Department of Electronics and Telecommunications

University of Moratuwa

Design Document for Soldering Station

EN2160

Engineering Design Realization

210169L- W.W.R.N.S. Fernando

210503H- I.P.D.D. Rajapaksha

Contents

1.0 Introduction

This document serves as a comprehensive guide detailing the design and implementation process of a soldering station, from its conceptualization phase to the development of the final prototype. It offers insights into the various components, functionalities, design principles, testing methodologies, and potential areas for optimization.

2.0 General Overview

The soldering station is envisioned as a versatile and reliable tool that addresses the gaps identified in current market offerings. By integrating advanced features with an ergonomic design, this project seeks to cater to various soldering applications, from intricate electronics manufacturing to simple repair tasks.

3.0 Objectives

The primary objectives of the soldering station design are:

1. **Enhanced Precision and Control**: Provide users with precise temperature control to ensure high-quality soldering results across different applications.
2. **Energy Efficiency**: Incorporate energy-saving features to reduce power consumption without compromising performance.
3. **User-Friendly Design**: Develop an intuitive interface and ergonomic form to enhance user comfort and productivity.
4. **Versatility**: Ensure compatibility with a wide range of soldering tips and accessories to accommodate various tasks.
5. **Reliability and Durability**: Use high-quality materials and robust construction to ensure long-term reliability.
6. **Safety**: Implement safety mechanisms to protect users from potential hazards associated with soldering.

4.0 Preliminary Steps for System Development

4.1 Marketing Department's Need for the System

The marketing department identified a gap in the market for a high-performance soldering station that balances precision, efficiency, and ease of use. This need arises from feedback collected through customer interactions, market research, and competitive analysis. The current market offerings either lack the desired precision for professional use or are too complex for beginners and educational purposes. Thus, there's a strong demand for a versatile soldering station that can cater to both ends of the user spectrum. This product aims to fill that gap, providing a reliable tool

that enhances productivity and user satisfaction in various applications such as electronics manufacturing, repair, and prototyping.

4.2 Detailed Data Sheets of Existing Products

1.Induction Soldering Stations

The induction heating soldering station offers precise temperature control and consistent heating across solder joints, minimizing risks of overheating or cold joints. While it requires specialized training due to its complex operation, it features adjustable power settings for varying soldering needs and is suitable for delicate components like SMDs and integrated circuits. Despite its higher initial cost, the technology's precision and reliability make it a preferred choice in industries such as electronics manufacturing and medical device assembly.

Data sheet: https://www.farnell.com/datasheets/1862062.pdf

2. Lead-Free Soldering Stations

Utilizing lead-free solder alloys composed of tin, silver, and copper significantly reduces environmental impact and produces less toxic fumes, enhancing workplace safety. Although higher soldering temperatures are required and adjustments in soldering techniques are necessary due to different flow characteristics, compliance with environmental regulations such as RoHS is achieved. This approach is ideal for applications where lead contamination is a concern, such as in consumer electronics manufacturing. Additionally, to compensate for the higher melting temperatures, additional flux may be needed to improve wetting.

Data sheet: https://www.farnell.com/datasheets/1719208.pdf

3. Infrared Soldering Stations

Utilizing infrared radiation for non-contact soldering provides precise temperature control and uniform heating distribution, reducing the risk of damage to sensitive components. This method is suitable for soldering heat-sensitive materials like plastics and ceramics, offering fast heating and cooling cycles that enhance soldering efficiency. Despite the higher initial cost due to specialized equipment and technology, infrared radiation soldering is commonly used in industries requiring high-quality soldering with minimal thermal stress on components, such as aerospace and automotive electronics assembly.

Data sheet: https://paceworldwide.com/sites/default/files/2019-07/IR1000_Manual.pdf

4. Hot Air Soldering Stations

Using heated air for soldering provides precise control of temperature and airflow, offering versatility for various applications including SMD soldering, rework, and desoldering. Adjustable temperature and airflow settings accommodate different soldering needs, making it suitable for electronics repair, prototyping, and small-scale production environments. Proper technique is essential to avoid damaging small or delicate components, and periodic calibration and maintenance are necessary to ensure consistent soldering performance.

Data sheet : https://www.farnell.com/datasheets/1724056.pdf

5. Ultrasonic Soldering Stations:

Ultrasonic soldering utilizes ultrasonic vibrations to generate frictional heat, allowing for soldering without direct contact. This method is suitable for joining dissimilar materials with different melting points, such as metals, ceramics, and glass, providing a fast and efficient process with minimal heat-affected zones. While it requires specialized equipment and training, ultrasonic soldering is ideal for applications where conventional methods are impractical, such as with fragile or heat-sensitive materials. Additionally, it offers potential for automation in high-volume manufacturing environments.

Data sheet : https://www.kuroda-techno.com/english/wp-content/themes/kurodatechno/images/product/download/usm-5-performance.pdf

By analyzing these products, the new soldering station aims to integrate the best features of each while addressing their limitations. This includes offering precise temperature control, user-friendly interface, portability, and affordability.

4.3 Write-Ups in Magazines

1. **New York Magazine: The Strategist**

    - **Article**: The Best Soldering Iron Stations
    - **Summary**: This article from New York Magazine's Strategist section offers a comprehensive review of various soldering iron stations available on the market. It discusses the features, pros, and cons of different models, providing valuable insights for professionals. The review highlights key aspects such as temperature control, ergonomics, and energy efficiency, which are critical for users looking to invest in a high-quality soldering station.

2. **DIYODE Magazine**

- **Article**: T2040 Micron Soldering Station Review
- **Summary**: DIYODE Magazine provides an in-depth review of the T2040 Micron Soldering Station. The article delves into the technical specifications, user experience, and performance metrics of the station. It also covers the advanced features and safety mechanisms that make this soldering station stand out in the market. This review is particularly useful for understanding how modern soldering stations incorporate new technologies to improve precision and efficiency.

3. **MagPi Magazine**

- **Article**: Tenma 60W Digital Soldering Station
- **Summary**: MagPi Magazine features a detailed review of the Tenma 60W Digital Soldering Station. This article explores the digital soldering station's capabilities, emphasizing its precise temperature control, ease of use, and suitability for both beginners and experienced users. The review also discusses the station's build quality and value for money, making it a relevant source for understanding user needs and market expectations in soldering technology.

These magazine write-ups offer valuable perspectives on current trends and innovations in the soldering technology space. By referencing these articles, the advanced soldering station project can better align its features with industry standards and user expectations, ensuring a competitive edge in the market.

4.4 User Identification

1. **Soldering Professionals**
- Use Case: High-precision work in electronics manufacturing, device repairs, jewelry-making, and stained-glass work.
- Requirements: Reliable, versatile, and capable of maintaining precise temperatures for different applications.
2. **Students and Beginners**
- Use Case: Learning soldering techniques in educational settings or as a hobby.
- Requirements: Easy to use, affordable, and equipped with safety features to prevent accidents.
3. **Small Business Owners**
- Use Case: Operating repair shops, prototyping services, or custom manufacturing.
- Requirements: Cost-effective, reliable, and versatile enough to handle various tasks.
4. **Professional Engineers and Designers**
- Use Case: Developing and testing electronic devices in industries such as aerospace, automotive, telecommunications, and consumer electronics.
- Requirements: Precise control, advanced features, and compatibility with a range of tools and materials.

4.5 Market Segment

The soldering station targets several market segments:

1. **Electronics Manufacturing:** Companies needing precise and reliable soldering tools for production lines.
2. **Repair Shops:** Businesses focused on repairing electronic devices, where efficiency and reliability are critical.
3. **Educational Institutions:** Schools and colleges teaching electronics and engineering, requiring safe and user-friendly equipment.
4. **Professional Engineering:** Engineers and designers in various industries needing advanced soldering stations for prototyping and testing.

4.6 Design Market Survey of the User Understanding User Needs

To ensure the soldering station design aligns with user requirements, the following steps are implemented:

4.6.1 Stakeholder Engagement

- Conduct interviews and focus groups with potential users to gather detailed insights into their needs, preferences, and pain points.
- Identify key features and functionalities that users expect from a soldering station.

4.6.2 Observation and Feedback Sessions

- Observe users in their actual work environments to understand the context in which the soldering stations are used.
- Collect feedback on existing soldering stations, focusing on areas for improvement and desired enhancements.

4.6.3 Iterative Design Process

- Develop initial prototypes of the soldering station.
- Conduct usability testing with real users, gathering feedback on both functional and ergonomic aspects.
- Refine the design iteratively based on user feedback, ensuring that the final product meets user expectations and requirements.

This comprehensive approach ensures that the soldering station is designed with a user-centric focus, addressing both functional and ergonomic requirements effectively.

4.7 Final Draft of Specification

Precise Temperature Control:

- Requirement: The ability to set and maintain exact temperatures to prevent damage to sensitive components.
- Implementation: Digital temperature control with real-time monitoring and adjustment.

Energy Efficiency:

- Requirement: Minimizing power consumption without compromising performance.
- Implementation: Features like automatic sleep mode and energy-efficient components.

Ease of Use:

- Requirement: Intuitive controls and ergonomic design to reduce user fatigue and enhance productivity.
- Implementation: Simplified user interface, comfortable grip, and clear display.

Versatility:

- Requirement: Compatibility with various soldering tips and accessories to handle different tasks.
- Implementation: Modular design allowing easy replacement and addition of components.

Reliability:

- Requirement: Durable construction and high-quality materials to ensure long-term use.
- Implementation: Robust housing, high-grade materials, and rigorous testing.

Safety:

- Requirement: Protecting users from potential hazards such as burns or electrical shocks.
- Implementation: Overheat protection, insulated handles, and secure power connections.

Compatibility:

- Requirement: Integration with standard power supplies and other equipment.
- Implementation: Universal power input and compatibility with industry-standard accessories.

5.0 Circuit Design, Simulation result, Specification, Selection Criteria, and Final Circuit Diagram

5.1 Power Supply Circuit

In the initial phase of our design methodology, we meticulously analyzed the power requirements of the soldering station, focusing on the specific needs of the soldering iron and hot air gun components. The soldering iron, with its replaceable design and a power consumption of 60W, necessitated a reliable 24V DC power supply. Similarly, the hot air gun also required a 24V DC power source for optimal performance. Additionally, the heating element of the hot air gun demanded a separate 230V AC supply due to its higher voltage requirement, necessitating the isolation of AC and DC circuits on separate PCBs to ensure safety and efficiency.

To address the power needs of the soldering station, we devised a comprehensive power supply circuit. Beginning with the AC-to-DC conversion process, we employed a transformer to downconvert the mains voltage from 230V AC to the required 24V AC output. This transformed AC voltage was then rectified using a bridge rectifier to convert it into a pulsating DC waveform. To stabilize the output voltage and reduce ripple, 24V voltage regulators and capacitors were integrated into the circuit, ensuring a smooth and consistent DC output. Recognizing the requirement for a minimum current output of 3A to power both the soldering iron and hot air gun, we introduced an NPN transistor to boost the current output as necessary, guaranteeing adequate power delivery to the load.

In addition to the 24V DC supply, we extended the power supply circuit to generate a separate 5V DC output required for the microcontroller operation. This involved incorporating additional regulators and capacitors into the circuit design to ensure a stable and regulated 5V output, essential for the reliable operation of the microcontroller responsible for temperature control and user interface functions.

By meticulously designing and implementing this power supply circuit, we ensure the consistent and reliable delivery of power to all components of the soldering station, laying a solid foundation for its overall performance and functionality. With a clear understanding of the circuitry and its components, readers will be equipped to replicate and assemble the power supply circuit for their own soldering station projects.

5.1.1 Power supply circuit Simulation

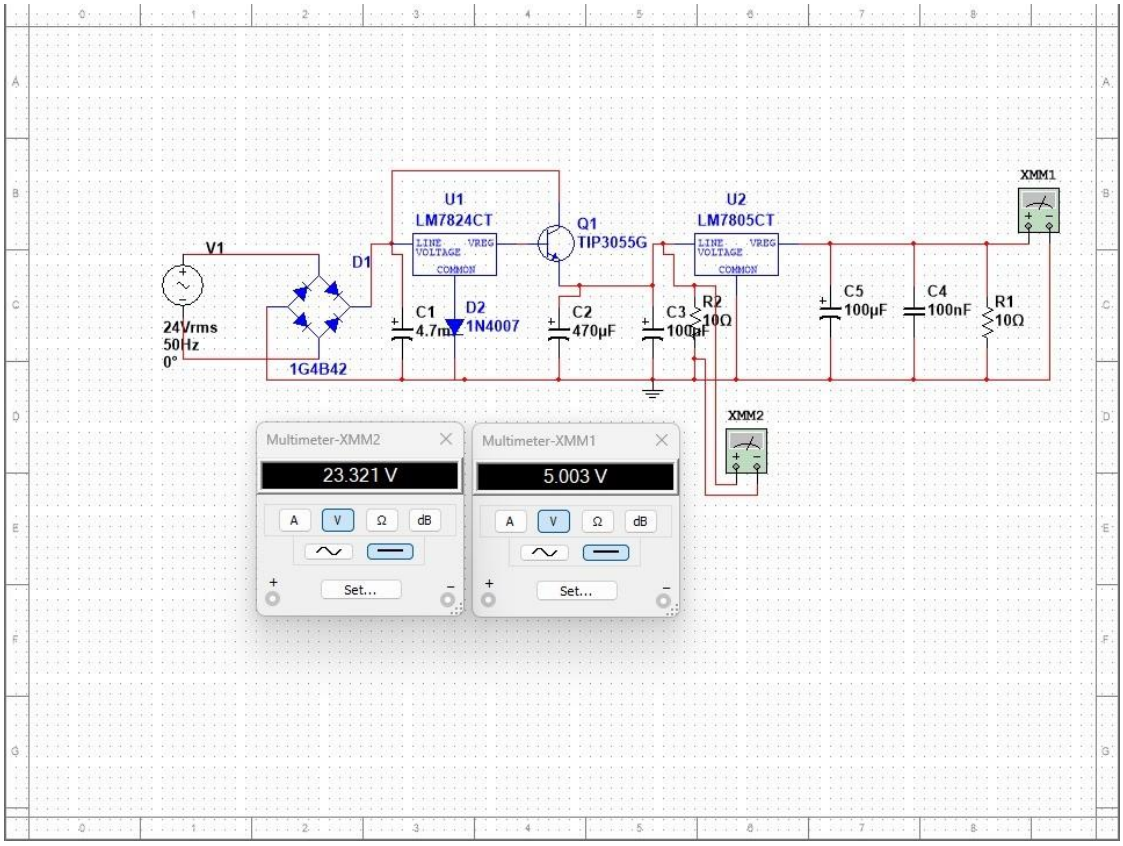


Figure 1-Power Supply circuit simulation

Simulation of the refined circuit showcased its capability to effectively fulfill the project requirements. Additionally, the subsequent dot board implementations validated the feasibility of the design in real-world applications.

5.1.2 Component selection

- Screw terminal
- Bridge ratifier – KBU808
- Diode-1n4007
- Transistor- TIP3055G
- Voltage regulator
  - 24V-LM7824CT
  - 5V-L7805CV

- Capacitors
  - 4.7mF 50V
  - 470uF 50V
  - 100uF 50V ×2

5.1.3 Final Circuit Diagram

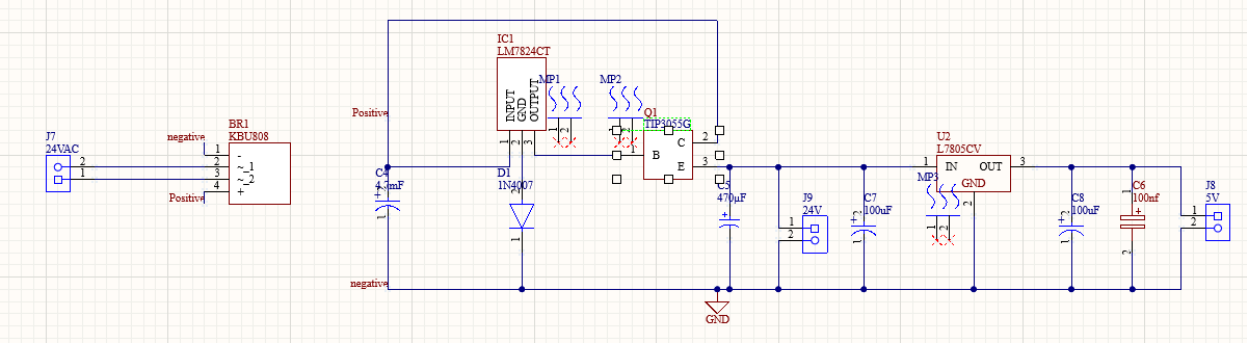


Figure 2-Final circuit diagram

5.2 Microcontroller circuit

For the microcontroller circuit, we employ the ATmega328P-PU chip due to its suitability for industrial applications and its rich array of digital, PWM, and analog pins, fulfilling our project requirements effectively. The ATmega328P-PU is a versatile microcontroller with the following specifications:

5.2.1 Features

Atmega328p-pu product details

The ATmega48A/PA/88A/PA/168A/PA/328/P is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega48A/PA/88A/PA/168A/PA/328/P achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimizepower consumption versus processing speed.

• High Performance, Low Power Atmel® AVR® 8-Bit Microcontroller

• Advanced RISC Architecture

– 131 Powerful Instructions – Most Single Clock Cycle Execution
– 32 x 8 General Purpose Working Registers
– Fully Static Operation
– Up to 20 MIPS Throughput at 20MHz
– On-chip 2-cycle Multiplier

• High Endurance Non-volatile Memory Segments

– 4/8/16/32KBytes of In-System Self-Programmable Flash program memory
– 256/512/512/1KBytes EEPROM
– 512/1K/1K/2KBytes Internal SRAM
– Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
– Data retention: 20 years at 85°C/100 years at 25°C(1)
– Optional Boot Code Section with Independent Lock Bits

• In-System Programming by On-chip Boot Program

• True Read-While-Write Operation

– Programming Lock for Software Security

• Atmel® QTouch ® library support

– Capacitive touch buttons, sliders and wheels
– QTouch and QMatrix® acquisition
– Up to 64 sense channels

– Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
– One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
– Real Time Counter with Separate Oscillator

• Per – Six PWM Channels
– 8-channel 10-bit ADC in TQFP and QFN/MLF package Temperature Measurement
– 6-channel 10-bit ADC in PDIP Package Temperature Measurement
– Programmable Serial USART
– Master/Slave SPI Serial Interface
– Byte-oriented 2-wire Serial Interface (Philips I2C compatible)
– Programmable Watchdog Timer with Separate On-chip Oscillator
– On-chip Analog Comparator
– Interrupt and Wake-up on Pin Change

• Special Microcontroller Features

– Power-on Reset and Programmable Brown-out Detection
– Internal Calibrated Oscillator
– External and Internal Interrupt Sources
– Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby

• I/O and Packages

– 23 Programmable I/O Lines
– 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF

• Operating Voltage:

– 1.8 - 5.5V

• Temperature Range:

– -40°C to 85°C

• Speed Grade:

– 0 - 4MHz@1.8 - 5.5V, 0 - 10MHz@2.7 - 5.5.V, 0 - 20MHz @ 4.5 - 5.5V

• Power Consumption at 1MHz, 1.8V, 25°C

– Active Mode: 0.2mA

– Power-down Mode: 0.1µA
– Power-save Mode: 0.75µA(Including 32kHz RTC)

The ATmega328P-PU is renowned for its reliability, robustness, and widespread industrial usage, making it an ideal choice for our project.

In the circuit design, the ATmega328P-PU chip serves as the central processing unit, controlling various functions of the soldering station. It interfaces with other components, including sensors, displays, and input devices, to execute the desired operations.

To program the ATmega328P-PU chip, we utilize the SPI (Serial Peripheral Interface) protocol, requiring connections for MOSI (Master Out Slave In), MISO (Master In Slave Out), SCK (Serial Clock), 5V, RESET, and GND pins. These connections establish communication between the microcontroller and the programming device, facilitating the uploading of firmware and program code.

Power for the ATmega328P-PU chip is sourced from the power supply circuit, ensuring a stable 5V supply with the required current capacity. This ensures optimal performance and reliable operation of the microcontroller and associated peripherals. By integrating the power supply and microcontroller circuits, we achieve a cohesive and efficient system design for our soldering station project.

Instead of that circuit contain a reset push button, buzzer and stabilize capacitor

5.2.2 Component selection

- AT mega 328p pu
- Resistor –10kohm
- 16Mhz crystal oscillator
- buzzer
- pushbutton
- Capacitor -
- 22pF×2
- 100nF×2

5.2.3 Final circuit diagram

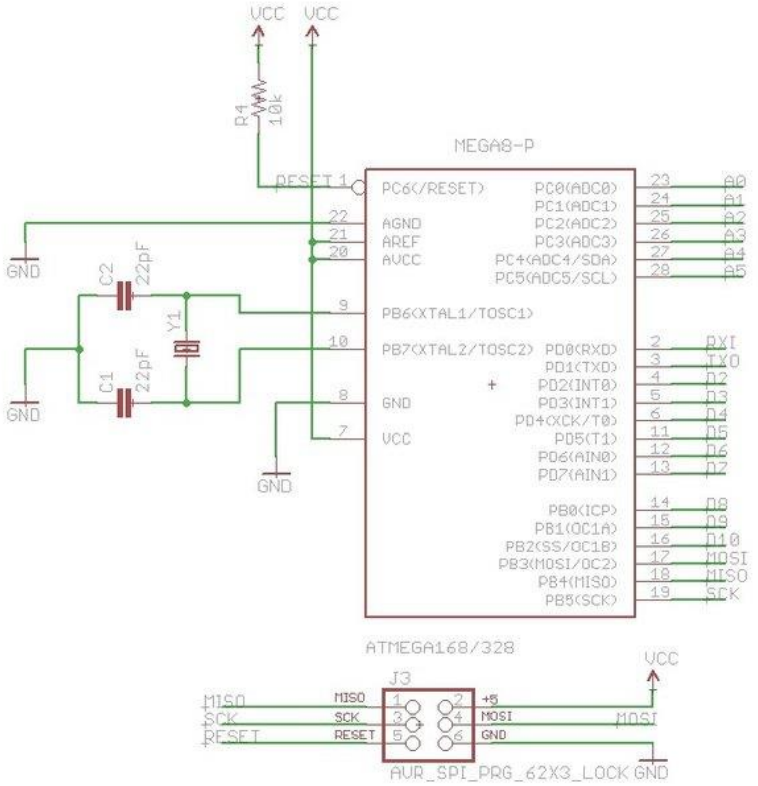


Figure 3-MC circuit

5.4 Soldering Iron and Hot Air gun Fan Current Control Circuits

For the current control circuits of the soldering iron and hot air gun fan, we utilize MOSFETs to regulate the current flow, enabling precise control over their operation. The choice of MOSFETs is critical to ensure efficient performance and reliable operation of the soldering station.

For the soldering iron, we selected the IRF540N MOSFET due to its robustness, high current handling capability, and low ON-resistance. The IRF540N MOSFET offers the following specifications:

5.4.1 Features

MOSFETs

- Maximum Drain-Source Voltage (Vds): 100V

- Continuous Drain Current (Id): 33A

- Low ON-Resistance (Rds(on)): 44mΩ at Vgs = 10V

- Fast Switching Speed

These features make the IRF540N MOSFET an ideal choice for controlling the current flow in the soldering iron circuit. Its high current handling capacity (up to 33A) ensures reliable operation even under heavy load conditions, while its low ON-resistance minimizes power dissipation and heat generation.

Similarly, for the hot air gun fan, we opted for the IRFZ44N MOSFET, complemented by a flyback diode for protection against high voltage spikes. The IRFZ44N MOSFET offers the following specifications.

MOSFETs

- Maximum Drain-Source Voltage (Vds): 55V

- Continuous Drain Current (Id): 49A

- Low ON-Resistance (Rds(on)): 17.5mΩ at Vgs = 10V

- Fast Switching Speed

The IRFZ44N MOSFET's high current handling capacity (up to 49A) and low ON-resistance make it well-suited for controlling the current flow in the hot air gun fan circuit. Additionally, the inclusion of a flyback diode ensures protection against voltage spikes generated during the fan's operation, enhancing overall reliability.

To dissipate heat generated during operation and prevent MOSFET damage, we incorporate heatsinks into the circuit design. Heatsinks efficiently dissipate heat away from the MOSFETs, ensuring optimal performance and longevity. Furthermore, the addition of an enclosure fan helps to mitigate heat buildup within the soldering station enclosure, ensuring safe and reliable operation of the circuitry.

By carefully selecting MOSFETs with appropriate specifications and implementing measures to manage heat dissipation, we ensure the efficient and reliable operation of the soldering iron and hot air gun fan circuits within the soldering station.

Component selection for Iron

- IRF540N
- Heat sink
- Resistor
  - 100ohm
  - 1Kohm

5.4.2 Final Circuit diagram for the Soldering Iron

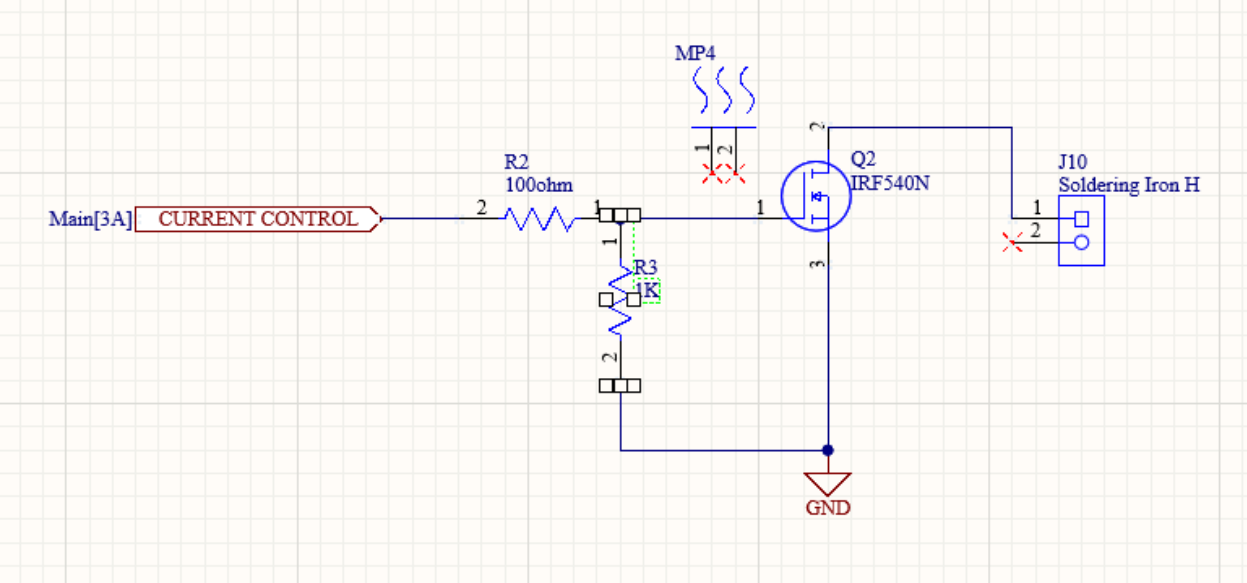


Figure 4-Soldering iron circuit

5.4.3 Component selection for Hot air fan

- Mosfet- IRFZ44N
- Heat sink
- Resistor –1Kohm
- Didoe-UF5408-E3/54

5.4.4Final Circuit diagram for Hot Air Gun

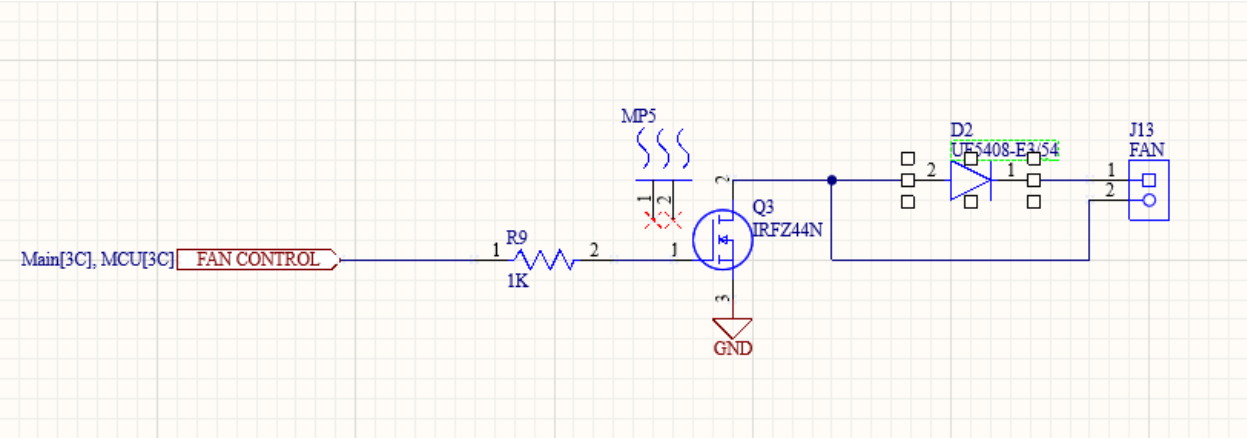


Figure 5- Hot air gun circuit

5.5 Soldering Iron and Hot Air gun Temperature sensing circuit

5.5.1 Soldering Iron Temperature sensing circuit

In our soldering station project, ensuring precise temperature control for the soldering iron is essential to achieve high-quality soldering results consistently. To accomplish this, we implement a sophisticated temperature sensing and amplification system tailored specifically for the soldering iron.

At the heart of our temperature sensing system is a thermocouple, chosen for its reliability and accuracy in measuring temperature variations. The thermocouple generates small voltage signals proportional to changes in temperature, but these signals require amplification to provide meaningful feedback for temperature regulation

Amplifier specifications

For amplification, we employ an inverting amplifier circuit configuration, utilizing the LM358MX operational amplifier (op-amp). The LM358MX was selected after careful consideration of its datasheet specifications, which include a wide supply voltage range of 3V to 26V, low input offset voltage of 2mV, and high common-mode rejection ratio (CMRR) of 100dB. These characteristics make the LM358MX well-suited for our application, ensuring stable and accurate amplification of the thermocouple signals.

By configuring resistor values according to the LM358MX datasheet recommendations, we achieve the desired amplification factor, ensuring precise temperature measurement and control for the soldering iron. With resistor values set at 120kΩ and 1kΩ, we achieve a gain of 120, enabling the system to accurately regulate the soldering iron's temperature within the desired range.

5.5.1.1 Temperature Sensor Circuit Simulation

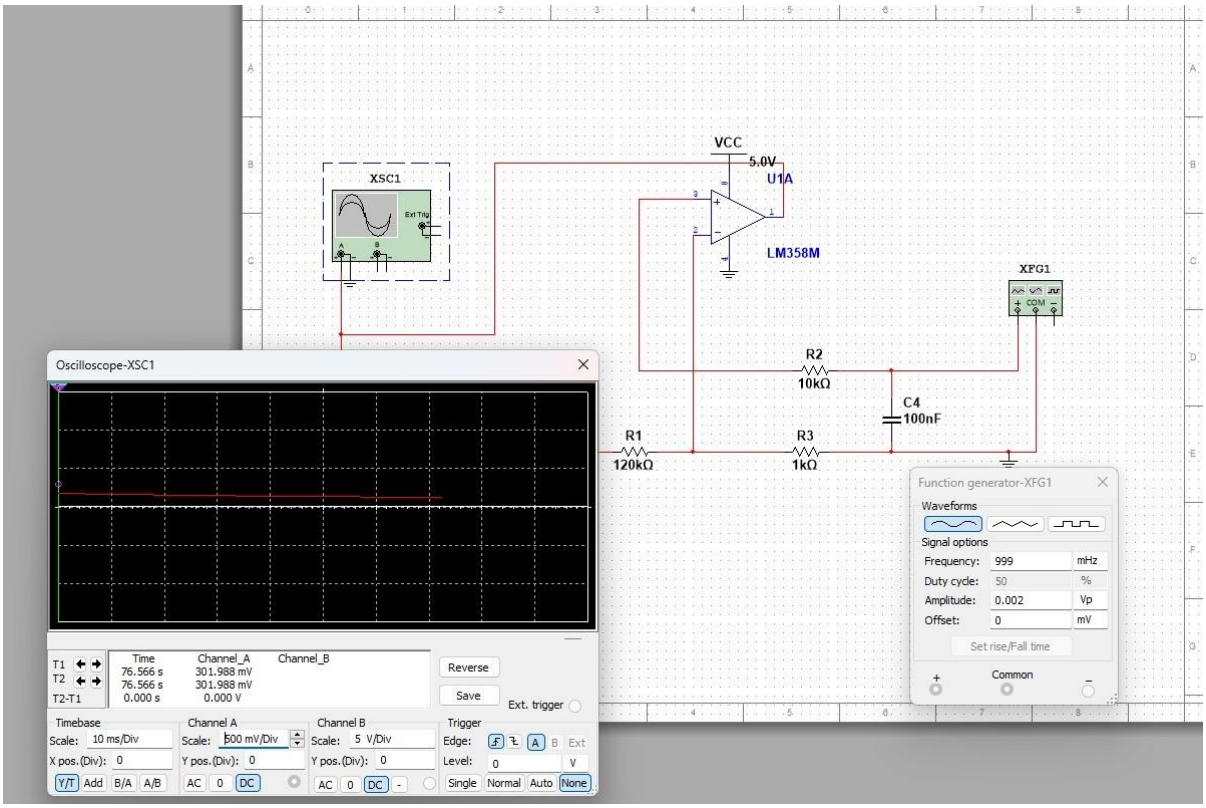
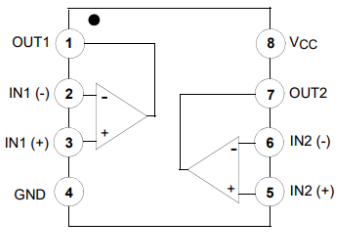


Figure 6-Temper Sensor circuit simulation

5.5.1.2 Features

- Internally Frequency Compensated for Unity Gain
- Large DC Voltage Gain: 100dB
- Wide Power Supply Range:
  LM258/LM258A, LM358/LM358A: 3V~32V (or ±1.5V ~ 16V)
  LM2904 : 3V~26V (or ±1.5V ~ 13V)
- Input Common Mode Voltage Range Includes Ground
- Large Output Voltage Swing: 0V DC to Vcc -1.5V DC
- Power Drain Suitable for Battery Operation.

Internal Block Diagram



Electrical Characteristics

(V_{CC} = 5.0V, V_{EE} = GND, T_A = 25°C, unless otherwise specified)

Parameter	Symbol	Conditions	LM258			LM358			LM2904			Unit
			Min.	Typ.	Max.	Min.	Typ.	Max.	Min.	Typ.	Max.	
Input Offset Voltage	V_{IO}	V_{CM} = 0V to V_{CC} -1.5V $V_{O(P)}$ = 1.4V, R_S = 0Ω	-	2.9	5.0	-	2.9	7.0	-	2.9	7.0	mV
Input Offset Current	I_{IO}	-	-	3	30	-	5	50	-	5	50	nA
Input Bias Current	I_{BIAS}	-	-	45	150	-	45	250	-	45	250	nA
Input Voltage Range	$V_{I(R)}$	V_{CC} = 30V (LM2904, V_{CC}=26V)	0	-	V_{CC} -1.5	0	-	V_{CC} -1.5	0	-	V_{CC} -1.5	V
Supply Current	I_{CC}	R_L = ∞, V_{CC} = 30V (LM2904, V_{CC}=26V)	-	0.8	2.0	-	0.8	2.0	-	0.8	2.0	mA
		R_L = ∞, V_{CC} = 5V	-	0.5	1.2	-	0.5	1.2	-	0.5	1.2	mA
Large Signal Voltage Gain	G_V	V_{CC} = 15V, R_L= 2kΩ $V_{O(P)}$ = 1V to 11V	50	100	-	25	100	-	25	100	-	V/mV
Output Voltage Swing	$V_{O(H)}$	V_{CC}=30V (V_{CC} =26V for LM2904) R_L = 2kΩ	26	-	-	26	-	-	22	-	-	V
		R_L= 10kΩ	27	28	-	27	28	-	23	24	-	V
	$V_{O(L)}$	V_{CC} = 5V, R_L= 10kΩ	-	5	20	-	5	20		5	20	mV
Common-Mode Rejection Ratio	CMRR	-	70	85	-	65	80	-	50	80	-	dB
Power Supply Rejection Ratio	PSRR	-	65	100	-	65	100	-	50	100	-	dB
Channel Separation	CS	f = 1kHz to 20kHz (Note1)	-	120	-	-	120	-	-	120	-	dB
Short Circuit to GND	I_{SC}	-	-	40	60	-	40	60	-	40	60	mA
Output Current	I_{SOURCE}	$V_{I(+)}$ = 1V, $V_{I(-)}$ = 0V, V_{CC} = 15V, $V_{O(P)}$ = 2V	20	30	-	20	30	-	20	30	-	mA
	I_{SINK}	$V_{I(+)}$ = 0V, $V_{I(-)}$ = 1V, V_{CC} = 15V, $V_{O(P)}$ = 2V	10	15	-	10	15	-	10	15	-	mA
		$V_{I(+)}$ = 0V,$V_{I(-)}$ =1V , V_{CC} = 15V, $V_{O(P)}$ = 200mV	12	100	-	12	100	-	-	-	-	µA
Differential Input Voltage	$V_{I(DIFF)}$	-	-	-	V_{CC}	-	-	V_{CC}	-	-	V_{CC}	V

5.5.1.3 Component selection for Soldering Iron Temperature sensor

- Opamp- LM358MX
- Capacitor -100nF
- Resisters -
  - 1kohm
  - 10kohm
  - 120kohm

5.5.1.4 Final Circuit diagram for Soldering Iron Temperature sensor

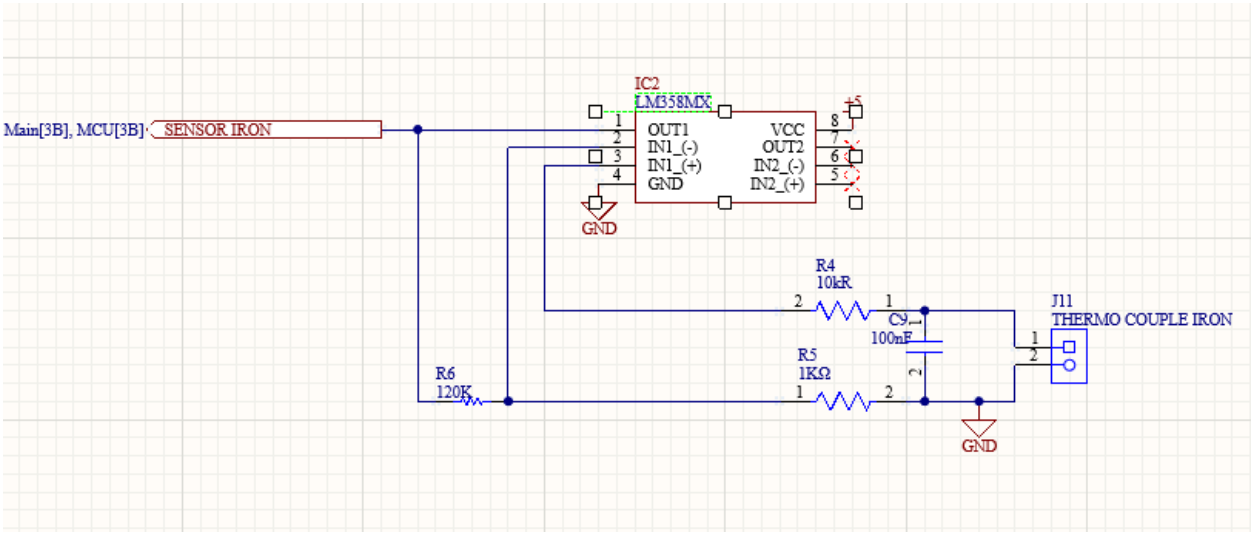


Figure 7-Iron temperature sensor

5.5.2 Hot Air Gun Temperature Sensing and Amplification

In addition to the soldering iron, the hot air gun integrated into our soldering station also requires precise temperature control for optimal performance. However, temperature sensing for the hot air gun presents unique challenges, particularly due to noise interference that can affect temperature readings.

To address these challenges, we incorporate a low-pass filter into the temperature sensing system before the amplification stage. This filter effectively attenuates high-frequency noise, ensuring that only the relevant temperature signals are amplified for accurate temperature measurement.

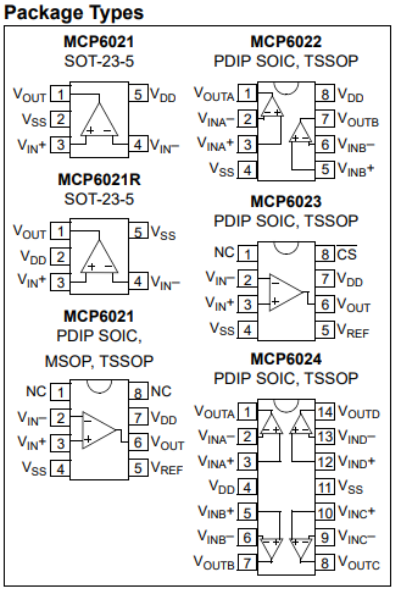
For amplification of the filtered temperature signals, we utilize the MCP6022-I_ST operational amplifier (op-amp). This op-amp was chosen for its exceptional characteristics, including a low input bias current of 1pA, low noise of 8nV/√Hz, and wide bandwidth of 10MHz, as outlined in its datasheet. These features make the MCP6022-I_ST well-suited for our application, allowing for precise amplification of the hot air gun temperature signals while minimizing noise interference.

By carefully referencing the MCP6022-I_ST datasheet and configuring the amplification circuit accordingly, we ensure accurate temperature measurement and control for the hot air gun, enhancing the overall performance and functionality of our soldering station.

5.5.2.1Features

- Rail-to-Rail Input/Output
- Wide Bandwidth: 10 MHz (typical)
- Low Noise: 8.7 nV/√Hz, at 10 kHz (typical)
- Low Offset Voltage:
  - Industrial Temperature: ±500 µV (maximum)
  - Extended Temperature: ±250 µV (maximum)
- Mid-Supply V_{REF}: MCP6021 and MCP6023
- Low Supply Current: 1 mA (typical)
- Total Harmonic Distortion:
  - 0.00053% (typical, G = 1 V/V)
- Unity Gain Stable
- Power Supply Range: 2.5V to 5.5V
- Temperature Range:
  - Industrial: -40°C to +85°C
  - Extended: -40°C to +125°C

Package Types



DC ELECTRICAL CHARACTERISTICS

Electrical Specifications: Unless otherwise indicated, T_A = +25°C, V_{DD} = +2.5V to +5.5V, V_{SS} = GND, V_{CM} = V_{DD}/2, $V_{OUT} \approx V_{DD}/2$ and R_L = 10 kΩ to $V_{DD}/2$.

Parameters	Sym	Min	Typ	Max	Units	Conditions
Input Offset						
Input Offset Voltage:						
Industrial Temperature Parts	V_{OS}	-500	—	+500	µV	V_{CM} = 0V
Extended Temperature Parts	V_{OS}	-250	—	+250	µV	V_{CM} = 0V, V_{DD} = 5.0V
Extended Temperature Parts	V_{OS}	-2.5	—	+2.5	mV	V_{CM} = 0V, V_{DD} = 5.0V T_A = -40°C to +125°C
Input Offset Voltage Temperature Drift	$\Delta V_{OS}/\Delta T_A$	—	±3.5	—	µV/°C	T_A = -40°C to +125°C
Power Supply Rejection Ratio	PSRR	74	90	—	dB	V_{CM} = 0V
Input Current and Impedance						
Input Bias Current	I_B	—	1	—	pA	
Industrial Temperature Parts	I_B	—	30	150	pA	T_A = +85°C
Extended Temperature Parts	I_B	—	640	5,000	pA	T_A = +125°C
Input Offset Current	I_{OS}	—	±1	—	pA	
Common-Mode Input Impedance	Z_{CM}	—	10^{13}||6	—	Ω||pF	
Differential Input Impedance	Z_{DIFF}	—	10^{13}||3	—	Ω||pF	
Common-Mode						
Common-Mode Input Range	V_{CMR}	V_{SS}-0.3	—	V_{DD}+0.3	V	
Common-Mode Rejection Ratio	CMRR	74	90	—	dB	V_{DD} = 5V, V_{CM} = -0.3V to 5.3V
	CMRR	70	85	—	dB	V_{DD} = 5V, V_{CM} = 3.0V to 5.3V
	CMRR	74	90	—	dB	V_{DD} = 5V, V_{CM} = -0.3V to 3.0V
Voltage Reference (MCP6021 and MCP6023 only)						
V_{REF} Accuracy (V_{REF} – V_{DD}/2)	V_{REF_ACC}	-50	—	+50	mV	
V_{REF} Temperature Drift	$\Delta V_{REF}/\Delta T_A$	—	±100	—	µV/°C	T_A = -40°C to +125°C
Open-Loop Gain						
DC Open-Loop Gain (Large Signal)	A_{OL}	90	110	—	dB	V_{CM} = 0V, V_{OUT} = V_{SS}+0.3V to V_{DD}-0.3V
Output						
Maximum Output Voltage Swing	V_{OL}, V_{OH}	V_{SS}+15	—	V_{DD}-20	mV	0.5V input overdrive
Output Short Circuit Current	I_{SC}	—	±30	—	mA	V_{DD} = 2.5V
	I_{SC}	—	±22	—	mA	V_{DD} = 5.5V

5.5.2.2 Component selection for Hot Air Gun Temperature sensor

- Operational amplifier - LM358MX
- Capacitor -100nF
- Resisters -
  - 1kohm
  - 10kohm
  - 120kohm

5.5.2.3 Final Circuit diagram for Hot Air Gun Temperature sensor

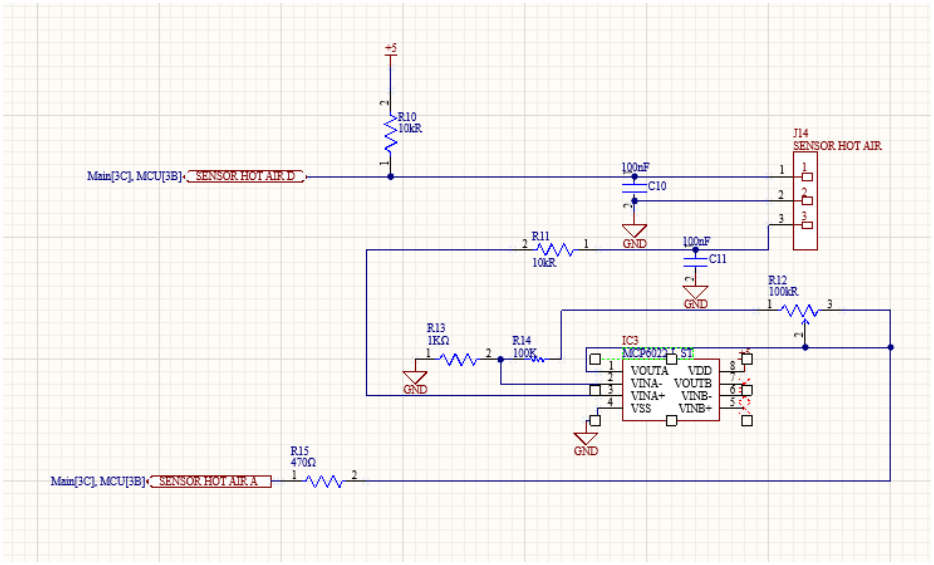


Figure 8-Hot air gun temperature sensor

5.4 AC PCB connecting circuit

The MCQ sub-circuit consists solely of resistors and serves the critical function of controlling the AC circuit. Its simplicity belies its importance, as it enables the connection of the AC circuit via a minimalistic 3-pin connector

5.4.1 Component selection

- Resister
  - 220ohm
  - 10kohm

5.4.2 Final circuit diagram

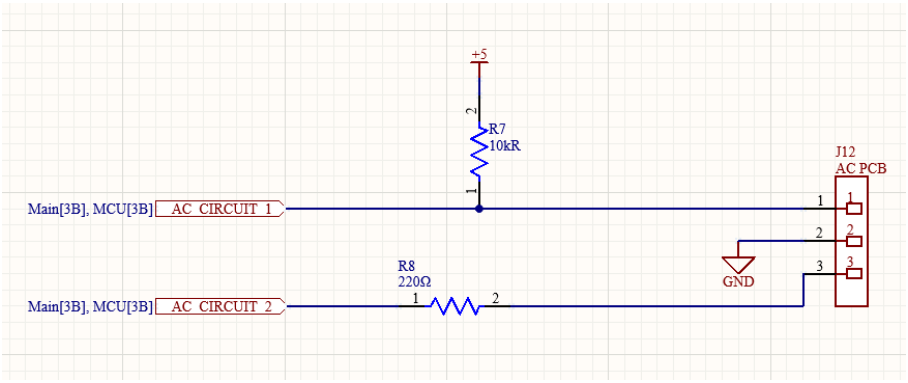


Figure 9-ac circuit

5.6 AC PCB

The PCB design for the AC supply in the soldering station serves a crucial role in safely managing the high-voltage AC current required for the hot air gun's heating element. The PCB consists of two main circuits: a zero-crossing detector using an AC optocoupler and an opto-isolated triac triggering circuit.

Zero-crossing detector

The zero-crossing detector circuit is vital for synchronizing the operation of the soldering station with the AC power supply. By detecting the precise moment when the AC voltage waveform crosses zero volts, this circuit ensures accurate timing for triggering the triac switch, minimizing electrical noise and ensuring efficient power control.

Optocoupler

To isolate the AC part from the rest of the circuitry and prevent potential damage, an AC optocoupler is employed. This optocoupler effectively separates the high-voltage AC portion of the circuit from the low-voltage components, providing a safe interface between them.

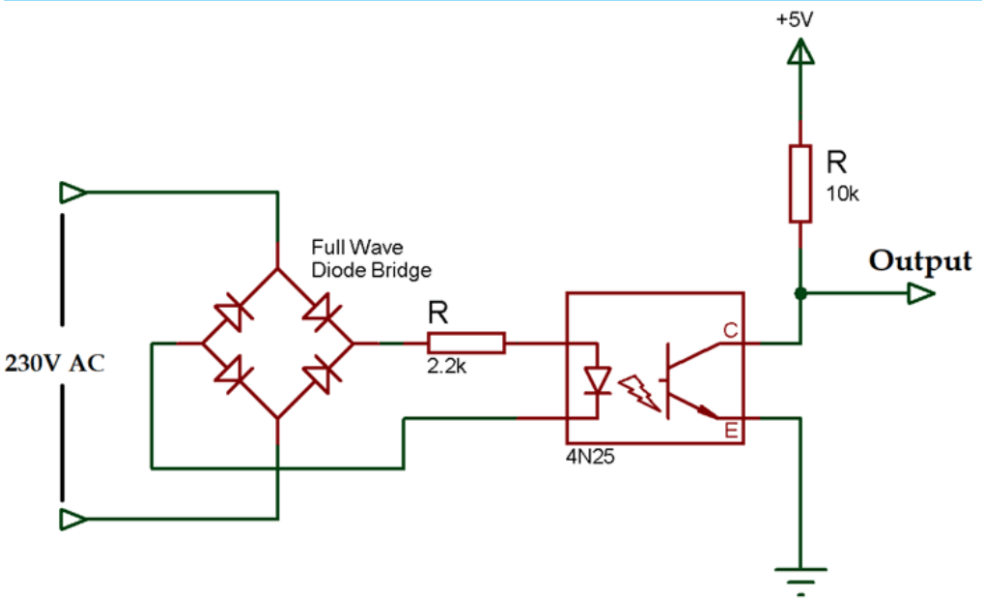


Figure 10-Zero crossing detector circuit

5.6.1 Features

4N25 Optocoupler

- Response time (tr: typ., 3 us at VCE = 10 V, IC = 2 mA, RL = 100 ohms)
- Current Transfer Ratio (CTR: min. 20% at IF = 10 mA, VCE = 10 V)
- Input-output isolation voltage (Viso = 2500 Vrms)
- Dual-in-line package
- UL approved
- CSA approved
- IEC/EN/DIN EN 60747-5-2 approved
- Options available are:
  060E = IEC/EN/DIN EN 60747-5-2 Option
  W00E = 0.4" Lead Spacing Option
  300E = Lead Bend SMD Option
  500E = Tape and Reel Packaging Option

BR1-2W10

- No. of Phases : Single
- Repetitive Reverse Peak Voltage(VRRM, VDC)
- 2W04 : 400V
- 2W10 : 1000V (1kV)
- Bridge Input Voltage (VRmS)
- 2W04 : 280V
- 2W10 : 700V
- Forward Current If (AV) : 2A
- Bridge Rectifier Case Style : Through Hole
- Forward Voltage VF Max : 1.1V
- No. of Pins : 4Pins
- Operating Temperature: 125°C to -55°C

5.6.2 Component selection for Zero-crossing detector

- ❖ BR1-2W10
- ❖ 47Kohm resister
- ❖ 4N25M optocoupler

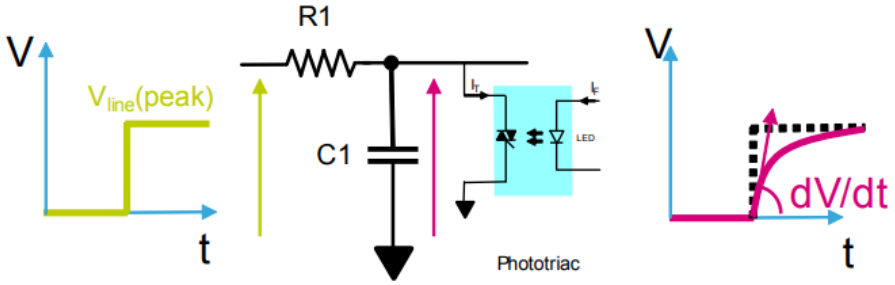
Opto-isolated triac triggering circuit

The opto-isolated triac triggering circuit acts as an isolated switch, enabling precise control over the AC power supplied to the hot air gun's heating element.

Bidirectional triode thyristor

Utilizing a bidirectional triode thyristor (triac) allows for efficient switching of AC power while ensuring isolation between the control circuitry and the high-voltage AC line.Due to the considerable heat generated during operation, especially by the triac, a large heatsink is incorporated into the PCB design. This heatsink efficiently dissipates heat away from the triac, maintaining optimal performance and reliability over extended periods of use.

5.6.3 Opto-isolated triac triggering circuit



$$\frac{dV_{C1}}{dt}(\max) = \frac{V_{line}(peak)}{\tau} = \frac{V_{line}(peak)}{R1 \times C1}$$

Figure 11-triac triggering circuit

MOC3021

* Isolation voltage between input and output V_{iso} : 5,000V_{rms} **Jan**
* 6pin DIP photocoupler, triac driver output
* High repetitive peak off-state voltage V_{DRM} : Min. 400V
* High critical rate of rise of off-state voltage
  (dV/dt : MIN. 100V / μs)
* Dual-in-line package :
  MOC3020, MOC3021, MOC3022, MOC3023
* Wide lead spacing package :
  MOC3020M, MOC3021M, MOC3022M, MOC3023M
* Surface mounting package :
  MOC3020S, MOC3021S, MOC3022S, MOC3023S
* Tape and reel packaging :
  MOC3020S-TA1, MOC3021S-TA1, MOC3022S-TA1, MOC3023S-TA1
* Safety approval
  UL / CSA / FIMKO / VDE* approved
  *Required "V" ordering option

5.6.4 Component selection for Opto-isolated triac triggering circuit

- 10nF capcitor
- BTA12-600B
- MOC3021 octocoupler
- Resistor
  - 39ohm (2W)
  - 330ohm

5.6.5 Final circuit diagram for AC PCB

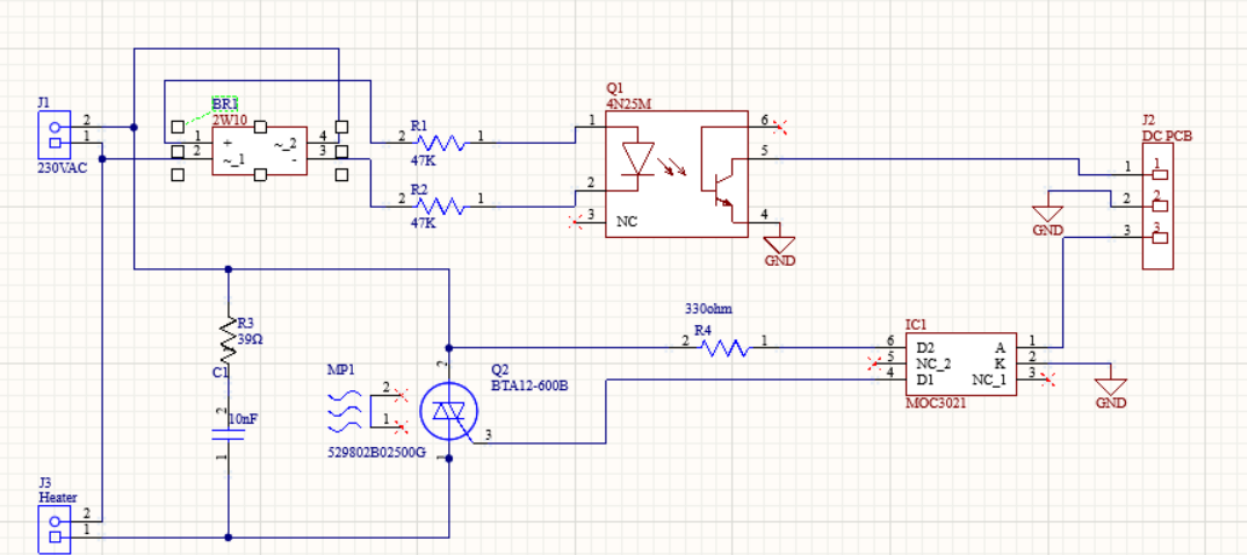


Figure 12-AC PCB circuit diagram

6.0 PCB Design -Schematic with final Circuit diagram, Routing, PCB 3D view, Drilling details and printed PCB

6.1 Schematic Design

For the schematic design of our soldering station project, a hierarchical approach is employed for the DC PCB, while the AC PCB follows a more straightforward layout. In the DC PCB schematic, a hierarchical structure divides the circuit into manageable modules, detailing the temperature control system, power supply circuitry, and sensor interfaces. Each functional block is represented as a hierarchical module, facilitating easier understanding and troubleshooting.

Conversely, the AC PCB schematic presents a clear representation of the zero-crossing detector and opto-isolated triac triggering circuit, focusing on simplicity and clarity. Both schematics prioritize clear connections between components to ensure proper signal flow and functionality, laying the foundation for the subsequent PCB design stages.

6.1.1 Hierarchical structure for DC PCB

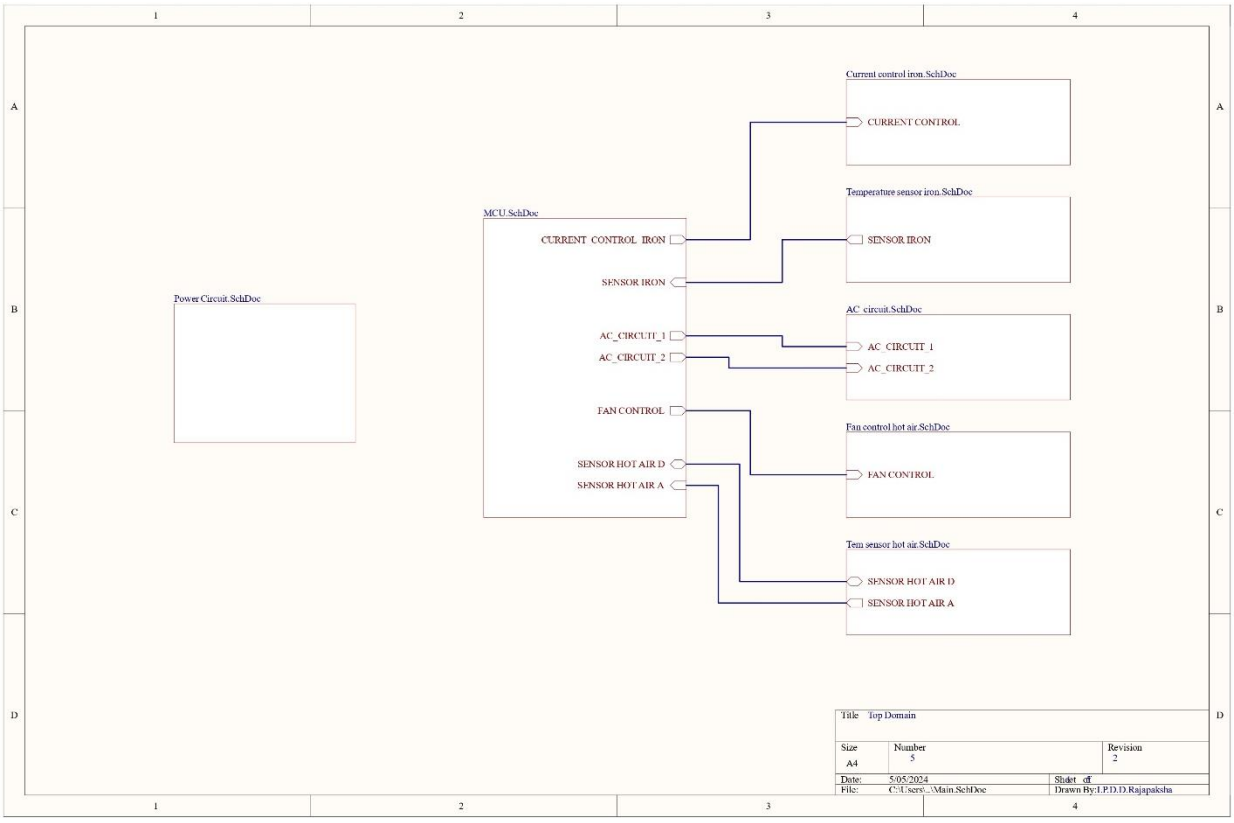
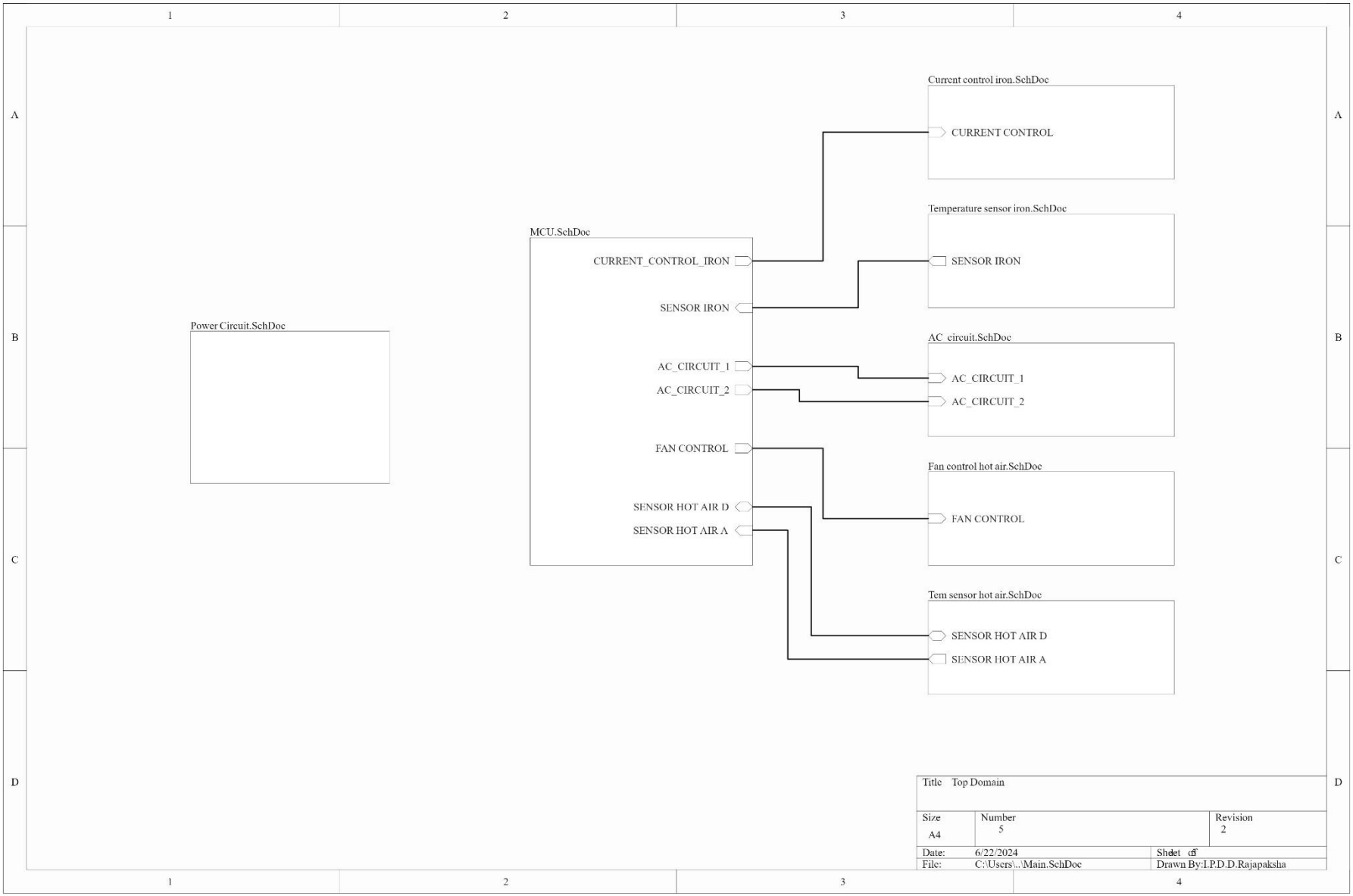
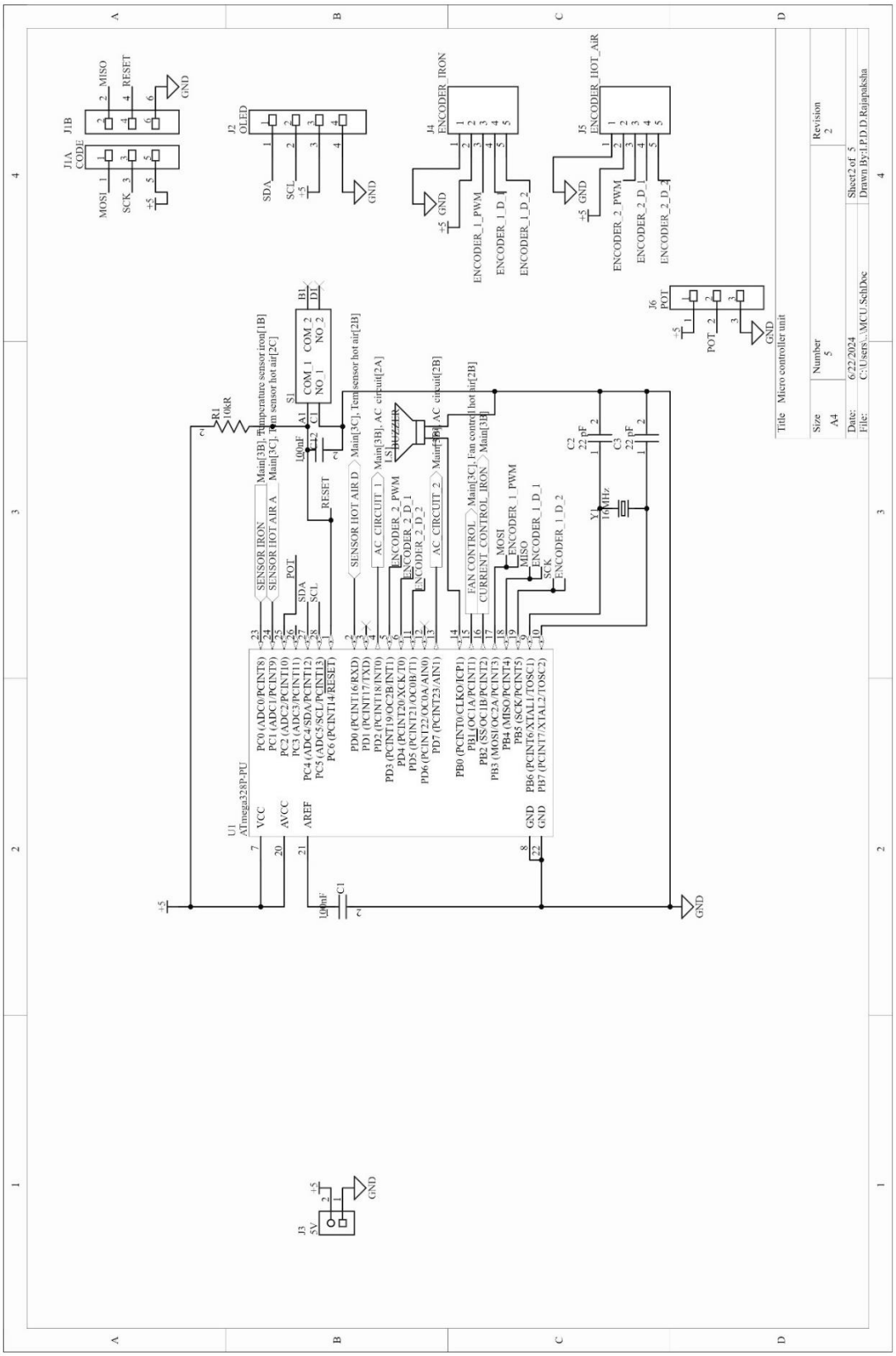


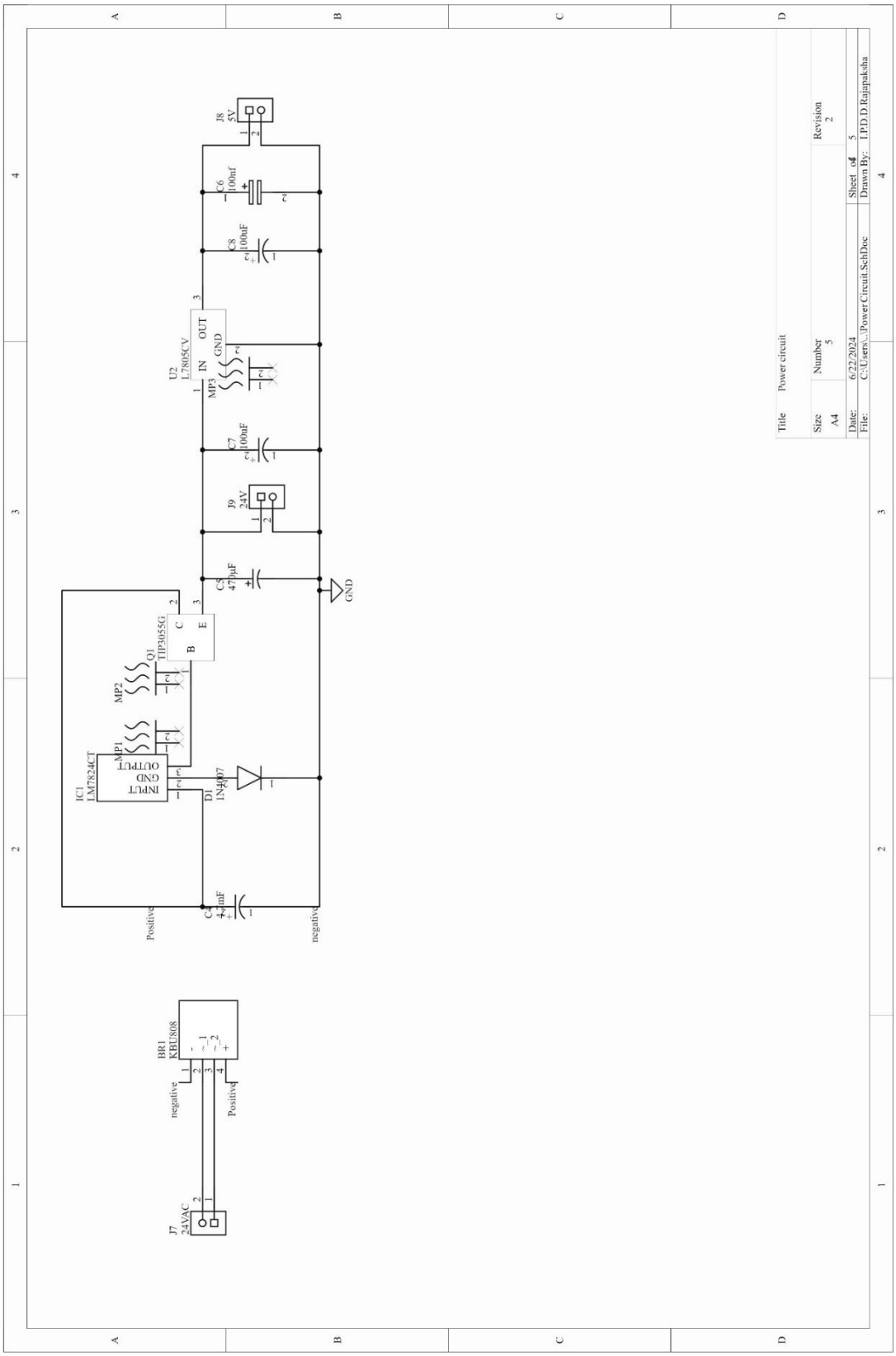
Figure 13-hierarchical design


Current control iron.SchDoc
CURRENT CONTROL
Temperature sensor iron.SchDoc
SENSOR IRON
MCU.SchDoc
CURRENT_CONTROL_IRON
SENSOR IRON
AC_CIRCUIT_1
AC_CIRCUIT_2
FAN CONTROL
SENSOR HOT AIR D
SENSOR HOT AIR A
Power Circuit.SchDoc
AC_circuit.SchDoc
AC_CIRCUIT_1
AC_CIRCUIT_2
Fan control hot air.SchDoc
FAN CONTROL
Tem sensor hot air.SchDoc
SENSOR HOT AIR D
SENSOR HOT AIR A
Title Top Domain
Size A4
Number 5
Revision 2
Date: 6/22/2024
Sheet of
File: C:\Users\..\Main.SchDoc
Drawn By:I.P.D.D.Rajapaksha

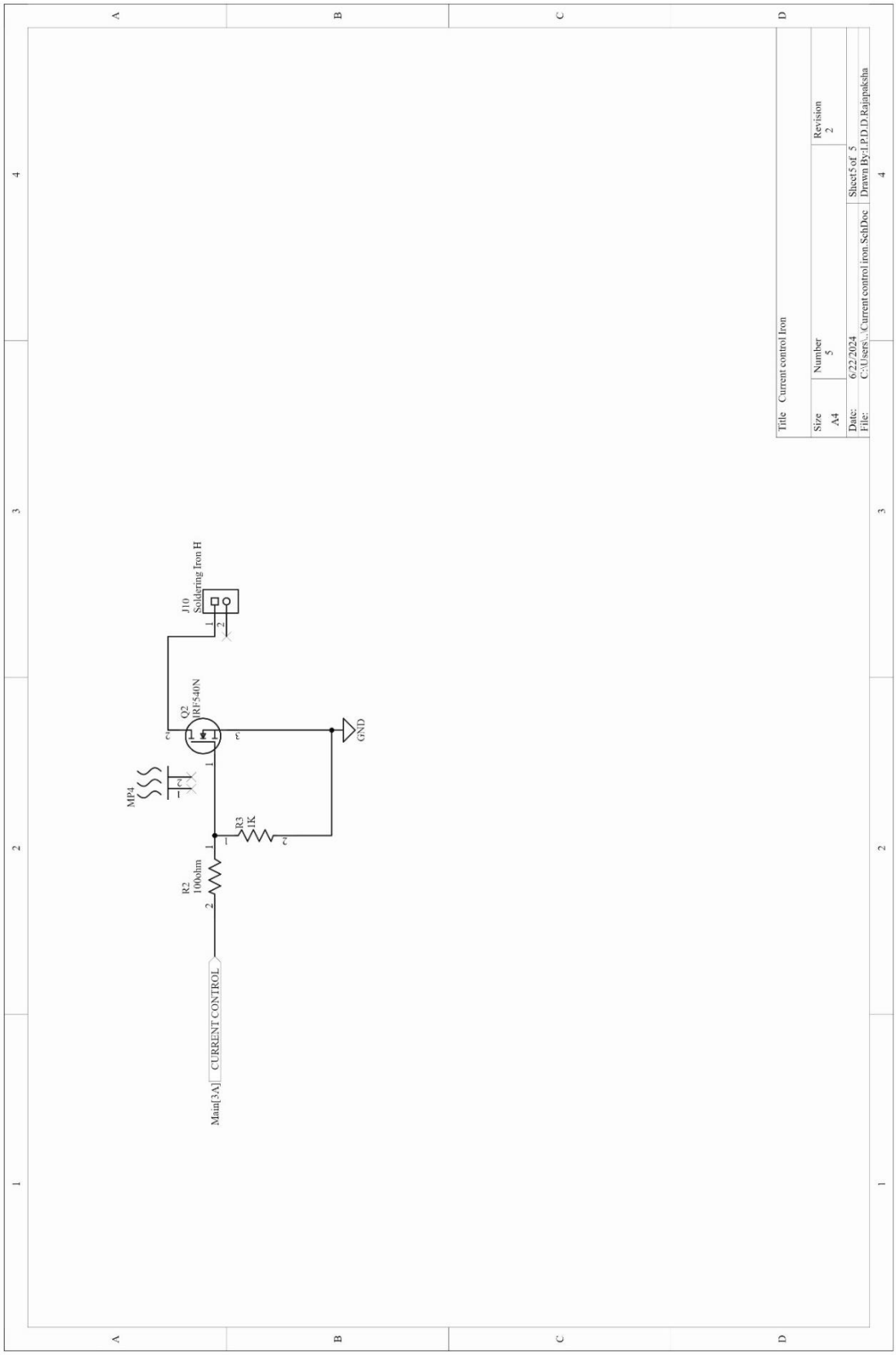
6.1.2 Microcontroller Unit



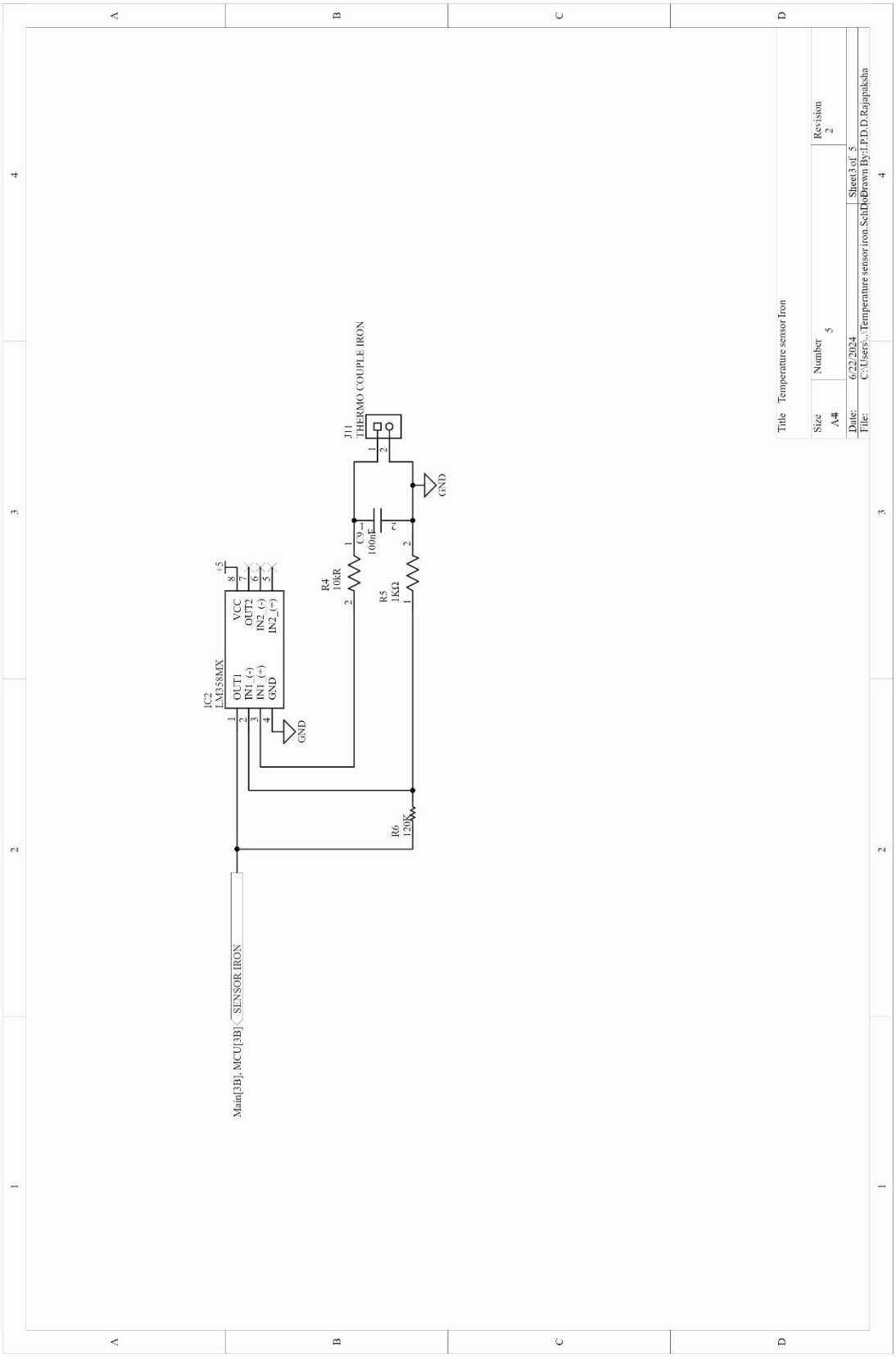
6.1.3 Power supply



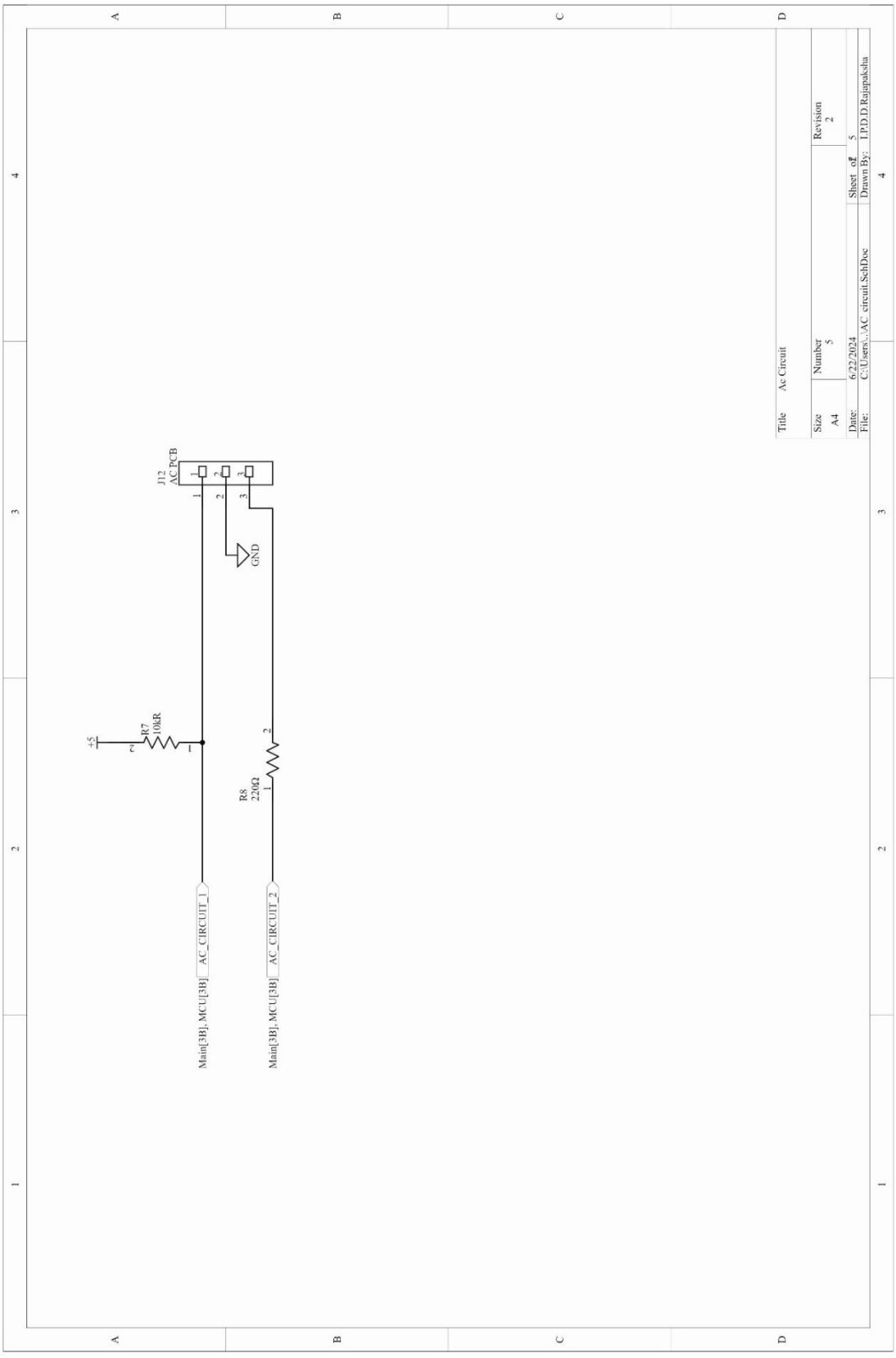
6.1.3 Current control Iron Circuit



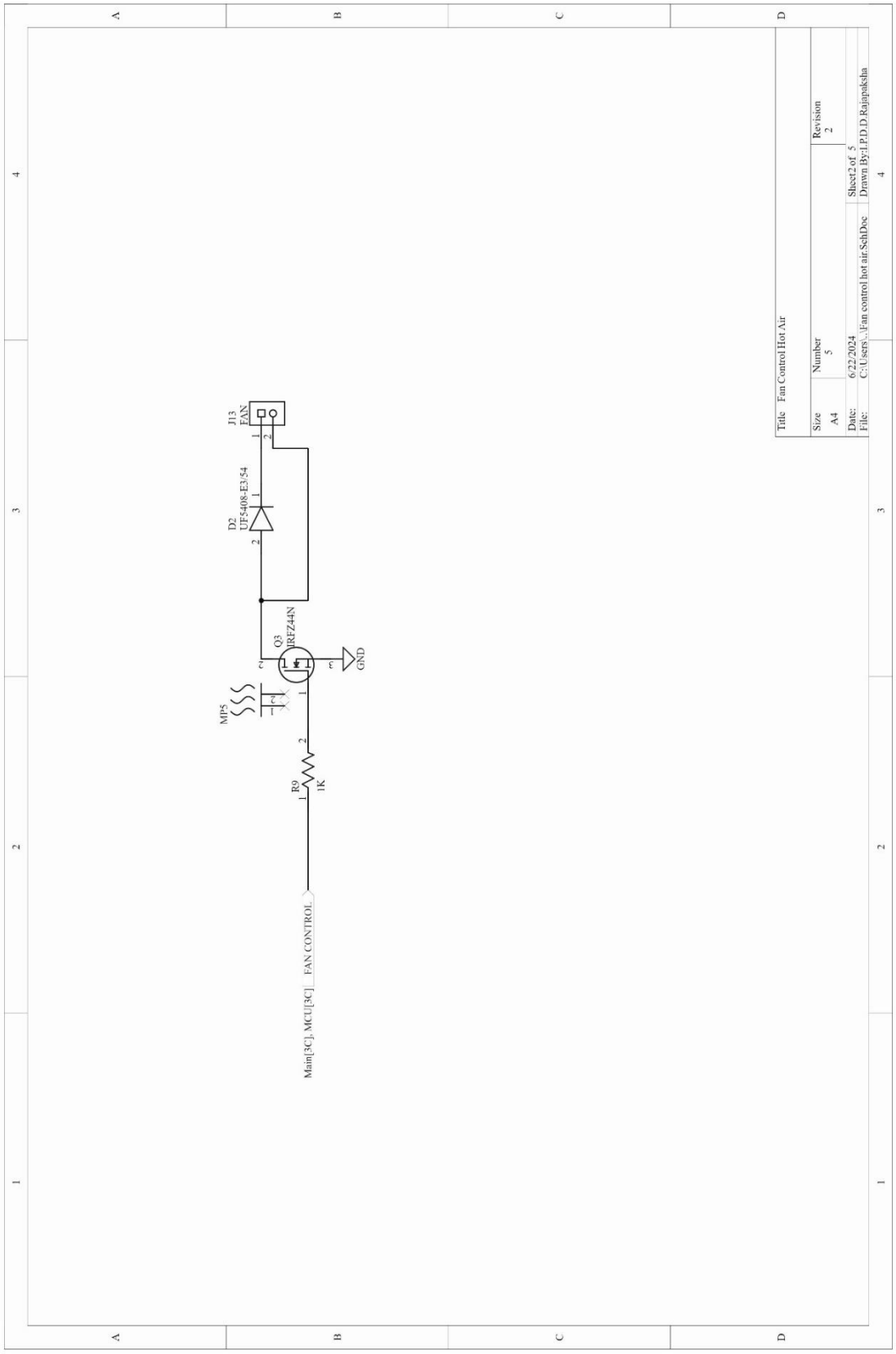
6.1.4 Temperature Sensor Iron



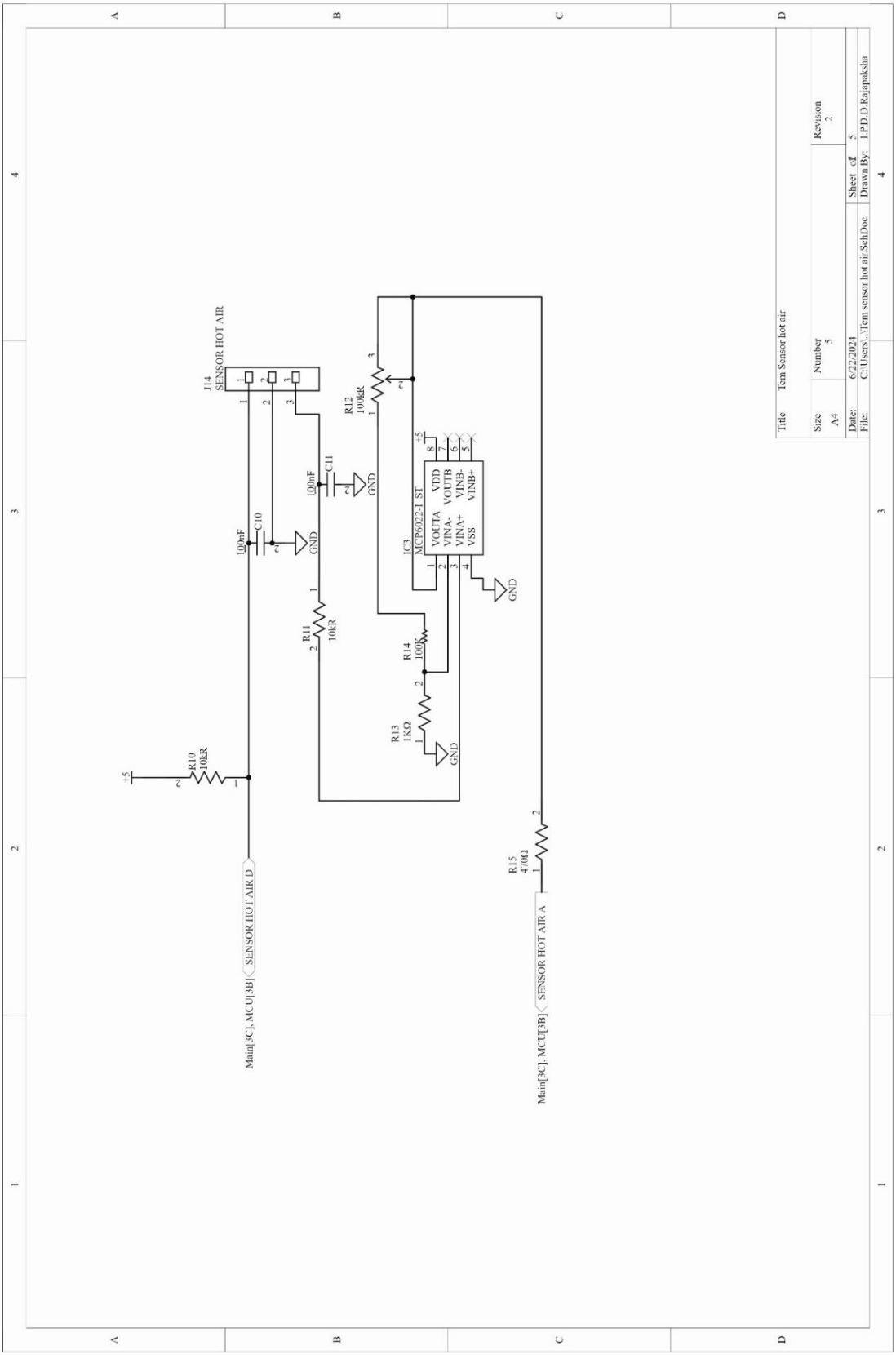
6.1.5 AC Circuit



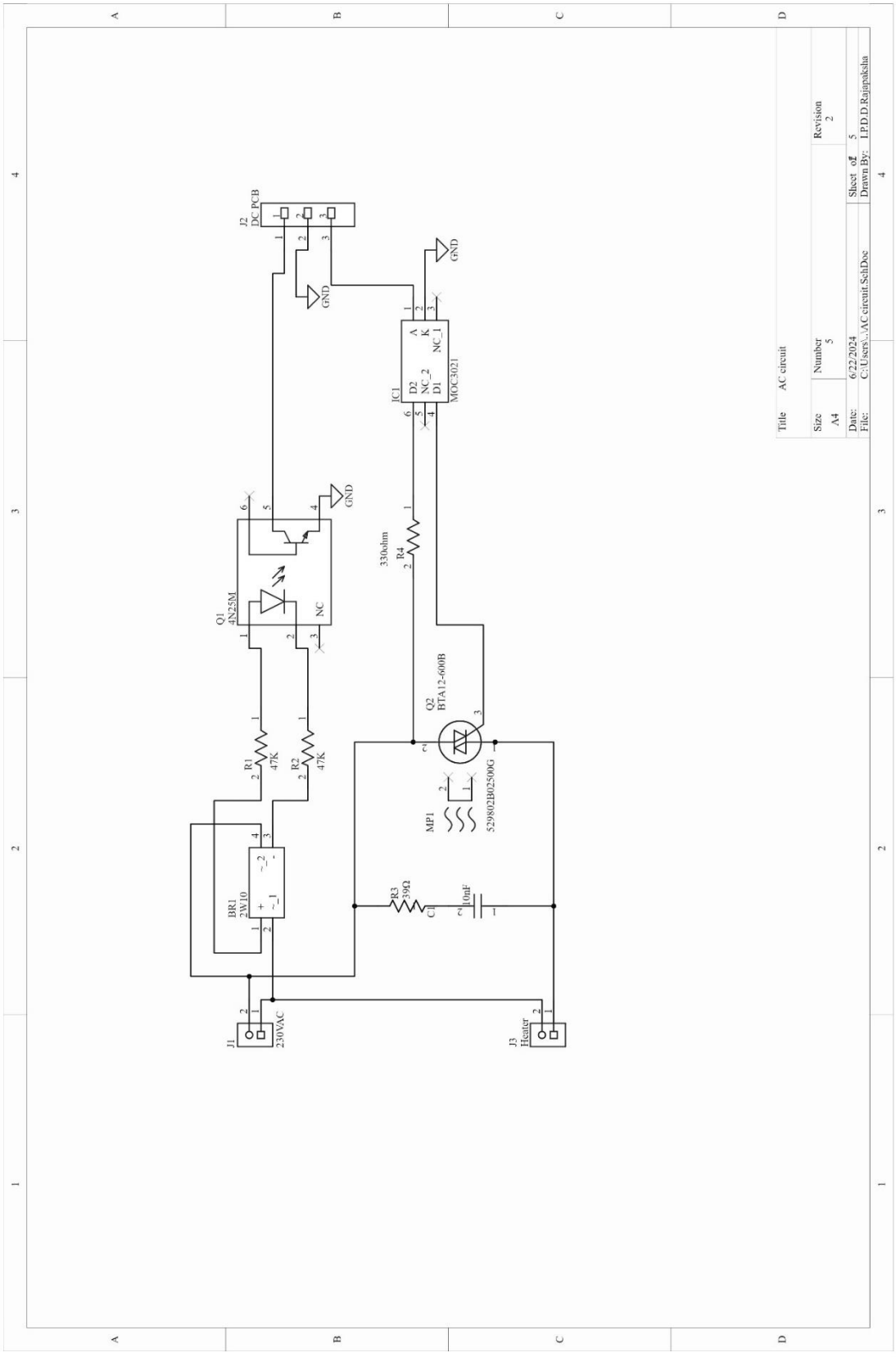
6.1.6 Fan Control Hot Air Gun



6.1.7 Temperature Sensor Hot Air Gun



6.1.8 AC PCB


J2
DC PCB
GND
GND
IC1
MOC3021
D2
NC_2
D1
A
K
NC_1
330ohm
R4
Q1
4N25M
NC
Q2
BTA12-600B
R1
47K
R2
47K
BR1
2W10
MP1
529802B02500G
R3
39Ω
C1
10nF
J1
230VAC
J3
Heater
Title
AC circuit
Size
A4
Number
5
Revision
2
Date:
6/22/2024
File:
C:\Users\...\AC circuit.SchDoc
Sheet of
5
Drawn By:
I.P.D.D.Rajapaksha

Once the schematic designs are finalized, the next step involves translating them into physical PCB layouts. This includes routing traces, placing components, defining drill hole locations, and generating the final PCB design files for manufacturing.

6.2 Routing

For routing the PCB design of our soldering station project, we adopt a cautious approach to ensure reliable current flow and minimize signal interference. Given that certain paths handle up to 3A of current, we opt for a trace width of 2.5mm in these critical areas, providing ample capacity to handle the current without overheating or voltage drops. This choice exceeds the typical doubling rule for trace widths, emphasizing an extra margin of safety. Additionally, for other routes carrying lower currents, we employ a trace width of 0.38mm, which exceeds the minimum requirement for signal integrity and durability. By carefully selecting trace widths based on current requirements and signal characteristics, we aim to optimize the performance and reliability of our PCB design for the soldering station.

6.2.1 Top layer –DC PCB

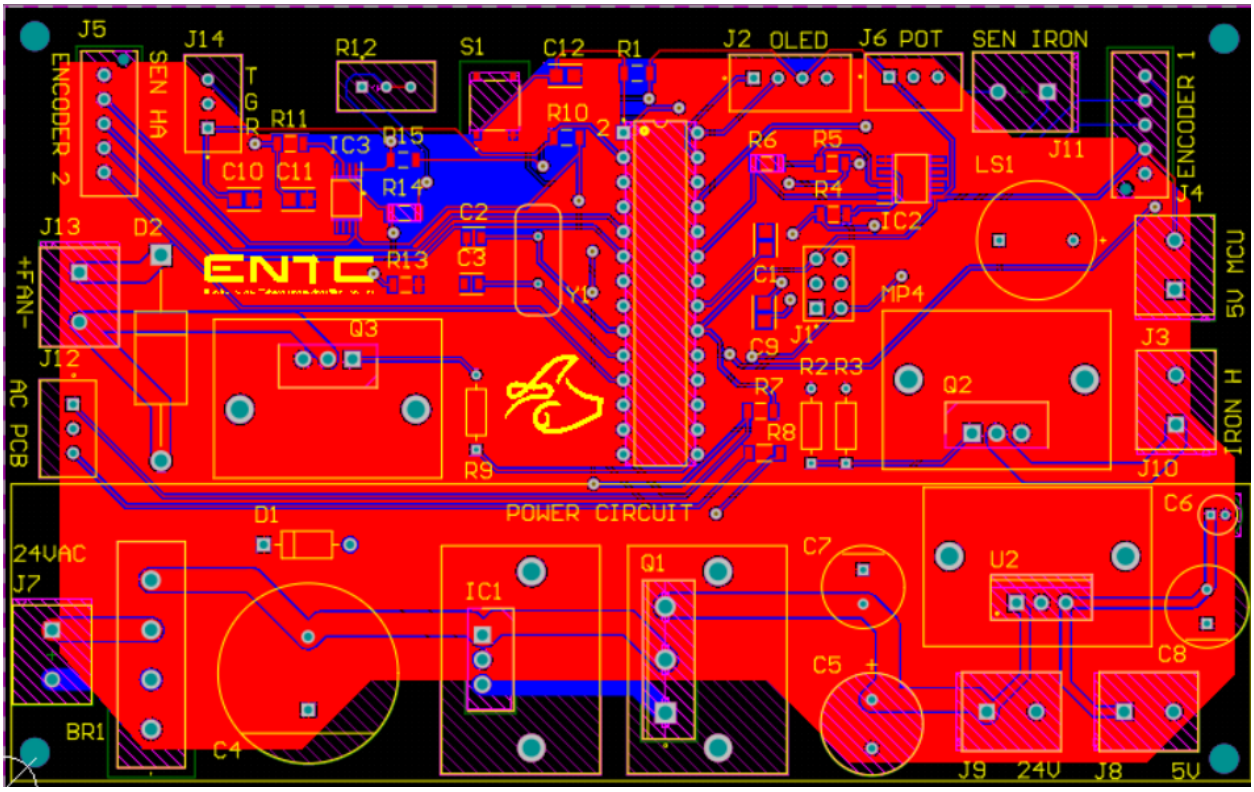


Figure 14-Top layer

6.2.2 Bottom layer – DC PCB

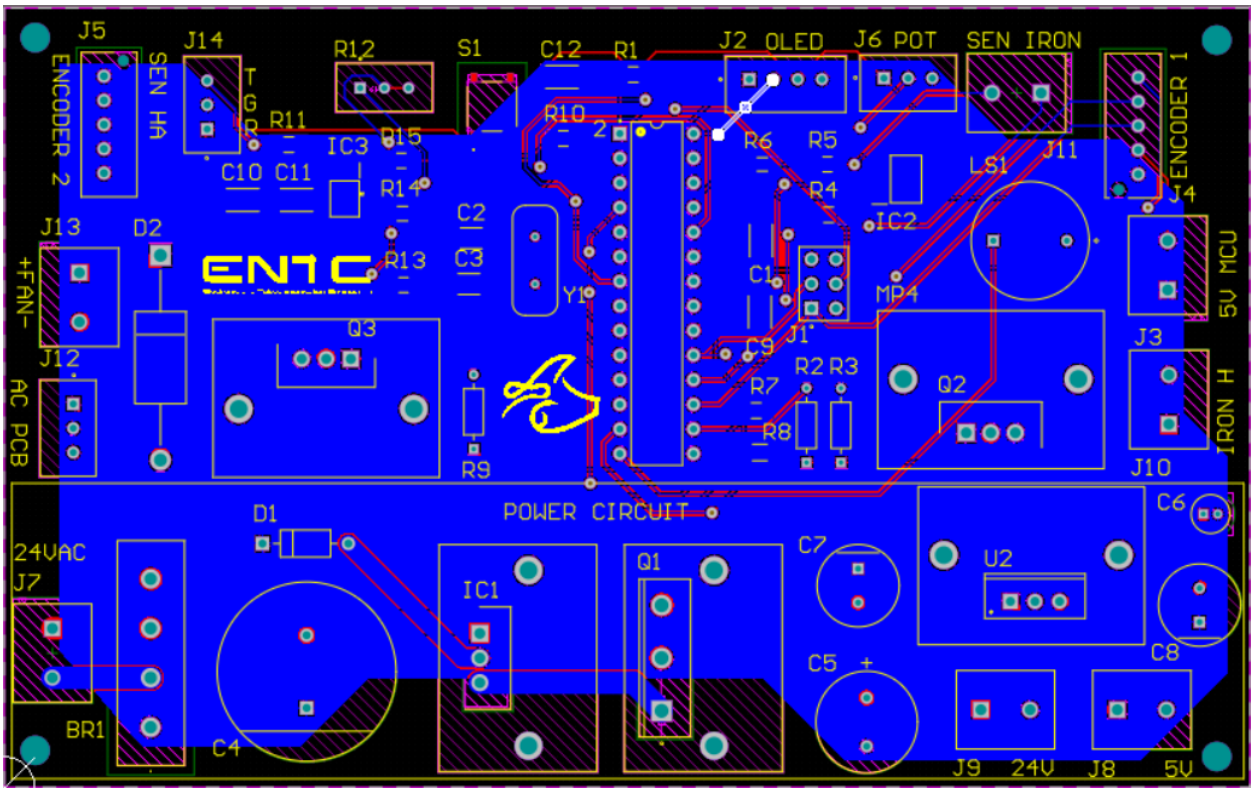


Figure 15-bottom layer

6.2.3 Top Layer – AC PCB

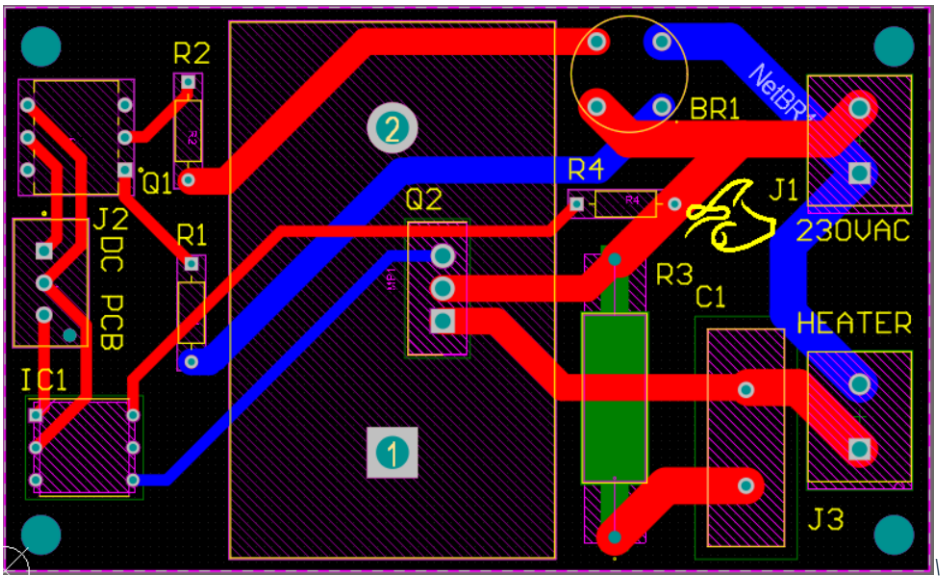


Figure 16-Top layer

6.2.4 Bottom Layer – AC PCB

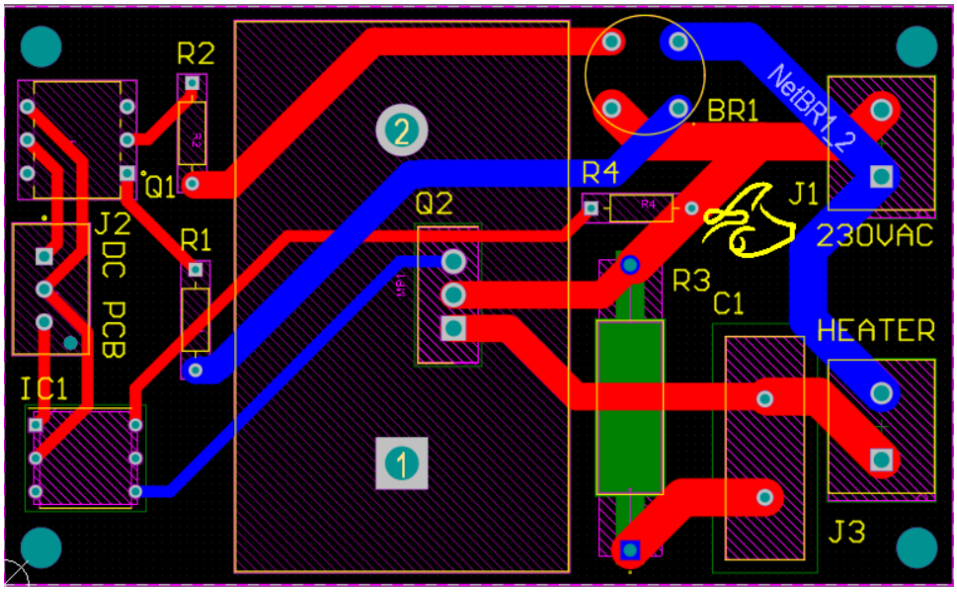


Figure 17-Bottom layer

In the routing phase, traces are carefully laid out to minimize signal interference and ensure proper electrical connections. Components are placed strategically on the PCB to optimize space utilization and facilitate efficient assembly.

6.3 3D view

6.3.1 DC PCB

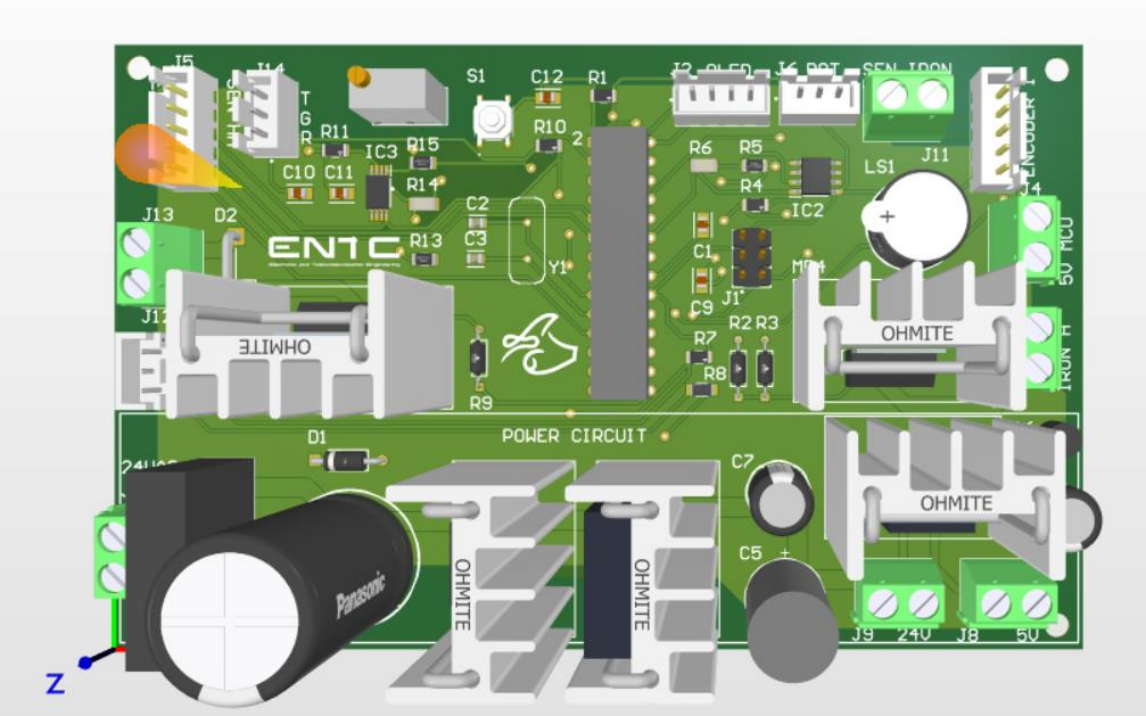


Figure 18-DC PCB 3D view

6.3.2 AC PCB

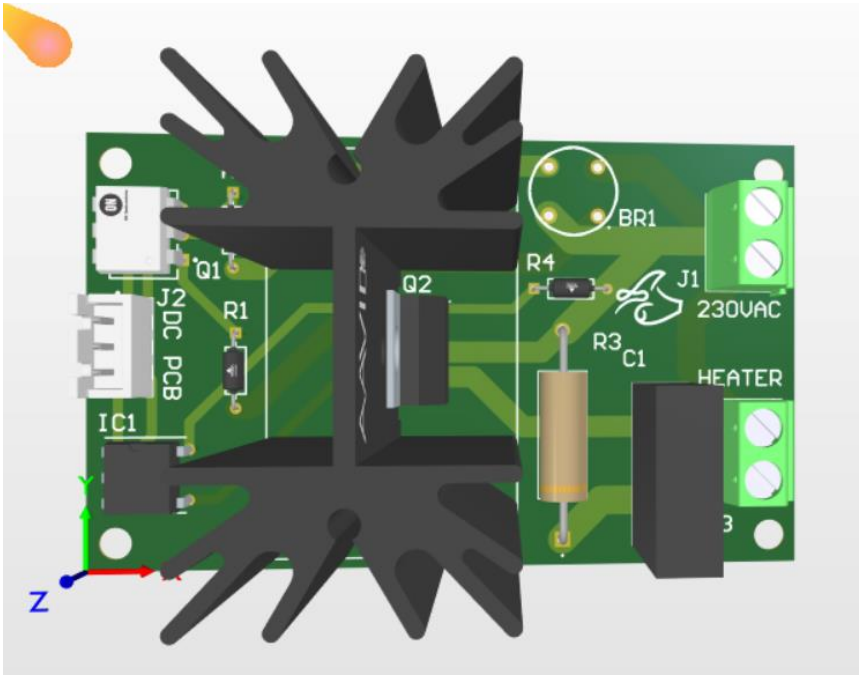


Figure 19-AC PCB 3D view

6.4 Drilling Details

Drilling details are specified to ensure accurate hole placement for mounting components and connectors. Hole sizes are determined based on component specifications and PCB fabrication requirements.

6.4.1 DC PCB

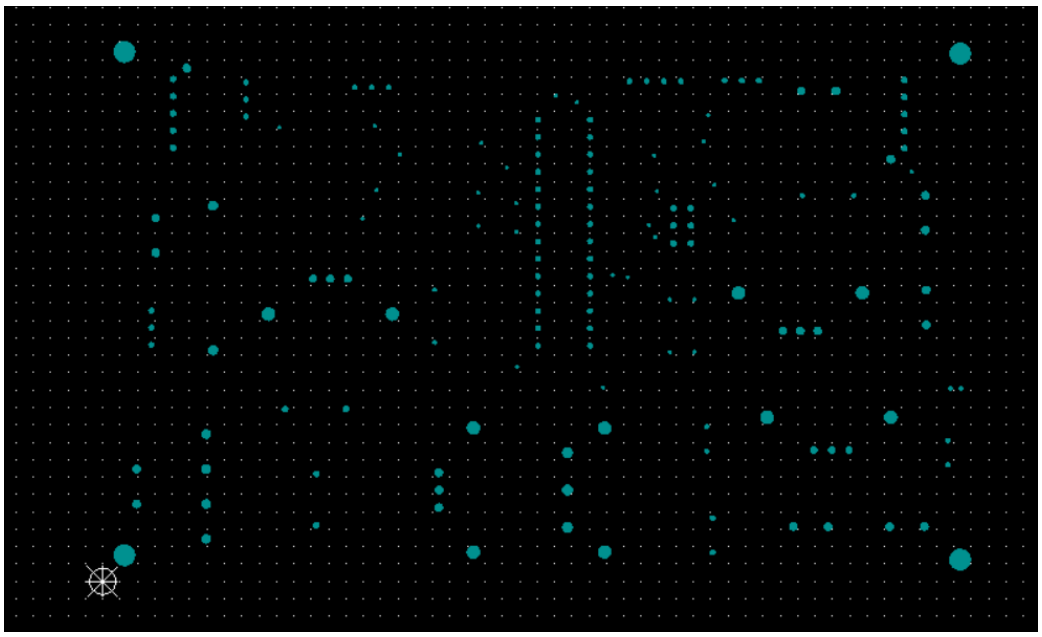


Figure 20-Drill details

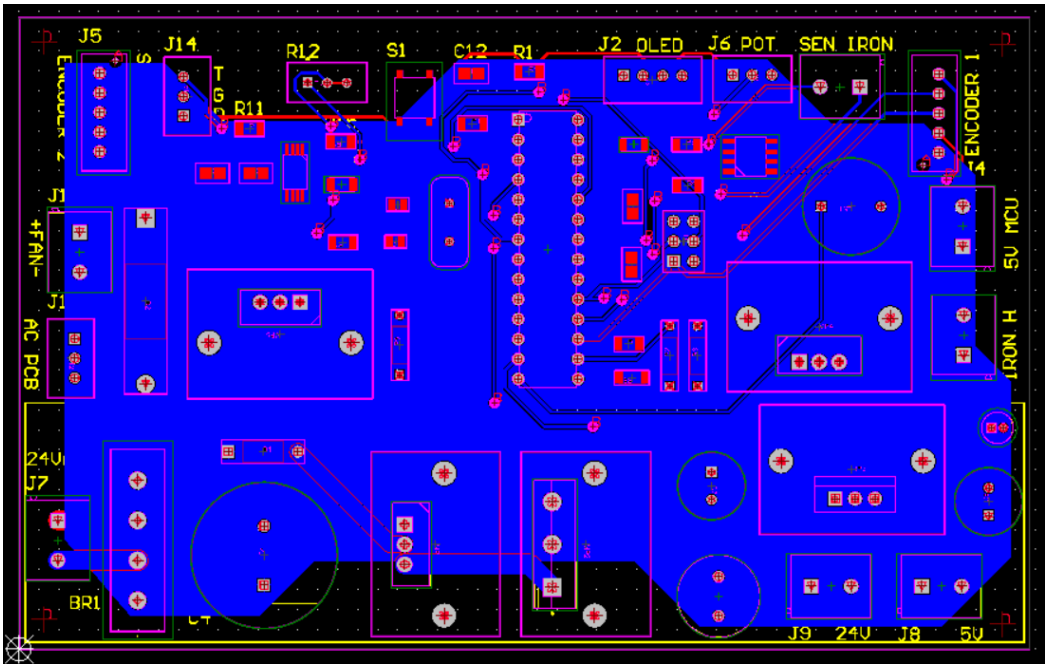


Figure 21-Drill details

6.4.2 AC PCB

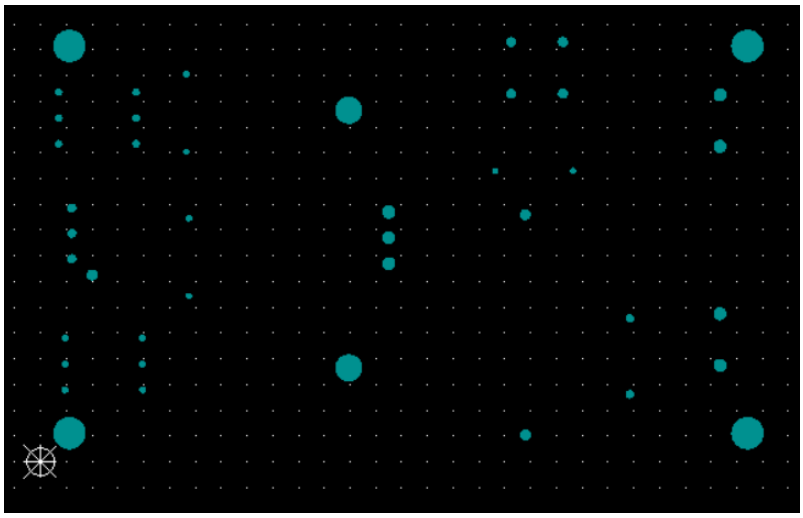


Figure 22-Drill details

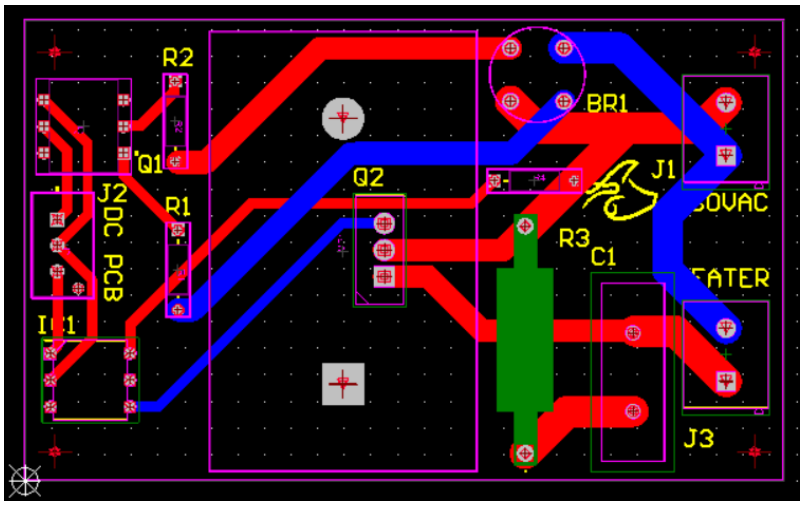


Figure 23-Drill details

After routing and drilling details are finalized, the PCB design files are generated and sent for manufacturing. The manufacturer uses these files to produce the printed PCB according to the specified design parameters.

6.5 Printed PCB

6.5.1 DC PCB

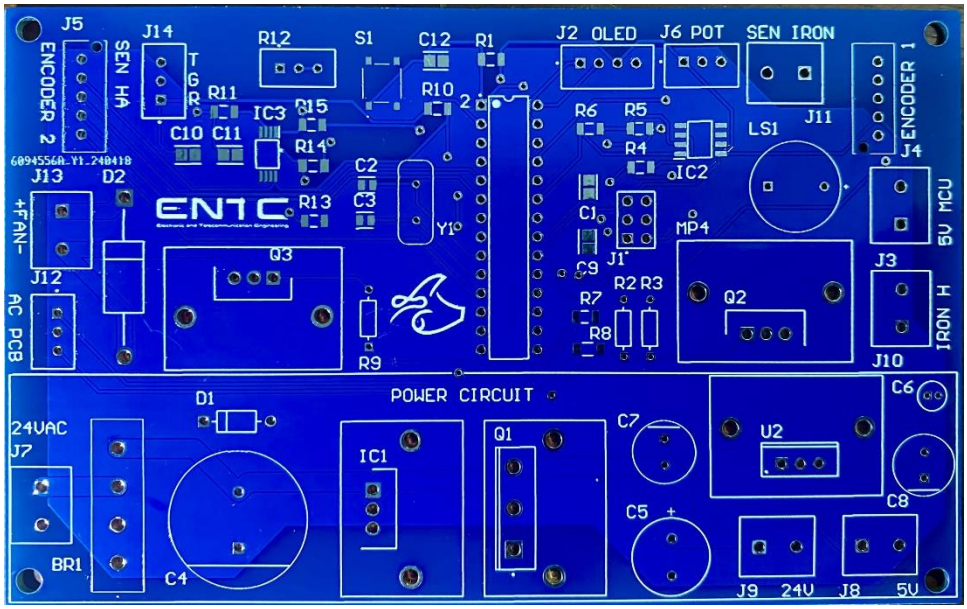


Figure 24-Front side

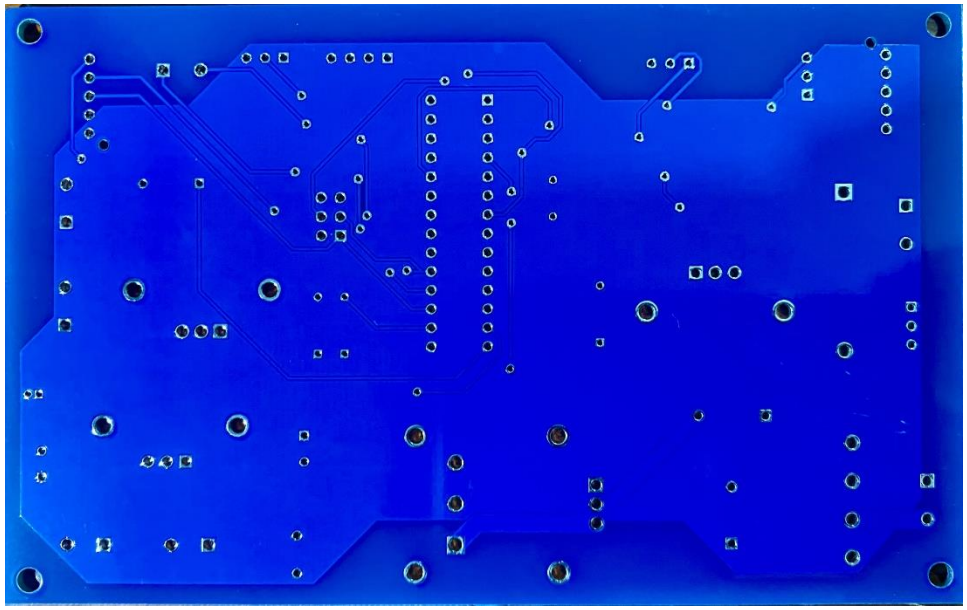


Figure 25-Back side

6.5.2 AC PCB

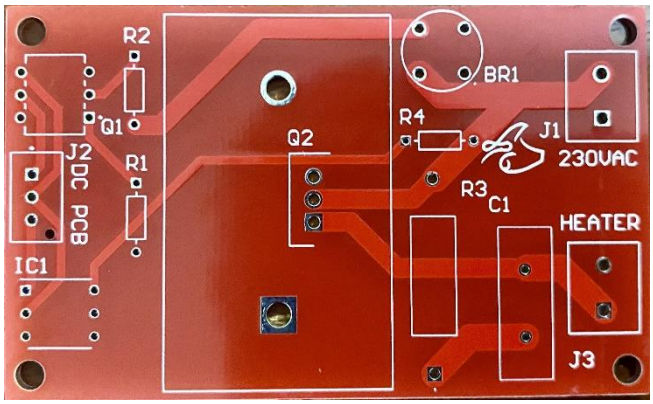


Figure 26-Front side

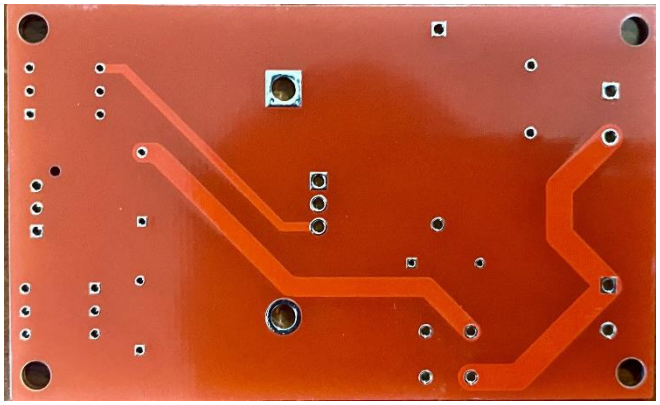


Figure 27-Back side

6.6 Soldered PCB

6.6.1 DC PCB

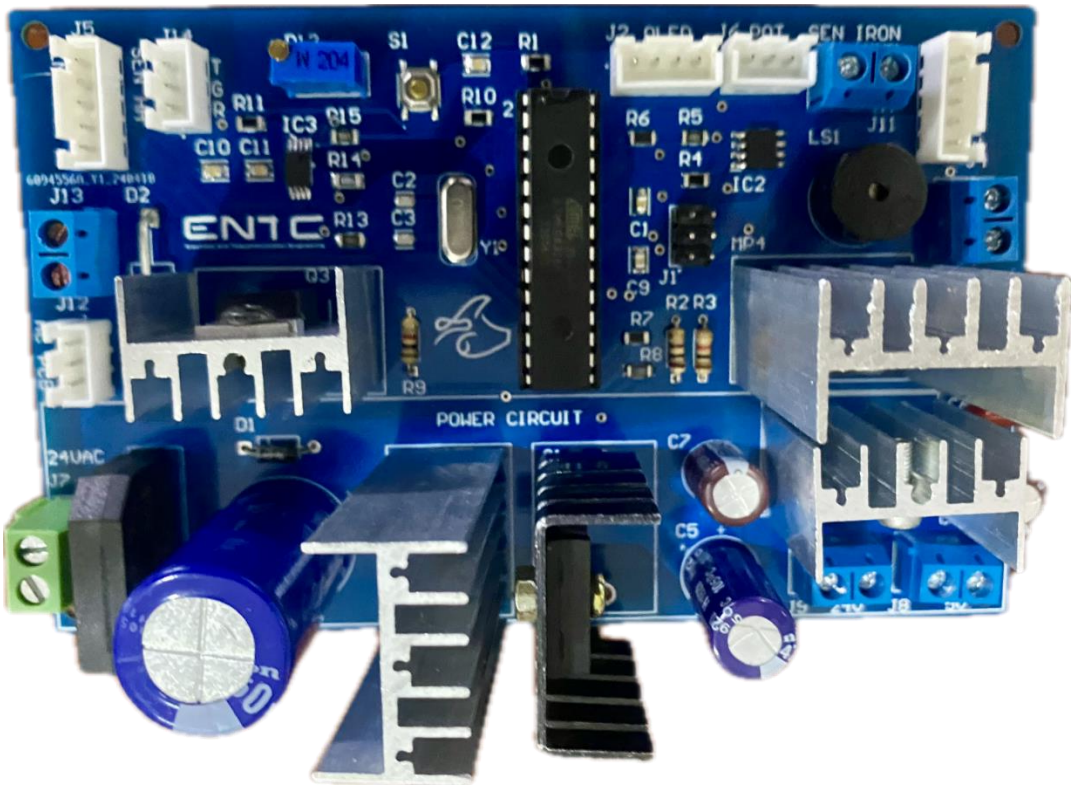


Figure 28-soldered DC PCB

6.6.2 AC PCB

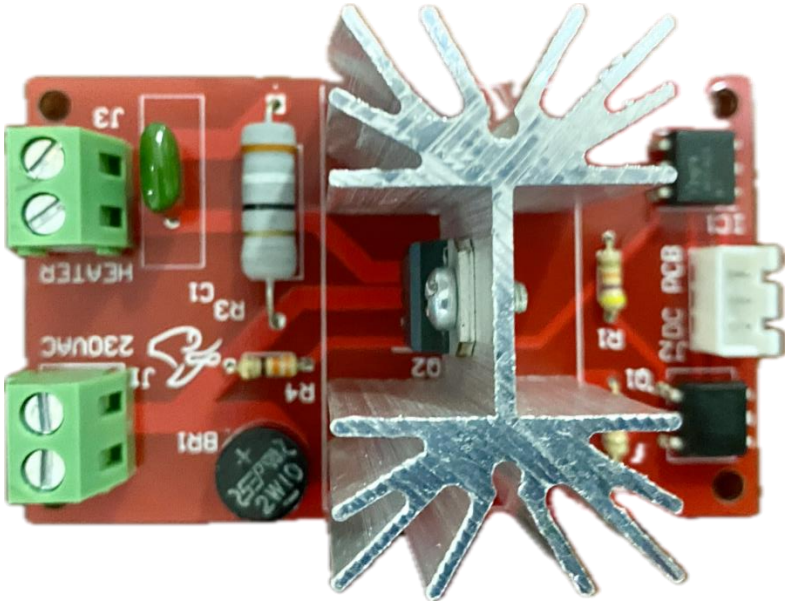


Figure 29-soldered AC PCB

6.7 Connector Details

6.7.1 Power Connectors

For the 230V AC power input, we utilize robust screw terminals with a larger size to accommodate higher voltage and current requirements safely. These screw terminals provide secure connections and facilitate easy wiring during assembly. Additionally, for the 24V DC power supply, we employ screw terminals, chosen for their compact size and suitability for lower voltage applications.

6.7.2 Data Line Connectors

To establish connections for data lines and signals, we employ 25 mm JST connectors of varying pin configurations, including 3-pin, 4-pin, and 5-pin variants. These JST connectors offer reliable and compact solutions for interconnecting components and peripheral devices on the PCB.

6.7.3 Component-Specific Connectors

For interfacing with external devices such as the soldering iron and hot air gun, we utilize standard 8-pin and 5-pin DIP (Dual In-line Package) connectors. These DIP connectors provide convenient interfaces for connecting and disconnecting components, facilitating ease of maintenance and replacement.

Figure 30-8 pin output

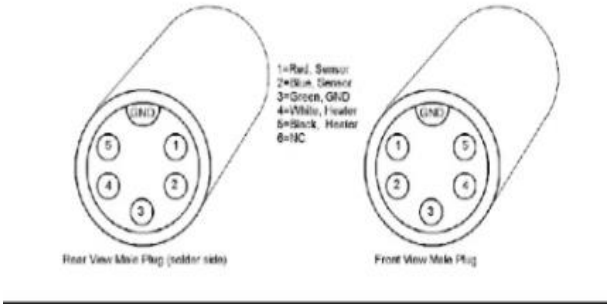
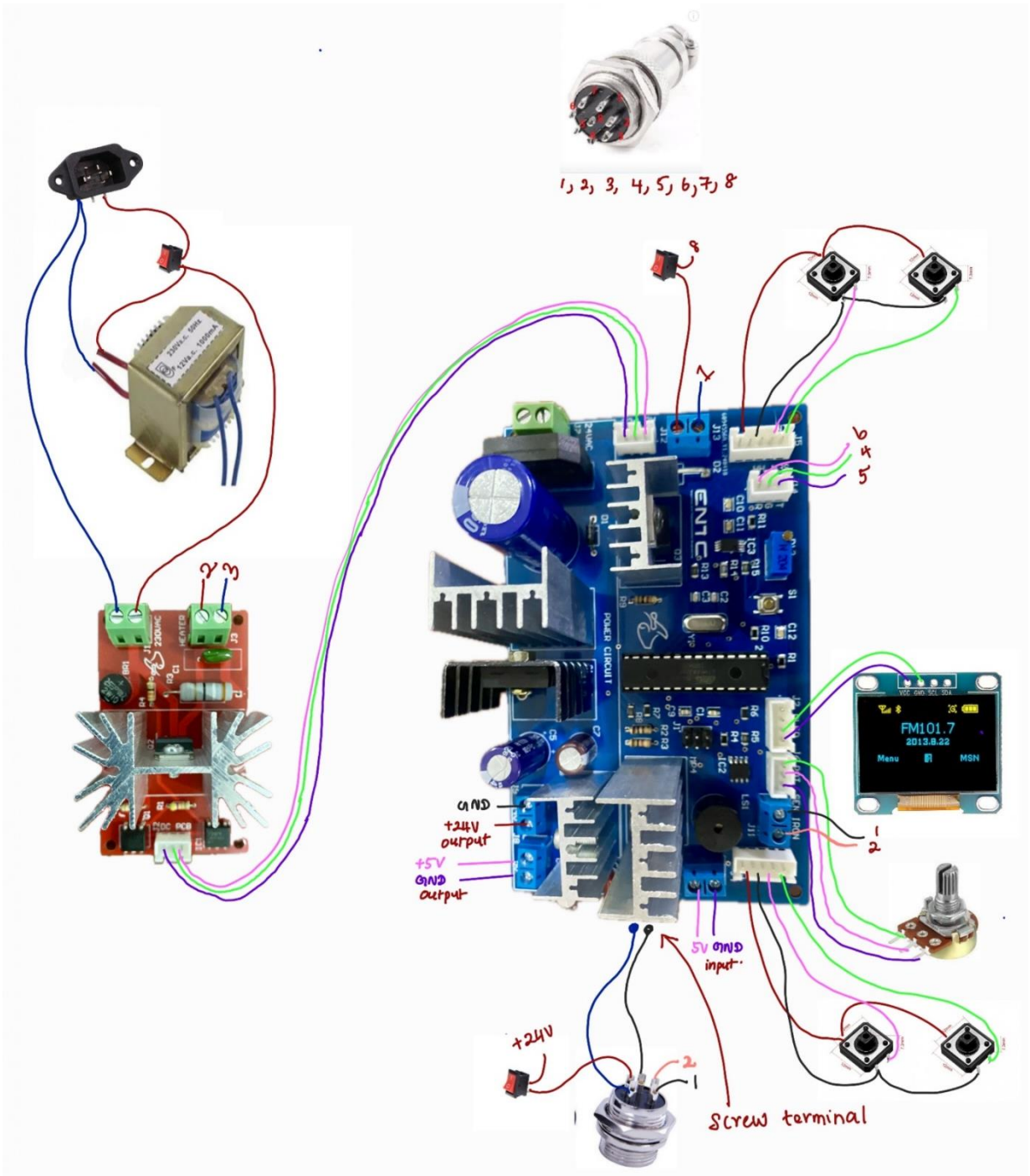


Figure 31-5 pin output

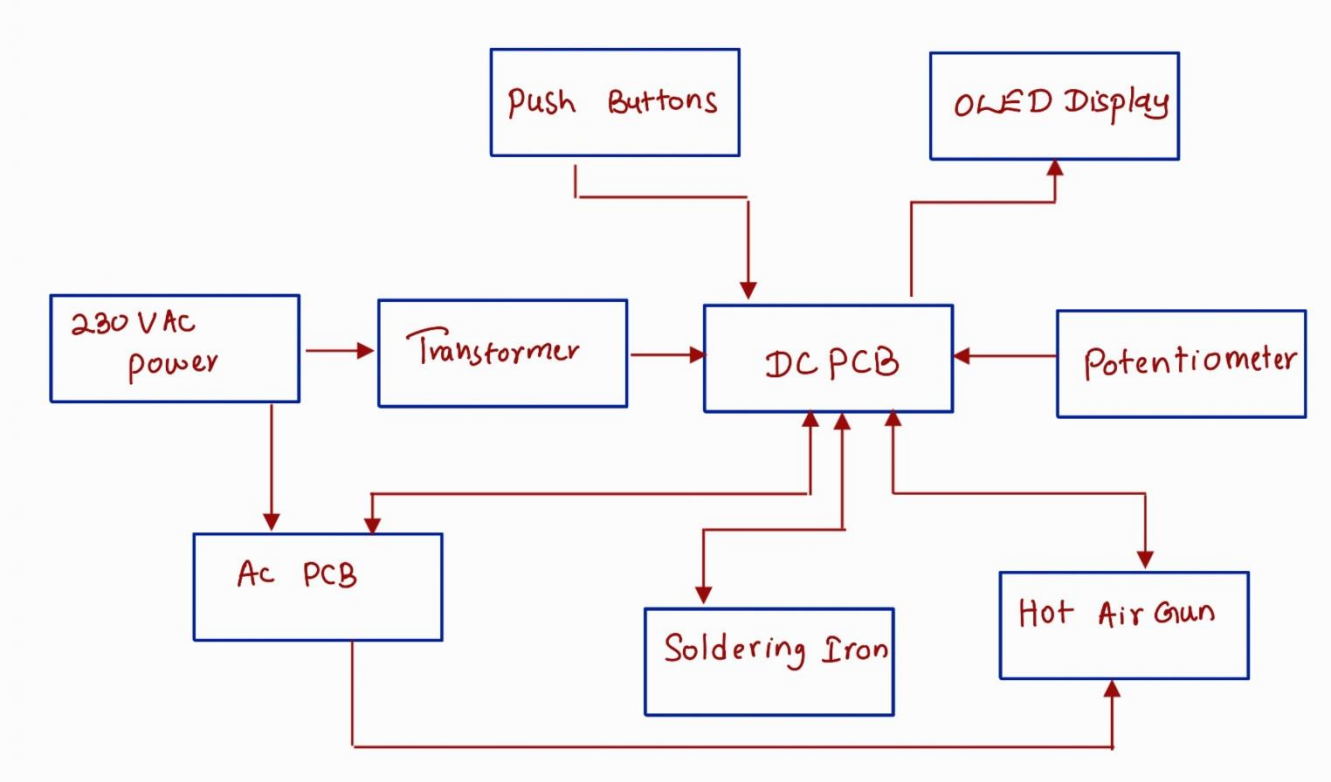
6.8 Wiring Diagram for the system

In wiring diagram, we meticulously plan and illustrate the interconnections between various components and circuits on the PCB. To ensure reliable operation and safety, we use different wire sizes tailored to specific current requirements. For circuits carrying higher currents, such as those connected to the soldering iron and hot air gun, we opt for larger gauge wires capable of

handling 2 to 3 amperes of current. This choice minimizes resistance and voltage drops, ensuring efficient power transmission and preventing overheating or damage to the wires. Additionally, we employ standard 8-pin and 5-pin DIP connectors to interface with the soldering iron and hot air gun consistently and securely. By adhering to standardized wiring practices and selecting appropriate wire sizes, we maintain optimal performance and reliability throughout the electronic system.



6.9 Block Diagram for the Complete System



6.10 Test reports and Test procedures

The power supply circuit, microcontroller circuit, current controlling circuit, and amplifier circuit were tested before the labs closed due to a non-academic staff strike. The AC PCB, hot air current control, and amplifier still need to be tested, and the PID code for the soldering iron needs tuning. These tasks require a signal generator, an oscilloscope, and a power supply.

3.1. Sensor (Amplifier) testing

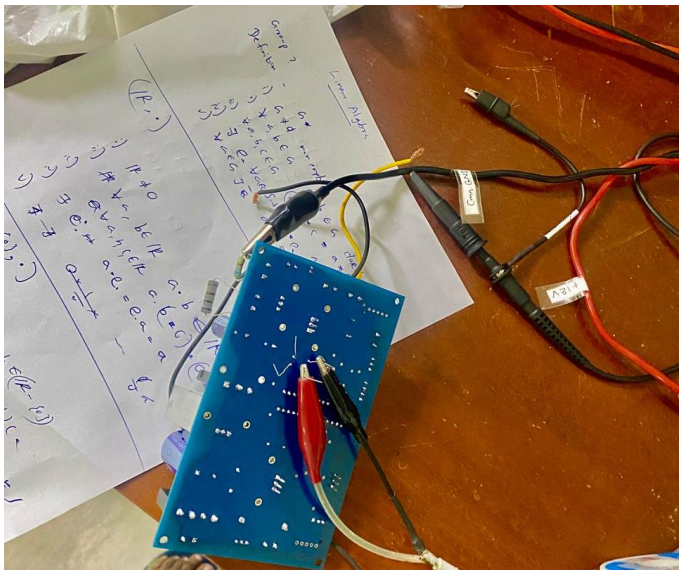


Figure 32 - Testing

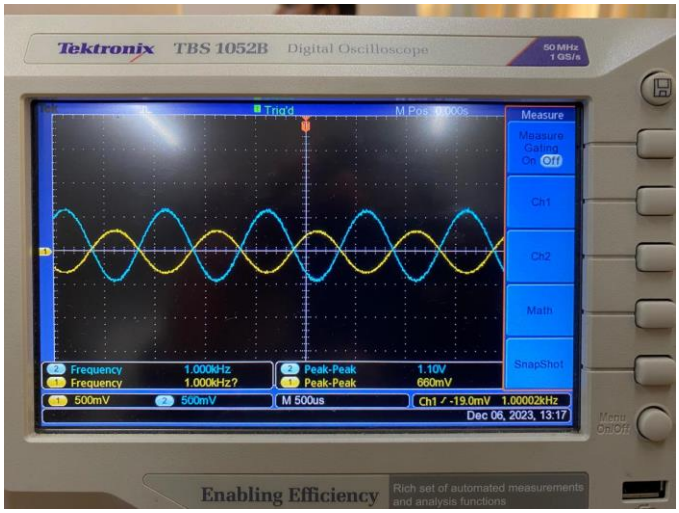


Figure 33 - Sensor output

7.0 Industrial design

7.1 Sub-assemblies and their specifications

1. Soldering Iron
   Name- Soldering Iron Handle for 907A (KAWH Spare Part)

   Specifications:

   - Model 907A
   - Wattage: 50W
   - Voltage: DC 24V
   - Connector: 5 holes
   - Heating element: 60W
   - Temperature: 200 ~ 480C

2. Hot air gun
   Name- Hot Air Gun 8586 (KAWH Spare Part)

   Specifications:

   - Thermal fuse resistance: Nickel alloy thermal fuse resistance ≈60 Ohm
   - Sensor type: K-type thermocouple (Characteristic resistance ≈3Ohm)
   - Built-in fan voltage: DC24V or 12V; the voltage is variable during air regulation.
   - Plug diameter: Approx. 0.49inch/12.5mm
   - Plug pin quantity: 8 female pins.

3. 5 pin connector.
   - GX16 XLR Aviation Plug Socket Metal Female Wire Panel Connector 16mm, 5 pin.
4. 8 pin connector.
   - GX16 XLR Aviation Plug Socket Metal Female Wire Panel Connector 16mm, 8 pin.
5. On/Off switches

   Specifications:

   - Red Mini 3-Pin On/Off Rectangular Rocker Switch SPST
   - LED illuminated switch.
   - Rating: 6A 250VAC /10A 125VAC
   - Mounting Hole Size: 3/4" x 1/2" (LxW)

6. 10k potentiometer

7. Push buttons

Specifications

- Contact Type: Momentary Contact
- Switch Size (Approx.): 12x12x7.3mm (LxWxH)
- Button Size (Approx.): 3.8x3.8mm (LxW)
- Number of pins: 4

8. OLED display

Specifications:

- Display: OLED Display
- Font color: White
- Dot matrix size: 128×64
- Display size: 0.96″”
- Controller IC: SSD1306
- Working Voltage: DC 3.3V – 5V
- Power consumption: 0.04W
- Communication way: IIC
- Working Temperature: -30 C ~ +80 C
- Dimension: 29.3 x 27.6mm

9. Power socket

Specifications:

- Type: 3P IEC 320 C14 Power Inlet Sockets Connectors Panel
- Voltage/Current Rating: AC 250V 10A
- Overall Size: 5 x 2.2 x 2.7cm / 2″ x 0.9″ x 1.1″ (LWT)

10. DC 24V fan
11. 24V, 5A transformer.
12. 2 PCBs

7.2 Requirements for Mechanical Parts such as Enclosures and Brackets:

Enclosures:

Material: High-quality ABS plastic for front face to prevent current leakage; durable painted metal for the body for robustness.

Dimensions: Compact design to minimize footprint while accommodating all components.

Heat Dissipation: Incorporation of ventilation and cooling systems such as a DC 24V fan and proper vent placements.

Mounting Points: Specific mounting points for PCB, transformers, switches, and connectors.

Brackets:

Material: Metal brackets with rust-resistant coating.

Size: Custom-fit to ensure secure attachment of the soldering iron and hot air gun handles.

Positioning: Designed to support easy access and ergonomic use of tools

7.3 Criteria for Selecting Materials and Finishes for Mechanical Components:

Material Selection:

ABS Plastic (for front face): Insulative properties to prevent electrical hazards.

Metal (for body and brackets): Strength and durability to withstand industrial usage.

Finish:

Painted Metal Parts: Rust-resistant and aesthetically pleasing finish for durability and professional appearance.

Plastic Parts: Smooth finish to avoid rough edges and enhance user safety.

7.4 Selection Criteria

1. Soldering Iron Handle (Model 907A):
   - Wattage and Voltage: Match the power requirements of the soldering station.
   - Connector Type: Compatibility with the soldering station's design.
   - Heating Element: Ensure efficient heat transfer and durability.
   - Temperature Range: Suitable for various soldering tasks.

2. Hot Air Gun (Model 8586):
   - Thermal Fuse Resistance and Sensor Type: Safety features to prevent overheating.
   - Built-in Fan Voltage: Compatibility with the soldering station's power supply.
   - Plug Diameter and Pin Quantity: Ensure compatibility with connectors and wiring.

3. 5 Pin and 8 Pin Connectors (GX16 XLR Aviation Plug):
   - Durability and Reliability: Needed for stable connections between components.
   - Pin Configuration: Match the requirements of the soldering station's circuitry.

4. On/Off Switches:

- Rating: Ensure switches can handle the voltage and current of the soldering station.
- Illumination: LED illuminated switches provide visual feedback for users.
- 10k Potentiometer:
- Resistance Value: Matches the requirements for controlling fan speed.
- Size and Mounting: Fit within the design of the soldering station's control panel.

5. Push Buttons:
- Contact Type and Size: Momentary contact buttons suitable for user interaction.
- Number of Pins: Compatible with the soldering station's circuitry.

6. OLED Display:
- Display Size and Resolution: Provides clear and readable information to users.
- Controller IC: Compatibility with the microcontroller or interface used in the soldering station.
- Power Consumption: Efficient use of energy resources.

7. Power Socket (3P IEC 320 C14 Power Inlet):
- Type and Rating: Matches the voltage and current requirements of the soldering station.
- Size and Mounting: Fits within the overall design of the soldering station's enclosure.

8. DC 24V Fan:
- Voltage and Current Rating: Matches the power supply of the soldering station.
- Size and Airflow: Provides sufficient cooling for the soldering station's components.

9. 24V, 5A Transformer:
- Voltage and Current Rating: Provides the necessary power for the soldering station's operation.
- Efficiency and Safety: Ensures reliable and safe conversion of grid power to the required voltage.

7.5 Design of Mechanical Sub-Assemblies

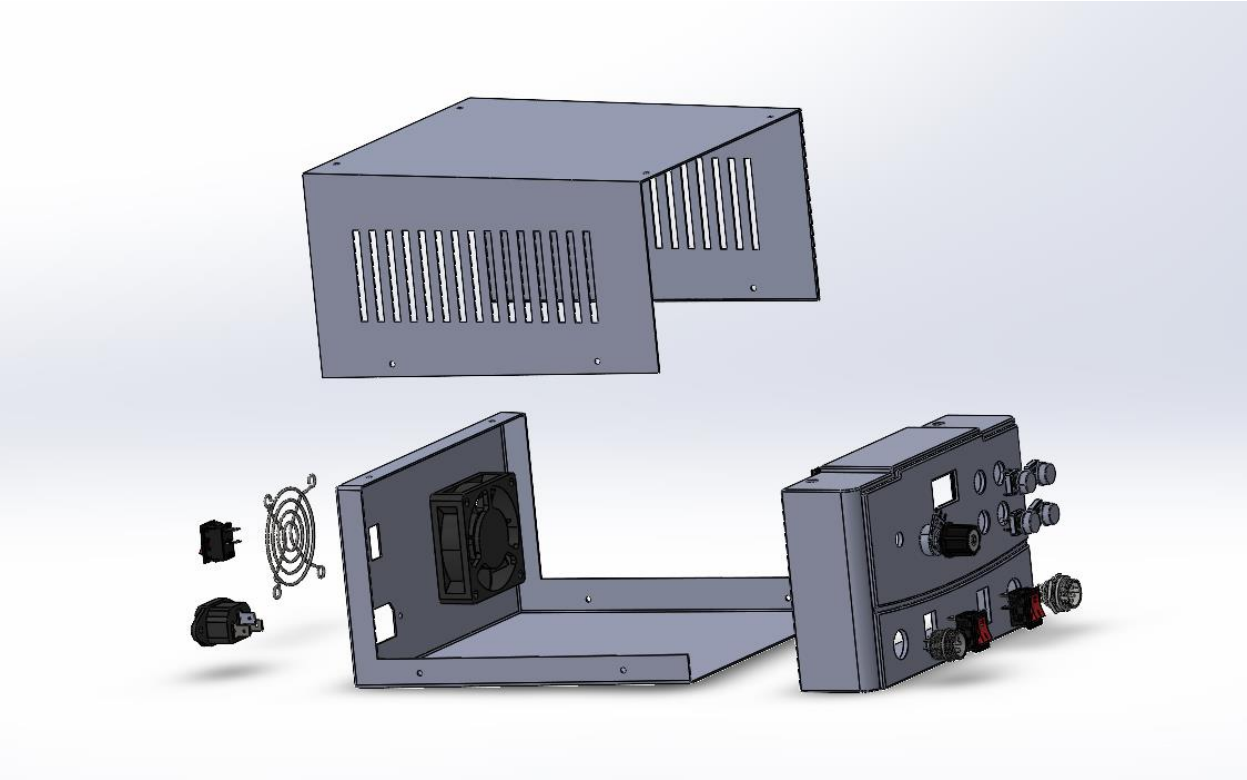


Figure 34- Final Design Exploded view

7.5.1 Front Face of the Enclosure:

- **Connectors and Switches Placement:** The positioning of the 5-pin and 8-pin connectors for the soldering iron and hot air gun, alongside the on/off switches, ensures easy accessibility and control for the user, enhancing convenience during operation.
- **User Interface Components:** Incorporating push buttons, potentiometer, and OLED display on the front face facilitates seamless interaction with the station, allowing users to adjust settings and monitor operation with ease.
- **Wiring:** Careful wiring ensures that all components are connected to the PCB following specified pin diagrams, guaranteeing smooth operation and optimal functionality.

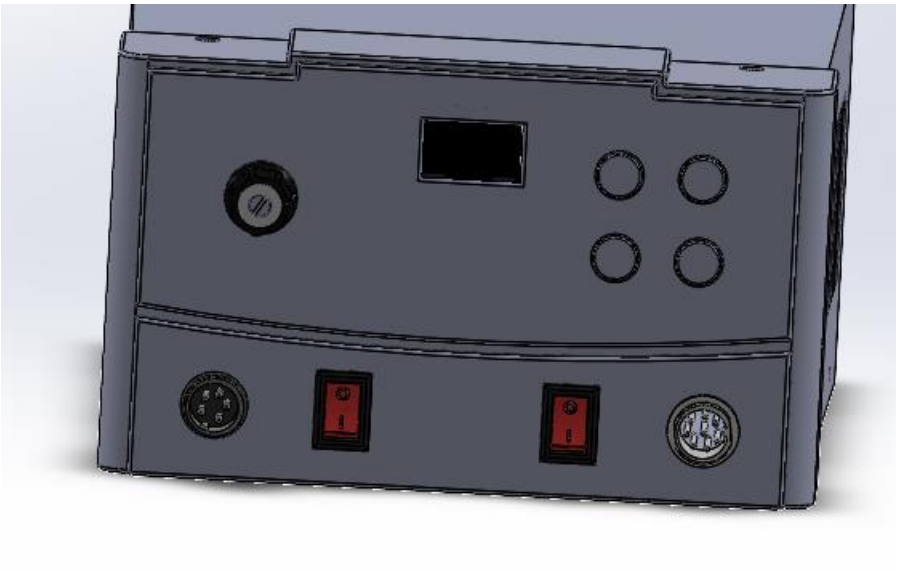


Figure 35- Front Face

7.5.2 Back of the Enclosure:

- **Power Inlet Socket and Switch:** Placing the power inlet socket and on/off switch at the back creates a clean and organized layout, simplifying power connection and control access while maintaining a streamlined appearance.
- **Fan Placement:** Positioning the fan at the back, accompanied by a fan guard, ensures efficient heat dissipation and enhances safety, contributing to the longevity and performance of internal components.

Figure 36- Back view

7.5.3 Design Considerations:

- **Compact Design:** The compact enclosure design optimizes space utilization while accommodating all necessary components, minimizing footprint without compromising functionality.
- **Heat Management:** Integration of a fan mitigates heat buildup within the enclosure, promoting optimal performance and prolonging the lifespan of internal components.
- **Switches for Protection:** Incorporating separate switches for the soldering iron and hot air gun enhances user safety by allowing easy powering off of individual tools when not in use, thereby extending tool lifespan.
- **Standard Power Inlet Socket:** Utilizing a standard power inlet socket ensures compatibility with common power cords, enhancing user accessibility and ease of use.

7.5.4 Assembly Details:

- **Wire Management:** Employing standard color-coded wires, heat shrinks, wire covers, and wire ties for wiring organization simplifies maintenance, reduces clutter, and enhances reliability, ensuring smooth operation over time.
- **Component Placement:** Strategically placing components within the enclosure minimizes wire requirements and isolates PCBs from metal surfaces, mitigating potential interference and electrical issues.
- **Shock Absorption:** Integration of rubber strips beneath the transformer absorbs shock and vibrations, safeguarding internal components from damage during operation and increasing overall durability.
- **Enclosure Material:** Constructing the enclosure face from ABS plastic prevents potential current leakage or shock to users, while cost-effective metal parts, painted for durability and aesthetics, ensure longevity and industrial-grade reliability.
- **Component Attachment:** Secure attachment of components within the enclosure prevents movement or dislodgment during operation, with plastic legs and knobs enhancing ease of handling and assembly.

7.5.5 Standardization and Industrial Design:

- By adhering to industry standards and principles of industrial design, the soldering station exhibits standardized features and ergonomic considerations, ensuring compatibility with user expectations and industry norms.
- The user-friendly wiring design, utilizing standardized components and thoughtful placement, enhances usability and accessibility, aligning with industrial design standards and promoting ease of maintenance and operation.

7.6 Standards Used

In the design and manufacturing of the soldering station, adherence to industry standards was essential to ensure safety, reliability, and overall quality. The following standards were taken into account:

1. Electrical Safety Standards:

- Compliance with general electrical safety standards to prevent risks such as electric shock and fire.
- Use of materials that meet safety ratings for flammability.

2. Electromagnetic Compatibility (EMC):

- Ensuring the soldering station does not emit electromagnetic interference that could affect other devices.
- Ensuring the device is immune to interference from other electronic equipment.

3. RoHS Compliance:

- Ensuring all components are free from hazardous substances like lead, mercury, and cadmium.

4. WEEE Compliance:

- Designing the product to facilitate recycling and proper disposal of electronic waste.

5. Mechanical Design Standards:

- Adhering to quality management principles to ensure consistent product quality.
- Following guidelines for tolerances and precision in manufacturing.

6. Thermal Management Standards:

- Ensuring the thermal management system, including fans and heat sinks, operates efficiently.
- Implementing guidelines for thermal management in PCB design.

7. Ergonomic Standards:

- Designing the soldering station to be user-friendly and ergonomic.
- Considering human factors to ensure ease of use and interaction.

8. Component and Connector Standards:

- Using connectors and components that meet safety and compatibility requirements.
- Ensuring reliability and durability of components under various conditions.

7.7 Reasons for selection and non-selection and changes

1. OLED Display

Selection Reasons:

- High Resolution: The OLED display provides a clear and sharp display, which is essential for showing detailed information and menus.
- Ease of Coding: The SSD1306 controller IC is well-supported with libraries, making it straightforward to integrate into the project.
- User-Friendly: The display can show menus and detailed information, enhancing user interaction and usability.

2. Power Supply
   - **Initial Plan:** Initially, the design planned to provide power using 24V DC directly.
   - **Change:** The power supply method was changed to using grid power (230V AC) and then converting it to the required voltage internally.
   - **Reason for Change:** This change simplifies the setup for the user, as they can plug the station directly into a standard power outlet without needing a separate DC power source.
3. Heat Management

Fan Integration:

- **Reason:** To manage heat generated by the internal components, a fan was included to ensure effective cooling, which helps in maintaining the performance and longevity of the station.

4. Connectors

Standard Connectors:

- **Power Sockets:** Using standardized power sockets ensures compatibility with common power cords and enhances safety.
- **5-pin and 8-pin Aviation Connectors:** These connectors were chosen for their durability, reliability, and ability to maintain stable connections under various conditions.

5. Switches

Separate Switches:

- **For Soldering Iron and Hot Air Gun**: Including separate switches allows the user to turn off each tool independently, reducing unnecessary power dissipation and enhancing safety.
- **Overall Power Switch:** An additional main power switch was included to easily power off the entire station when not in use.

6. Potentiometer

Use for Fan Speed Control:

- **Reason:** A potentiometer was used to allow the user to adjust the fan speed, enabling customized cooling based on the operational requirements and helping to reduce noise and power consumption when full cooling is not needed.

7. Tactile Push Buttons

Four Push Buttons:

- **Reason:** These buttons are used to navigate the menu on the OLED display, providing a simple and intuitive interface for users to interact with the station's functions.

8. Plastic Knobs as Stand Legs

Selection Reason:

- **Stability:** Plastic knobs provide stable support for the enclosure.
- **Electrical Isolation**: They prevent electrical conductivity, ensuring user safety.
- **Durability:** They are durable and can withstand repeated use.

9. Enclosure Design
    - **Initial Design:** The initial design of the enclosure was in two parts: a lid and a boxed body.
    - **Change:** The design was modified to a three-part enclosure (lid, and the body split into two parts).
    - **Reason for Change:** This change was made to simplify assembly and adhere to design standards, making the enclosure easier to handle and assemble.

10. Hot Air Gun and Soldering Iron Connections

Use of Male and Female Connectors:

- **Reason:** Instead of connecting the tools directly, using connectors allows for easy replacement and maintenance. Users can replace just the soldering iron or hot air gun if they fail, without needing to buy a whole new station. This flexibility enhances the product's usability and longevity.

7.8 Rough Sketches to Final Design

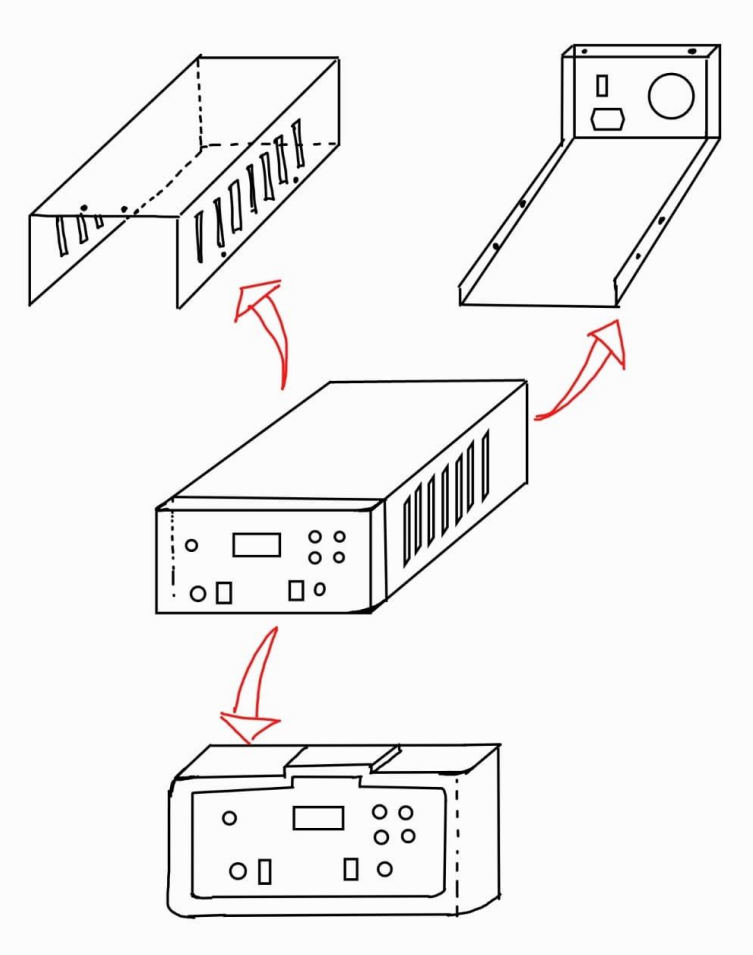


Figure 37-Sketches

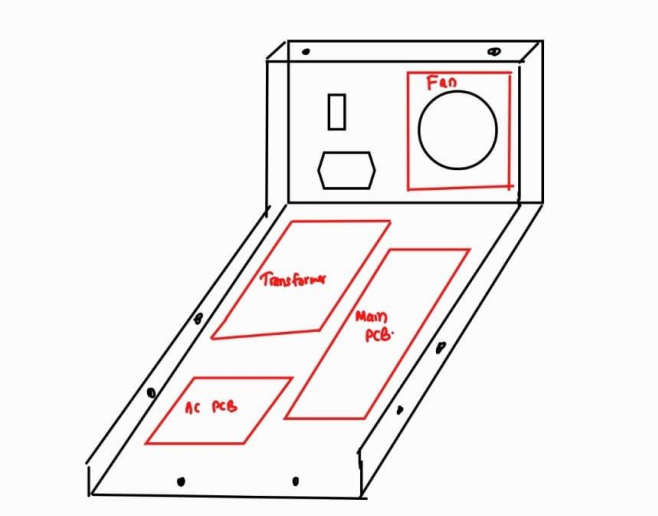
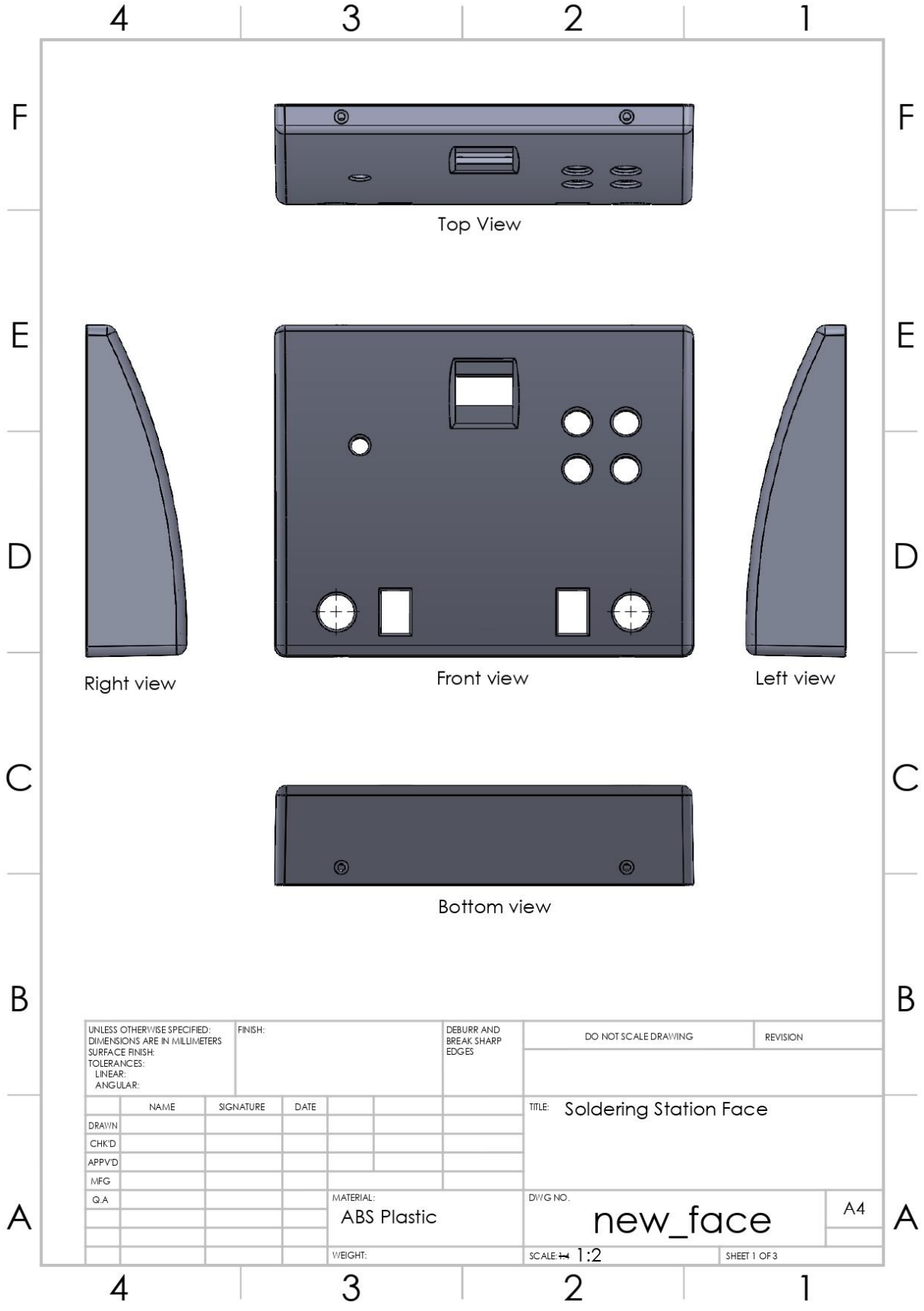
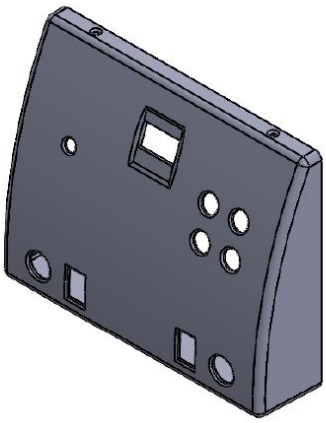
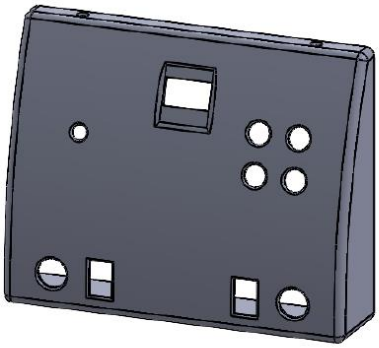


Figure 38- Component placement

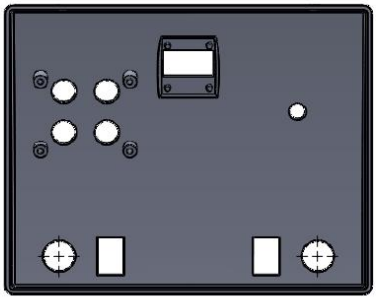

Top View
Right view
Front view
Left view
Bottom view
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
SURFACE FINISH:
TOLERANCES:
LINEAR:
ANGULAR:
FINISH:
DEBURR AND BREAK SHARP EDGES
DO NOT SCALE DRAWING
REVISION
NAME
SIGNATURE
DATE
DRAWN
CHK'D
APPV'D
MFG
Q.A
TITLE: Soldering Station Face
MATERIAL: ABS Plastic
DWG NO. new_face
A4
WEIGHT:
SCALE: 1:2
SHEET 1 OF 3



Isometric View



Dimetric View

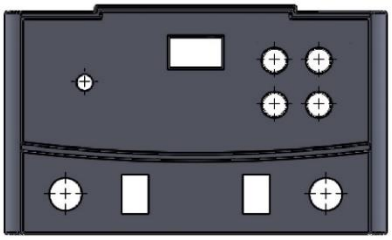


Back view

UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:		FINISH:			DEBURR AND BREAK SHARP EDGES		DO NOT SCALE DRAWING	REVISION
	NAME	SIGNATURE	DATE				TITLE: Soldering Station Face	
DRAWN								
CHK'D								
APPV'D								
MFG								
Q.A				MATERIAL: ABS plastic			DWG NO. new_face	A4
				WEIGHT:			SCALE: ~~1:4~~ 1:3	SHEET 2 OF 3

Top

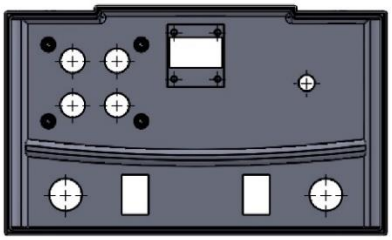
Left



Front

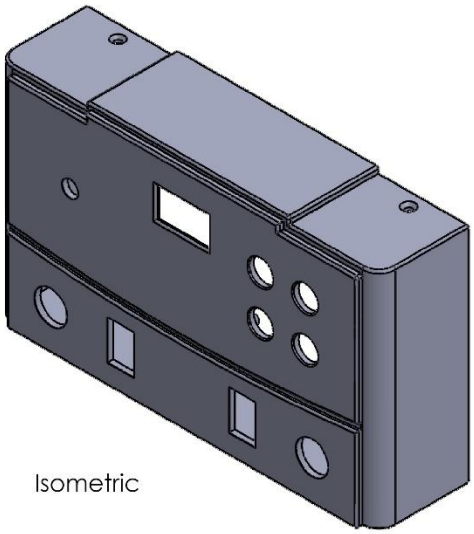
Right

Bottom

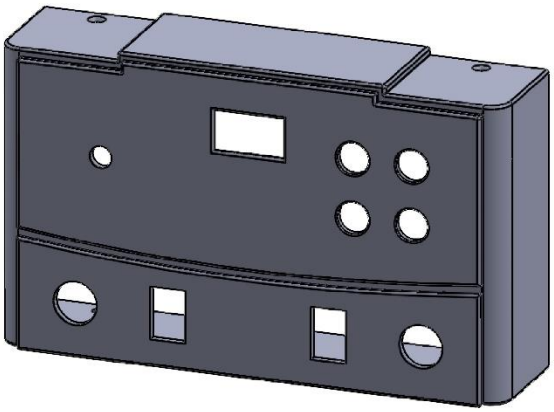


Back

UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:		FINISH:			DEBURR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING	REVISION
	NAME	SIGNATURE	DATE			TITLE: Face	
DRAWN							
CHK'D							
APPV'D							
MFG							
Q.A				MATERIAL: ABS		DWG NO. metal final face	A4
				WEIGHT:		SCALE: 1:3	SHEET 1 OF 2



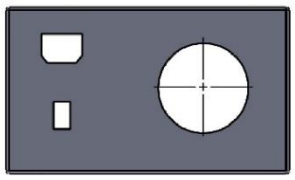
Isometric



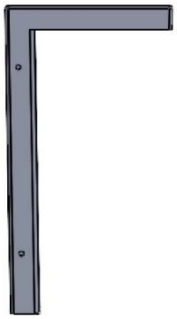
Dimetric

UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:		FINISH:			DEBURR AND BREAK SHARP EDGES		DO NOT SCALE DRAWING	REVISION
	NAME	SIGNATURE	DATE				TITLE: Face	
DRAWN								
CHK'D								
APPV'D								
MFG								
Q.A				MATERIAL: ABS			DWG NO. metal final face	A4
				WEIGHT:			SCALE: ~~1:4~~ 1:2	SHEET 2 OF 2

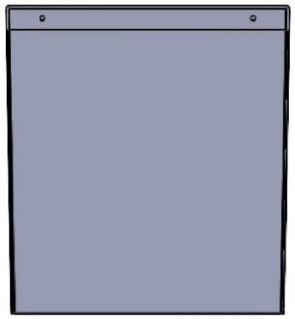
Back



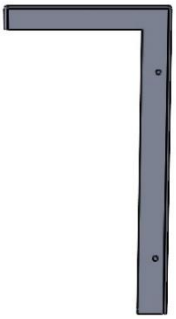
Right



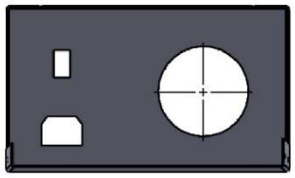
Top



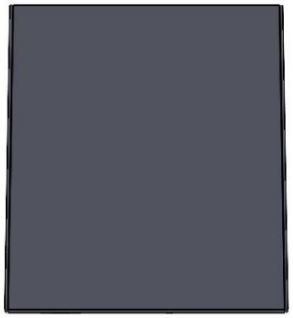
Left



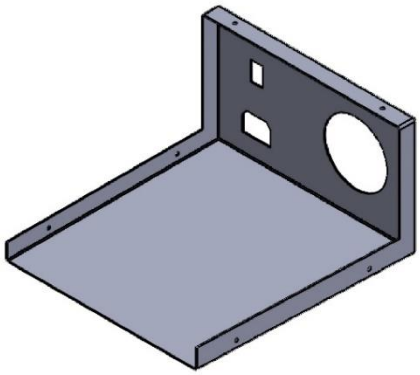
Front



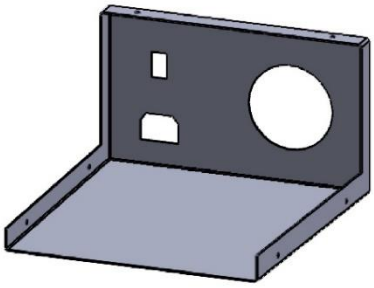
Bottom



UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:		FINISH:			DEBURR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING	REVISION
	NAME	SIGNATURE	DATE			TITLE: Body part 2	
DRAWN							
CHK'D							
APPV'D							
MFG							
Q.A				MATERIAL: Metal		DWG NO. box2_all	A4
				WEIGHT:		SCALE:1:4	SHEET 1 OF 2

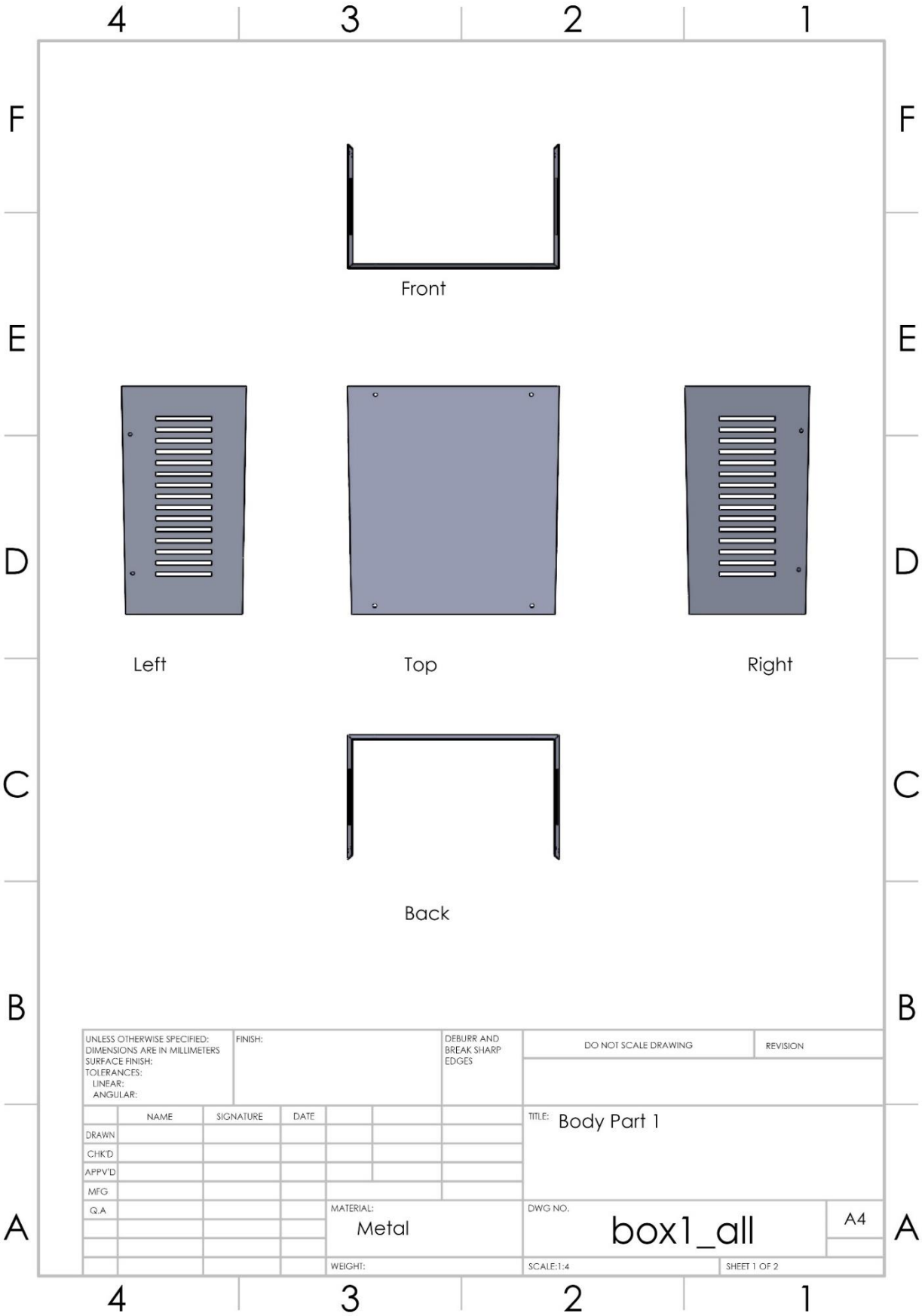


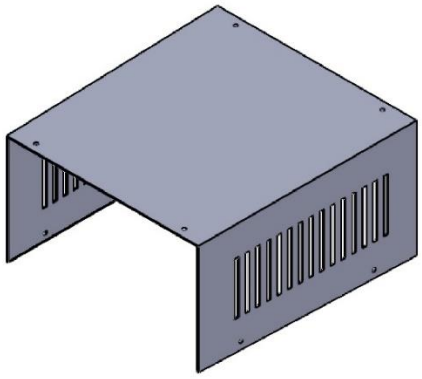
Isometric



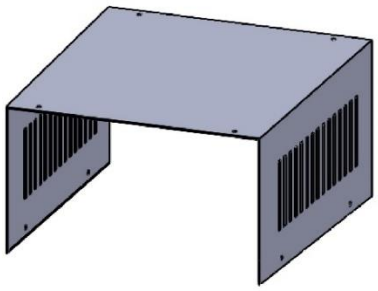
Dlametric

UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:		FINISH:			DEBURR AND BREAK SHARP EDGES		DO NOT SCALE DRAWING	REVISION
	NAME	SIGNATURE	DATE				TITLE: Body part 2	
DRAWN								
CHK'D								
APPV'D								
MFG								
Q.A				MATERIAL: Metal			DWG NO. box2_all	A4
				WEIGHT:			SCALE:1:4	SHEET 2 OF 2


4
3
2
1
F
E
D
C
B
A
Front
Left
Top
Right
Back
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
SURFACE FINISH:
TOLERANCES:
LINEAR:
ANGULAR:
FINISH:
DEBURR AND BREAK SHARP EDGES
DO NOT SCALE DRAWING
REVISION
NAME
SIGNATURE
DATE
DRAWN
CHK'D
APPV'D
MFG
Q.A
TITLE: Body Part 1
MATERIAL: Metal
DWG NO. box1_all
A4
WEIGHT:
SCALE:1:4
SHEET 1 OF 2



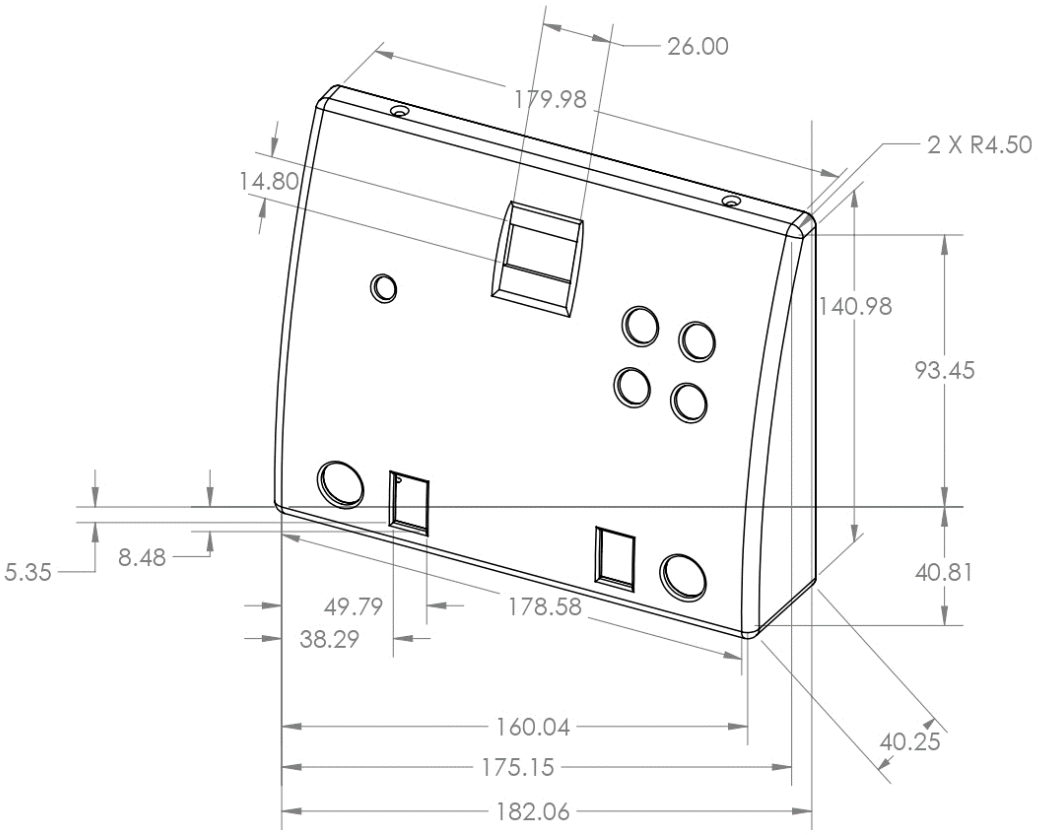
Isometric



Dimetric

UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:		FINISH:			DEBURR AND BREAK SHARP EDGES		DO NOT SCALE DRAWING	REVISION
	NAME	SIGNATURE	DATE				TITLE: Body part 1	
DRAWN								
CHK'D								
APPV'D								
MFG								
Q.A				MATERIAL: metal			DWG NO. box1_all	A4
				WEIGHT:			SCALE:1:4	SHEET 2 OF 2

Dimentions



UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:		FINISH:		DEBURR AND BREAK SHARP EDGES		DO NOT SCALE DRAWING		REVISION
	NAME	SIGNATURE	DATE			TITLE: Soldering station face		
DRAWN								
CHK'D								
APPV'D								
MFG								
Q.A				MATERIAL: ABS plastic		DWG NO. new_face		A4
				WEIGHT:		SCALE: ~~1:4~~ 1:2	SHEET 3 OF 3	

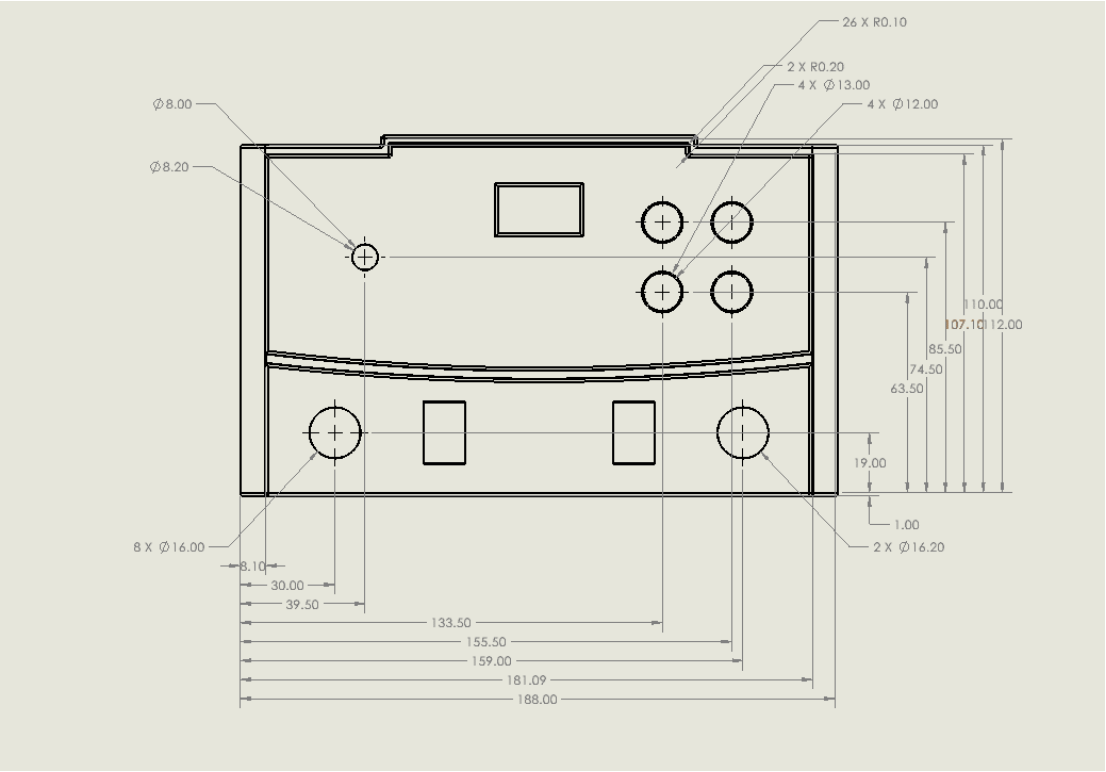


Figure 39

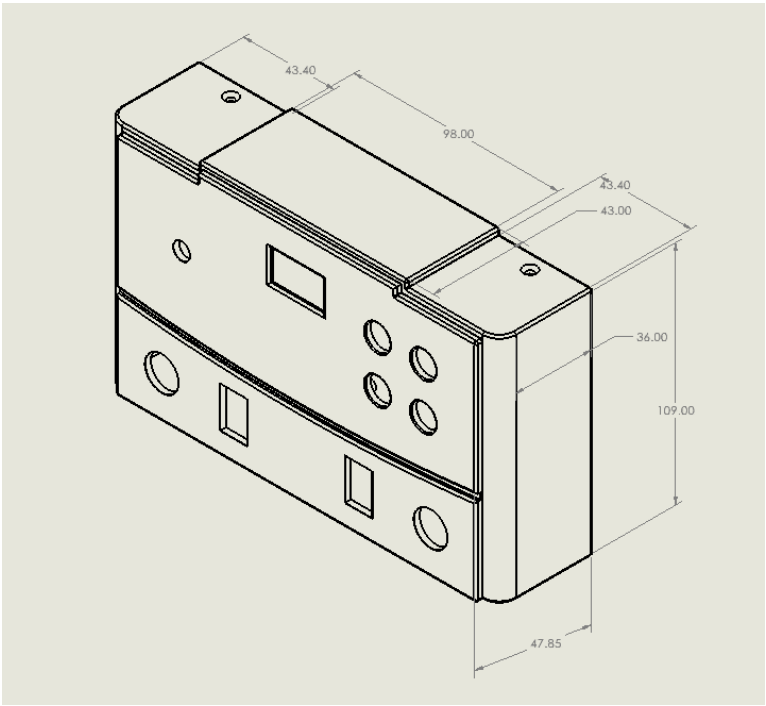


Figure 40- Face from side

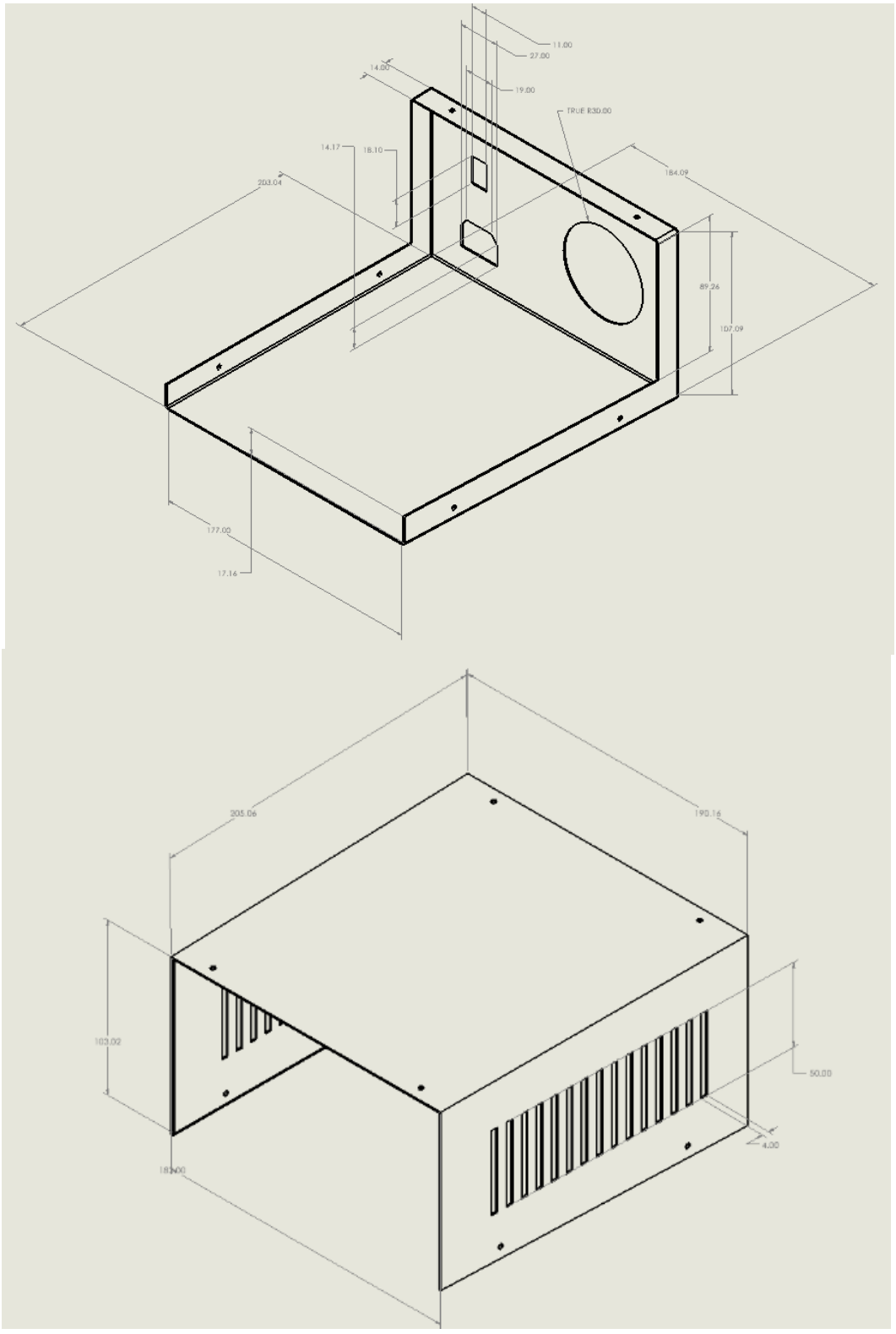


Figure 41- Body part 2

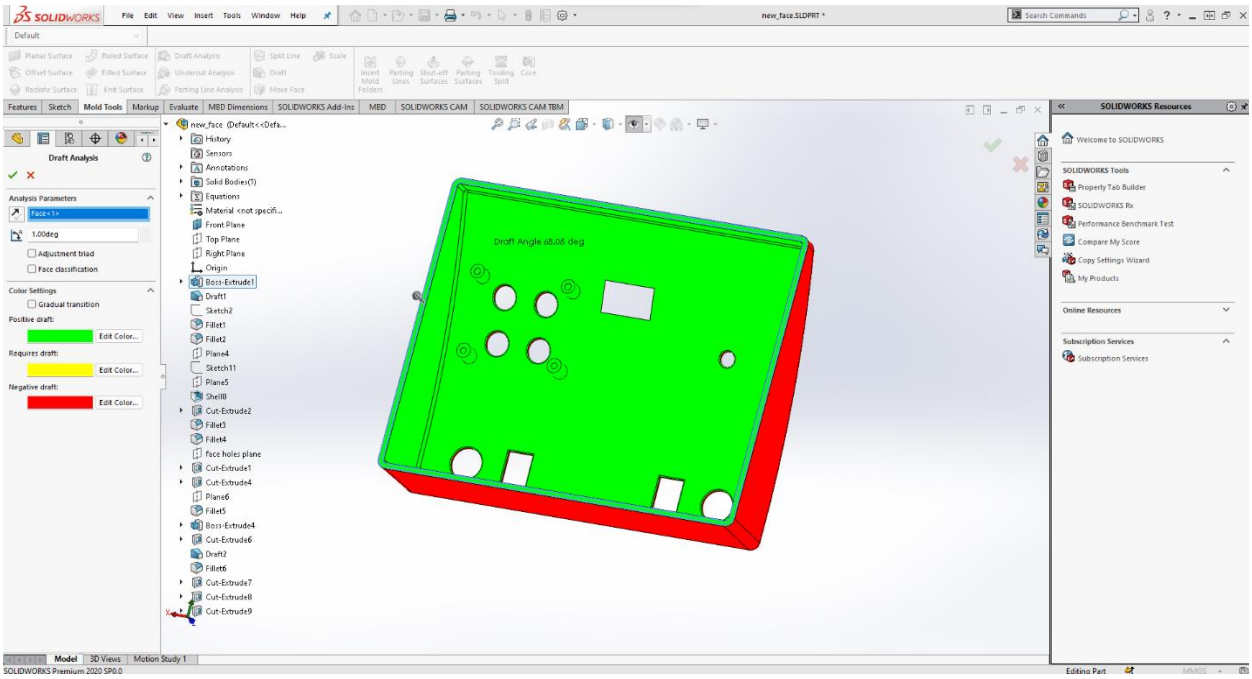


Figure 42 - draft analysis of face

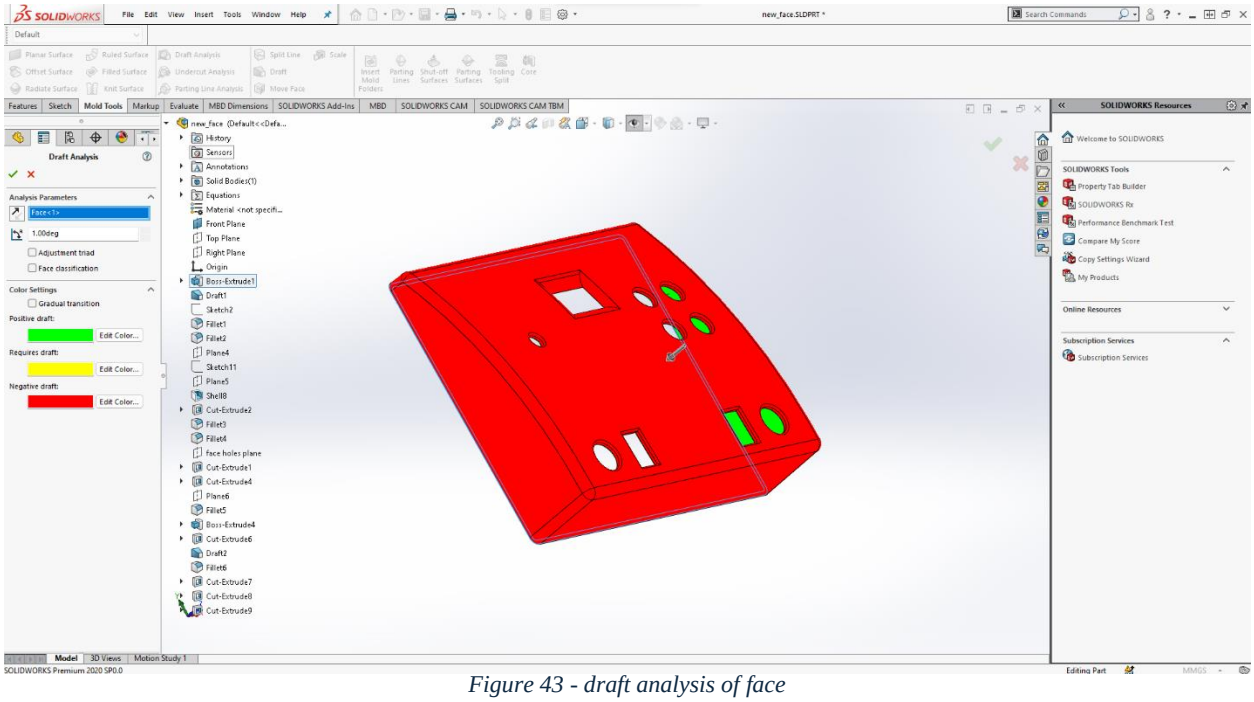


Figure 43 - draft analysis of face

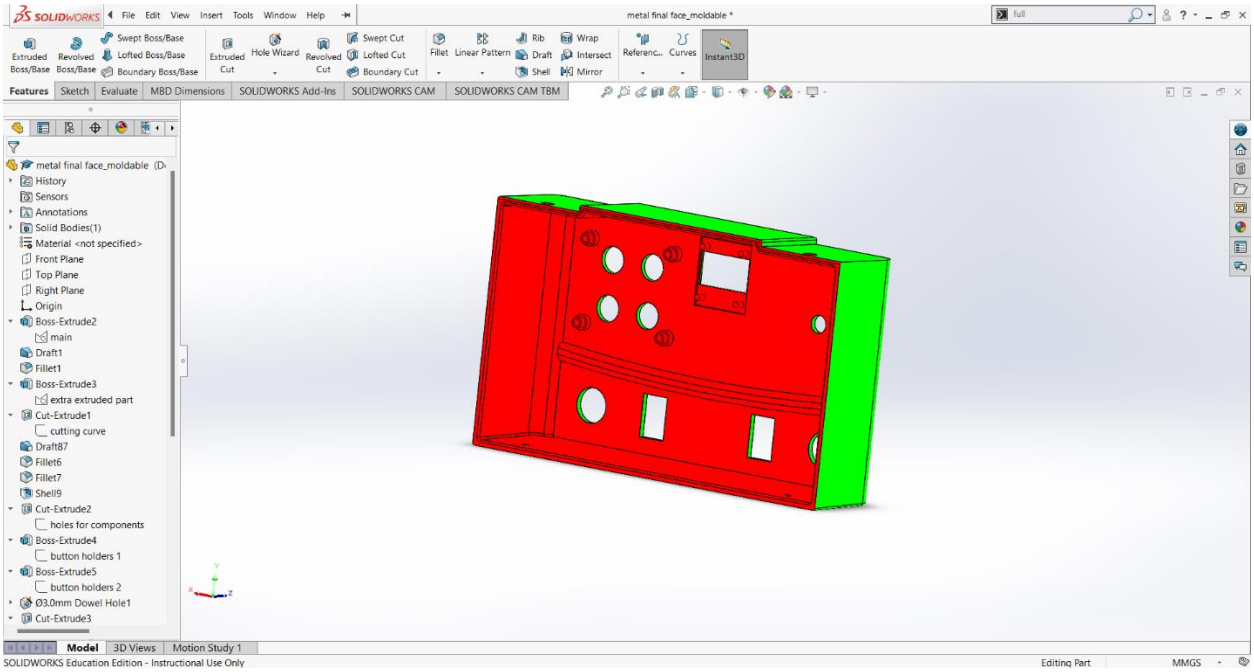


Figure 44- Draft Analysis of face

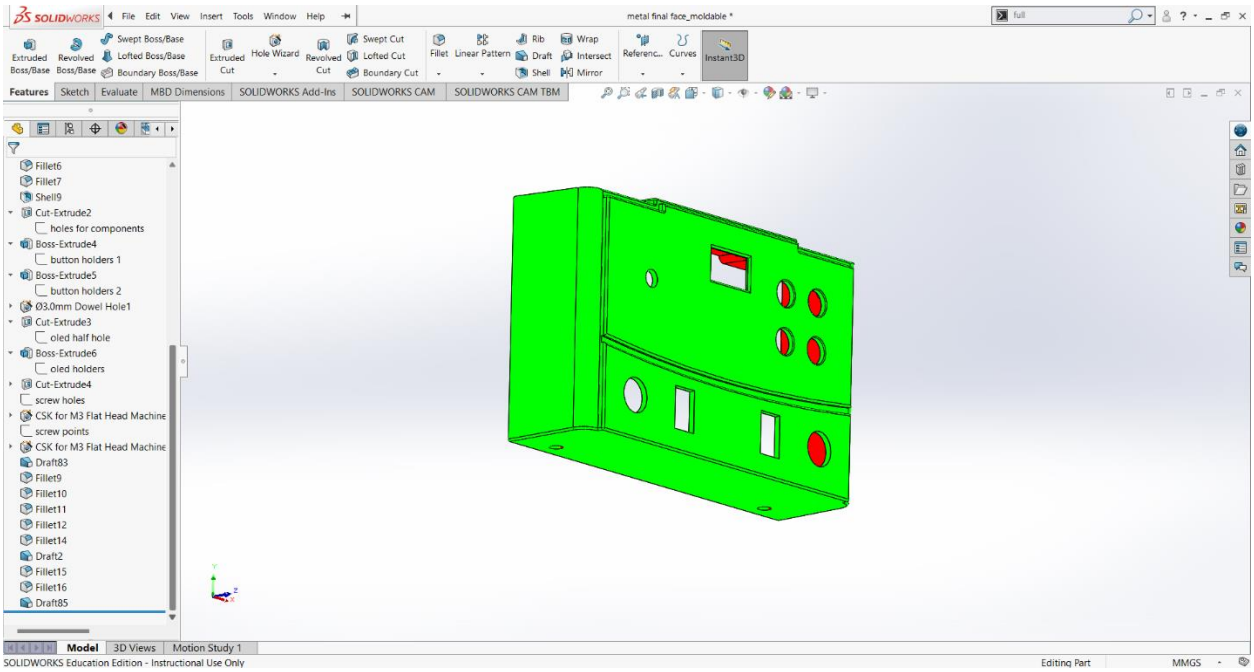


Figure 45- Draft Analysis of face

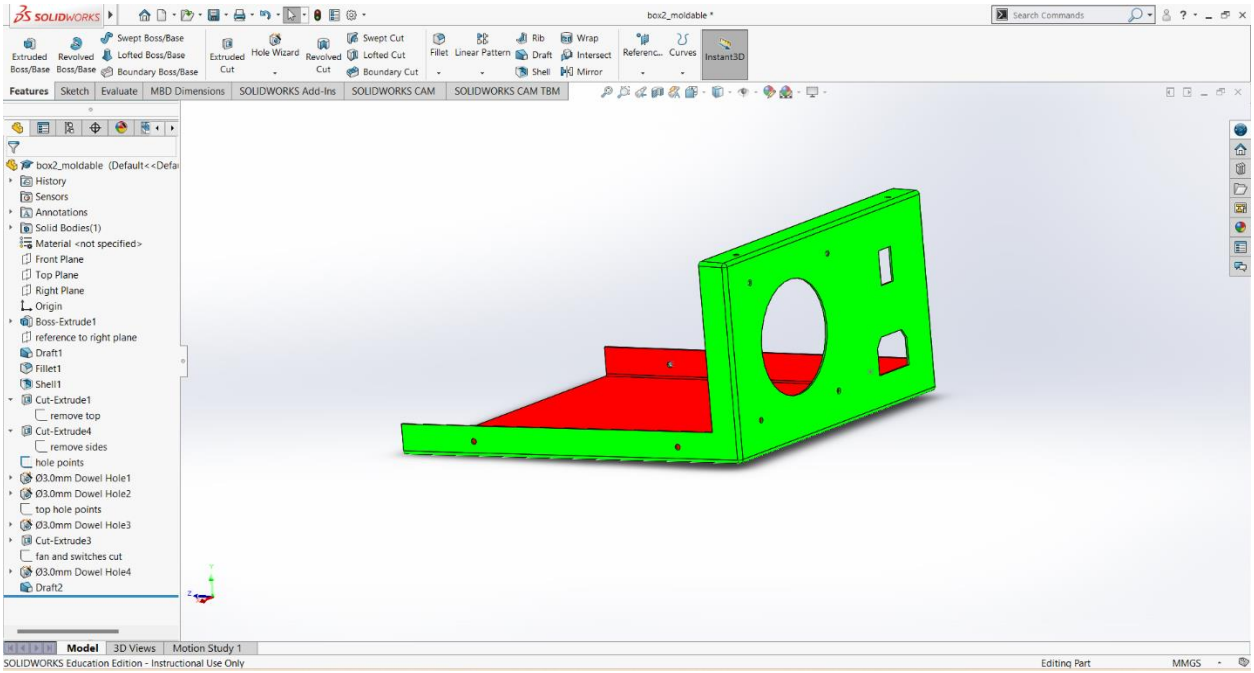


Figure 46- Draft analysis of body part 1

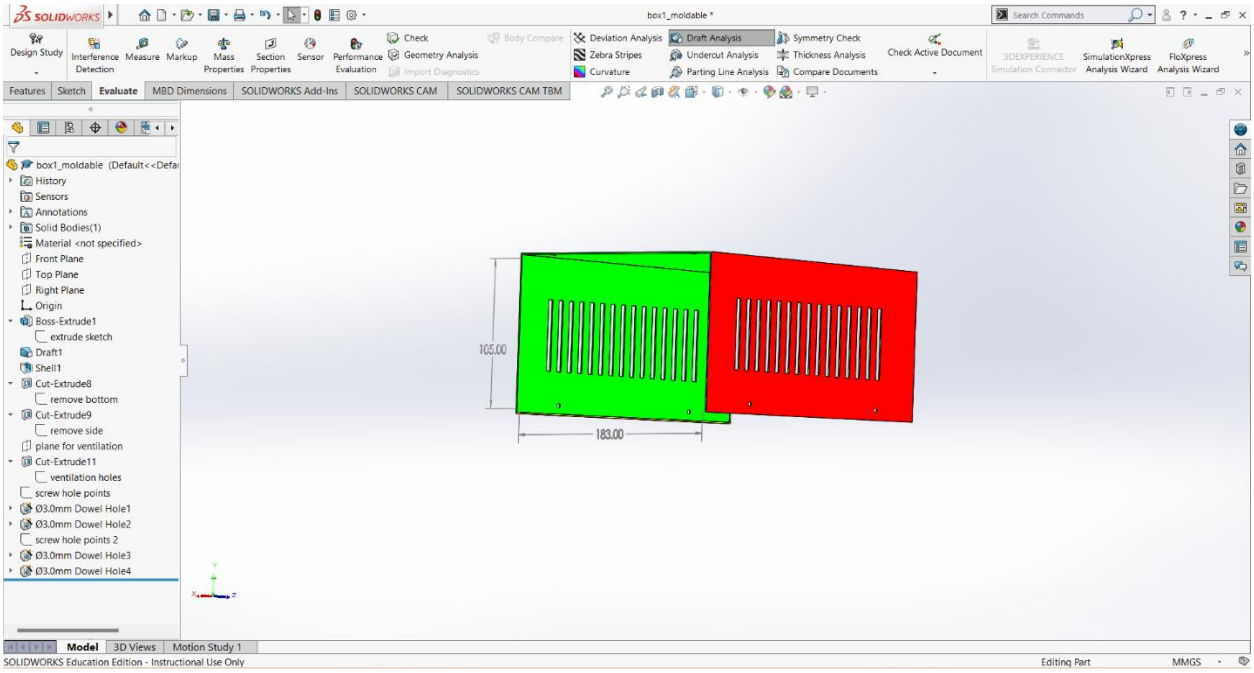


Figure 47- Draft analysis of body part 2

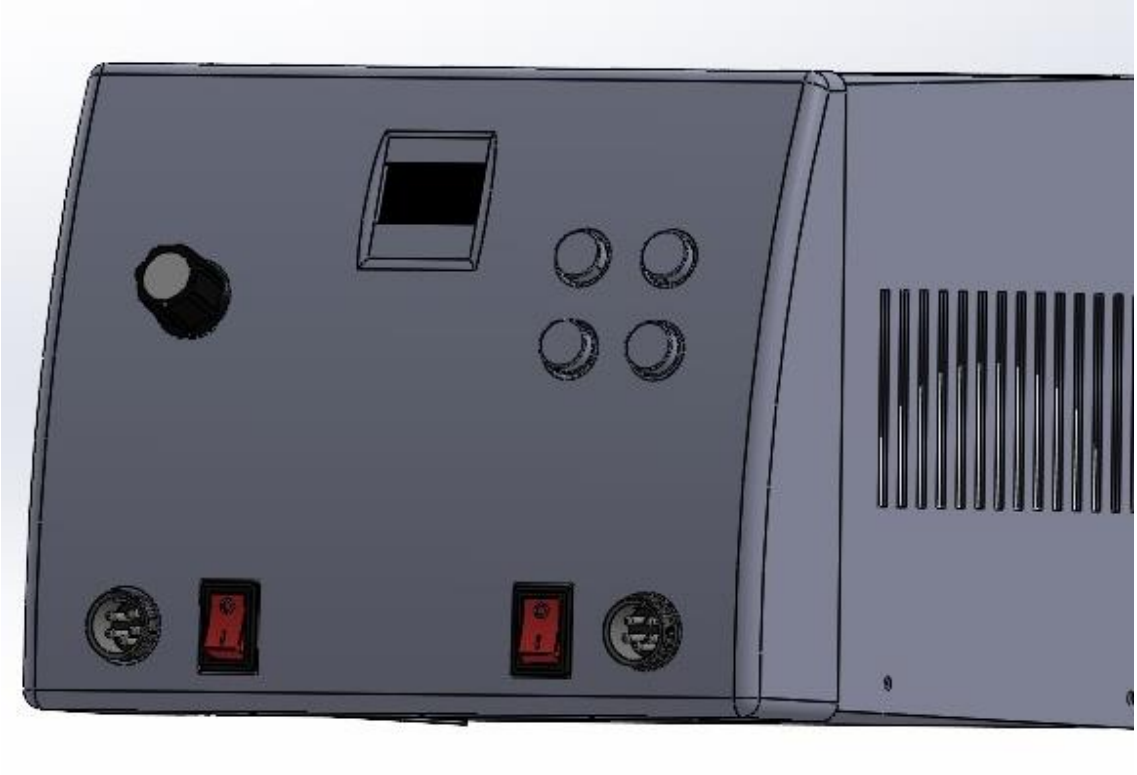


Figure 48 – final side view

Figure 49 - final front view

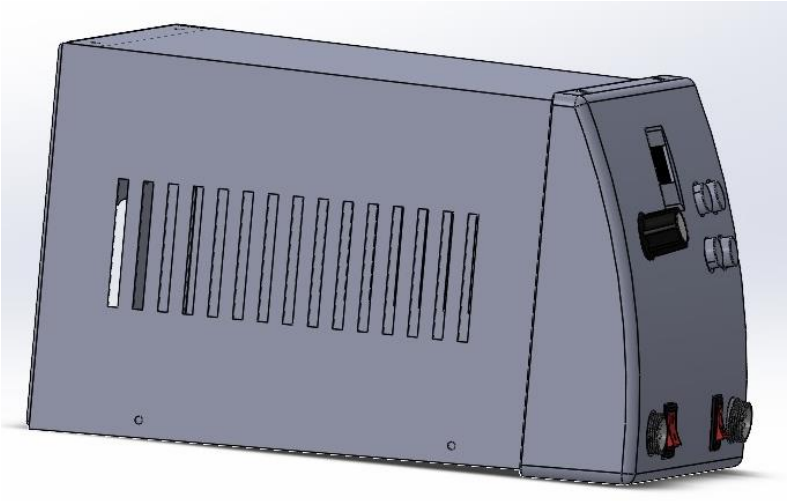


Figure 50-final side view

Figure 51- Final design

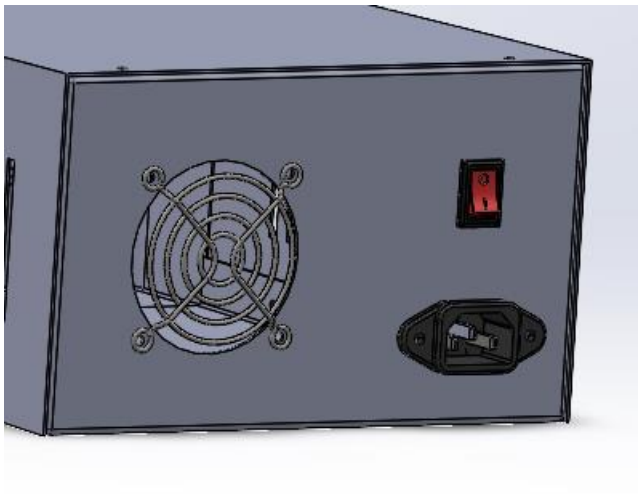


Figure 52- Back view

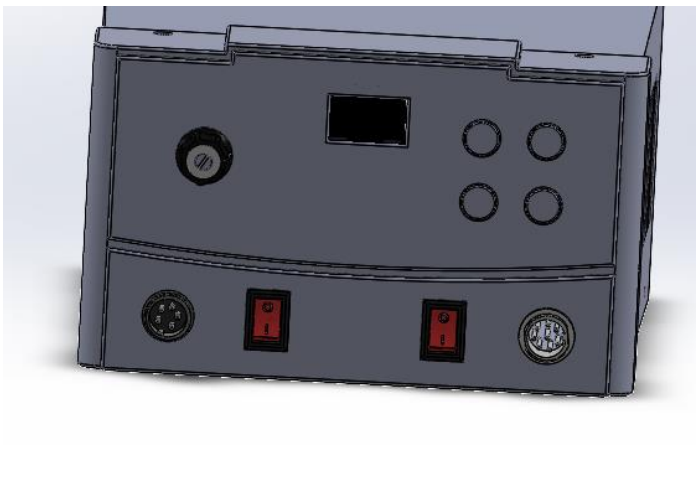


Figure 53- Front face

7.9 Mold design

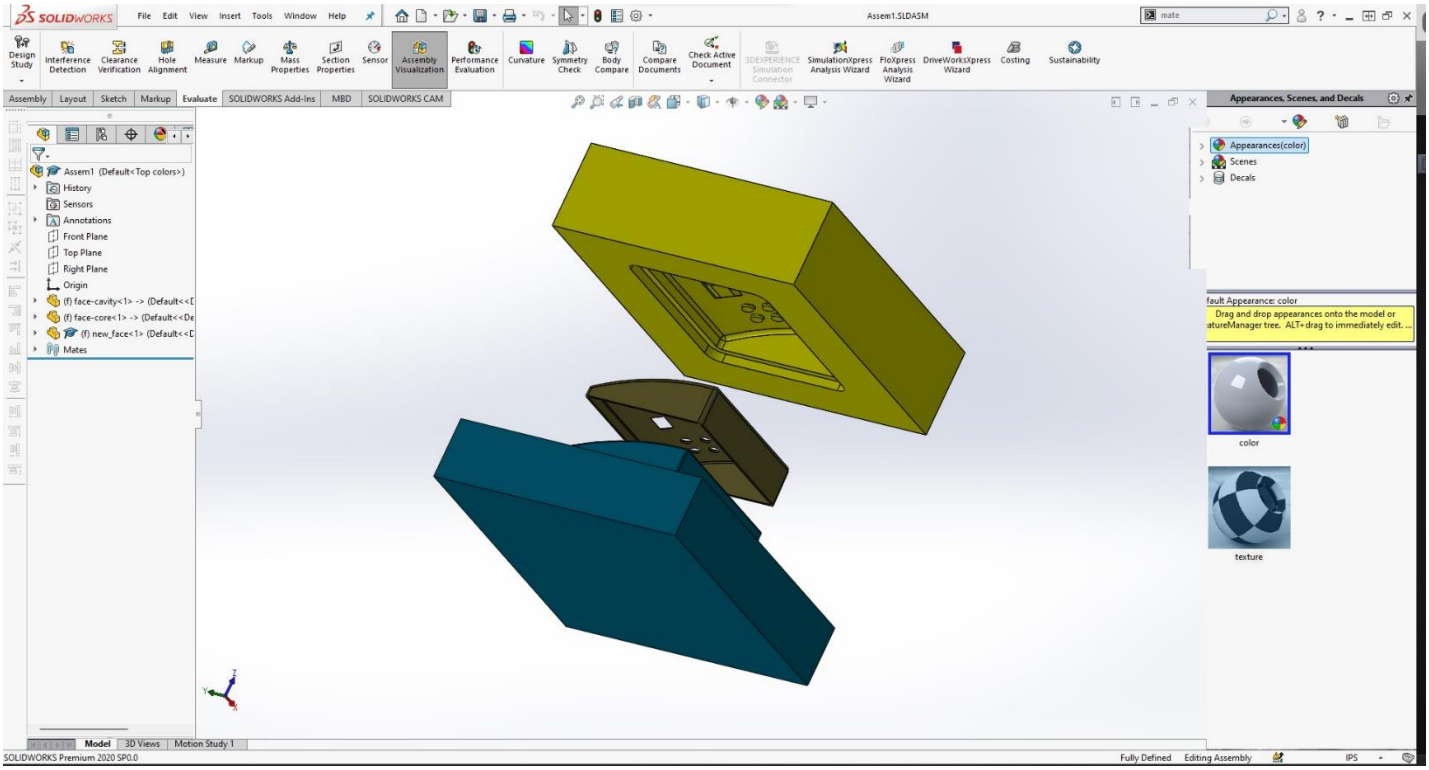


Figure 54 - Mold design exploded view

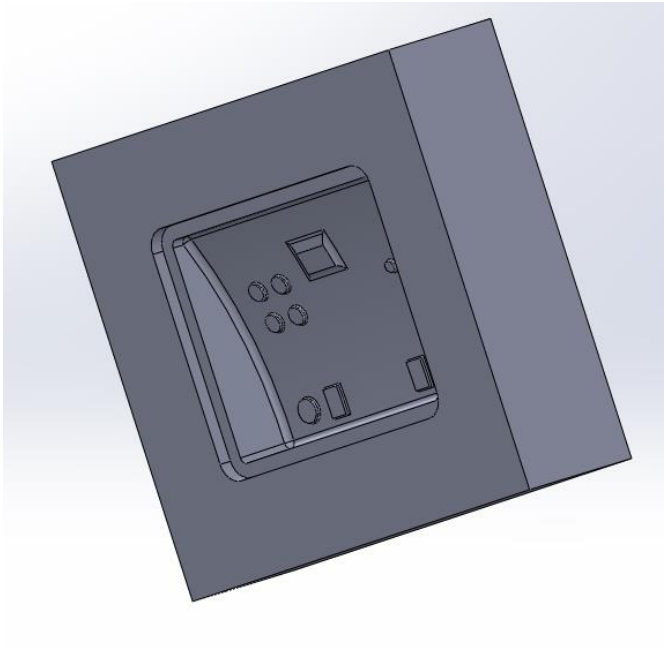


Figure 55– Cavity

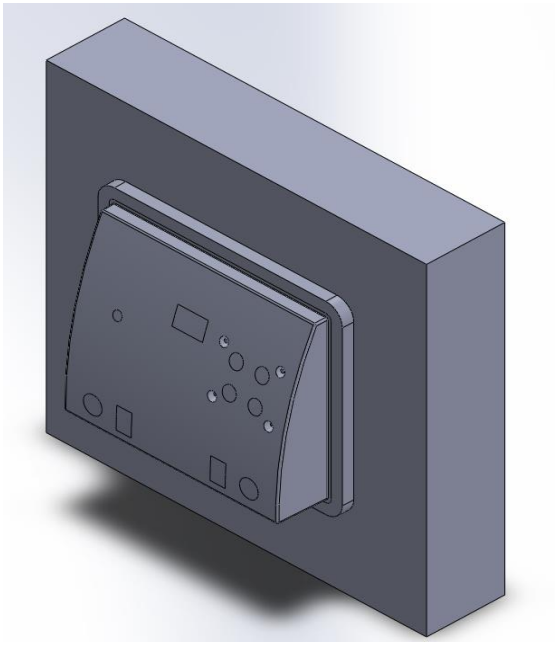


Figure 56- Core

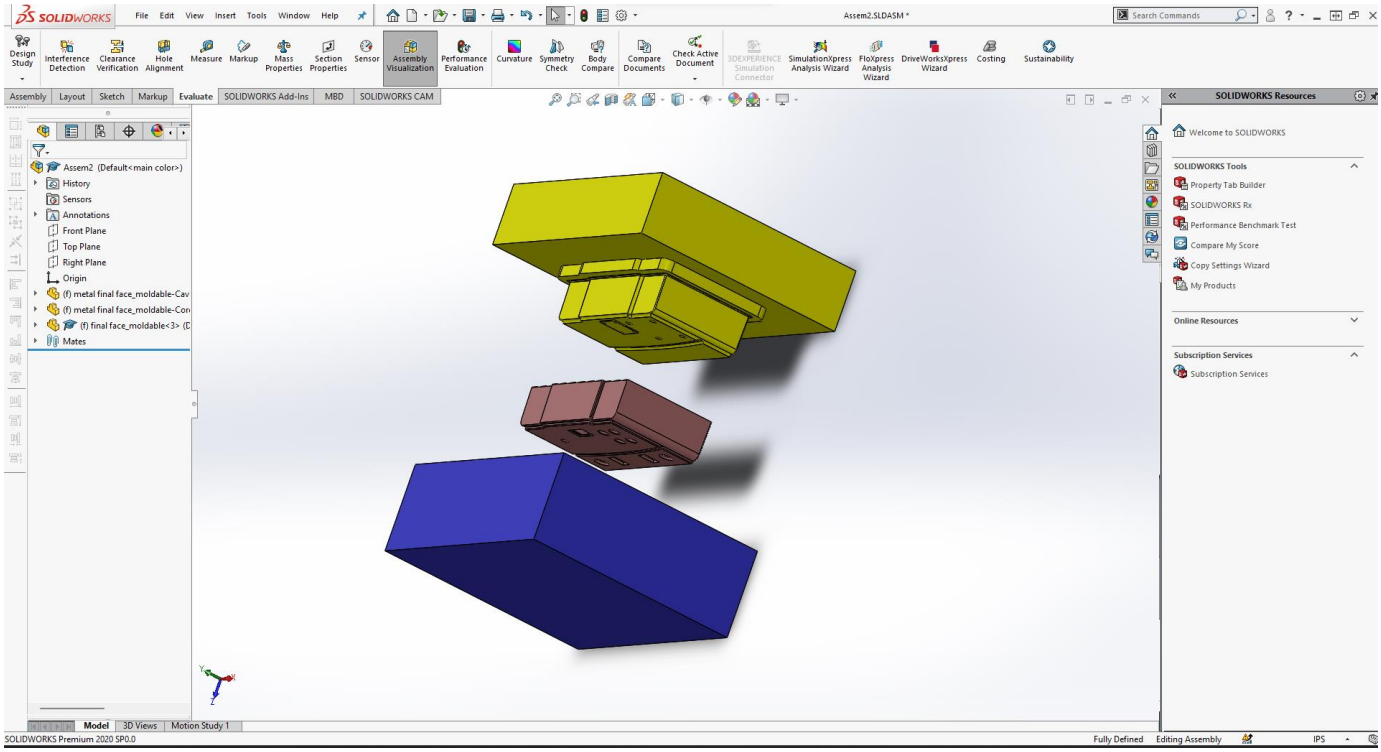


Figure 57-Mold design exploded view

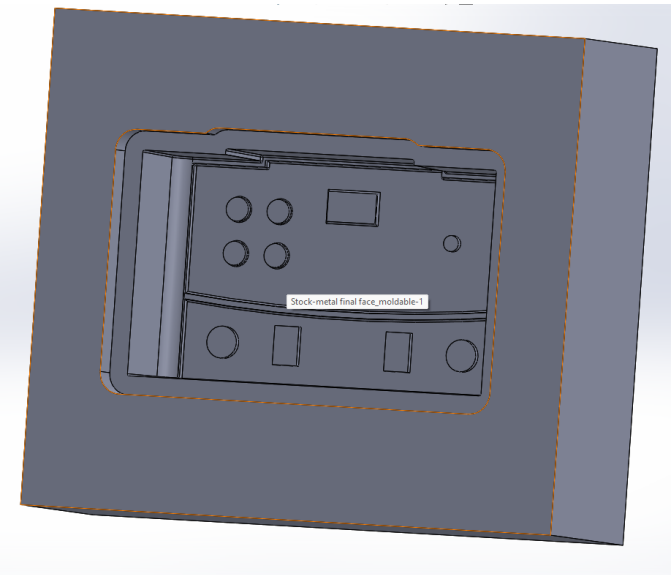


Figure 58-Cavity

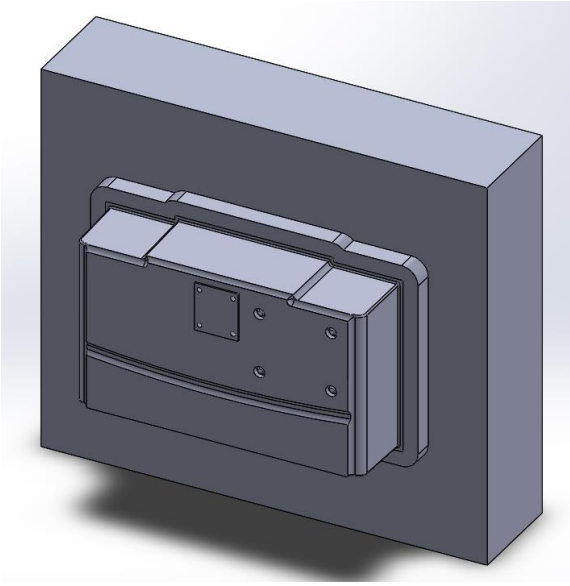


Figure 59-Core

7.9 PCB Dimensions

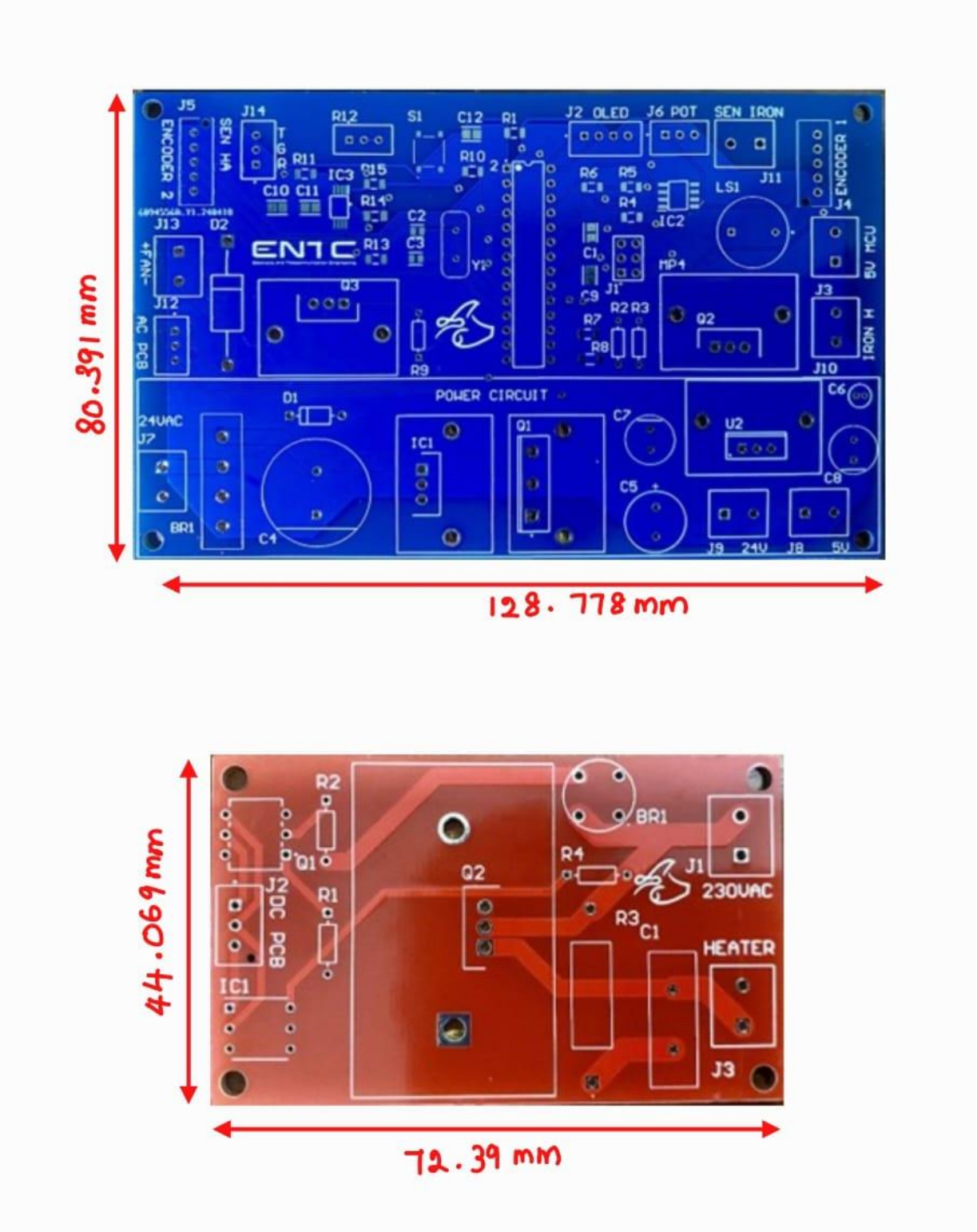


Figure 60- PCB dimensions

7.10 Final design outcome

Figure 61- Final design front view

Figure 62- Final design back view

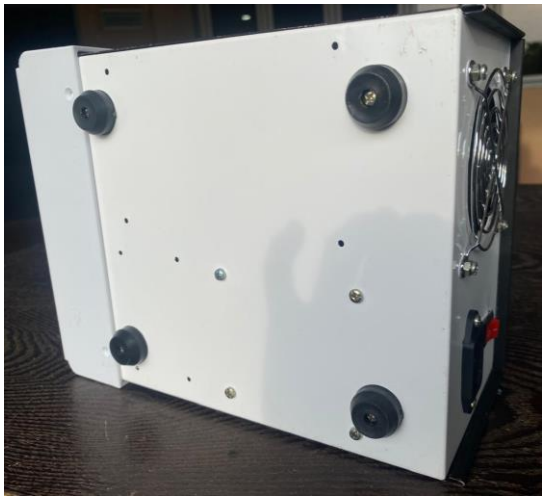


Figure 63- Final design Bottom view

Figure 64- Final design side view

Figure 65- Final design outcome

8.0 Photograph showing the system integration

Use heat sleeves, expandable braided PET cable sleeves, and wire ties for the wiring. Although the PCB testing is not yet complete, proceed with connecting the other main components of the system.

8.1 Photo collection of integration step-by-step

Figure 66- Soldering

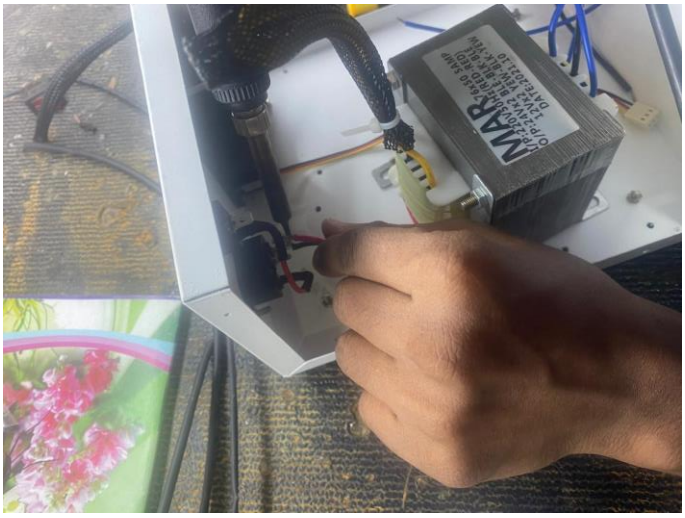


Figure 67-connecting transformer

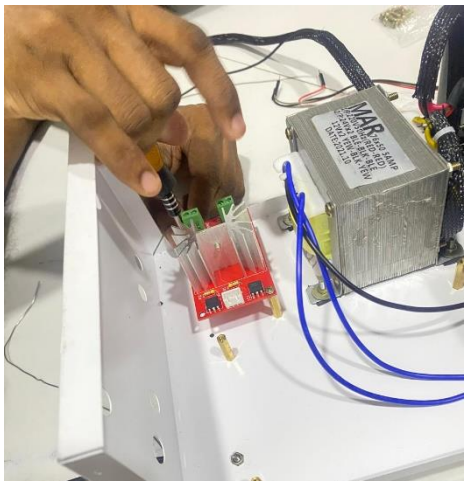


Figure 68 - mounting pcb

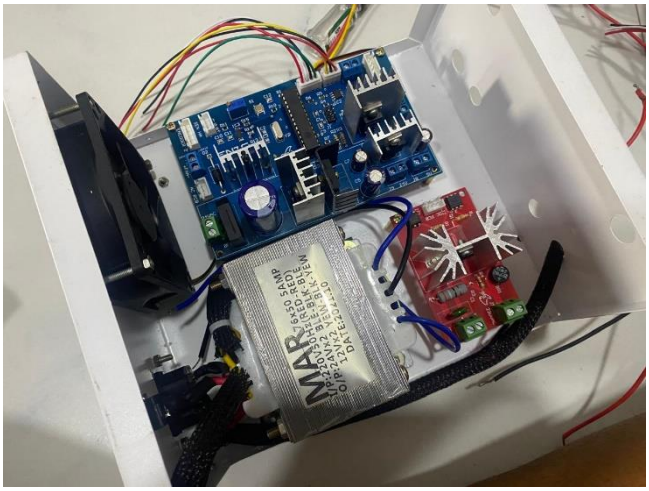


Figure 69 - placing

Figure 70 – transformer placing and wiring

Figure71 - integration 1

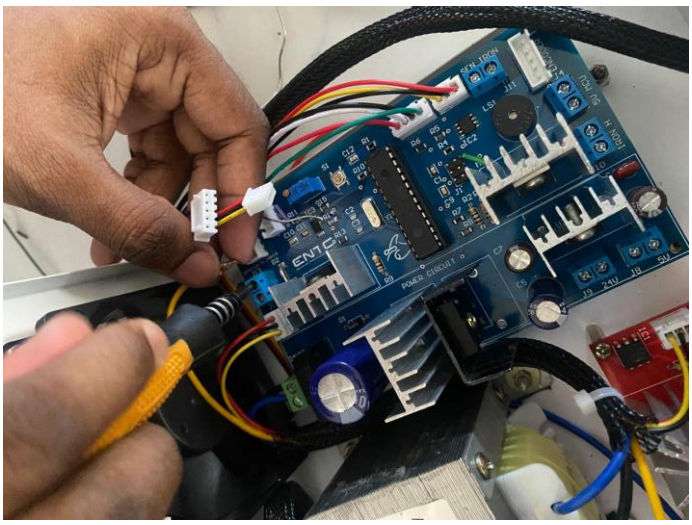


Figure 72 - integration 2

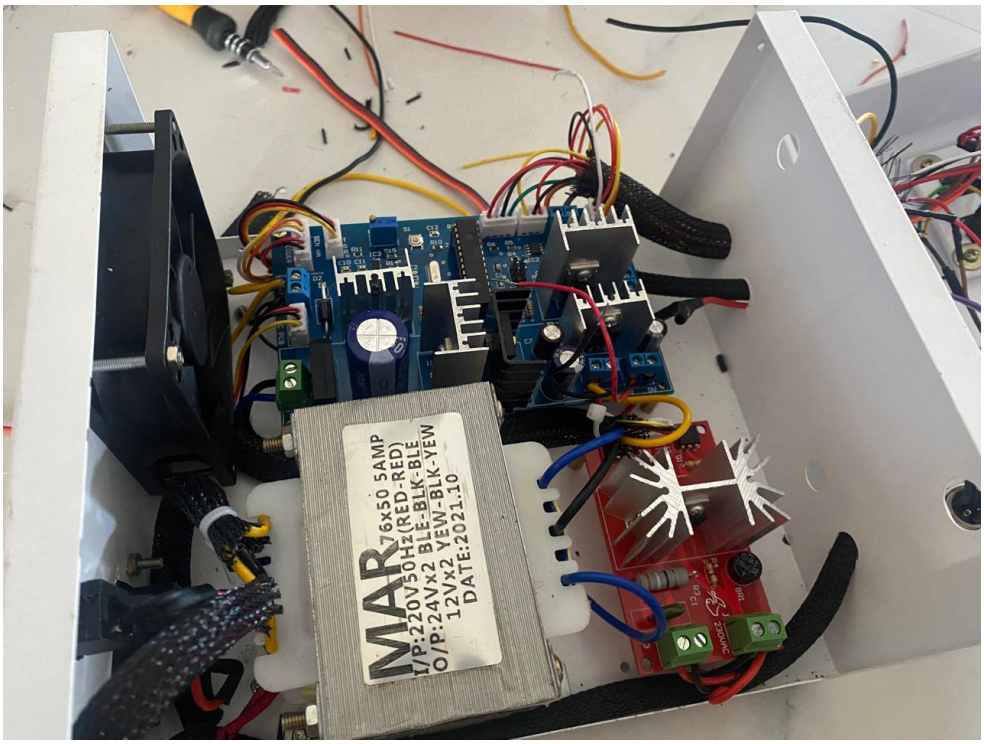


Figure 73 - integration 3

8.2 Outcome

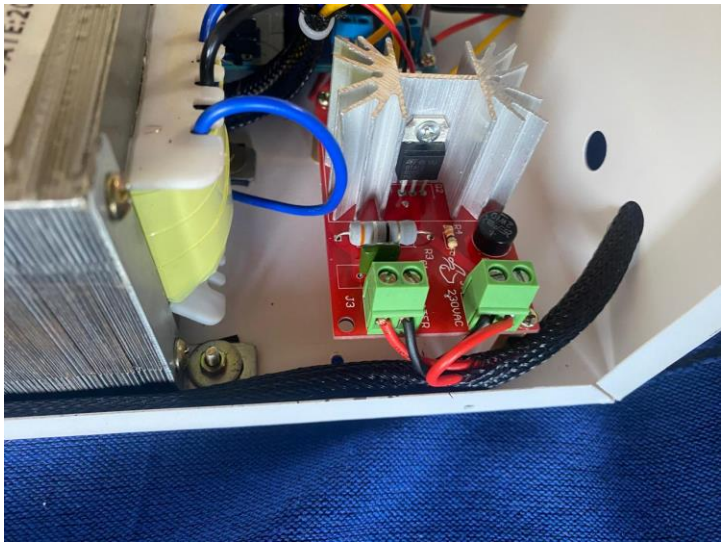


Figure 74- expandable braided PET cable sleeves

Figure 75- expandable braided PET cable sleeves

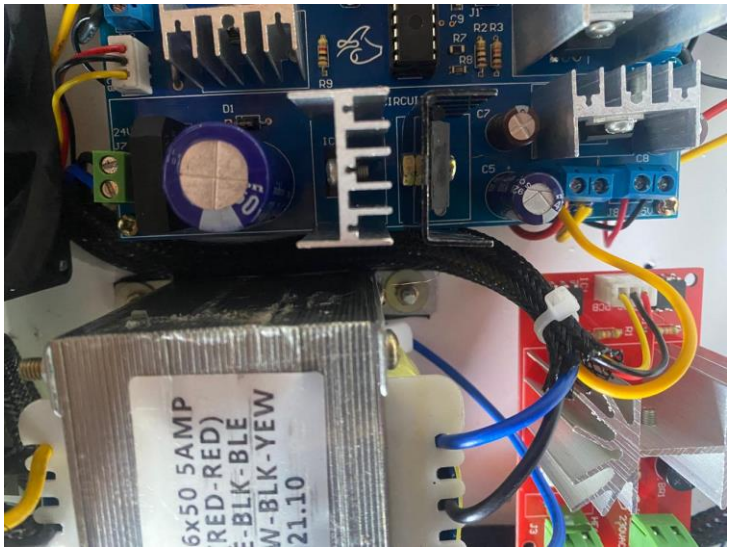


Figure 76 - wire ties

Figure 77 - JST

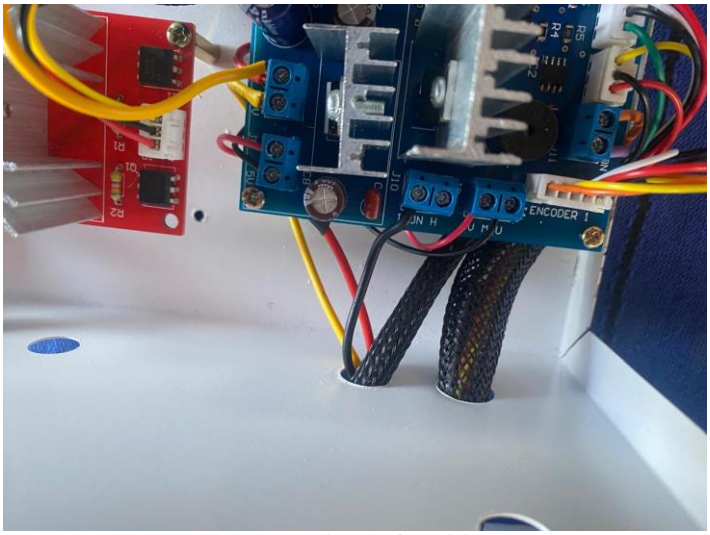


Figure 78 – wiring 1

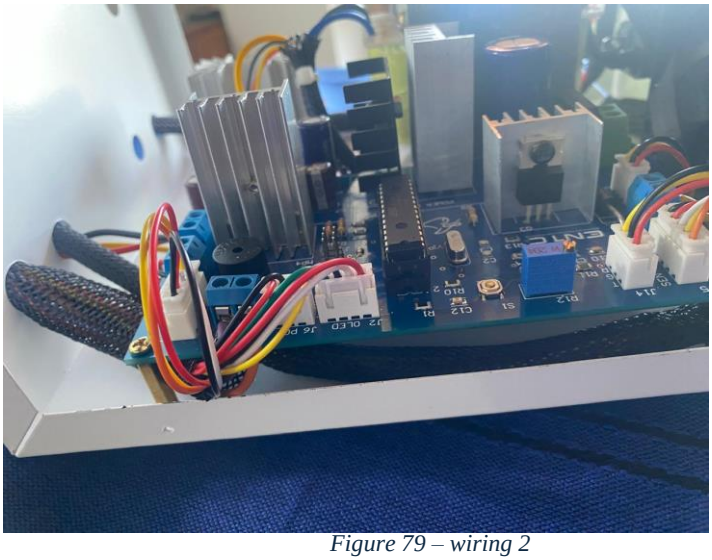


Figure 79 – wiring 2

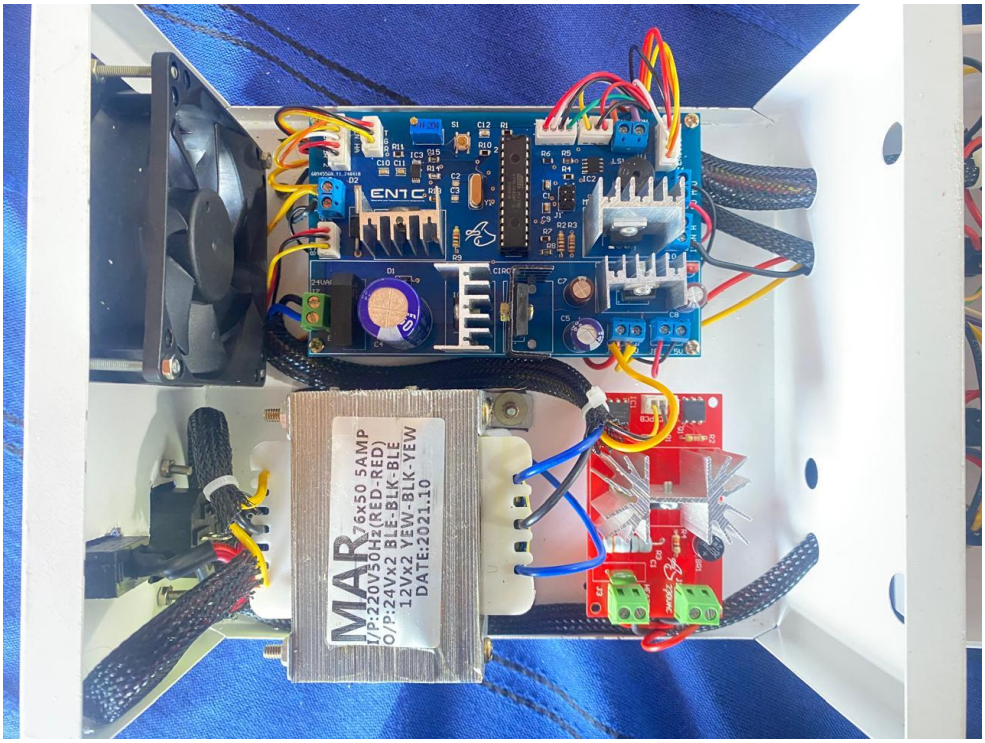


Figure 80 – final outcome

8.3 Photographs of display

8.0 Software Details

8.1. Setting Up the Environment

Software Details for Atmel Studio and ATmega328P Chip

8.1.1 Installing Atmel Studio

1. **Download Atmel Studio:**
    - Visit the Microchip website and navigate to the Atmel Studio download page.
    - Download the latest version of Atmel Studio.
2. **Install Atmel Studio:**
    - Run the downloaded installer.
    - Follow the installation prompts, accepting the license agreement and choosing the default installation options unless specific customization is required.

8.1.2. Setting Up a New Project

1. **Create a New Project:**
    - Open Atmel Studio.
    - Go to `File > New > Project`.
    - Select `GCC C Executable Project`, give your project a name, and choose a location to save it.
2. **Select Device:**
    - In the Device Selection window, type `ATmega328P` in the search bar and select it.
    - Click `OK`.

8.1.3. Configuring Project Settings

1. **Project Configuration:**
    - Navigate to the `Solution Explorer` pane.
    - Right-click on your project name and select `Properties`.
2. **Toolchain Settings:**
    - Configure the toolchain settings as per your project requirements. This typically includes setting optimization levels, enabling/disabling certain compiler warnings, etc.

8.1.4. Writing the Code

1. **Add `main.c` File:**
    - Right-click on the `Source Files` folder in the `Solution Explorer`.
    - Select `Add > New Item`, choose `C File (.c)`, and name it `main.c`.
    - Copy and paste your entire code, including all custom libraries, into this `main.c` file.

8.1.5. Compiling the Code

1. **Build Solution:**
    - Go to `Build > Build Solution` or press `F7`.
    - Check the `Output` pane at the bottom of Atmel Studio to ensure there are no errors.

8.1.6. Uploading the code

1. **Preparing the PCB and Programmer:**
    - **PCB Layout:** Identify necessary headers of the for connecting the programmer.

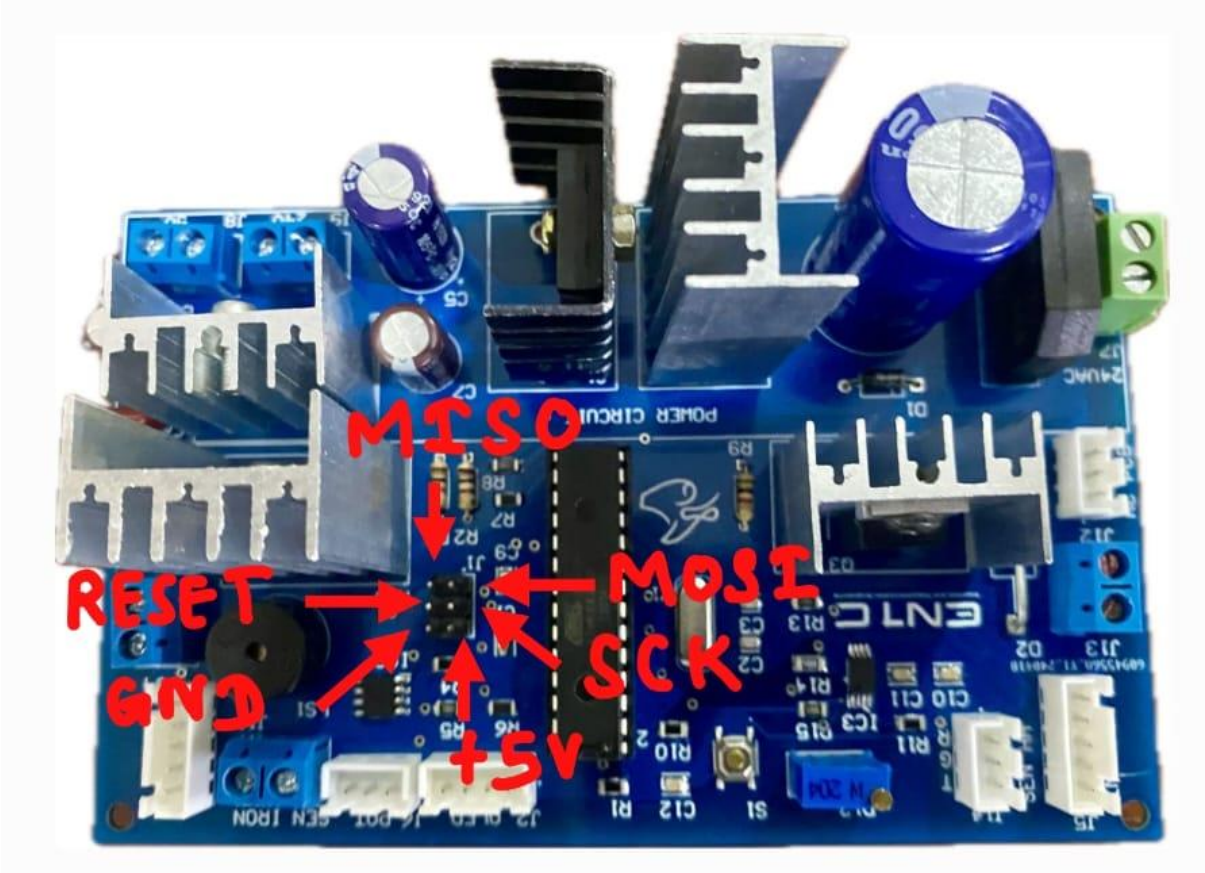


Figure 81 – pin configuration

    - **Connect Programmer:** Identify the appropriate header pins on your PCB for the ISP (In-System Programming) interface. These typically include MISO, MOSI, SCK, RESET, VCC, and GND.

2. **Connecting the Programmer:**
    - **AVRISP mkII:**
        - Connect the AVRISP mkII to your computer via USB.
        - Connect the AVRISP mkII to the ISP header on your PCB, ensuring proper orientation of the connector. The pin layout is usually as follows:
            - **MISO (Master In Slave Out)**
            - **MOSI (Master Out Slave In)**
            - **SCK (Serial Clock)**
            - **RESET**
            - **VCC (Power Supply)**
            - **GND (Ground)**

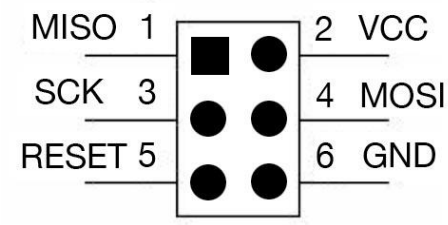


Figure 82 - AVRISP mkII pinout

    - **USBasp:**
        - Connect the USBasp to your computer via USB.
        - Connect the USBasp to the ISP header on your PCB. Ensure the correct orientation by matching the pins:
            - **MISO**
            - **MOSI**
            - **SCK**
            - **RESET**
            - **VCC**
            - **GND**

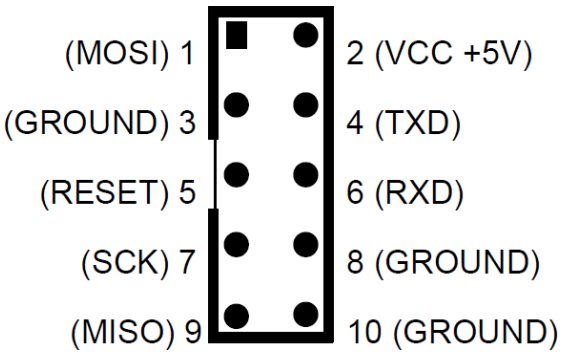


Figure 83- AVRISP USBasp pinout

3. **Set Up Programming Tool in Atmel Studio:**
    - Go to `Tools > Device Programming`.
    - In the `Tool` dropdown menu, select your programmer (AVRISP mkII or USBasp).
    - In the `Device` dropdown menu, select `ATmega328P`.
    - In the `Interface` dropdown menu, select `ISP`.
    - Click `Apply`.
4. **Read Device Signature:**
    - Click `Read` to ensure that the programmer can communicate with the ATmega328P chip.
5. **Upload Code:**
    - Navigate to the `Memories` tab.
    - In the `Flash` section, click `Browse` and select the `.hex` file generated by your build process.
    - Click `Program` to upload the code to the ATmega328P chip.
6. **Verify Code Upload:**
    - After programming, you can verify the upload by clicking `Verify`.

8.1.7. Testing and Deployment

Explanation: After uploading the code, test the soldering station's functionality to ensure proper operation and calibrate temperature settings if necessary.

Steps:

- Safely disconnect the programmer from the PCB.
- Ensure that the ATmega328P chip on the PCB is properly powered for standalone operation.
- Power on the soldering station and observe the display for any initialization messages or errors.
- Test the functionality of input buttons to navigate through menus and adjust temperature settings.
- Use temperature sensors to monitor temperature readings and verify that the PID control maintains stable temperature control.
- If necessary, adjust temperature setpoints or PID parameters in the code and re-upload to the Arduino board.
- Repeat testing and calibration until satisfied with the soldering station's performance.

8.2. Code and code description

8.2.1. Code with Arduino libraries in c++ language

Initially, we developed the code using Arduino libraries. The following code demonstrates our implementation for the soldering station utilizing these Arduino libraries.

```
#include <PID_v1.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <EEPROM.h>
#include <Wire.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1
#define SCREEN_ADDRESS 0x3C

#define PB_BACK 8
#define PB_OK 7
#define PB_UP 4
#define PB_DOWN 2

bool in_temp_display = false;
bool in_menu1 = false;
bool in_choice = false;

String gun_iron[2] = {"Both", "Hot air gun"};
int selected = 0;
String menu1[2] = {"Sleep Mode", "Tip Change"};
int select_m1 = 0;

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

int temperaturePin_iron = 23; // Pin for iron temperature sensor
int temperaturePin_gun = 24;  // Pin for hot air gun temperature sensor
int controlPinIron = 16;      // Pin for iron control
int controlPinGun = 17;       // Pin for hot air gun control
int Setpoint_iron = 150;
int Setpoint_gun = 150;

bool sleeping = false;
unsigned long last_activity_time = 0;
unsigned long sleep_duration = 60000; // 1 minute for testing

double aggKp = 4, aggKi = 0.2, aggKd = 1;
```

```
double consKp = 1, consKi = 0.05, consKd = 0.25;

double Input_iron; // Moved to global scope
double Output_iron;
double Setpoint;

double Input_gun;
double Output_gun;

PID myPIDIron(&Input_iron, &Output_iron, &Setpoint_iron, consKp, consKi, consKd,
DIRECT);
PID myPIDGun(&Input_gun, &Output_gun, &Setpoint_gun, consKp, consKi, consKd,
DIRECT);

void setup() {
    Serial.begin(9600);
    if (!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
        Serial.println(F("SSD1306 allocation failed"));
        for (;;)
            ;
    }
    EEPROM.put(0, Setpoint_iron);

    pinMode(controlPinIron, OUTPUT);
    pinMode(controlPinGun, OUTPUT);
    pinMode(temperaturePin_iron, INPUT);
    pinMode(temperaturePin_gun, INPUT);
    pinMode(PB_BACK, INPUT);
    pinMode(PB_OK, INPUT);
    pinMode(PB_UP, INPUT);
    pinMode(PB_DOWN, INPUT);

    display.display();
    delay(2000);
    display.clearDisplay();
    display.setTextSize(3);
    display.setTextColor(SSD1306_WHITE);
    display.println("Welcome!");
    display.display();
    delay(2000);
    in_choice = true;
    last_activity_time = millis();

    // Initialize the PID instances
    myPIDIron.SetMode(AUTOMATIC);
```

```
    myPIDGun.SetMode(AUTOMATIC);
}

void loop() {
    if (in_choice) {
        go_to_menu();
    }
    if (selected && in_temp_display) {
        display_temp(Setpoint_gun);
    } else if (!selected && in_temp_display) {
        int set = EEPROM.get(0, Setpoint_iron);
        display_temp(set);
    }

    if (in_menu1) {
        go_to_menu1();
    }

    checkSleepMode();

    // PID for iron
    Input_iron = readTemperature(temperaturePin_iron);
    int fSetpointIron = EEPROM.get(0, Setpoint_iron);
    double gapIron = abs(fSetpointIron - Input_iron);

    if (gapIron < 10) {
        myPIDIron.SetTunings(consKp, consKi, consKd);
    } else {
        myPIDIron.SetTunings(aggKp, aggKi, aggKd);
    }

    myPIDIron.Compute();
    analogWrite(controlPinIron, Output_iron);

    // PID for hot air gun
    Input_gun = readTemperature(temperaturePin_gun);
    double gapGun = abs(Setpoint_gun - Input_gun);

    if (gapGun < 10) {
        myPIDGun.SetTunings(consKp, consKi, consKd);
    } else {
        myPIDGun.SetTunings(aggKp, aggKi, aggKd);
    }

    myPIDGun.Compute();
```

```
    analogWrite(controlPinGun, Output_gun);
}

void display_A_Menu(int arrowIndex, String Arr[], int Arr_len, int textSize) {
    display.clearDisplay();
    display.setTextSize(textSize);
    for (int i = 0; i < Arr_len; i++) {
        display.setCursor(0, i * 10);
        if (i == arrowIndex) {
            display.print("> ");
        } else {
            display.print("  ");
        }
        display.println(Arr[i]);
    }
    display.display();
}

int wait_for_button_press() {
    while (true) {
        if (digitalRead(PB_UP) == LOW) {
            delay(200);
            return PB_UP;
        }
        if (digitalRead(PB_DOWN) == LOW) {
            delay(200);
            return PB_DOWN;
        }
        if (digitalRead(PB_OK) == LOW) {
            delay(200);
            return PB_OK;
        }
        if (digitalRead(PB_BACK) == LOW) {
            delay(200);
            return PB_BACK;
        }
    }
}

void go_to_menu() {
    while (digitalRead(PB_BACK) == HIGH) {
        display.clearDisplay();
        display_A_Menu(selected, gun_iron, 2, 1);
        int pressed = wait_for_button_press();
        if (pressed == PB_UP) {
```

```
            delay(200);
            selected += 1;
            if (selected > 1) {
                selected = 0;
            }
            display_A_Menu(selected, gun_iron, 2, 1);
        } else if (pressed == PB_DOWN) {
            delay(200);
            selected -= 1;
            if (selected < 0) {
                selected = 1;
            }
            display_A_Menu(selected, gun_iron, 2, 1);
        } else if (pressed == PB_OK) {
            delay(200);
            in_choice = false;
            in_temp_display = true;
            last_activity_time = millis(); // Update activity time
            break;
        } else if (pressed == PB_BACK) {
            delay(200);
            // Do something if needed
        }
    }
}

void display_temp(int setpoint) {
    display.clearDisplay();
    display.setTextSize(2);
    display.setTextColor(SSD1306_WHITE);
    display.setCursor(0, 20);
    if (selected) {
        display.print("Set Gun Temp: ");
    } else {
        display.print("Set Iron Temp: ");
    }
    display.print(setpoint);
    display.println(" C");
    display.display();

    int pressed = wait_for_button_press();
    if (pressed == PB_BACK) {
        delay(200);
        in_choice = true;
    } else if (pressed == PB_UP) {
```

```
            delay(200);
            if (selected) {
                Setpoint_gun += 1;
                if (Setpoint_gun > 400) {
                    Setpoint_gun = 400;
                }
            } else {
                setpoint += 1;
                if (setpoint > 400) {
                    setpoint = 400;
                }
                EEPROM.update(0, setpoint);
            }
            last_activity_time = millis(); // Update activity time
        } else if (pressed == PB_DOWN) {
            delay(200);
            if (selected) {
                Setpoint_gun -= 1;
                if (Setpoint_gun < 150) {
                    Setpoint_gun = 150;
                }
            } else {
                setpoint -= 1;
                if (setpoint < 150) {
                    setpoint = 150;
                }
                EEPROM.update(0, setpoint);
            }
            last_activity_time = millis(); // Update activity time
        } else if (pressed == PB_OK) {
            delay(200);
            if (!selected) {
                in_choice = false;
                in_temp_display = false;
                in_menu1 = true;
            }
        }
    }

    void go_to_menu1() {
        while (digitalRead(PB_BACK) == HIGH) {
            display.clearDisplay();
            display_A_Menu(select_m1, menu1, 2, 1);
            int pressed = wait_for_button_press();
            if (pressed == PB_UP) {
```

```
            delay(200);
            select_m1 += 1;
            if (select_m1 > 1) {
                select_m1 = 0;
            }
            display_A_Menu(select_m1, menu1, 2, 1);
        } else if (pressed == PB_DOWN) {
            delay(200);
            select_m1 -= 1;
            if (select_m1 < 0) {
                select_m1 = 1;
            }
            display_A_Menu(select_m1, menu1, 2, 1);
        } else if (pressed == PB_OK) {
            delay(200);
            int mode = select_m1;
            run_mode(mode);
        } else if (pressed == PB_BACK) {
            delay(200);
            in_choice = false;
            in_menu1 = false;
            in_temp_display = true;
            break;
        }
    }
}

void run_mode(int mode) {
    if (mode == 0) {
        sleepMode();
    } else if (mode == 1) {
        tipChange();
    }
}

double readTemperature(int pin) {
    double Input = analogRead(pin);
    Input = map(Input, 0, 450, 25, 350);
    return Input;
}

void checkSleepMode() {
    if (millis() - last_activity_time > sleep_duration && !sleeping) {
        sleepMode();
        sleeping = true;
```

```
    }
}

void wakeUp() {
    display.clearDisplay();
    display.setTextSize(2);
    display.setTextColor(SSD1306_WHITE);
    display.setCursor(0, 20);
    display.println("Waking up...");
    display.display();
    delay(1000);
    sleeping = false;
    last_activity_time = millis();
    display.clearDisplay();
}

void sleepMode() {
    display.clearDisplay();
    display.setTextSize(2);
    display.setTextColor(SSD1306_WHITE);
    display.setCursor(0, 20);
    display.println("Sleeping...");
    display.display();
    delay(1000);
    // Enter low power mode or stop heating elements
    analogWrite(controlPinIron, 0);
    analogWrite(controlPinGun, 0); // Stop the hot air gun as well
    while (true) {
        if (digitalRead(PB_OK) == LOW || digitalRead(PB_UP) == LOW ||
digitalRead(PB_DOWN) == LOW || digitalRead(PB_BACK) == LOW) {
            wakeUp();
            break;
        }
    }
}

void tipChange() {
    display.clearDisplay();
    display.setTextSize(2);
    display.setTextColor(SSD1306_WHITE);
    display.setCursor(0, 20);
    display.println("Tip Changed");
    display.display();
    delay(1000);
    // Lower temperature for safe tip change
```

```
    int original_setpoint = EEPROM.get(0, Setpoint_iron);
    Setpoint_iron = 100; // Example lower temperature
    EEPROM.update(0, Setpoint_iron);
    delay(5000); // Wait for 5 seconds
    Setpoint_iron = original_setpoint;
    EEPROM.update(0, Setpoint_iron);
}
```

8.2.2. Code in C language

As coding with Arduino is non engineering, the next step involved implementing the soldering station code in C language for AVR microcontroller programming. The code below demonstrates this implementation.

```
/**
 * ------------------------------------------------------------------------------------+
 * @desc        LCD FONT 5x8
 * ------------------------------------------------------------------------------------+
 * @source
 *              Copyright (C) 2020 Marian Hrinko.
 *              Written by Marian Hrinko (mato.hrinko@gmail.com)
 *
 * @author      Marian Hrinko
 * @date        08.12.2020
 * @update      08.12.2022
 * @file        font.h
 * @version     1.0
 * @tested      AVR Atmega328p
 *
 * @depend
 * ------------------------------------------------------------------------------------+
 * @descr       LCD pixel fonts
 * ------------------------------------------------------------------------------------+
 * @usage       Display characters
 * ------------------------------------------------------------------------------------+
 *
 * Modified by Rebecca Fernando
 * Date: 2024/07/4
 * Description: Edited for specific project requirements
 */

/**
 * ------------------------------------------------------------------------------------+
 * @brief       SSD1306 OLED Driver
 * ------------------------------------------------------------------------------------+
 *              Copyright (C) 2020 Marian Hrinko.
 *              Written by Marian Hrinko (mato.hrinko@gmail.com)
 *
 * @author      Marian Hrinko
 * @date        06.10.2020
 * @file        ssd1306.h
 * @version     2.0.0
 * @test        AVR Atmega328p
 *
```

```
 * @depend      string.h, font.h, twi.h
 * ------------------------------------------------------------------------------------+
 * @brief       Version 1.0 -> applicable for 1 display
 *              Version 2.0 -> rebuild to 'cacheMemLcd' array
 *              Version 3.0 -> simplified alphanumeric version for 1 display
 * ------------------------------------------------------------------------------------+
 * @usage       Basic Setup for OLED Display
 * ------------------------------------------------------------------------------------+
 *
 * Modified by Rebecca Fernando
 * Date: 2024/07/4

/**
 * ------------------------------------------------------------------------------------+
 * @brief       I2C Library
 * ------------------------------------------------------------------------------------+
 *              Copyright (C) 2017 Michael Köhler
 *
 * @author      Michael Köhler
 * @date        09.10.2017
 * @file        i2c.c
 * ------------------------------------------------------------------------------------+
 *
 * Modified by Rebecca Fernando
 * Date: 2024/07/4

/**********************************************************************
 * Arduino PID Library - Version 1.2.1
 * by Brett Beauregard <br3ttb@gmail.com> brettbeauregard.com
 *
 * This Library is licensed under the MIT License
 **********************************************************************
 *
 * For an ultra-detailed explanation of why the code is the way it is, please visit:
 * http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/
 *
 * For function documentation see:
 * http://playground.arduino.cc/Code/PIDLibrary
 * (Click "Libraries" on the left panel. The link to the documentation is listed as
"PIDLibrary - Provides basic feedback control".)
 **********************************************************************
 *
 * Modified by Rebecca Fernando
 * Date: 2024/07/4
 *Description: Edited for specific project Soldering Station
*/

#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/eeprom.h>
#include <stdbool.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <util/atomic.h>
#define F_CPU 16000000UL
#include <util/delay.h>
```

```
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define SCREEN_ADDRESS 0x3C

// Pin definitions
#define PB_BACK    PB0   // Pin 8 on Arduino, mapped to Port B, Pin 0 (PB0)
#define PB_OK      PD7   // Pin 7 on Arduino, mapped to Port D, Pin 7 (PD7)
#define PB_UP      PD4   // Pin 4 on Arduino, mapped to Port D, Pin 4 (PD4)
#define PB_DOWN    PD2   // Pin 2 on Arduino, mapped to Port D, Pin 2 (PD2)

#define CONTROL_PIN_IRON    PB2   // Pin 16 on Arduino, mapped to Port B, Pin 2 (PB2)
#define CONTROL_PIN_GUN     PB3   // Pin 17 on Arduino, mapped to Port B, Pin 3 (PB3)

#define TEMP_PIN_IRON       PC1   // Pin A1 on Arduino, mapped to Port C, Pin 1 (PC1)
#define TEMP_PIN_GUN        PC2   // Pin A2 on Arduino, mapped to Port C, Pin 2 (PC2)

// Global variables
bool in_temp_display = false;
bool in_menu1 = false;
bool in_choice = false;

const char* gun_iron[2] = {"Both", "Hot air gun"};
int selected = 0;
const char* menu1[2] = {"Sleep Mode", "Tip Change"};
int select_m1 = 0;

int Setpoint_iron = 150;
int Setpoint_gun = 150;

bool sleeping = false;
volatile unsigned long last_activity_time = 0;
unsigned long sleep_duration = 60000; // 1 minute for testing

double aggKp = 4, aggKi = 0.2, aggKd = 1;
double consKp = 1, consKi = 0.05, consKd = 0.25;
double outMin, outMax;

double Input_iron;
double Output_iron;
double Setpoint;
double Input_gun;
double Output_gun;

void I2C_Init();
void I2C_Start();
void I2C_Write(uint8_t data);
void SCREEN_Command(uint8_t command);
void SCREEN_Data(uint8_t data);
void SCREEN_Init();
void SCREEN_Clear();
void SCREEN_SetCursor(uint8_t row, uint8_t col);
void SCREEN_WriteChar(uint8_t ch, uint8_t size);
void SCREEN_Printf(const char *str, uint8_t size);
void go_to_menu();
void display_temp(int setpoint);
void go_to_menu1();
void run_mode(int mode);
double readTemperature(int pin);
```

```
void checkSleepMode();
void wakeUp();
void sleepMode();
void tipChange();
void display_A_Menu(int arrowIndex, const char* Arr[], int Arr_len, int textSize);
int wait_for_button_press();
void millis_init();
unsigned long millis();


void millis_init() {
	// Configure Timer0 for 1ms interrupt using prescaler 64
	TCCR0A = 0x00;             // Normal mode
	TCCR0B = (1 << CS01) | (1 << CS00); // Prescaler 64
	TIMSK0 = (1 << TOIE0);     // Enable overflow interrupt for Timer0
	TCNT0 = 0;                 // Initialize counter
}

// Interrupt Service Routine for Timer0 Overflow
ISR(TIMER0_OVF_vect) {
	last_activity_time++; // Increment millis_count in the ISR
}

unsigned long millis()
{
	unsigned long millis_value;
	// Ensure atomic access to millis_count
	ATOMIC_BLOCK(ATOMIC_FORCEON) {
		millis_value = last_activity_time;
	}
	return millis_value;
}

// Font array for ASCII characters
const uint8_t font[][8] = {
	{0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},  // Space (32)
	{0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},  // ! (33)
	{0x00, 0x00, 0x07, 0x00, 0x00, 0x07, 0x00, 0x00},  // ' (39)
	{0x00, 0x1C, 0x1C, 0x00, 0x1C, 0x1C, 0x00, 0x00},  // ( (40)
	{0x00, 0x1C, 0x1C, 0x00, 0x1C, 0x1C, 0x00, 0x00},  // ) (41)
	{0x00, 0x00, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x00},  // , (44)
	{0x00, 0x00, 0x00, 0x1C, 0x1C, 0x00, 0x00, 0x00},  // - (45)
	{0x00, 0x00, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x00},  // . (46)
	{0x00, 0x00, 0x00, 0x1C, 0x1C, 0x1C, 0x00, 0x00},  // / (47)
	{0x00, 0x1C, 0x1C, 0x1C, 0x1C, 0x1C, 0x00, 0x00},  // 0 (48)
	{0x00, 0x07, 0x07, 0x00, 0x07, 0x07, 0x00, 0x00},  // 1 (49)
	{0x00, 0x1C, 0x07, 0x1C, 0x1C, 0x1C, 0x00, 0x00},  // 2 (50)
	{0x00, 0x1C, 0x07, 0x00, 0x07, 0x1C, 0x00, 0x00},  // 3 (51)
	{0x00, 0x07, 0x07, 0x1C, 0x07, 0x07, 0x00, 0x00},  // 4 (52)
	{0x00, 0x1C, 0x1C, 0x00, 0x1C, 0x07, 0x00, 0x00},  // 5 (53)
	{0x00, 0x1C, 0x1C, 0x1C, 0x1C, 0x1C, 0x00, 0x00},  // 6 (54)
	{0x00, 0x07, 0x07, 0x00, 0x07, 0x07, 0x00, 0x00},  // 7 (55)
	{0x00, 0x1C, 0x1C, 0x1C, 0x1C, 0x1C, 0x00, 0x00},  // 8 (56)
	{0x00, 0x1C, 0x1C, 0x00, 0x1C, 0x1C, 0x00, 0x00},  // 9 (57)
	{0x00, 0x00, 0x1C, 0x1C, 0x1C, 0x00, 0x00, 0x00},  // : (58)
	{0x00, 0x00, 0x00, 0x1C, 0x1C, 0x00, 0x00, 0x00},  // ; (59)
	{0x00, 0x00, 0x07, 0x1C, 0x07, 0x00, 0x00, 0x00},  // < (60)
	{0x00, 0x00, 0x1C, 0x1C, 0x1C, 0x1C, 0x00, 0x00},  // = (61)
```

```
        {0x00, 0x00, 0x07, 0x1C, 0x07, 0x00, 0x00, 0x00},  // > (62)
        {0x00, 0x00, 0x1C, 0x07, 0x1C, 0x00, 0x00, 0x00},  //? (63)
        {0x00, 0x07, 0x07, 0x1C, 0x07, 0x07, 0x00, 0x00},  // A (65)
        {0x00, 0x1C, 0x1C, 0x1C, 0x1C, 0x1C, 0x00, 0x00},  // B (66)
        {0x00, 0x1C, 0x07, 0x07, 0x07, 0x1C, 0x00, 0x00},  // C (67)
        {0x00, 0x1C, 0x1C, 0x1C, 0x1C, 0x1C, 0x00, 0x00},  // D (68)
        {0x00, 0x1C, 0x1C, 0x00, 0x1C, 0x1C, 0x00, 0x00},  // E (69)
        {0x00, 0x1C, 0x1C, 0x00, 0x1C, 0x1C, 0x00, 0x00},  // F (70)
        {0x00, 0x1C, 0x1C, 0x1C, 0x1C, 0x1C, 0x00, 0x00},  // G (71)
        {0x00, 0x07, 0x07, 0x1C, 0x07, 0x07, 0x00, 0x00},  // H (72)
        {0x00, 0x1C, 0x00, 0x1C, 0x00, 0x1C, 0x00, 0x00},  // I (73)
        {0x00, 0x1C, 0x07, 0x07, 0x07, 0x07, 0x00, 0x00},  // J (74)
        {0x00, 0x07, 0x1C, 0x1C, 0x1C, 0x07, 0x00, 0x00},  // K (75)
        {0x00, 0x00, 0x1C, 0x1C, 0x1C, 0x1C, 0x00, 0x00},  // L (76)
        {0x00, 0x07, 0x1C, 0x1C, 0x07, 0x07, 0x00, 0x00},  // M (77)
        {0x00, 0x1C, 0x1C, 0x1C, 0x1C, 0x1C, 0x00, 0x00},  // N (78)
        {0x00, 0x1C, 0x07, 0x07, 0x07, 0x1C, 0x00, 0x00},  // O (79)
        {0x00, 0x07, 0x07, 0x1C, 0x1C, 0x1C, 0x00, 0x00},  // P (80)
        {0x00, 0x1C, 0x07, 0x07, 0x07, 0x07, 0x00, 0x00},  // Q (81)
        {0x00, 0x1C, 0x1C, 0x07, 0x1C, 0x07, 0x00, 0x00},  // R (82)
        {0x00, 0x1C, 0x1C, 0x1C, 0x1C, 0x1C, 0x00, 0x00},  // S (83)
        {0x00, 0x1C, 0x00, 0x1C, 0x00, 0x1C, 0x00, 0x00},  // T (84)
        {0x00, 0x1C, 0x07, 0x07, 0x07, 0x1C, 0x00, 0x00},  // U (85)
        {0x00, 0x1C, 0x1C, 0x1C, 0x1C, 0x1C, 0x00, 0x00},  // V (86)
        {0x00, 0x1C, 0x1C, 0x07, 0x1C, 0x07, 0x00, 0x00},  // W (87)
        {0x00, 0x07, 0x1C, 0x1C, 0x1C, 0x07, 0x00, 0x00},  // X (88)
        {0x00, 0x1C, 0x1C, 0x1C, 0x1C, 0x1C, 0x00, 0x00},  // Y (89)
        {0x00, 0x1C, 0x07, 0x1C, 0x07, 0x1C, 0x00, 0x00},  // Z (90)
        {0x00, 0x1C, 0x07, 0x1C, 0x07, 0x1C, 0x00, 0x00},  // a (97)
        {0x00, 0x07, 0x1C, 0x1C, 0x1C, 0x07, 0x00, 0x00},  // b (98)
        {0x00, 0x1C, 0x1C, 0x1C, 0x1C, 0x1C, 0x00, 0x00},  // c (99)
        {0x00, 0x1C, 0x1C, 0x1C, 0x1C, 0x1C, 0x00, 0x00},  // d (100)
        {0x00, 0x1C, 0x1C, 0x00, 0x1C, 0x1C, 0x00, 0x00},  // e (101)
        {0x00, 0x1C, 0x1C, 0x00, 0x1C, 0x1C, 0x00, 0x00},  // f (102)
        {0x00, 0x1C, 0x1C, 0x1C, 0x1C, 0x1C, 0x00, 0x00},  // g (103)
        {0x00, 0x07, 0x07, 0x1C, 0x07, 0x07, 0x00, 0x00},  // h (104)
        {0x00, 0x1C, 0x00, 0x1C, 0x00, 0x1C, 0x00, 0x00},  // i (105)
        {0x00, 0x1C, 0x07, 0x07, 0x07, 0x07, 0x00, 0x00},  // j (106)
        {0x00, 0x07, 0x1C, 0x1C, 0x1C, 0x07, 0x00, 0x00},  // k (107)
        {0x00, 0x00, 0x1C, 0x1C, 0x1C, 0x1C, 0x00, 0x00},  // l (108)
        {0x00, 0x07, 0x1C, 0x1C, 0x07, 0x07, 0x00, 0x00},  // m (109)
        {0x00, 0x1C, 0x1C, 0x1C, 0x1C, 0x1C, 0x00, 0x00},  // n (110)
        {0x00, 0x1C, 0x07, 0x07, 0x07, 0x1C, 0x00, 0x00},  // o (111)
        {0x00, 0x07, 0x07, 0x1C, 0x1C, 0x1C, 0x00, 0x00},  // p (112)
        {0x00, 0x1C, 0x07, 0x07, 0x07, 0x07, 0x00, 0x00},  // q (113)
        {0x00, 0x1C, 0x1C, 0x07, 0x1C, 0x07, 0x00, 0x00},  // r (114)
        {0x00, 0x1C, 0x1C, 0x1C, 0x1C, 0x1C, 0x00, 0x00},  // s (115)
        {0x00, 0x1C, 0x00, 0x1C, 0x00, 0x1C, 0x00, 0x00},  // t (116)
        {0x00, 0x1C, 0x07, 0x07, 0x07, 0x1C, 0x00, 0x00},  // u (117)
        {0x00, 0x1C, 0x1C, 0x1C, 0x1C, 0x1C, 0x00, 0x00},  // v (118)
        {0x00, 0x1C, 0x1C, 0x07, 0x1C, 0x07, 0x00, 0x00},  // w (119)
        {0x00, 0x07, 0x1C, 0x1C, 0x1C, 0x07, 0x00, 0x00},  // x (120)
        {0x00, 0x1C, 0x1C, 0x1C, 0x1C, 0x1C, 0x00, 0x00},  // y (121)
        {0x00, 0x1C, 0x07, 0x1C, 0x07, 0x1C, 0x00, 0x00},  // z (122)
};

void I2C_Init() {
        // Initialize I2C with 100 kHz baud rate
```

```
        TWBR = 72;  // Calculated value for 100 kHz at 16 MHz CPU clock
        TWSR &= ~(1<<TWPS0) & ~(1<<TWPS1);  // Prescaler = 1
}

void I2C_Start() {
        TWCR = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN);  // Start condition
        while (!(TWCR & (1<<TWINT)));  // Wait for start condition to be transmitted
}

void I2C_Write(uint8_t data) {
        TWDR = data;
        TWCR = (1<<TWINT) | (1<<TWEN);  // Clear TWINT to start transmission of data
        while (!(TWCR & (1<<TWINT)));  // Wait for transmission to complete
}

void SCREEN_Command(uint8_t command) {
        I2C_Start();
        I2C_Write(SCREEN_ADDRESS << 1);  // Slave address + write mode
        I2C_Write(0x00);            // Command mode
        I2C_Write(command);
        TWCR = (1<<TWINT) | (1<<TWEN) | (1<<TWSTO);  // Stop condition
        _delay_us(10);  // Small delay
}

void SCREEN_Data(uint8_t data) {
        I2C_Start();
        I2C_Write(SCREEN_ADDRESS << 1);  // Slave address + write mode
        I2C_Write(0x40);            // Data mode
        I2C_Write(data);
        TWCR = (1<<TWINT) | (1<<TWEN) | (1<<TWSTO);  // Stop condition
        _delay_us(10);  // Small delay
}

void SCREEN_Init() {
        I2C_Init();  // Initialize I2C communication
        _delay_ms(100);  // Delay for stable power-on
        SCREEN_Command(0xAE);  // Display off
        SCREEN_Command(0x20);  // Set Memory Addressing Mode
        SCREEN_Command(0x10);  // 00,Horizontal Addressing Mode;01,Vertical Addressing
Mode;
        // 10,Page Addressing Mode (RESET);11,Invalid
        SCREEN_Command(0xB0);  // Set Page Start Address for Page Addressing Mode,0-7
        SCREEN_Command(0xC8);  // Set COM Output Scan Direction
        SCREEN_Command(0x00);  // ---set low column address
        SCREEN_Command(0x10);  // ---set high column address
        SCREEN_Command(0x40);  // --set start line address
        SCREEN_Command(0x81);  // Set contrast control register
        SCREEN_Command(0xFF);  // --set segment re-map 0 to 127
        SCREEN_Command(0xA1);  // Set Display RAM Horizontal Flip (0xA0,0xA1)
        SCREEN_Command(0xA6);  // Set display mode (0xA6,0xA7)
        SCREEN_Command(0xA8);  // Set Multiplex Ratio (1 to 64)
        SCREEN_Command(0x3F);  // 1/64 duty
        SCREEN_Command(0xA4);  // Output follows RAM content;0xa4;Output ignores RAM
content
        SCREEN_Command(0xD3);  //-set display offset
        SCREEN_Command(0x00);   //-not offset
        SCREEN_Command(0xD5);  //--set display clock divide ratio/oscillator frequency
        SCREEN_Command(0xF0);  //--set divide ratio, Set Clock as 100 Frames/Sec
```

```
	SCREEN_Command(0xD9);  //--set pre-charge period
	SCREEN_Command(0x22);   //
	SCREEN_Command(0xDA);  //--set com pins hardware configuration
	SCREEN_Command(0x12);
	SCREEN_Command(0xDB);  //--set vcomh
	SCREEN_Command(0x20);  //0x20,0.77xVcc
	SCREEN_Command(0x8D);  //--set DC-DC enable
	SCREEN_Command(0x14);
	SCREEN_Command(0xAF);  //--turn on SCREEN panel
	SCREEN_Command(0xAF);
	SCREEN_Command(0xB1);  // Set Phase Length
	SCREEN_Command(0x1D);  // Select External Vcc Supply

	SCREEN_Command(0xB3);  // Set Display Clock Divide Ratio/Oscillator Frequency
	SCREEN_Command(0xF1);  // Set Display Divide Ratio 3e, 3f
}

void SCREEN_Clear() {
	uint8_t i, j;
	for (j = 0; j < 8; j++) {
		SCREEN_Command(0xB0 + j);  // Set page address (0 to 7)
		SCREEN_Command(0x00);      // Set low column address
		SCREEN_Command(0x10);      // Set high column address
		for (i = 0; i < 128; i++) {
			SCREEN_Data(0x00);     // Clear entire page
		}
	}
}

void SCREEN_SetCursor(uint8_t row, uint8_t col) {
	SCREEN_Command(0xB0 + row);             // Set page address
	SCREEN_Command(0x00 + (8*col & 0x0F)); // Set column lower address
	SCREEN_Command(0x10 + ((8*col>>4)&0x0F));// Set column higher address
}

void SCREEN_WriteChar(uint8_t ch, uint8_t size) {
	uint8_t i, j;
	for (i = 0; i < 8; i++) {
		for (j = 0; j < size; j++) {
			SCREEN_Data(font[ch - 32][i]);  // Print each row of the character
		}
	}
}

void SCREEN_Printf(const char *str, uint8_t size) {
	while (*str) {
		SCREEN_WriteChar(*str++, size);
	}
}

// PID control class
typedef struct {
	double *myInput;    // pointer to input value
	double *myOutput;   // pointer to output value
	double *mySetpoint; // pointer to setpoint value

	double kp;          // proportional gain
	double ki;          // integral gain
```

```
        double kd;          // derivative gain

        bool inAuto;        // flag indicating whether the controller is in automatic mode
        unsigned long SampleTime; // sample time in milliseconds
        unsigned long lastTime;   // last time the controller was updated

        double outputSum;   // sum of errors for integral term
        double lastInput;   // last input for derivative term

        double outMin;      // minimum output limit
        double outMax;      // maximum output limit
} PID;

void PID_Init(PID *pid, double *input, double *output, double *setpoint, double Kp,
double Ki, double Kd, int direction) {
        pid->myInput = input;
        pid->myOutput = output;
        pid->mySetpoint = setpoint;
        pid->kp = Kp;
        pid->ki = Ki;
        pid->kd = Kd;
        pid->inAuto = true;
        pid->SampleTime = 100;
        pid->lastTime = millis() - pid->SampleTime;
        pid->outputSum = 0;
        pid->lastInput = 0;
}

void PID_Compute(PID *pid) {
        if (!pid->inAuto) return;

        unsigned long now = millis();
        unsigned long timeChange = (now - pid->lastTime);
        if (timeChange >= pid->SampleTime) {
                double input = *(pid->myInput);
                double error = *(pid->mySetpoint) - input;
                pid->outputSum += (pid->ki * error);

                double output = pid->kp * error + pid->outputSum - pid->kd * (input - pid-
>lastInput);
                *(pid->myOutput) = output;

                if (*(pid->myOutput) > pid->outMax) *(pid->myOutput) = pid->outMax;
                else if (*(pid->myOutput) < pid->outMin) *(pid->myOutput) = pid->outMin;

                pid->lastInput = input;
                pid->lastTime = now;
        }
}

void PID_SetTunings(PID *pid,double Kp,double Ki,double Kd) {
        pid->kp = Kp;
        pid->ki = Ki;
        pid->kd = Kd;
}

void PID_SetOutputLimits(PID *pid,double Min,double Max) {
        if (Min >= Max) return;
```

```
	pid->outMin = Min;
	pid->outMax = Max;

	if (*(pid->myOutput) > pid->outMax) *(pid->myOutput) = pid->outMax;
	else if (*(pid->myOutput) < pid->outMin) *(pid->myOutput) = pid->outMin;

	if (pid->outputSum > pid->outMax) pid->outputSum = pid->outMax;
	else if (pid->outputSum < pid->outMin) pid->outputSum = pid->outMin;
}

void PID_SetMode(PID *pid, int Mode) {
	pid->inAuto = (Mode == 1);
}

PID myPIDIron, myPIDGun;

void setup()
{
	// Initialize GPIO pins
	DDRB &= ~(1 << PB_BACK);    // PB_BACK_PIN as input
	DDRD &= ~((1 << PB_OK) | (1 << PB_UP) | (1 << PB_DOWN));  // PB_OK_PIN, PB_UP_PIN,
PB_DOWN_PIN as inputs

	DDRB |= (1 << CONTROL_PIN_IRON) | (1 << CONTROL_PIN_GUN);  // CONTROL_PIN_IRON,
CONTROL_PIN_GUN as outputs

	// Enable pull-up resistors so that the pin reads as high when the button is not
pressed.
	PORTB |= (1 << PB_BACK);    // PB_BACK_PIN with pull-up
	PORTD |= (1 << PB_OK) | (1 << PB_UP) | (1 << PB_DOWN);  // PB_OK_PIN, PB_UP_PIN,
PB_DOWN_PIN with pull-ups

	DDRC &= ~((1 << TEMP_PIN_IRON) | (1 << TEMP_PIN_GUN));  // Set TEMP_PIN_IRON and
TEMP_PIN_GUN as inputs
	PORTC |= (1 << TEMP_PIN_IRON) | (1 << TEMP_PIN_GUN);   // Enable pull-up resistors
for these pins

	// Initialize ADC for analogRead
	ADMUX |= (1 << REFS0); // Set reference voltage to AVcc
	ADCSRA |= (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0); // Set ADC prescaler to 128
	ADCSRA |= (1 << ADEN); // Enable ADC

	// Initialize PWM for analogWrite
	// Timer 1 for CONTROL_PIN_IRON (PB2)
	TCCR1A |= (1 << WGM10) | (1 << WGM11) | (1 << COM1B1); // Fast PWM, 8-bit
	TCCR1B |= (1 << WGM12) | (1 << CS11); // Fast PWM, prescaler 8

	// Timer 2 for CONTROL_PIN_GUN (PB3)
	TCCR2A |= (1 << WGM20) | (1 << WGM21) | (1 << COM2A1); // Fast PWM
	TCCR2B |= (1 << CS22); // Prescaler 64

	// Initialize necessary components
	millis_init();
	sei(); // Enable global interrupts

	// Set up PID controller
	PID_Init(&myPIDIron,&Input_iron, &Output_iron, &Setpoint,consKp,consKi,consKd,1);
	PID_Init(&myPIDGun,&Input_gun, &Output_gun, &Setpoint,consKp,consKi,consKd,1);
```

```
	PID_SetOutputLimits(&myPIDIron, 0,225);
	PID_SetMode(&myPIDIron, 1);// AUTOMATIC mode
	PID_SetOutputLimits(&myPIDGun, 0,225);
	PID_SetMode(&myPIDGun, 1);// AUTOMATIC mode

	last_activity_time = millis();

	eeprom_update_word((uint16_t*)0, Setpoint_iron);

	SCREEN_Init();    // Initialize the SCREEN
	SCREEN_Clear();   // Clear the display
	// Display "Welcome!" message
	SCREEN_SetCursor(0, 0);      // Set cursor to start of first line
	SCREEN_Printf("Welcome!",3);         // Print text
	_delay_ms(3000);
	SCREEN_Clear();

	in_choice = true;
}

void loop()
{
	if (in_choice) {
		go_to_menu();
	}
	if (selected && in_temp_display) {
		display_temp(Setpoint_gun);
		} else if (!selected && in_temp_display) {
		int set = eeprom_read_word((uint16_t*)0); // Read Setpoint_iron from EEPROM
		display_temp(set);
	}

	if (in_menu1) {
		go_to_menu1();
	}

	checkSleepMode(); // Check for sleep mode based on inactivity

	// PID for iron
	Input_iron = readTemperature(TEMP_PIN_IRON);
	int fSetpointIron = eeprom_read_word((uint16_t*)0);
	double gapIron = abs(fSetpointIron - Input_iron);

	if (gapIron < 10) {
		PID_SetTunings(&myPIDIron,consKp,consKi,consKd);
		} else {
		PID_SetTunings(&myPIDIron,aggKp,aggKi,aggKd);
	}

	PID_Compute(&myPIDIron);
	OCR1B =Output_iron;//set PWM duty cycle on PB2 (analogWrite(controlPinIron,
Output_iron);)

	// PID for hot air gun
	Input_gun = readTemperature(TEMP_PIN_GUN);
	double gapGun = abs(Setpoint_gun - Input_gun);

	if (gapGun < 10) {
```

```
                PID_SetTunings(&myPIDGun, consKp, consKi, consKd);
                } else {
                PID_SetTunings(&myPIDGun, aggKp, aggKi, aggKd);
        }

        PID_Compute(&myPIDGun);
        OCR2A =Output_gun;//set PWM duty cycle on PB3 (analogWrite(controlPinGun,
Output_gun);)
}

int main()
{
        setup();
        while (1)
        {
                loop();
        }
}

void display_A_Menu(int arrowIndex, const char* Arr[], int Arr_len, int textSize) {
        SCREEN_Clear();
        for (int i = 0; i < Arr_len; i++) {
                SCREEN_SetCursor(i, 0);
                if (i == arrowIndex) {
                        SCREEN_Printf("> ",1);
                        } else {
                        SCREEN_Printf("  ",1);
                }
                SCREEN_Printf(Arr[i],1);
        }
}

int wait_for_button_press()
{
        while (1) {
                if (!(PIND & (1 << PB_UP))) {
                        _delay_ms(200);
                        return PB_UP;
                }
                if (!(PIND & (1 << PB_DOWN))) {
                        _delay_ms(200);
                        return PB_DOWN;
                }
                if (!(PIND & (1 << PB_OK))) {
                        _delay_ms(200);
                        return PB_OK;
                }
                if (!(PIND & (1 << PB_BACK))) {
                        _delay_ms(200);
                        return PB_BACK;
                }
        }
}

void go_to_menu()
{
        while (PIND & (1 << PB_BACK)) {
                SCREEN_Clear();
```

```
                display_A_Menu(selected, gun_iron, 2, 1);
                int pressed = wait_for_button_press();
                if (pressed == PB_UP) {
                        _delay_ms(200);
                        selected = (selected + 1) % 2;
                        display_A_Menu(selected, gun_iron, 2, 1);
                        } else if (pressed == PB_DOWN) {
                        _delay_ms(200);
                        selected = (selected - 1 + 2) % 2;
                        display_A_Menu(selected, gun_iron, 2, 1);
                        } else if (pressed == PB_OK) {
                        _delay_ms(200);
                        in_choice = false;
                        in_temp_display = true;
                        last_activity_time = millis(); // Update activity time
                        break;
                        } else if (pressed == PB_BACK) {
                        _delay_ms(200);
                        // Do something if needed
                }
        }
}

void display_temp(int setpoint) {
        SCREEN_Clear();
        SCREEN_SetCursor(0,20);

        if (selected) {
                SCREEN_Printf( "Set Gun Temp: ",2);
                } else {
                SCREEN_Printf(  "Set Iron Temp: ",2);
        }

        char* temp_str = malloc(12);
        snprintf(temp_str, 12, "%d", setpoint);
        const char* const_temp_str = temp_str;

        SCREEN_Printf(const_temp_str,2);
        SCREEN_WriteChar('c',2);


        int pressed = wait_for_button_press();
        if (pressed == PB_BACK) {
                _delay_ms(200);
                in_choice = true;
                } else if (pressed == PB_UP) {
                _delay_ms(200);
                if (selected) {
                        Setpoint_gun += 1;
                        if (Setpoint_gun > 400) {
                                Setpoint_gun = 400;
                        }
                        } else {
                        setpoint += 1;
                        if (setpoint > 400) {
                                setpoint = 400;
                        }
                        eeprom_update_word((uint16_t*)0, setpoint);
```

```
                }
                last_activity_time = millis(); // Update activity time
                } else if (pressed == PB_DOWN) {
                _delay_ms(200);
                if (selected) {
                        Setpoint_gun -= 1;
                        if (Setpoint_gun < 150) {
                                Setpoint_gun = 150;
                        }
                        } else {
                        setpoint -= 1;
                        if (setpoint < 150) {
                                setpoint = 150;
                        }
                        eeprom_update_word((uint16_t*)0, setpoint);
                }
                last_activity_time = millis(); // Update activity time
                } else if (pressed == PB_OK) {
                _delay_ms(200);
                if (!selected) {
                        in_choice = false;
                        in_temp_display = false;
                        in_menu1 = true;
                }
        }
}

void go_to_menu1() {
        while (PIND & (1 << PB_BACK)) {
                SCREEN_Clear();
                display_A_Menu(select_m1, menu1, 2, 1);
                int pressed = wait_for_button_press();
                if (pressed == PB_UP) {
                        _delay_ms(200);
                        select_m1 = (select_m1 + 1) % 2;
                        display_A_Menu(select_m1, menu1, 2, 1);
                        } else if (pressed == PB_DOWN) {
                        _delay_ms(200);
                        select_m1 = (select_m1 - 1 + 2) % 2;
                        display_A_Menu(select_m1, menu1, 2, 1);
                        } else if (pressed == PB_OK) {
                        _delay_ms(200);
                        int mode = select_m1;
                        run_mode(mode);
                        } else if (pressed == PB_BACK) {
                        _delay_ms(200);
                        in_choice = false;
                        in_menu1 = false;
                        in_temp_display = true;
                        break;
                }
        }
}

void run_mode(int mode) {
        if (mode == 0) {
                sleepMode();
                } else if (mode == 1) {
```

```
                tipChange();
        }
}

void tipChange() {
        SCREEN_Clear();
        SCREEN_SetCursor(0,20);
        SCREEN_Printf("Tip Changed",2);
        _delay_ms(1000);
        // Lower temperature for safe tip change
        int original_setpoint = eeprom_read_word((uint16_t*)0); // Read from EEPROM
        Setpoint_iron = 100; // Example lower temperature
        eeprom_update_word((uint16_t*)0, Setpoint_iron);
        _delay_ms(5000); // Wait for 5 seconds
        Setpoint_iron = original_setpoint;
        eeprom_update_word((uint16_t*)0, Setpoint_iron);
        SCREEN_Clear();
}

void checkSleepMode() {
        if (millis() - last_activity_time > sleep_duration && !sleeping) {
                sleepMode();
                sleeping = true;
        }
}

void wakeUp() {
        SCREEN_Clear();
        SCREEN_SetCursor(0,20);
        SCREEN_Printf("Waking up...",2);
        _delay_ms(1000);
        sleeping = false;
        last_activity_time = millis();
        SCREEN_Clear();
}

void sleepMode() {
        SCREEN_Clear();
        SCREEN_SetCursor(0,20);
        SCREEN_Printf("Sleeping...",2);
        _delay_ms(1000);
        // Enter low power mode or stop heating elements
        OCR0A = 0;  // Stop heating elements of iron
        OCR0B = 0;  // Stop heating elements of gun
        while (1) {
                if (!(PIND & (1 << PB_OK)) || !(PIND & (1 << PB_UP)) || !(PIND & (1 <<
PB_DOWN)) || !(PIND & (1 << PB_BACK))) {
                        wakeUp();
                        break;
                }
        }
}

double readTemperature(int pin) {
        // Select ADC channel
        ADMUX = (ADMUX & 0xF0) | (pin & 0x0F);
        // Start conversion
        ADCSRA |= (1 << ADSC);
```

```
        // Wait for conversion to complete
        while (ADCSRA & (1 << ADSC));
        double input = ADC;
        input = ((input / 1024.0) * 500.0);  // Assuming a 5V reference

        // Map the input value from 0-450 to 25-350
        double mappedInput = (input - 0) * (350.0 - 25.0) / (450.0 - 0) + 25.0;
        return mappedInput;
}
```

8.3. Includes and Defines

The code begins with the inclusion of necessary AVR libraries and defines various constants:

- **Libraries**: Standard AVR libraries for IO, interrupts, EEPROM access, and utility functions.
- **Constants**: Definitions for CPU frequency, screen dimensions, I2C address, and pin mappings for buttons and control lines.

Pin Definitions

- **Buttons**: `PB_BACK`, `PB_OK`, `PB_UP`, `PB_DOWN`
- **Control Pins**: `CONTROL_PIN_IRON`, `CONTROL_PIN_GUN`
- **Temperature Sensor Pins**: `TEMP_PIN_IRON`, `TEMP_PIN_GUN`

Global Variables

- Boolean flags for UI states (`in_temp_display`, `in_menu1`, `in_choice`).
- String arrays for menu options.
- Temperature setpoints and PID parameters.
- Variables to manage sleep mode and time tracking (`last_activity_time`, `sleep_duration`).

Function Prototypes

The code provides prototypes for various functions handling:

- I2C communication (`I2C_Init`, `I2C_Start`, `I2C_Write`).
- OLED screen commands and data transmission (`SCREEN_Command`, `SCREEN_Data`, `SCREEN_Init`, `SCREEN_Clear`, `SCREEN_SetCursor`, `SCREEN_WriteChar`, `SCREEN_Printf`).
- UI interactions (`go_to_menu`, `display_temp`, `go_to_menu1`, `run_mode`, `tipChange`, `display_A_Menu`).
- System state management (`checkSleepMode`, `wakeUp`, `sleepMode`).
- Helper functions (`wait_for_button_press`, `millis_init`, `millis`).

Millis Function Implementation

The `millis_init` and `millis` functions provide a millisecond counter using Timer0, enabling time-based functionalities such as sleep mode.

Interrupt Service Routine

The ISR for Timer0 overflow increments a global time counter (`last_activity_time`), providing the foundation for the `millis` function.

Font Array

A custom 8x8 font array is defined for ASCII characters, allowing the OLED screen to display text.

I2C Functions

- **I2C_Init**: Initializes the I2C interface for 100 kHz communication.
- **I2C_Start**: Sends the start condition on the I2C bus.
- **I2C_Write**: Transmits a byte over I2C.

OLED Display Functions

- **SCREEN_Command**: Sends a command to the OLED screen.
- **SCREEN_Data**: Sends data to the OLED screen.
- **SCREEN_Init**: Initializes the OLED screen with a series of configuration commands.
- **SCREEN_Clear**: Clears the OLED screen.
- **SCREEN_SetCursor**: Sets the cursor position on the screen.
- **SCREEN_WriteChar**: Writes a character to the screen at the current cursor position.
- **SCREEN_Printf**: Prints a formatted string to the screen.

Usage and Flow

1. **Initialization**: The code initializes the I2C bus and OLED screen.
2. **UI Interaction**: Functions manage user interactions, updating display based on button presses.
3. **Time Management**: The `millis` function provides timing capabilities for features like sleep mode.
4. **Sleep Mode**: The code includes functionality to enter and exit sleep mode based on user inactivity.

9.0 Appendix

26-27 Jan 2024: Research:
During these days, we conducted independent searches for resources related to our project. These resources included existing products, research papers, and videos demonstrating these products in use within the industry. This research phase provided us with valuable insights into the current market offerings and technological advancements, helping us identify potential areas for innovation and improvement in our own design.

30 Jan 2024: Project Proposal Finalized:
We finalized our project proposal, outlining the objectives, scope, and expected outcomes of our Digital Soldering Station project. This included defining the user requirements and initial conceptual designs. The proposal served as a roadmap for our project, ensuring all team members were aligned on the project goals and methodologies.

17 Feb 2024: Review Progress, Plan Next Steps, and Create Stakeholder Map:
We reviewed our progress to date, assessed our achievements, and identified areas needing further attention. During this time, we also planned the next steps in our project timeline and created a stakeholder map to identify all parties involved and their interests in the project. This stakeholder analysis was crucial for ensuring that our design met all stakeholder needs and expectations.

20 Feb 2024: Research on Circuit:
We conducted detailed research on the circuit design required for our soldering station, focusing on temperature control mechanisms and power management. This research helped us understand the technical requirements and challenges associated with developing a reliable and efficient circuit for our soldering station.

8 March 2024: Conceptual Design:
We developed initial conceptual designs for the soldering station, including functional block diagrams and key features. These designs were evaluated for feasibility and effectiveness. This phase involved brainstorming and creative thinking to propose innovative solutions that met the user requirements.

7-12 March 2024: Circuit Design:
We worked on detailed circuit designs, creating schematics and outlining the components required. This phase involved significant collaboration to ensure accuracy and functionality. We focused on integrating the temperature control system, power management, and additional features like sleep mode and tip change mode into our circuit design.

14 March 2024: Simulation:
We performed simulations of our circuit designs to test their functionality and identify any potential issues. These simulations allowed us to validate our design choices and make necessary adjustments before moving to the prototype phase. This step was critical for ensuring that our designs worked as intended and met the performance criteria.

15 March 2024: Prototype Simulation:
We developed a prototype simulation, testing the overall system integration and performance. This was a critical step in validating our design choices and ensuring that all components worked together seamlessly. The prototype simulation helped us identify and address any integration issues early in the development process.

17-20 March 2024: Circuit Design Finalized:
We finalized the circuit design, incorporating feedback from simulations and ensuring that all components worked together seamlessly. This involved fine-tuning the design to meet all technical specifications and user requirements, ensuring a robust and reliable circuit for our soldering station.

22-23 March 2024: Component Selection:
We selected the components needed for our soldering station, considering factors such as availability, cost, and performance. This phase involved evaluating different components to ensure they met our design specifications and budget constraints. Component selection was crucial for the overall performance and reliability of our soldering station.

21-23 March 2024: Enclosure Design:
We worked on the enclosure design, ensuring it was ergonomic, aesthetically pleasing, and functional. This included considering heat dissipation, ease of use, and overall design aesthetics. The enclosure design aimed to enhance the user experience by providing a comfortable and intuitive interface for the soldering station.

28-29 March 2024: Schematic and PCB Design:
We created the final schematics and PCB designs, ready for printing. This step was crucial for transitioning from design to physical prototype. We focused on ensuring that the PCB layout was optimized for performance, reliability, and ease of assembly.

05 April 2024: Sending PCB Design to Print:
We sent our PCB design to be printed, marking the beginning of the physical prototyping phase. This step involved coordinating with PCB manufacturers to ensure timely and accurate production of our circuit boards.

25 April 2024: Main Code Finalized:
We finalized the main code for controlling the soldering station, ensuring it was robust and met all functional requirements. This involved extensive testing and debugging to ensure the software controlled the hardware effectively and provided a seamless user experience.

01 May 2024: Sending Enclosure to Print:
We sent our enclosure design to be printed, moving closer to assembling the final prototype. The enclosure printing was coordinated with 3D printing services to ensure high-quality production of our design.

01 May 2024: Soldering PCB:
We soldered the components onto the PCB, assembling the electronic core of our soldering

station. This step involved careful assembly and soldering to ensure all components were securely and correctly attached to the PCB.

02-03 May 2024: Testing PCB:
We tested the assembled PCB to ensure all components functioned correctly and the overall system met our design specifications. This phase involved thorough testing and validation to identify and fix any issues before the final assembly and deployment of the soldering station.

Tuning PID and amplifier testing

22-24 June 2024: Assembling and Wiring:
During these days, we focused on assembling the various components of our soldering station and completing the necessary wiring. This involved integrating the PCB, enclosure, soldering iron, and other parts to form a complete, functional unit. Careful attention was given to ensuring all connections were secure and correctly wired to ensure optimal performance and reliability.

Final Testing

10.0 Reference

Atmega328p pu data sheet- https://www.alldatasheet.com/datasheet-pdf/pdf/392289/ATMEL/ATMEGA328P-PU.html

Zero crossing circuit- https://circuitdigest.com/electronic-circuits/zero-crossing-detector-circuit-diagram

opto-isolated triac triggering circuit -https://www.st.com/resource/en/application_note/an5114-controlling-a-triac-with-a-phototriac-stmicroelectronics.pdf

PID Reference- https://www.finepowertools.com/diy/soldering-iron-tip/

https://shedheads.net/best-soldering-station/tips/

Arduino-PID-Library/README.txt at master · br3ttb/Arduino-PID-Library (github.com)

Tip change references- https://aixuntech.com/newsinfo/the-ultimate-guide-to-soldering-iron-tips-types-sizes-and-uses/

https://www.arduino.cc/reference/en/libraries/pid/

https://github.com/br3ttb/Arduino-PID-Library/blob/master/PID_v1.h

Amplifier - https://www.electronics-tutorials.ws/amplifier/amp_1.html

Display -OLED SSD1306 - SH1106 - Arduino Reference

https://github.com/Sylaina/oled-display

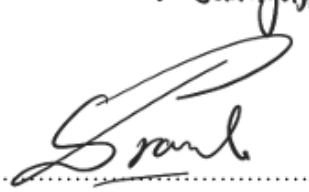
SSD1306/lib/ssd1306.h at master · Matiasus/SSD1306 (github.com)

1. Checked by: A.K.N. De Zoysa

..... 23/06/2024

Signature Date

2. Checked by: Saumyajith S.A.D.P



..... 22/06/2024

Signature Date

3. Checked by: Senaviratnne I.U.B

..... 22/06/2024

Signature Date

4. Checked by: Silva. M.K.Y.U.N.

..... 22/06/2024

Signature Date