table1
                    string query = "INSERT INTO Table1 VALUES (@DateAndTime,@AnalogIN)";
                    conn.Open();
                    SqlCommand cmd = new SqlCommand(query, conn);
                    cmd = new SqlCommand(query, conn);
                    cmd.Parameters.AddWithValue("@DateAndTime", DateAndTime);
                    cmd.Parameters.AddWithValue("@AnalogIN", numdata);

                    int records = cmd.ExecuteNonQuery();                // Execute insert
                }
```

```
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                }
                finally
                {
                    conn.Close();
                }
                break;

            case 'y':
                IndicatorHeating.BackColor = Color.Green;               // Turn ON heating
indicator
                break;
            case 'Y':
                IndicatorHeating.BackColor = Color.Red;                 // Turn OFF heating
indicator
                break;
            case 'z':
                IndicatorMachineStatus.BackColor = Color.Green;         // Turn ON machine
status indicator
                break;
            case 'Z':
                IndicatorMachineStatus.BackColor = Color.Red;   // Turn OFF machine status
indicator
                break;
            case 'T':
                Tem_value1 = Convert.ToString(RXData.Substring(1));  // Temperature reading
received, store
                lblCurrentTemp.Text = (Tem_value1);                  // Show Temperature Value
                break;

            case 'R':
                try
                {
                    // Prepare SQL query to select Recipe_Name, Temp, and Time from
RecipeDB_Table
                    string query = @"SELECT Recipe_Name, Temp,Time FROM RecipeDB_Table WHERE
CAST(Recipe_Name AS NVARCHAR(MAX)) = @Recipe_Name";

                    conn.Open();                                                // Open the
database connection
                    using (SqlCommand cmd = new SqlCommand(query, conn))        // Create a
SqlCommand
                    {
                        cmd.Parameters.Add("@Recipe_Name", SqlDbType.NVarChar).Value =
lblRecipeName.Text;     // Add the parameter @Recipe_Name

                        // Execute the query and get the result using SqlDataReader
                        using (SqlDataReader reader1 = cmd.ExecuteReader())
                        {
                            if (reader1.Read())                                 // If a record
is found
                            {
                                Rtemp = reader1["Temp"]?.ToString();        // Retrieve the
Temp value and store it in Rtemp
```

```
                                RTime = reader1["Time"]?.ToString();       // Retrieve the
Time value and store it in RTime

                                SerialPort1.Write("T" + Rtemp);  // Send the temperature to
the serial device with a 'T' prefix
                                SerialPort1.Write("C" + RTime);  // Send the time to the
serial device with a 'C' prefix
                            }
                            else
                            {

                            }
                        }
                    }
                }
                catch (Exception ex)
                {
                    MessageBox.Show($"Error: {ex.Message}\nStack Trace: {ex.StackTrace}");
                }
                finally
                {
                    conn.Close();
                }

                break;
            case 'Q':
               // Exitbutton();
                break;
        }
    }
}

private void timer1_Tick(object sender, EventArgs e)
{

    if (SerialPort1.IsOpen)
    {
        Indicator_Controller.BackColor = Color.FromArgb(140, 220, 120);
    }
    if (!SerialPort1.IsOpen)
    {
        Connect();
        Indicator_Controller.BackColor = Color.Red;
    }
}
```

Connect Disconnect Problem solution

```
//Add a Try-Catch inside SerialPort1_DataReceived_1
private void SerialPort1_DataReceived_1(object sender, SerialDataReceivedEventArgs e)
{
    try
    {
        if (SerialPort1.IsOpen)
        {
```

```
            string RXData = SerialPort1.ReadLine();          // Read line from serial
            d1 writeit = new d1(SerialDataReceived2Form);    // Delegate to update UI
            Invoke(writeit, RXData);                         // Invoke on UI thread
        }
    }
    catch (IOException ioex)
    {
        // Log or show a warning message
        Console.WriteLine("Serial Port I/O Exception: " + ioex.Message);
    }
    catch (InvalidOperationException ioex)
    {
        // This occurs if you access SerialPort when it's closed
        Console.WriteLine("Serial Port Invalid Operation: " + ioex.Message);
    }
    catch (Exception ex)
    {
        Console.WriteLine("Unexpected error reading serial data: " + ex.Message);
    }
}

//Properly Close the Serial Port Before Reconnecting
void Connect()
{
    try
    {
        if (SerialPort1.IsOpen)
        {
            SerialPort1.Close();
            SerialPort1.DataReceived -= SerialPort1_DataReceived_1; // Unsubscribe to avoid
multiple handlers
        }

        SerialPort1.PortName = "COM15";          // Update with your actual COM port
        SerialPort1.BaudRate = 9600;
        SerialPort1.DataReceived += SerialPort1_DataReceived_1; // Attach event handler
        SerialPort1.Open();                              // Open serial port
    }
    catch (Exception ex)
    {
        MessageBox.Show("Serial connection failed: " + ex.Message);
    }
}
//Check Connection Before Accessing SerialPort
if (SerialPort1.IsOpen)
{
    try
    {
        Indicator_Controller.BackColor = Color.FromArgb(140, 220, 120);
        TempGauge.Value = float.TryParse(lblCurrentTemp.Text, out float temp) ? temp : 0;
        PressureGauge.Value = float.TryParse(lblCurrentPressure.Text, out float pressure) ?
pressure : 0;
    }
    catch (Exception ex)
    {
        SerialPort1.Close();
        Connect(); // Attempt reconnect
```

```
        Indicator_Controller.BackColor = Color.Red;
        Console.WriteLine("Serial I/O error: " + ex.Message);
    }
}
else
{
    Connect();  // Try reconnecting
    Indicator_Controller.BackColor = Color.Red;
}
```

Send Serial Data by enter

```
private void textBox2_KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Enter)
    {
        if (SerialPort1.IsOpen)                                  // Check if serial port is open
        {
            string serialdatasend = TextBoxSendData.Text;      // Get the text from the TextBox
and send it via serial port
            SerialPort1.Write(serialdatasend);
        }
        TextBoxSendData.Clear();
    }
}
```

Tracker

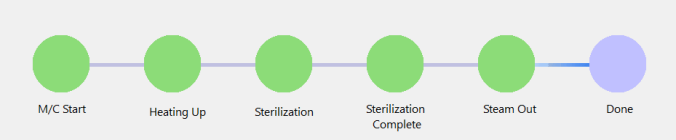


Figure 6. 19 Tracker

Step1: Add 6 buttons (Normal button after coding will circular)
Step2: add 5 panels
Step3: Add 5 picture box
Step4: add image into picturbox (gif)

Receive data form controller-Home.cs

```
//Variables

public static String M_C_Start;
public static String Heating_Up;
public static String Sterilization;
public static String Sterilization_Complete;
public static String Steam_Out;
public static String Done;
public static int slider;
```

```
//After receive data
case 'f':
    M_C_Start = "Remaining";
    break;
case 'F':
    M_C_Start = "Done";
    break;
case 'g':
    Heating_Up = "Remaining";
    slider = 1;
    break;
case 'G':
    Heating_Up = "Done";
    slider = 0 ;
    break;
case 'h':
    Sterilization = "Remaining";
    slider = 2;
    break;
case 'H':
    Sterilization = "Done";
    slider = 0;
    break;
case 'i':
    Sterilization_Complete = "Remaining";
    slider = 3;
    break;
case 'I':
    Sterilization_Complete = "Done";
    slider = 0;
    break;
case 'j':
    Steam_Out = "Remaining";
    slider = 4;
    break;
case 'J':
    Steam_Out = "Done";
    slider = 0;
    break;
case 'k':
    Done = "Remaining";
    slider = 5;
    break;
case 'K':
    Done = "Done";
    slider = 0;
    break;
```

start.cs Load code

Call Update Tracker in timer

```
private void UpdateTracker()
 {
     SetStepColor(btnStep1, Home.M_C_Start);
     SetStepColor(btnStep2, Home.Heating_Up);
     SetStepColor(btnStep3, Home.Sterilization);
     SetStepColor(btnStep4, Home.Sterilization_Complete);
     SetStepColor(btnStep5, Home.Steam_Out);
     SetStepColor(btnStep6, Home.Done);
 }
// Set button color based on status
 private void SetStepColor(Button btn, string status)
 {
     switch (status)
     {
         case "Done":
             btn.BackColor = Color.FromArgb(140,220,120);
             break;
         case "Remaining":
         case "Working":
             btn.BackColor = Color.FromArgb(192,192,255);
             break;
         default:
             btn.BackColor = Color.Silver;
             break;
     }
 }

private void sliderTimer_Tick(object sender, EventArgs e)  // timer interval 1000
 {
     slider();
 }
 private void slider()
 {
     // Create an array of your PictureBoxes
     PictureBox[] pictureBoxes = { pictureBoxStep1, pictureBoxStep2, pictureBoxStep3,
pictureBoxStep4, pictureBoxStep5 };

     // Define all panels in stage order
     Panel[] panels = { panelStep1, panelStep2, panelStep3, panelStep4, panelStep5 };


     // Hide all and make background transparent (optional)
     foreach (var pic in pictureBoxes)
     {
         pic.Visible = false; // ← This hides all previous ones
         pic.BackColor = Color.Transparent;
     }

     // Show the correct one based on slider index
     int index = Home.slider - 1; // Because array is 0-based

     if (index >= 0 && index < pictureBoxes.Length)
```

```
    {
        pictureBoxes[index].Visible = true; // Only show this one
    }
    // Apply light blue color up to the current slider stage
    int stageIndex = Home.slider - 1;

    if (stageIndex >= 0 && stageIndex < panels.Length)
    {
        for (int i = 0; i <= stageIndex; i++)
        {
            panels[i].BackColor = Color.FromArgb(192, 192, 225);  // Light blue
        }
    }
}
```

Count Down Timer

```
int totalSeconds = 0;
int remainingSeconds = 0;
private void PrepareCountdown()
{
    string input = lblSetTime.Text.Trim();

    if (double.TryParse(input, out double setMinutes))
    {
        int minutes = (int)setMinutes;
        int seconds = (int)Math.Round((setMinutes - minutes) * 100); // decimal as seconds

        // Ensure seconds don't exceed 59
        if (seconds >= 60)
        {
            minutes += seconds / 60;
            seconds = seconds % 60;
        }

        totalSeconds = (minutes * 60) + seconds;
        remainingSeconds = totalSeconds;

        TimerCountDown.Start();
    }
    else
    {
        MessageBox.Show("Invalid time format in lblSetTime. Use format like 10.35 for 10 min
35 sec.");
    }
}

private void TimerCountDown_Tick(object sender, EventArgs e)
{
    if (remainingSeconds > 0)
    {
        remainingSeconds--;

        int mins = remainingSeconds / 60;
        int secs = remainingSeconds % 60;

        lblRemainingTime.Text = $"{mins:D2}:{secs:D2}";
```

```
    }
    else
    {
        TimerCountDown.Stop();
        lblRemainingTime.Text = "00:00";
        //  Send message back to serial
        if (SerialPort1.IsOpen)
        {
            SerialPort1.WriteLine("Remaining_Time_Complete");
        }
    }}
```

Searchable Dropdown

Just add those two line code into load data of combo box (when combo box data will load)

```
Recipe_Name_comboBox.AutoCompleteMode = AutoCompleteMode.SuggestAppend;
Recipe_Name_comboBox.AutoCompleteSource = AutoCompleteSource.ListItems;
```

6.2.1.9 Page-09 (Crystal Report)

SelectBatch.cs

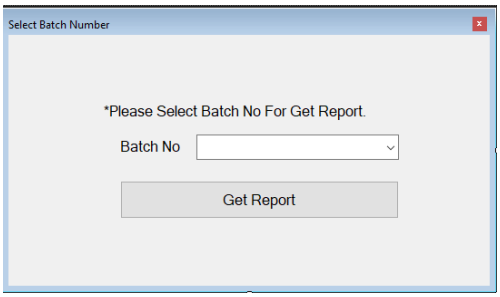


Figure 6. 20 Select Batch

Make select Batch number page

Create a Reusable Searchable ComboBox

1. Add New User Control
In your project:

- Right-click the project > Add > User Control...
- Name it: SearchableComboBox.cs

2. Design (Drag and Drop)
Open the designer of SearchableComboBox:

- Drag a **ComboBox** from toolbox.
- Rename it: comboBox

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Windows.Forms;

namespace Steam_Sterilizer
{
    public partial class SearchableComboBox : UserControl
```

```
    {
        private List<string> originalItems = new List<string>();

        public SearchableComboBox()
        {
            InitializeComponent();

            comboBox.DropDownStyle = ComboBoxStyle.DropDown;
            comboBox.AutoCompleteMode = AutoCompleteMode.None;
            comboBox.TextChanged += ComboBox_TextChanged;
        }

        public ComboBox InnerComboBox => comboBox;

        // Method to load items externally
        public void LoadItems(List<string> items)
        {
            originalItems = items;
            comboBox.Items.Clear();
            comboBox.Items.AddRange(originalItems.ToArray());
        }

        private void ComboBox_TextChanged(object sender, EventArgs e)
        {
            string typedText = comboBox.Text;
            int selStart = comboBox.SelectionStart;

            var filtered = originalItems
                .Where(x => x.IndexOf(typedText, StringComparison.OrdinalIgnoreCase) >= 0)
                .ToList();

            comboBox.Items.Clear();
            comboBox.Items.AddRange(filtered.ToArray());

            comboBox.DroppedDown = true;
            comboBox.Text = typedText;
            comboBox.SelectionStart = selStart;
            Cursor.Current = Cursors.Default;
        }

        // Allow getting selected text/value externally
        public string SelectedText => comboBox.Text;
    }
}
```

4. Use it in a Form

In your SelectReport form:

Add control to designer:

- Build the solution first
- Then open the form designer
- Find SearchableComboBox in toolbox, drag it onto your form
- Rename it to searchBatchCombo

ShowReport.cs

Desing Show Report Page

Load data in your Form code:

```
private void ShowReport_Load(object sender, EventArgs e)
{
    LoadBatchIDs();
}
private void LoadBatchIDs()
{
    using (SqlConnection conn = DatabaseHelper.GetConnection())
    {
        string query = "SELECT DISTINCT Batch_ID FROM StartDB_Table";

        try
        {
            conn.Open();
            using (SqlCommand cmd = new SqlCommand(query, conn))
            {
                using (SqlDataReader reader = cmd.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        cmb_batchID.Items.Add(reader["Batch_ID"].ToString());
                    }
                }
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Error loading Batch IDs: " + ex.Message, "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        finally
        {
            conn.Close();
        }
    }
}
```

Show Report Button click Event

```
private void btn_showReport_Click(object sender, EventArgs e)
 {
     if (string.IsNullOrWhiteSpace(cmb_batchID.Text))
     {
         MessageBox.Show("Please select a Batch ID.", "Validation", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
         return;
     }

     using (SqlConnection conn = DatabaseHelper.GetConnection())
     {
         try
         {
             string sql = @"
```

```
        SELECT
            StartDB_Table.Batch_ID,
            CAST(StartDB_Table.Recipe_Name AS VARCHAR(MAX)) AS Recipe_Name,
            StartDB_Table.DateAndTime,
            StartDB_Table.Temperature,
            StartDB_Table.Pressure,
            StartDB_Table.Status,
            CAST(StartDB_Table.Operator AS VARCHAR(MAX)) AS Operator,
            StartDB_Table.MFG,
            StartDB_Table.EXP,
            MIN(StartDB_Table.DateAndTime) OVER(PARTITION BY StartDB_Table.Batch_ID) AS
StartTime,
            MAX(StartDB_Table.DateAndTime) OVER(PARTITION BY StartDB_Table.Batch_ID) AS
EndTime,
            RecipeDB_Table.Time AS RecipeTime,
            RecipeDB_Table.Temp AS RecipeTemp,
            RecipeDB_Table.Pressure AS RecipePressure
        FROM StartDB_Table
        INNER JOIN RecipeDB_Table
            ON CAST(StartDB_Table.Recipe_Name AS VARCHAR(MAX)) =
CAST(RecipeDB_Table.Recipe_Name AS VARCHAR(MAX))
        WHERE StartDB_Table.Batch_ID = @BatchID
        ORDER BY StartDB_Table.DateAndTime;
    ";

            SqlCommand cmd = new SqlCommand(sql, conn);
            cmd.Parameters.Clear();
            cmd.Parameters.AddWithValue("@BatchID", cmb_batchID.Text);
            SqlDataAdapter adp = new SqlDataAdapter(cmd);
            DataSet ds = new DataSet();
            conn.Open();
            adp.Fill(ds, "BatchReport");

            if (ds.Tables["BatchReport"].Rows.Count == 0)
            {
                MessageBox.Show("No data found for the selected Batch ID.");
                return;
            }

            // Extract values dynamically from first row
            DataRow row = ds.Tables["BatchReport"].Rows[0];
            string recipeName = row["Recipe_Name"].ToString();
            string mfg = Convert.ToDateTime(row["MFG"]).ToString("yyyy-MM-dd");
            string exp = Convert.ToDateTime(row["EXP"]).ToString("yyyy-MM-dd");

            DateTime startTime = Convert.ToDateTime(row["StartTime"]);
            DateTime endTime = Convert.ToDateTime(row["EndTime"]);
            string recipeTime = row["RecipeTime"].ToString();
            string recipeTemp = row["RecipeTemp"].ToString();
            string recipePressure = row["RecipePressure"].ToString();
            string operatorName = row["Operator"].ToString();

            // Prepare and show report
            ReportViewScreen viewScreen = new ReportViewScreen
            {
                ReportName = Path.Combine(Application.StartupPath,
@"Reporting/SteamReport.rpt"),
```

```
                ReportData = ds,
                Batch_ID = cmb_batchID.Text,

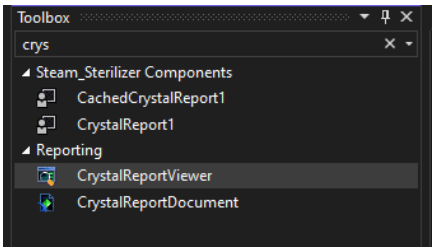
                // Pass dynamic values via public properties
                Recipe_Name = recipeName,
                MFG = mfg,
                EXP = exp,
                StartTime = startTime,
                EndTime = endTime,
                RecipeTime = recipeTime,
                RecipeTemp = recipeTemp,
                RecipePressure = recipePressure,
                OperatorName = operatorName

            };

            viewScreen.ShowDialog();
            this.Close();
        }
        catch (Exception ex)
        {
            MessageBox.Show("Error: " + ex.Message);
        }
    }
}
```

Create a crystalreport.cs

For report viewer



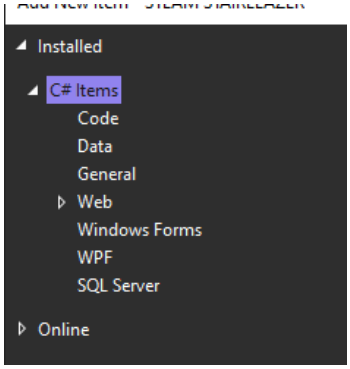
Right click on your project name > Add > Add new item > crystalreport.cs (windows form)
In toolbox search for CryatalReportViwer under reporting section
Drag and drop crystal report viwer
form tool box toolpanel view (None)

Make Report in CrystalReport.rpt

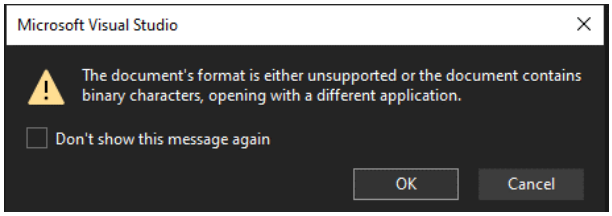
Right click on your project name > Add New item > Report

Download Crystal Report

If report is not present



If you found this problem that mean you don't have download the Crystal Reports for Visual Studio
Then make sure you have downloaded the CR for visual studio SP38 32b Installer vs 2019 or above below have download link



https://origin.softwaredownloads.sap.com/public/site/index.html

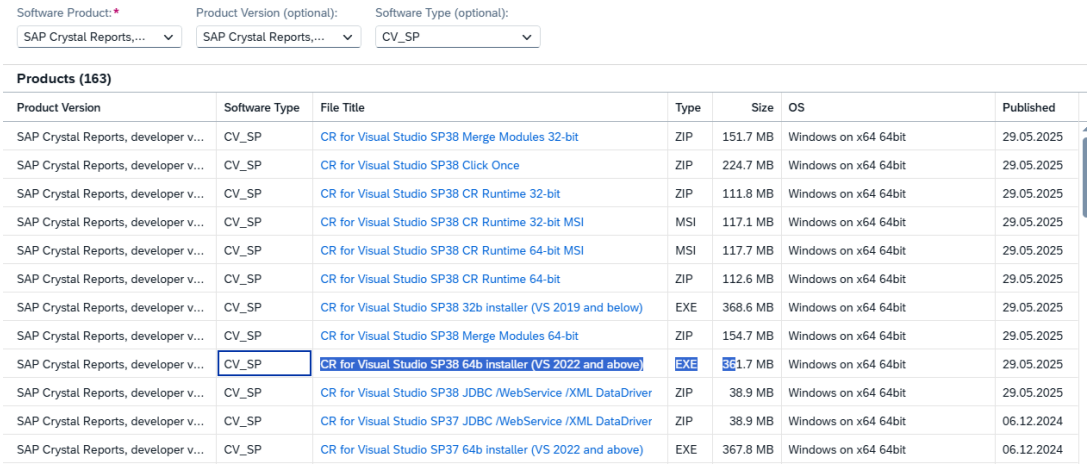
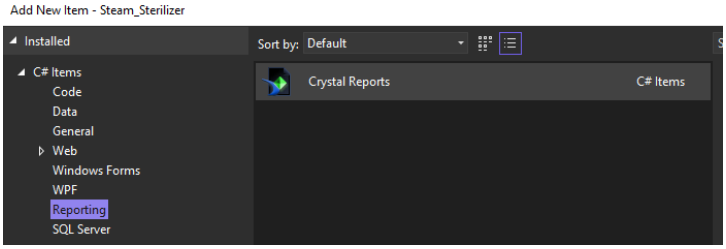


Figure 6. 21 Download Crystal Report

after download and install this, you can find reporting option
under reporting add crystal report for make the report
after add you will see the blank report page



Customize Report

How to add parameter

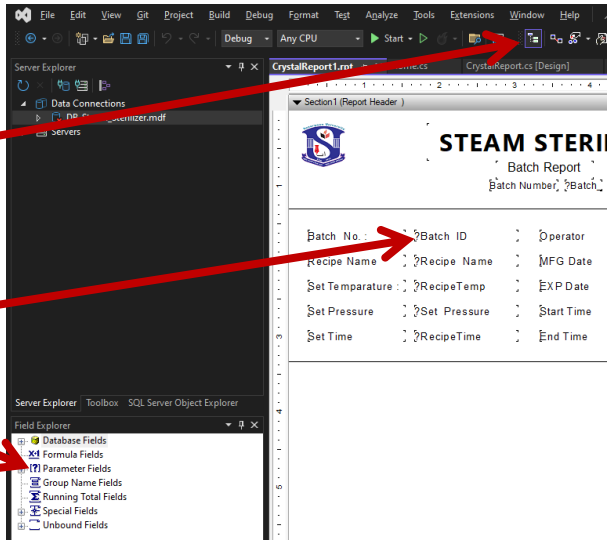
?Batch

Click on **Field Explorer**

Right click on **parameter Fields**

Add new parameter

Drag and drop the parameter on the report



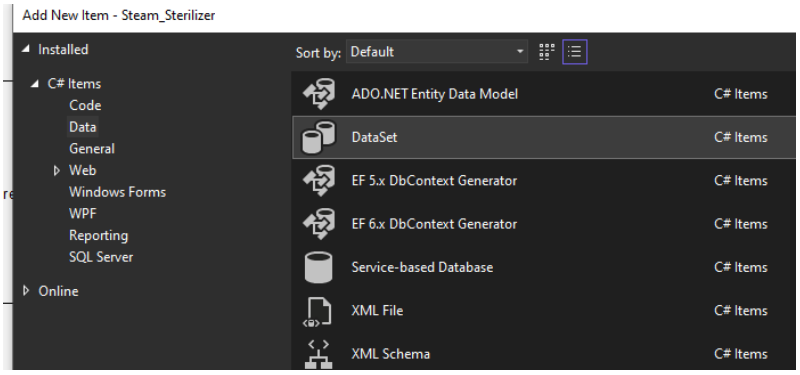
Add Chart

Right click on Report Header Section
Insert > Chart > place the chart in your report

In our case we need to a formula field
For date temp and pressure because our data is insert into databse like string type so we need to convert it into number for chart.

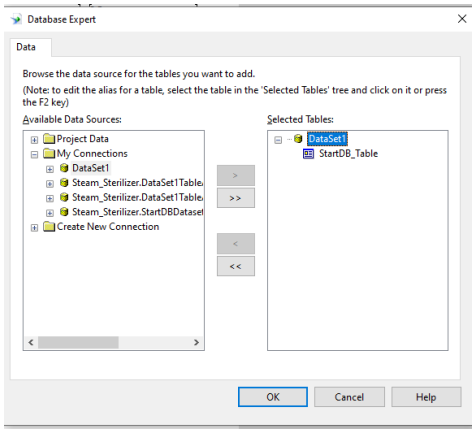
How to connect database with report by Dataset

Create Dataset if not

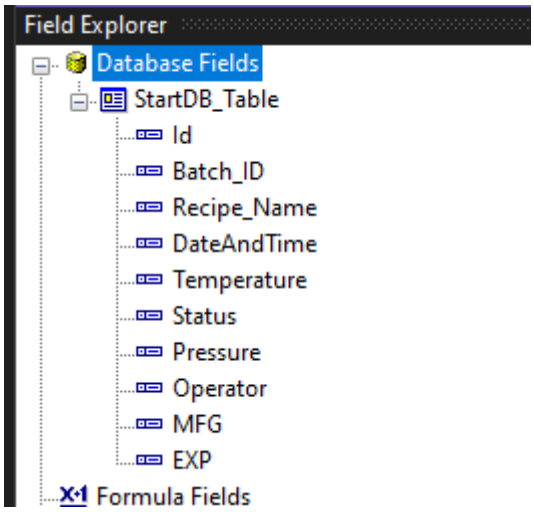


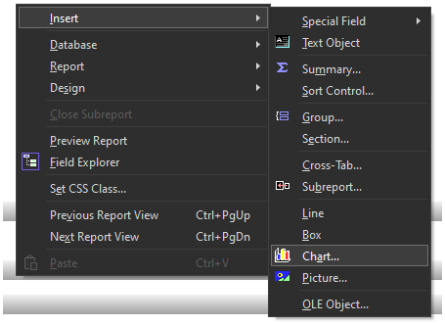
Right click on your project name from solution explorer
Add > Add new item > Data > Dataset

Connect with dataset

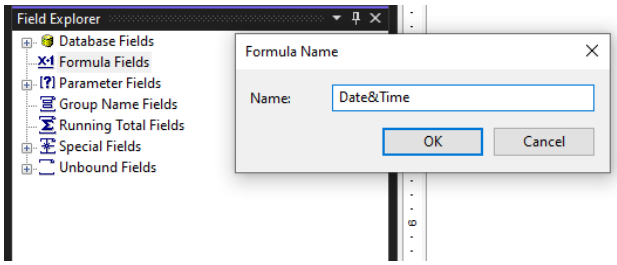


Right click on Database field form Field Explorer in report (crystalreport.rpt)
Database Expert > My connection > Select your dataset name and then click “>>” > ok
Then you can see your database Data table field In Database Fields





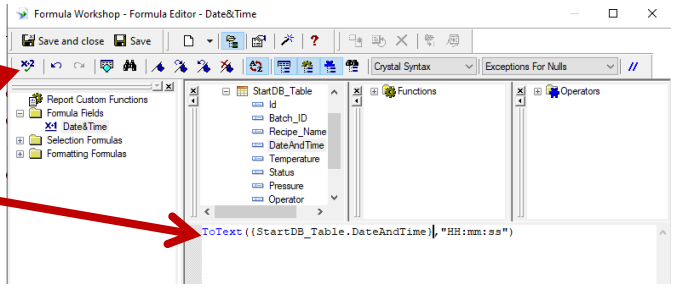
How to create formula field



Right click on Formula field from field Explorer
New> give the name of formula field “Date&Time” > Ok

You will see now this page

Here **Write the formula**
And then **check the formula** if any error in formula it detects.
Click Save and close.



Date&Time Formula

```
ToText({StartDB_Table.DateAndTime},"HH:mm:ss")
```

TemparatureValue Formula

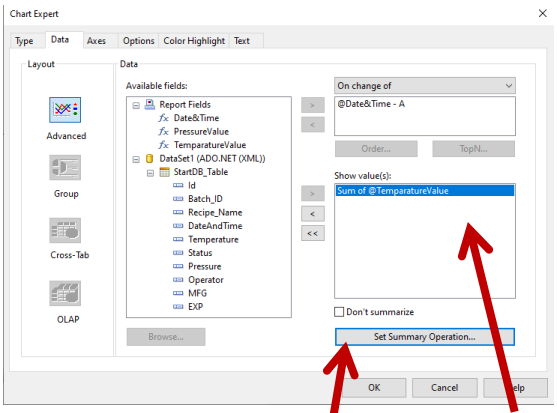
```
ToNumber ({StartDB_Table.Temperature})
```

PressureValue Formula

```
ToNumber ({StartDB_Table.Pressure})
```

Now add chart

Right click on Report Header Section
Insert > Chart > place the chart in your report

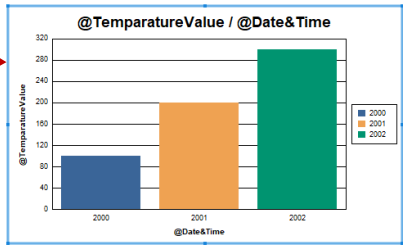


Now you will see a **Chart Expert** page
Add data from your table for chart

Drag and drop Date&Time and TemparatureValue for Date vs temperature chart .

When you drag and drop the temperatureValue in show values you will see it automatically detect **sum of**
But we don't need sum of we need only value show so that's why just **check the box Don't summarize.**

Click ok
You will see the **column chart**



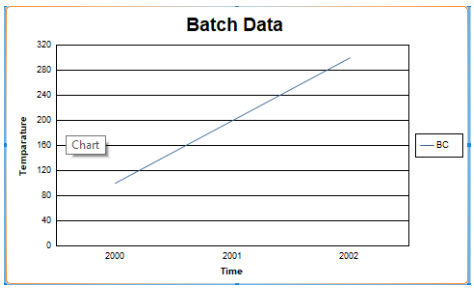
If you want to change it in line chart
Do
Right click in chart > chart Expert > select line > ok

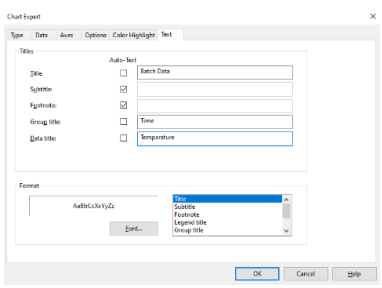
if you change the chart title , subtitle , or other thing

Right click > chart Expert > Text .....

Now chart look like this

If you need more customize your chart then Right click > chart option > here you will find all option to customize your chart.





Create Crystal Report in Visual Studio

Create a new window form for select BatchID “ShowReport.cs”

Add new combobox “cmb_batchID”
Add new button “btn_showReport”

LoadBatchIDs()

```
private void LoadBatchIDs()
{
    using (SqlConnection conn = DatabaseHelper.GetConnection())
    {
        string query = "SELECT DISTINCT Batch_ID FROM StartDB_Table";

        try
        {
            conn.Open();
            using (SqlCommand cmd = new SqlCommand(query, conn))
            {
                using (SqlDataReader reader = cmd.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        cmb_batchID.Items.Add(reader["Batch_ID"].ToString());
                    }
                }
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Error loading Batch IDs: " + ex.Message, "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        finally
        {
            conn.Close();
        }
    }
}
```

Button click event (btn_showReport)

```
private void btn_showReport_Click(object sender, EventArgs e)
 {
     if (string.IsNullOrWhiteSpace(cmb_batchID.Text))
     {
         MessageBox.Show("Please select a Batch ID.", "Validation", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
         return;
     }

     using (SqlConnection conn = DatabaseHelper.GetConnection())
     {
```

```
        try
        {
            string sql = @"
        SELECT
            StartDB_Table.Batch_ID,
            CAST(StartDB_Table.Recipe_Name AS VARCHAR(MAX)) AS Recipe_Name,
            StartDB_Table.DateAndTime,
            StartDB_Table.Temperature,
            CAST(StartDB_Table.Operator AS VARCHAR(MAX)) AS Operator,
            StartDB_Table.MFG,
            StartDB_Table.EXP,
            MIN(StartDB_Table.DateAndTime) OVER(PARTITION BY StartDB_Table.Batch_ID) AS
StartTime,
            MAX(StartDB_Table.DateAndTime) OVER(PARTITION BY StartDB_Table.Batch_ID) AS
EndTime,
            RecipeDB_Table.Time AS RecipeTime,
            RecipeDB_Table.Temp AS RecipeTemp,
            RecipeDB_Table.Pressure AS RecipePressure
        FROM StartDB_Table
        INNER JOIN RecipeDB_Table
            ON CAST(StartDB_Table.Recipe_Name AS VARCHAR(MAX)) =
CAST(RecipeDB_Table.Recipe_Name AS VARCHAR(MAX))
        WHERE StartDB_Table.Batch_ID = @BatchID
        ORDER BY StartDB_Table.DateAndTime;
    ";

            SqlCommand cmd = new SqlCommand(sql, conn);
            cmd.Parameters.Clear();
            cmd.Parameters.AddWithValue("@BatchID", cmb_batchID.Text);
            SqlDataAdapter adp = new SqlDataAdapter(cmd);
            DataSet ds = new DataSet();
            conn.Open();
            adp.Fill(ds, "BatchReport");

            if (ds.Tables["BatchReport"].Rows.Count == 0)
            {
                MessageBox.Show("No data found for the selected Batch ID.");
                return;
            }

            // Extract values dynamically from first row
            DataRow row = ds.Tables["BatchReport"].Rows[0];

            string recipeName = row["Recipe_Name"].ToString();
            string mfg = row["MFG"].ToString();
            string exp = row["EXP"].ToString();
            DateTime startTime = Convert.ToDateTime(row["StartTime"]);
            DateTime endTime = Convert.ToDateTime(row["EndTime"]);
            string recipeTime = row["RecipeTime"].ToString();
            string recipeTemp = row["RecipeTemp"].ToString();
            string recipePressure = row["RecipePressure"].ToString();
            string operatorName = row["Operator"].ToString();

            // Prepare and show report
            ReportViewScreen viewScreen = new ReportViewScreen
            {
```

```
                ReportName = Path.Combine(Application.StartupPath,
@"Reporting/CrystalReport5.rpt"),
                ReportData = ds,
                Batch_ID = cmb_batchID.Text,

                // Pass dynamic values via public properties
                Recipe_Name = recipeName,
                MFG = mfg,
                EXP = exp,
                StartTime = startTime,
                EndTime = endTime,
                RecipeTime = recipeTime,
                RecipeTemp = recipeTemp,
                RecipePressure = recipePressure,
                OperatorName = operatorName

            };

            viewScreen.ShowDialog();
            this.Close();
        }
        catch (Exception ex)
        {
            MessageBox.Show("Error: " + ex.Message);
        }
    }
}
```

Add new windows form for **show report**

ReportViewScreen.cs

Add crystal report viewer form toolbox
From properties (tool panel box = none)

Code for reportviewscreen

```
using CrystalDecisions.CrystalReports.Engine;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Steam_Sterilizer
{
    public partial class ReportViewScreen : Form
    {
        public ReportViewScreen()
        {
            InitializeComponent();
        }
```

```
        public string ReportName { get; set; }
        public DataSet ReportData { get; set; }
        public string Batch_ID { get; set; } // For Batch_ID
        public string Recipe_Name { get; set; }
        public string MFG { get; set; }
        public string EXP { get; set; }
        public DateTime StartTime { get; set; }
        public DateTime EndTime { get; set; }
        public string RecipeTime { get; set; }
        public string RecipeTemp { get; set; }
        public string RecipePressure { get; set; }
        public string OperatorName { get; set; }

        private void ReportViewScreen_Load(object sender, EventArgs e)
        {
            try
            {
                ReportDocument rdd = new ReportDocument();
                rdd.Load(ReportName);
                rdd.SetDataSource(ReportData);

                // Set Crystal Report parameters
                rdd.SetParameterValue("Batch_ID", Batch_ID);
                rdd.SetParameterValue("Recipe_Name", Recipe_Name);
                rdd.SetParameterValue("Username", "UnknownUser");
                rdd.SetParameterValue("MFG", MFG);
                rdd.SetParameterValue("EXP", EXP);
                rdd.SetParameterValue("StartTime", StartTime);
                rdd.SetParameterValue("EndTime", EndTime);
                rdd.SetParameterValue("RecipeTime", RecipeTime);
                rdd.SetParameterValue("RecipeTemp", RecipeTemp);
                rdd.SetParameterValue("Pressure", RecipePressure);
                rdd.SetParameterValue("Operator", OperatorName);

                crystalReportViewer1.ReportSource = rdd;
                crystalReportViewer1.Refresh();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Failed to load report: " + ex.Message);
            }
        }
    }
}
```

Chapter 7: Implementation & Testing

7.1 Implementation of the Project:

Initially, all the components required for the project were collected from multiple sources. Some parts were ordered online, others were purchased from local shops, and a few were custom-designed and fabricated by manufacturers. For each component, datasheets and technical PDFs were carefully reviewed, and relevant research was consulted to understand their functionality and how they interface with the controller. During assembly, some components were damaged at various stages and had to be repaired or reassembled.

To construct the machine, all components were brought to the workshop. The mechanical body was prepared, and each component was carefully attached to the main structure. The PCB was designed and fabricated externally, after which all electrical components were assembled onto the control board. Subsequently, all components were connected to the motherboard. The machine was divided into various sections, such as the main body and control panel, to organize the assembly effectively.

For controlling the machine, a SCADA software system was developed to enable smooth operation, user-friendly interaction, and real-time data recording. This software ensures that the machine can be controlled efficiently, with reliable monitoring and process recording for future analysis. Overall, the project involved integrating mechanical, electrical, and software components to create a fully functional, controlled system.

7.2 Calculation

A major part of this project's control system is the PID (Proportional–Integral–Derivative) controller, which ensures that the heater maintains the desired temperature precisely. For instance, if the set temperature is 121°C, the PID controller continuously adjusts the heater output to keep the temperature stable, avoiding overshoot or fluctuations. This is critical for processes that require accurate and consistent thermal control.

The PID system works by measuring the current temperature and comparing it to the set point. Based on this difference (the error), the controller calculates adjustments to the heater output, ensuring smooth operation and precise temperature regulation. The following code snippet demonstrates the PID implementation for the heater:

```
// PID for heater output
void PID_Section(void) {
  _PID_1_In = current;           // Current temperature input
  _PID_1_SP = Temp_Set_Point;    // Desired temperature set point
  _PID_1.Compute();              // Compute PID output

  _tempVariable_unsignedlong = millis();
  if(_tempVariable_unsignedlong - _PID_1_StartTime > 100) {
    _PID_1_StartTime += 100;
  }

  // Determine relay output based on PID result
```

```
  if(_PID_1_Out > _tempVariable_unsignedlong - _PID_1_StartTime) {
    _PID_1_RelOut = 1;   // Turn heater ON
  } else {
    _PID_1_RelOut = 0;   // Turn heater OFF
  }

  // Activate heater if system is enabled
  if (digitalRead(11) == HIGH) {
    digitalWrite(9, _PID_1_RelOut);
    last__PID_1_RelOut = _PID_1_RelOut;
  } else {
    digitalWrite(9, 0);  // Ensure heater is OFF when disabled
  }
}
```

This PID implementation ensures precise temperature control, maintaining the set-point at 121°C. Similar applications of PID control for temperature regulation have been discussed in the literature, including an IEEE study on optimizing heat supply systems using PID and fuzzy logic, confirming that PID-based systems are widely recognized for maintaining stable and accurate temperatures in heating applications [9].

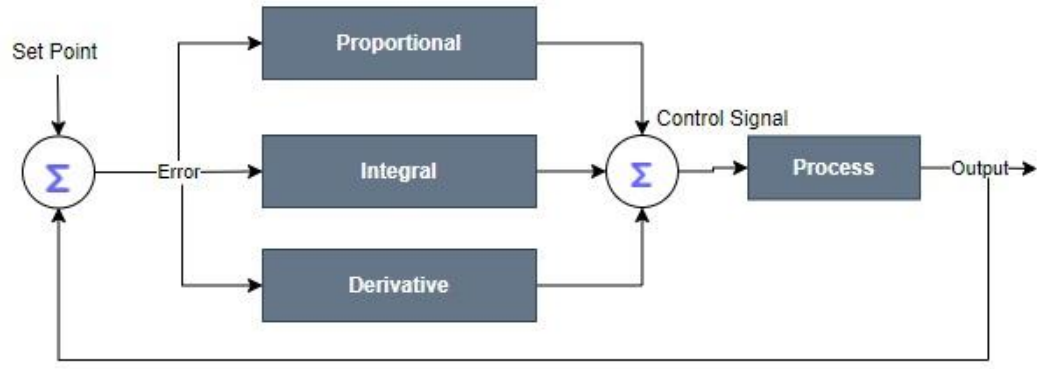


Figure 7. 1 Logic for PID

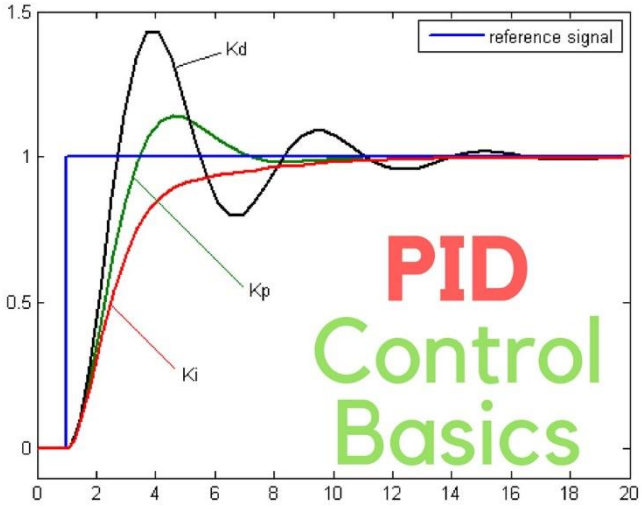


Figure 7. 2 PID controlling Output

7.3 Testing

During testing, the first step was to verify the temperature readings. We checked whether the temperature displayed on the SCADA system matched the readings from the temperature gauge. Similarly, pressure verification was performed to ensure the pressure gauge readings corresponded with SCADA system values.

We then examined whether the heating was stable and whether the system could reach and maintain the set temperature accurately. Potential points of leakage or pressure loss were inspected to ensure system integrity.

The SCADA system control was tested to confirm that: Communication between the motherboard and SCADA is functioning properly, Data is being recorded correctly in the database, Batch-wise reports can be generated accurately, The PID controller maintains the set temperature precisely throughout the process.

For sterilization validation, the Bowie-Dick test pack was used to confirm that the sterilization cycle was effective and that items were free from contamination after the process. The test ensures that the machine achieves proper steam penetration and meets operational safety standards.

Overall, these tests ensured that the system performs reliably, maintains accurate control, records data correctly, and completes sterilization effectively under the set parameters.

7.4 Result

The testing and validation of the system confirmed that the project performed reliably across all parameters.

Temperature Control:

The set temperature of the heater was 120°C. During the testing, the temperature readings varied between a minimum of 118°C and a maximum of 123°C, with an average temperature maintained around 120–121°C. The system took approximately 10 minutes to reach the set temperature from 30°C. Once near the set temperature, the PID controller activated, with the motherboard sending signals to the SSR. The SSR performed on/off switching (blinking) to adjust the heater output, maintaining stable temperature with minimal fluctuations.
During initial testing, some false temperature readings were observed due to the long distance of the temperature sensor from the main system. After reducing the sensor distance, readings stabilized, and the temperature displayed on the SCADA system matched the actual sensor values perfectly.

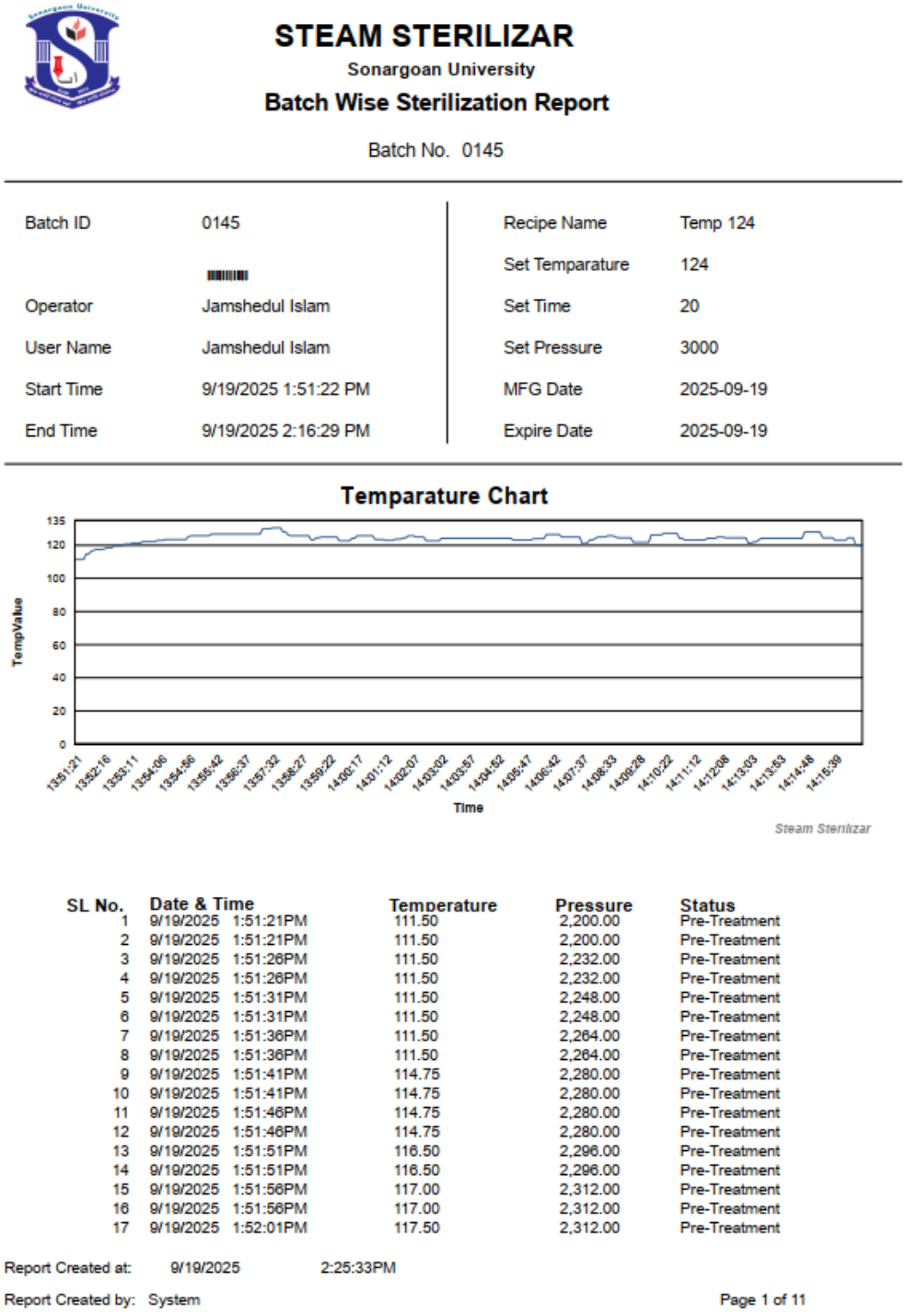
Pressure Control:

Pressure readings were monitored both via the pressure gauge and the SCADA system. For a set pressure of 2 kg/cm², the SCADA system displayed 1960 bar, confirming close agreement

with the physical gauge. Initially, false readings were noted because the pressure sensor was directly attached to the body. This issue was resolved by adding a non-metallic socket pipe between the sensor and the body, ensuring stable and accurate readings.

SCADA System Performance:

Communication between the motherboard and SCADA was verified and fully functional. All process parameters were correctly recorded in the database. Batch-wise reports were generated successfully, demonstrating reliable system logging and traceability.



STEAM STERILIZAR

Sonargoan University

Batch Wise Sterilization Report

Batch No. 0145

Batch ID	0145	Recipe Name	Temp 124
		Set Temparature	124
Operator	Jamshedul Islam	Set Time	20
User Name	Jamshedul Islam	Set Pressure	3000
Start Time	9/19/2025 1:51:22 PM	MFG Date	2025-09-19
End Time	9/19/2025 2:16:29 PM	Expire Date	2025-09-19

Temparature Chart

SL No.	Date & Time	Temperature	Pressure	Status
1	9/19/2025 1:51:21PM	111.50	2,200.00	Pre-Treatment
2	9/19/2025 1:51:21PM	111.50	2,200.00	Pre-Treatment
3	9/19/2025 1:51:26PM	111.50	2,232.00	Pre-Treatment
4	9/19/2025 1:51:26PM	111.50	2,232.00	Pre-Treatment
5	9/19/2025 1:51:31PM	111.50	2,248.00	Pre-Treatment
6	9/19/2025 1:51:31PM	111.50	2,248.00	Pre-Treatment
7	9/19/2025 1:51:36PM	111.50	2,264.00	Pre-Treatment
8	9/19/2025 1:51:36PM	111.50	2,264.00	Pre-Treatment
9	9/19/2025 1:51:41PM	114.75	2,280.00	Pre-Treatment
10	9/19/2025 1:51:41PM	114.75	2,280.00	Pre-Treatment
11	9/19/2025 1:51:46PM	114.75	2,280.00	Pre-Treatment
12	9/19/2025 1:51:46PM	114.75	2,280.00	Pre-Treatment
13	9/19/2025 1:51:51PM	116.50	2,296.00	Pre-Treatment
14	9/19/2025 1:51:51PM	116.50	2,296.00	Pre-Treatment
15	9/19/2025 1:51:56PM	117.00	2,312.00	Pre-Treatment
16	9/19/2025 1:51:56PM	117.00	2,312.00	Pre-Treatment
17	9/19/2025 1:52:01PM	117.50	2,312.00	Pre-Treatment

Report Created at: 9/19/2025 2:25:33PM

Report Created by: System

Page 1 of 11

Figure 7. 3 Batch Report

PID Performance:
The PID controller effectively maintained the temperature near the set-point with minimal overshoot. The system responded dynamically to fluctuations, keeping the heating stable and precise throughout the sterilization cycle.

Sterilization Validation:

We performed sterilization validation using a Bowie-Dick test pack. The principle of this test is that if sterilization is carried out properly, the indicator sheet inside the pack will show a complete colour change from its initial colour (red/light shade) to black.

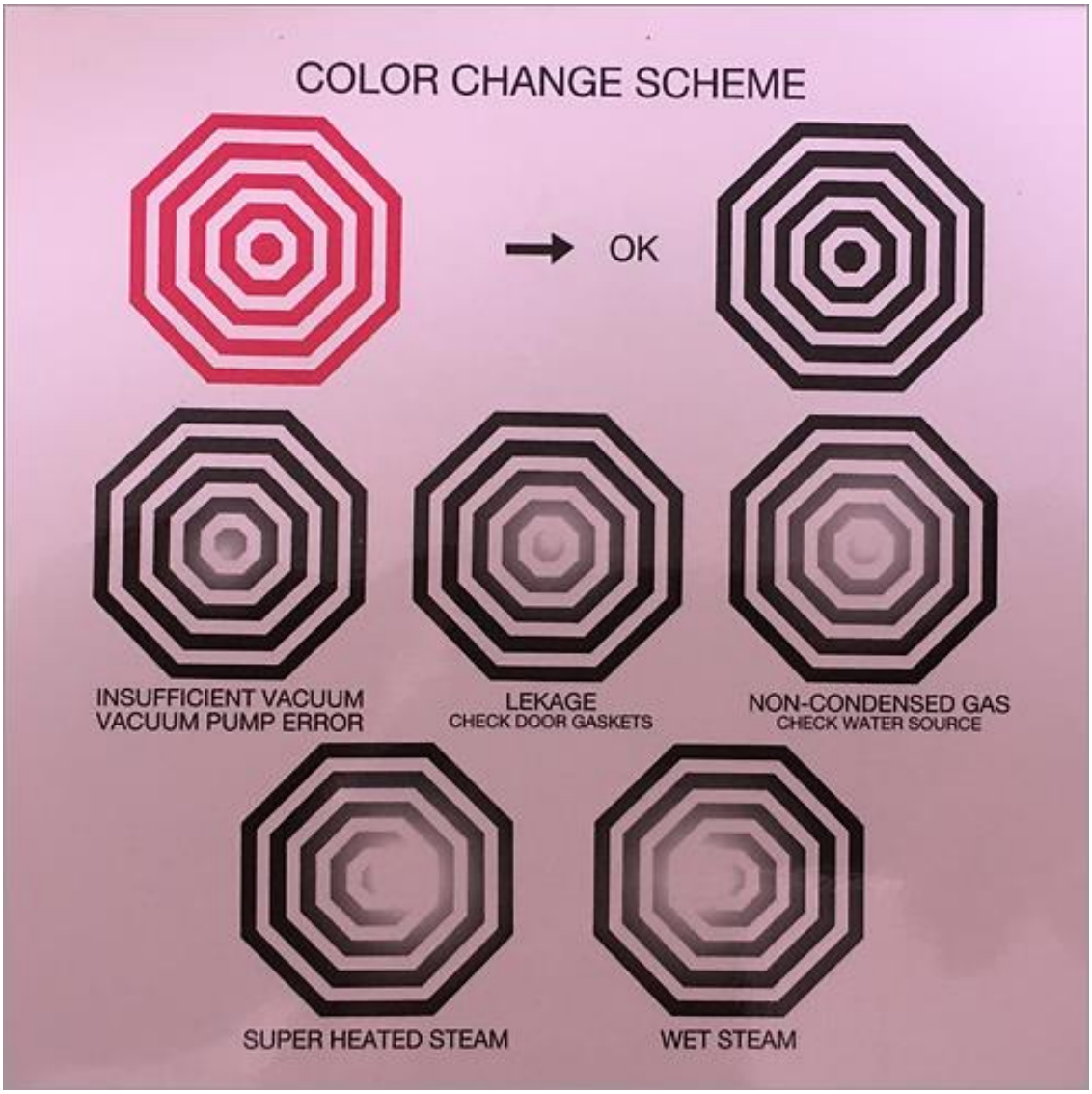


Figure 7. 4 Example of Bowie-Dick Test Pack showing proper and improper colour change results.

For a valid test result, the pack must be exposed under the following conditions:

- 121°C – 125°C for 15–30 minutes, or
- 134°C for 3–5 minutes.

If there are any problems during the sterilization process, such as:

- Insufficient vacuum (vacuum pump error),
- Leakage (defective door gasket),
- Non-condensable gases (improper water source),
- Superheated steam, or
- Wet steam,

then the indicator sheet will not show a uniform colour change. Instead, there will be separate spots or uneven patches, indicating improper steam penetration.

After performing the test, we observed a complete and uniform colour change with no spots or irregularities. This confirms that our sterilization process was effective. The corresponding test image is attached below.



Figure 7. 5 Actual Bowie-Dick Test Pack result showing complete uniform colour change.

Market Analysis:

Market analysis shows that imported automatic steam sterilizers available in Bangladesh are priced between 84,800 – 260,000 BDT, depending on brand and automation level.

In contrast, our locally fabricated prototype costs about only ~30,000 BDT, while incorporating advanced features.

Table 7. 1 Price of Existing Sterilizer

Model / Source	Key Features	Price (BDT)	Reference
HY-260 Digital Autoclave (Digisystem)	Stainless steel, computer-controlled automatic cycles, auto safety (over-temp, low water, pressure)	~135,000	Isa Healthcare BD. *Digisystem Autoclave 24.4L (HY-260)*. Available at: isahealthcare.com.bd (Accessed: Sep 2025)
SS-V50HD Digital Autoclave	Auto shut-off, alarm, safety lock, over-temp/pressure protection	~195,000	Lab Asia BD. *Digital Autoclave Machine Sterilizer SS-V50HD*. Available at: labasiabd.com (Accessed: Sep 2025)
HY-230 Portable Steam Sterilizer	SUS304 stainless steel, automatic controls, compact design	~84,800	HealthAid BD. *Portable Steam Sterilizer Autoclave Machine (HY-230)*. Available at: healthaid.com.bd (Accessed: Sep 2025)
STV-75I (Infitek USA)	Microcomputer automatic control, digital display, automatic exhaust & safety interlocks	~260,000	Lab Equipment BD. *Semi-Automated Autoclave Machine STV-75I*. Available at: labequipment.com.bd (Accessed: Sep 2025)

This demonstrates that our system is approximately 3–8 times more affordable than comparable market models, making it a practical low-cost alternative for small-scale hospitals and laboratories.

Capacity:

The sterilizer machine is designed with a sufficient water reservoir to support proper steam generation during the sterilization process. The total water capacity of the machine is 5 litters. Furthermore, the chamber capacity 8 inches of usable space above the tray, which provides adequate space for accommodating surgical instruments, laboratory glassware, or other materials that require sterilization. This capacity ensures proper steam circulation around the load for effective sterilization.

Overall Observations:

The system successfully integrated temperature and pressure control, PID regulation, SCADA monitoring, and batch reporting. Minor issues with sensor placement were corrected, resulting in accurate readings and reliable operation. The testing confirmed that the system is capable of performing stable and effective sterilization under controlled conditions. Market analysis indicates the machine is low-cost and affordable, and for bulk-size applications, the design can be scaled up while maintaining the same principle. Bowie-Dick test pack confirm valid sterilization.

Chapter 8: Discussion & Conclusion

8.1 Conclusion

This project successfully designed and implemented a steam sterilization system with integrated PID temperature control and SCADA-based monitoring. The system demonstrated stable and reliable performance, maintaining the set temperature of 120°C and pressure of 2 kg/cm² within narrow tolerance limits. The PID controller ensured accurate temperature regulation by dynamically adjusting heater output, while the SCADA interface provided smooth system control, real-time monitoring, and batch-wise data recording.

Testing validated the effectiveness of the system through both temperature–pressure stability checks and the Bowie-Dick test pack, confirming that the sterilization process achieved the required standards of safety and contamination-free results. Initial challenges with false readings from sensors were resolved by optimizing sensor placement and using non-metallic connections, ensuring accurate and reliable data collection.

Overall, the project achieved its objectives by developing a cost-effective, user-friendly, and reliable sterilization system suitable for healthcare applications. The integration of automation and data-driven monitoring enhances safety, traceability, and operational efficiency compared to traditional methods.

8.2 Limitation

1. Manual Loading and Unloading

- The sterilizer chamber must be opened and closed manually by the operator during loading and unloading of instruments.
- Since the chamber becomes extremely hot during operation, this manual handling may expose the operator to burn risks or other safety hazards.
- A fully automated door-locking and unloading mechanism would reduce this risk and improve user safety.

2. Limited Chamber Capacity

- The present design has a relatively small chamber size, which restricts the sterilization load.
- While suitable for small hospitals, laboratories, and clinics, the current capacity is insufficient for large-scale medical facilities or industrial applications that require sterilization of bulk instruments.
- Scaling up the chamber size and heating system would be necessary for wider adoption.

8.3 Future Scope

While the system performed effectively during testing, several improvements and future developments are possible:

1. Enhanced Sensor Accuracy:
   Using higher-grade sensors with built-in noise filtering can further improve the accuracy of temperature and pressure readings.
2. IoT Integration:
   Adding cloud-based monitoring and remote access features would allow real-time data logging and remote supervision of the sterilization process.
3. Mobile Application:
   A mobile app interface could be developed for operators to control and monitor sterilization cycles directly from smartphones or tablets.
4. Energy Optimization:
   Implementing advanced control algorithms such as fuzzy-PID or adaptive controllers could minimize energy consumption and further stabilize the heating process.
5. Scalability:
   The design can be scaled for larger sterilization chambers, making it applicable in large hospitals or centralized sterilization departments.
6. Safety Enhancements:
   Future versions could integrate automatic fault detection and alarm systems to warn operators of leaks, overheating, or incomplete sterilization cycles.

By addressing these future scopes, the system could be expanded into a more intelligent, sustainable, and widely applicable sterilization solution, ensuring higher reliability in healthcare and industrial applications.

References

[1] Mukbarak, M.T., Ozsahin, I.and Ozsahin, D.U., 2019, March. Evaluation of sterilization methods for medical device. *In 2019 Advances in Science and Engineering Technology International Conferences (ASET)* (pp. 1-4). IEEE.

[2] Donskey, C.J., Yowler, M., Falck-Ytter, Y., Kundrapu, S., Salata, R.A. and Rutala, W.A., 2014. A case study of a real-time evaluation of the risk od disease transmission associated with a failure to follow recommended sterilization procedures. *Antimicrobial Resistance and Infection Control,* ***3***(1), p.4.

[3] Zhang, Y., Hu, R., Chen, Y., Liu, X., Wu, J. and Yi, L., 2025. Improving Pressure Steam Sterilization Quality Through Healthcare Failure Mode and Effects Analysis: A Pre-Post Intervention Study in Central Sterile Supply Departments. *Risk Management and Healthcare Policy*, pp.2313-2321.

[4] DiTullio, B.L., Immediate Use Steam Sterilization Optimization And A Methodology to Forecast and Justify Instrumentation.

[5] Vasquez, J., 2015. BAKERSFIELD MEMORIAL HOSPITAL IMMEDIATE USE STEAM STERILIZATION (IUSS) QUALITY IMPROVEMENT PROJECT.

[6] Zhao, X., Liu, T., Ma, N., Wang, R., Li, Y. and Shen, T., 2025. Construction of management tools for early warning of prevacuum steam sterilization failure. *PloS one*, *20*(1), p.e0316322.

[7] Rishworth, H.R., 1938. Efficiency of Steam Sterilizers. *British Medical Journal*, *2*(4053), p.574.

[8] Zhao, L., Bhatia, B., Zhang, L., Strobach, E., Leroy, A., Yadav, M.K., Yang, S., Cooper, T.A., Weinstein, L.A., Modi, A. and Kedare, S.B., 2020. A passive high-temperature high-pressure solar steam generator for medical sterilization. *Joule*, *4*(12), pp.2733-2745.

[9] J. Zhang, H. Li, and Y. Wang, "Research on Optimization of Heat Supply System of Heat Exchanger Station Based on Fuzzy PID Control," 2023 IEEE International Conference on Smart Energy Systems and Technologies (SEST), 2023, pp. 1-5, doi: 10.1109/10603903.

Notes:

i “SDGs” means Sustainable Development Goals. They are 17 global goals set by the United Nations (UN) in 2015, designed to be achieved by 2030. These goals aim to address global challenges like poverty, health, education, environment, and economic growth. Each goal has specific targets and indicators.

ii “Penetrates” means the steam goes inside every small gap or layer of the instrument to kill germs completely.

iii “Denaturing” means changing the natural structure of proteins so they stop working. Microorganisms (like bacteria, viruses, fungi, spores) have proteins that are essential for their survival. When exposed to high heat and steam under pressure, those proteins lose their natural shape (they "denature"). Once the proteins are denatured, the microorganisms can’t function or reproduce, so they die.

iv “PCB” (Printed Circuit Board).

v “Actuators” are devices that perform physical actions when they receive signals from a controller.

vi “SCADA” is the acronym for Supervisory Control and Data Acquisition. SCADA is a computer-based system for gathering and analysing real-time data to monitor and control equipment that deals with critical and time-sensitive materials or events.

vii “GUI” (Graphical user interface).

viii Robust: In software terminology, a robust system or environment is one that is reliable, stable, and capable of performing consistently under various conditions without failure. In this project, Microsoft Visual Studio Community 2022 provides a robust platform for designing and controlling the SCADA interface, ensuring smooth and dependable operation.