EXECUTIVE SUMMARY

The objective of this report is to document the complete process of designing and implementing the Capstone Project for the Electronics Engineering Technologist (ENT) program. This report covers the project development of a concept *Supervisory Control and Data Acquisition System (SCADA)*. As expected of students of the ENT program, the development of the project requires the utilization of the theoretical skills and knowledge learned over the duration of the program backed by the experience gained in electronics laboratory experiments. The project requires the designing and prototyping of conceptual ideas and functional circuits as well as the experiments used to test, troubleshoot and characterize the systems. The process of the project is documented in the body of the report and describes how the above methods are utilized to successfully develop the project as outlined in the requirements section. The final system achieves the required design constraints, requirements and concept goals elicited. The report documents the procedures and techniques used and the resulting system is a showcase of the applied attributes required of an ENT upon graduation. The electronics and digital fundamentals display the skills learned from the program, whereas the adaptation of techniques and technologies such as the implementation of *XBee* wireless *transceivers* and custom protocol confirm the readiness of the students for the ever-changing electronics industry.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

1. INTRODUCTION

1.1. Purpose

This report will discuss the design and implementation of a concept model *Supervisory Control and Data Acquisition (SCADA)* system that monitors and controls a Remote Terminal Unit (RTU) consisting of a security system unit and a temperature controller unit. The project has been completed by students in order to demonstrate essential skills and knowledge obtained from their program.

1.2. Background

The report covers the process of the Electronics Engineering Technology (ENT) ‘capstone’ project required for completion of the course. The procedure requires the application of fundamental theories and concepts in electrical and electronics engineering, including thermal and control systems, wireless communication protocols, digital concepts and *embedded systems* programming. All aspects of the project will be completed at the Southern Alberta Institute of Technology (SAIT) under the guidance and instructions of the program instructors. All defined terms will be provided in italicized text and appended in the glossary for reference.

1.3. Scope

This report will cover the design and implementation process of the *SCADA* system. We are limited primarily to knowledge and skills acquired directly from the ENT program. As laid out in the methods we have supplemented the design with extra research and experimentation where concepts are not included in the curriculum. The materialization will require the design of schematics, software simulations and breadboard prototyping. The final product will require: a Printed Circuit Board (PCB), hardware and software interfaced with the ARM Cortex M4 microprocessor, an Liquid crystal display (LCD), and raw analog sensors.

1.4. Methods

The project is guided by the experience and knowledge learned through the ENT program; the majority of the design is applied directly from this education. However, we have modified designs and created novel implementation techniques in order to develop a unique and effective system. For any concept that falls outside of the curriculum, further research has been conducted using textbooks, academic papers and manufacturer implementation reports. All designs have been backed by reference material, and have been followed by prototyping and experimentation using the SAIT electronics lab facilities. When prototyping is finalized, a printed circuit board (PCB) is developed in the lab using Altium Designer™ and the finalized PCBs have

been ordered and assembled before being presented to the class and instructors as a complete product.

Preview

To complete the *SCADA* system as it is defined in the project proposal, the following phases are completed and documented. In the requirements elicitation process, the conceptual goals of the system are parameterized and narrowed down to practical constraints and definitions that can be achieved. Research is required to determine the feasibility of the proposal and the physical limitations that exist.

The problem domain description is required to give literal context to the system in order to better understand the interactions of the system and subsystems with all internal and external sources of interaction. Without the problem domain description, many physical and technical aspects of the project are overlooked and as a result, interfaces and hardware that are required to satisfy the requirements can be missed. The requirements and constraints are required to define practical limitations on the system due to financial and time constraints as well as managerial and theoretical customer constraints that are introduced for practical purposes by the Capstone instructor and SAIT ENT program.

The specifications section provides an accurate technical assessment of the finalized system design including electronic signals, voltage and current levels, power dissipation, protocols and component connectivity. The specifications are intended to be a valid design for a system that is followed exactly and is based on research and applied knowledge from the ENT program. The human-machine interface section details the system interface used to provide inputs and read outputs as completed in the final project, design art and graphical concepts were not used due to the simplicity of the design. In the electronics system design section, the physical systems are developed through a combination of applied research, and base knowledge used for preliminary calculations and circuit design as well as prototyping and trial and error used to determine the performance of the designs.

The software design section states and describes the software that is implemented in order to provide the functionality required by the previous sections. The testing and characterization section includes the results of and the methods used to gathered data that confirms the system operation is as designed and required by the previous sections. The final sections include pictorial demonstrations of the completed systems, and the details of the work and efforts expended in the development process. The conclusion states the success of the project and the final outcomes.

2. PROJECT PROPOSAL

Strategic Objective: Develop a proof of concept *SCADA* system with an RTU that is comprised of both a Temperature Control Unit (TC1) and a Security System Unit (SS1).

Product Overview: The *SCADA* system is used to control TC1 temperature, monitor TC1 temperature, detect SS1 access attempts, alarm unauthorized access attempts, and display lights on/off status at the RTU. The *SCADA* system is controlling temperature through a heating element and fan system and will be capable of controlling temperature between 25℃ to 75℃.

The monitored/measured data collected from the RTU will be put in timestamp pair format and stored at the workstation. The scanning period will be 30 seconds, as to allow for the possibility of multiple RTUs to be added to the system in future development. The MTU is connected to a laptop workstation for the *SCADA* system.

The MTU and RTU subsystems (SS1 and TC1) are all using the ARM M4 Cortex MCU on the STM32F405RG development boards. The communications system between the RTU and Master is wireless through an *XBee transceiver* utilizing a custom protocol.

3. REQUIREMENTS ELICITATION

This section defines the parameters that are required to develop a product that matches a clearly defined concept and design goal for the system and each subsystem.

The *SCADA* System will be composed of a Master Terminal Unit (MTU) and two RTUs).

The *SCADA* MTU will:

- Monitor temperature at the MTU.
- Display the current MTU temperature on a LCD.
- Use the Olimex STM32F405RG development board.
- Communicate RTU data with the Operator Workstation through RS-232 **[1]**.
- Have wireless *XBee* communication with the RTU temperature controller and RTU security system through radio modules.
- Follow a user defined protocol for radio communication.

The *SCADA* Operator Workstation will:

- Communicate with the MTU through RS-232.
- Store the data gathered from the RTU in timestamp pairs **[1]**.
- Allow the operator to view the stored data gathered from the RTU.

- Show alarms and statuses of RTU **[1]**.
- Show the temperature and set temperature of TC1.
- Create the temperature set point of TC1.
- Allow the operator to change the password for the security system.

The *SCADA* RTU temperature controller (TC1) will:

- Have wireless *XBee* communication with MTU.
- Allow the MTU to enter the temperature set point.
- Provide data to the MTU on request from MTU.
- Monitor the airflow and element temperature.
- Maintain the airflow set temperature between 25℃ and 75℃.
- Use a heating element and fan system for temperature control.
- Use a power transistor and power resistor as the heating elements.
- Use 120VAC mains power supply for input power.
- Be a complete unit with chassis.
- T.C will have two temperature sensors (one raw analog for airflow temperature and one digital/IC for element temperature).
- Have a fixed tolerance +/- 0.2 degrees Celsius
- Use the STM32F405RG microprocessor development board by Olimex.
- Will use an LCD to display set point, sensor temperature, and fan speed.
- Use a Keypad interface to allow for a set point input.

The *SCADA* RTU Security System will:

- Have wireless *XBee* communication with MTU.
- Monitor the brightness/illuminance level at the RTU.
- Control access to the RTU via password activated entry.
- Use a keypad for password entry.
- Communicate with the MTU regarding status of the system(illuminance level and authorization entry).
- Record the illuminance levels on a predetermined timely manner.
- Sound an alarm if entry is unauthorized.
- Record entry into the RTU.

The *SCADA* System has a development time of 3 months.
The *SCADA* System costs will not exceed the budget of $1000.

4. PROBLEM DOMAIN DESCRIPTION

The problem domain of the *SCADA* System includes scenarios at both the MTU and RTU domains. The interactions at the MTU include any interactions with both the MTU and the laptop workstation by a designated operator. At the RTU, all interactions with the temperature controller

is to be by a designated RTU technician due to the security system unit limiting the access to the TC1.

4.1. EVENTS IN THE PROBLEM DOMAIN

Events at the MTU:

1. MTU operator connects the laptop workstation to the MTU via RS-232 cable.
2. MTU operator sets up the MTU.
3. MTU operator turns on the MTU.
4. MTU operator opens previous data logs.
5. MTU operator changes TC1 set temperature.
6. MTU operator changes SS1 password.
7. MTU operator monitors temperature readings, RTU statuses, and alarms.
8. MTU operator handles alarms:
    a. Over temperature heater at TC1.
    b. Access attempt through SS1.
    c. Unauthorized access through SS1.

Events at the RTU:

1. RTU technician installs RTU package in the desired remote location.
2. RTU technician turns on the TC1.
3. RTU technician changes the TC1 set temperature.
4. TC1 temperature control adjusts to match current temperature to set temperature.
5. RTU technician enters password to access RTU:
    a. With correct password entry.
    b. With incorrect password entry.
6. The door is opened:
    a. Forcibly without correct password entry.
    b. With correct password entry.
7. The lights are on at the RTU location.
8. The lights are off at the RTU location.

The problem domain is summarized in the problem frame diagram of *Fig.1* for the *SCADA* system, including the TC1 and SS1 subsystems.

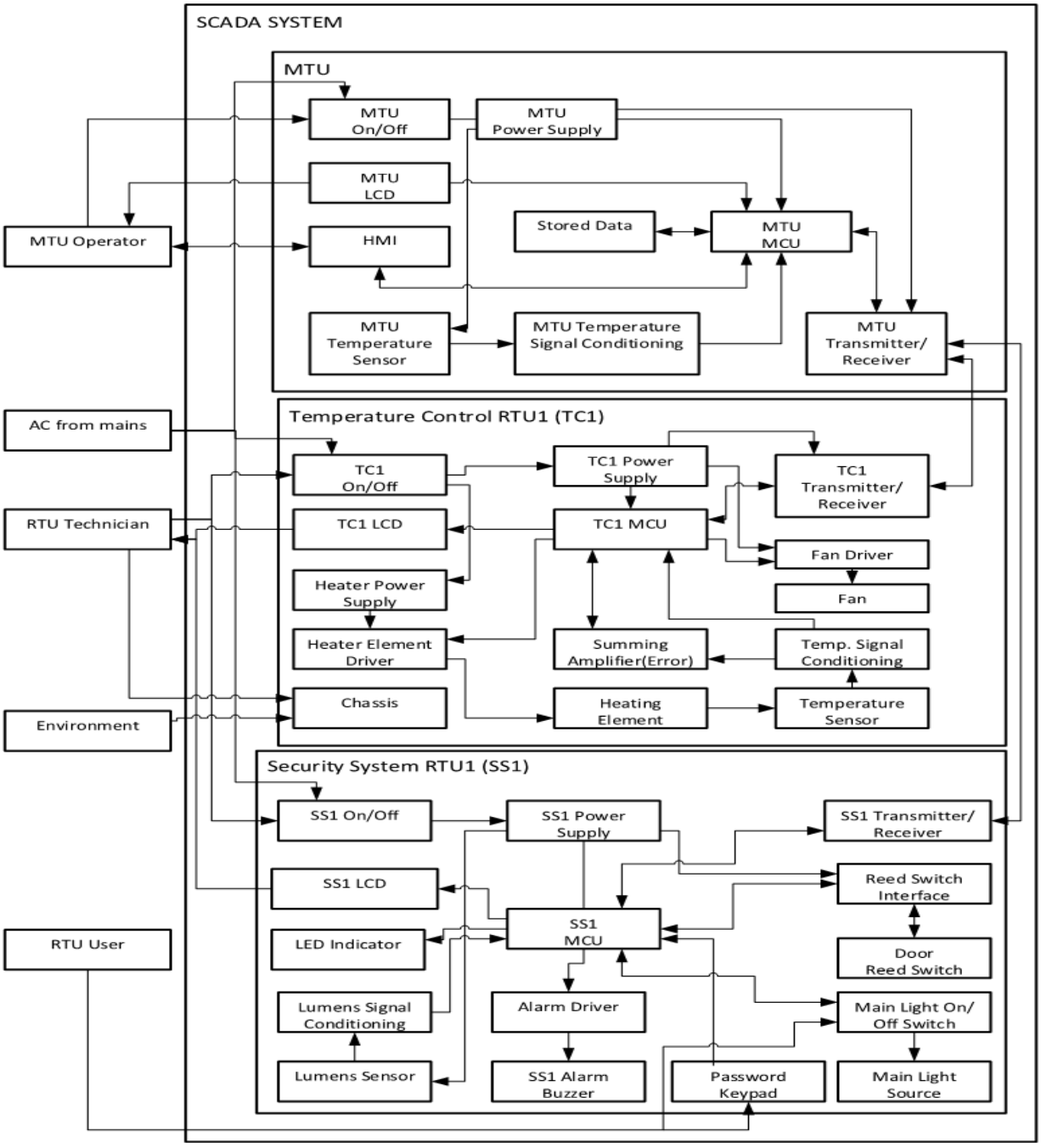


Figure 1 Problem Frame Diagram

For more details of the problem frame systems (MTU, TC1 and SS1) see sections **6.1**, **6.2** and **6.3** respectively.

4.2. CONTEXT DIAGRAMS RELATED TO EACH MCU

The context diagram describes the system interactions and direct connections of the Micro Controller Units (MCU) (STM32F405RG microprocessor) of each system.

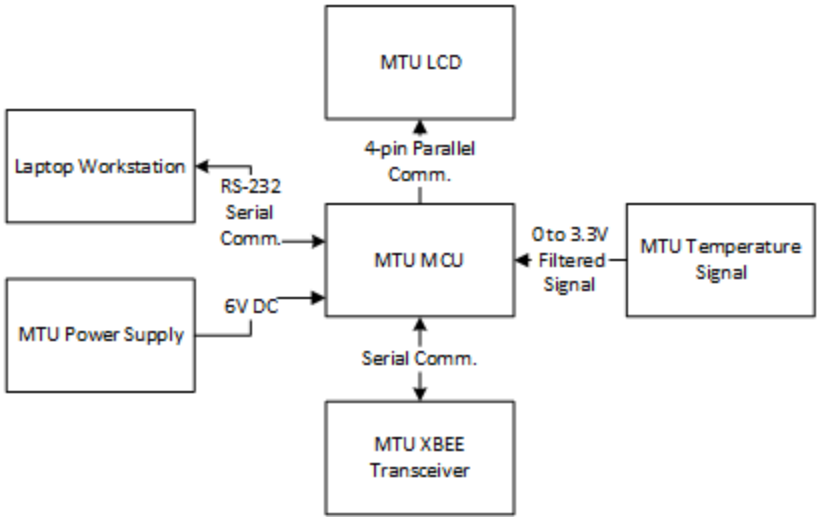


Figure 2 MTU Context Diagram

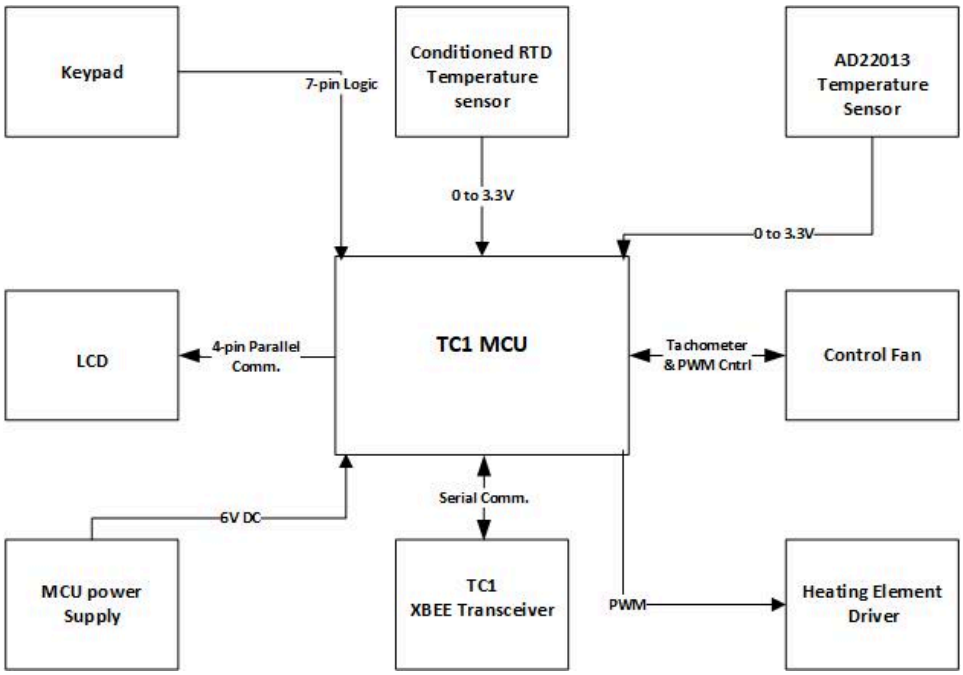


Figure 3 TC1 Context Diagram

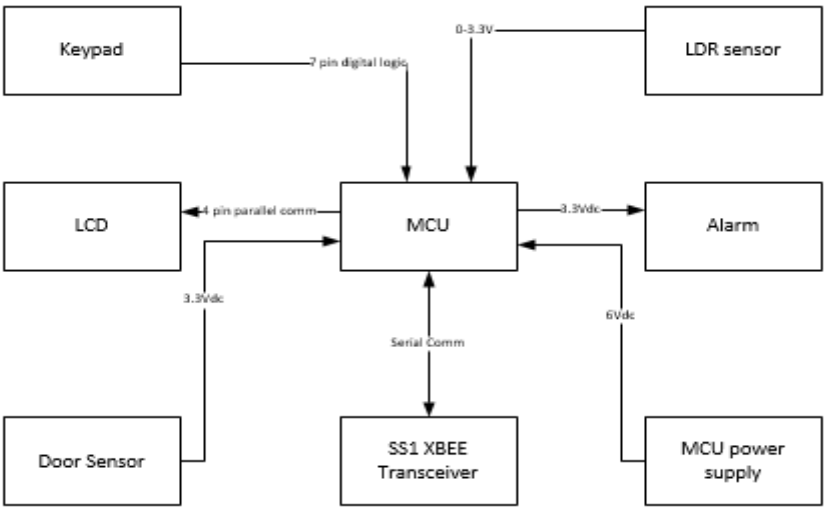


Figure 4 SS1 Context Diagram

5. REQUIREMENTS AND CONSTRAINTS

Table 1 Design Constraints

#	Hard/Software	Description
GDC1	HW	The MTU Olimex STM32F405 must be used.
GDC2	HW	The AC from mains is 110VAC.
GDC3	SW	Crossworks IDE must be used for development.
GDC4	HW	At least 1 raw sensor must be used individually.
GDC5	SW	A serial communication protocol is to be used
GDC6	HW	At least 1 *span and zero* circuit must be used.
GDC7	HW	Components in RTU must function between 0-100°C
GDC8	SW	The RTU datalog must be in timestamp pairs.
GDC9	SW	The MTU scan interval must allow for an effective real-time system.

Table 2 Commercial Constraints

#	Description
CC1	The time frame is 4 months start to finish.
CC2	The *SCADA* System is not to exceed $1000.
CC3	The *SCADA* System is to be designed and built on SAIT premises

Table 3 Functional and Performance Requirements

#	Hard/Software	Description
FR1	HW	The SCADA system must monitor the temperature of enclosure of the RTU.
PR1.1	HW/SW	The temperature measurements must be to tenths °Celsius.
PR1.2	HW	The temperature measurements must be at least 95% accurate.
FR2	HW	The *SCADA* system must condition the temperature signal from the *Modbus* temperature sensor.
FR2.1	HW	The conditioned temperature signal must interface with the ADC of the RTU MCU.
FR3	HW/SW	The *SCADA* system must control temperature at the RTU.
FR3.1	HW	The temperature must be controlled through a heating element.
FR3.2	HW	A fan must be used to allow for the transfer of heat.
FR4	SW	The *SCADA* system must store data received from RTU.
PR4.1	SW	The measured data must be collected every 30 seconds.
FR4.2	SW	The measured data must be stored with a timestamp.
FR5	SW	The *SCADA* system master must communicate with the RTU.
FR5.1	SW	The communication must utilize *Modbus* protocol
FR6	HW	The *SCADA* system master must have a Human Machine Interface (HMI).
FR6.1	SW	The HMI must be able to access the stored data-log and display them.
FR6.2	SW	The HMI must be able to create a new temperature set-point for the RTU.
FR7	HW	The RTU must have a power supply for the MCU, heating element, fan, and sensors.
FR8	HW	The RTU must have a reed switch to detect whether or not the enclosure is open.
FR9	HW	The RTU must have a light sensor the determine the light status in the enclosure.

6. SCADA SYSTEM SPECIFICATION

The *SCADA* system specifications are broken down into three subsections:

- Master Terminal Unit (MTU)
- Temperature Control Unit
- Security System Unit

SS1 and TC1 make up the Remote Terminal Unit 1 (RTU1).

6.1. SCADA Master Terminal Unit Specification

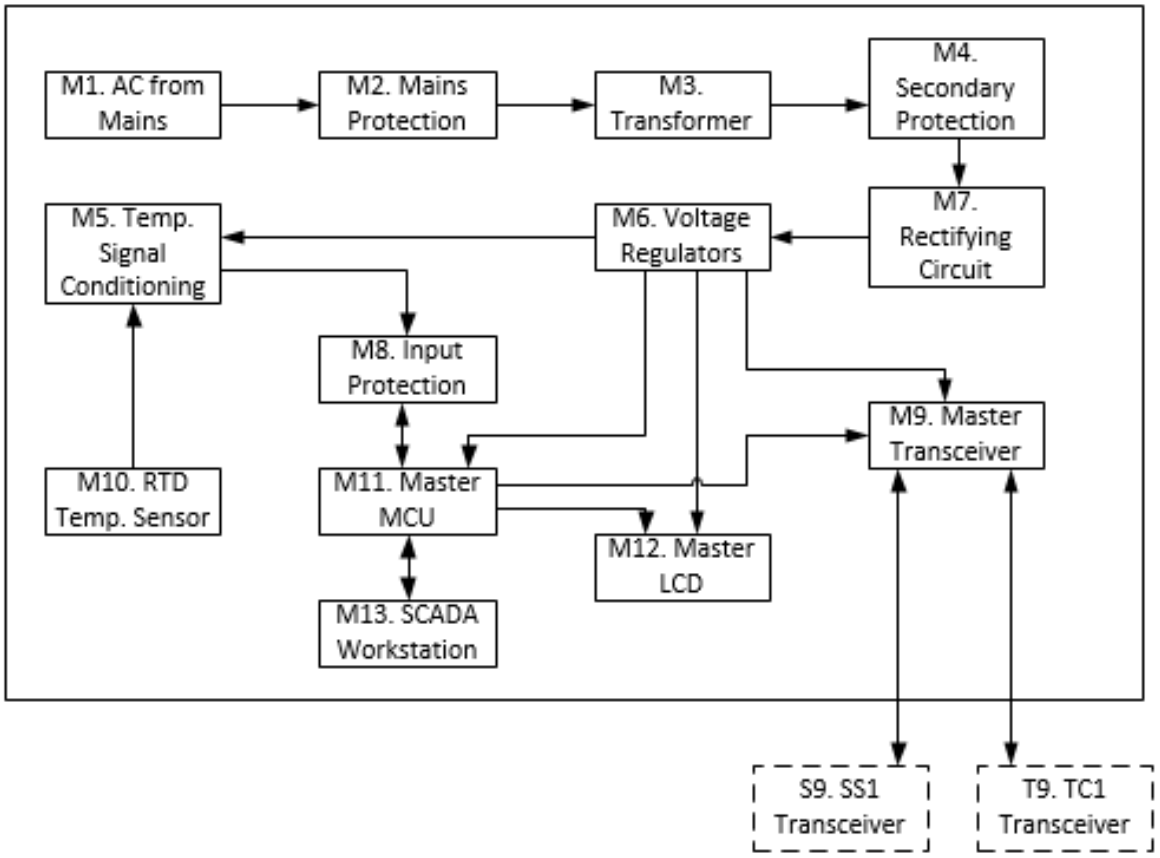


Figure 5 MTU Function Block Diagram

Table 4 MTU Function Block Descriptions

Functional Block Name	**Block Functions**
M1. AC from Mains	1. The power cord will plug into a socket for $120V_{rms}$ @ 60Hz to power the MTU circuitry.
M2. Mains Protection	1. For circuitry protection and safety. 2. 0.5 amp Slo-Blo fuse for current protection. 3. Thermal fuse and metal oxide varistor to handle voltage surges. 4. SPDT switch for on/off.
M3. Transformer	1. Convert $120V_{rms}$ @60Hz to $20V_{rms}$ @60Hz.
M4. Secondary Protection	1. 1 amp Slo-Blo fuses for both sides of secondary. 2. Shunt series resistor and capacitor across secondary.
M5. Temp. Signal Conditioning	1. 1mA constant current circuit for Resistance Temperature Detector (RTD). 2. Instrumentation fully differential amplifier. 3. First order low-pass filter. 4. *Span and zero* circuit to convert signal for MCU ADC.
M6. Voltage Regulators	1. Adjustable regulator for +11.6V DC. 2. Adjustable regulator for -11.6V DC. 3. Fixed regulator for +6V DC. 4. Fixed regulator for +3.3V DC.
M7. Rectifying Circuit	1. Diode full-wave rectifier circuit. 2. Smoothing capacitors for positive and negative voltages.
M8. Input Protection	1. Schottky diodes across MCU 3.3V and ground. 2. Resistors to limit current into MCU.
M9. Master Transceiver	1. XBee PRO S3B *transceiver* for wireless communications.
M10. RTD Temp. Sensor	1. RTD to detect temperature levels at the MTU.
M11. Master MCU	1. Calculates local temperature through ADC and displays to the LCD. 2. Polling information to and from *transceiver* network. - SS1 access, password and lumens level. - TC1 set temperature, current temperature and over-temp 3. Transmits data-log to *SCADA* workstation. 4. Receives TC1 set temperature and SS1 password reset from *SCADA* workstation.
M12. Master LCD	1. Displays local temperature and local RTD resistance.
M13. SCADA Workstation	1. Shows data-log in a specific GUI format. 2. Allow operator to change temperature set-point. 3. Allow operator to change SS1 password.
S9. SS1 Transceiver	1. Relays information to MTU *transceiver* about SS1 access and lumens level.
T9. TC1 Transceiver	1. Relays information to MTU *transceiver* about temperature set point, current temperature and over-temperature.

Table 5 MTU Function Block Interconnections and Signal Description

Source Block	Destination Block	Signals	Wires/ Terminals	Protocol
M1. AC from Mains	M2. Mains Protection	AC $120V_{rms}$ @60Hz	power cord to screw terminals	N/A
M2. Mains Protection	M3. Transformer	AC $120V_{rms}$ @60Hz w/ protection	two wire to screw terminals	N/A
M3. Transformer	M4. Rectifier Circuit	AC $20V_{rms}$ @60Hz	three wires to screw terminals (1 to ground)	N/A
M4. Secondary Protection	M7. Rectifier Circuit	AC $20V_{rms}$ @60Hz w/ protection	PCB traces	N/A
M7. Rectifier Circuit	M6. Voltage Regulators	+/-$14V_{DC}$ smoothed voltages	PCB traces	N/A
M6. Voltage Regulators	M5. Temp. Signal Conditioning	+/-$11.6V_{DC}$ +$3.3V_{DC}$	PCB traces	N/A
M6. Voltage Regulators	M11. Master MCU	+$6V_{DC}$	PCB traces	N/A
M6. Voltage Regulators	M12. Master LCD	+$3.3V_{DC}$	PCB traces to header pin connector	N/A
M6. Voltage Regulators	M9. Master Transceiver	+$3.3V_{DC}$	PCB traces to header to wire connector	N/A
M5. Temp. Signal Conditioning	M8. Input Protection	0.25V to 3.25V filtered signal	PCB traces	N/A
M8. Input Protection	M11. Master MCU	0.25V to 3.25V filtered signal w/ protection	PCB traces	N/A
M11. Master MCU	M8. Input Protection	+3.3V from MCU for protection	PCB traces	N/A
M10. RTD Temp. Sensor	M5. Temp. Signal Conditioning	+100mV to +139mV unfiltered signal	PCB traces	N/A

Table 5 MTU Function Block Interconnections and Signal Description (continued)

Source Block	Destination Block	Signals	Wires/ Terminals	Protocol
M11. Master MCU	M13. *SCADA* Workstation	RS232 signals	DB9 to USB connector	RS232
M13. *SCADA* Workstation	M11. Master MCU	RS232 signals	DB9 to USB connector	RS232
M11. Master MCU	M12. Master LCD	Parallel digital signal	header to wire connectors	N/A
M11. Master MCU	M9. Master Transceiver	Serial digital signal	header to wire connectors	User defined packet protocol
M9. Master Transceiver	S9. SS1 Transceiver	Modulated wireless communication	wireless @2.4GHz	*XBee* protocol
M9. Master Transceiver	T9. TC1 Transceiver	Modulated wireless communication	wireless @2.4GHz	*XBee* protocol
T9. Master Transceiver	M9. TC1 Transceiver	Modulated wireless communication	wireless @2.4GHz	*XBee* protocol
S9. Master Transceiver	M9. TC1 Transceiver	Modulated wireless communication	wireless @2.4GHz	*XBee* protocol

6.2. SCADA Temperature Control Unit 1 (TC1) Specification

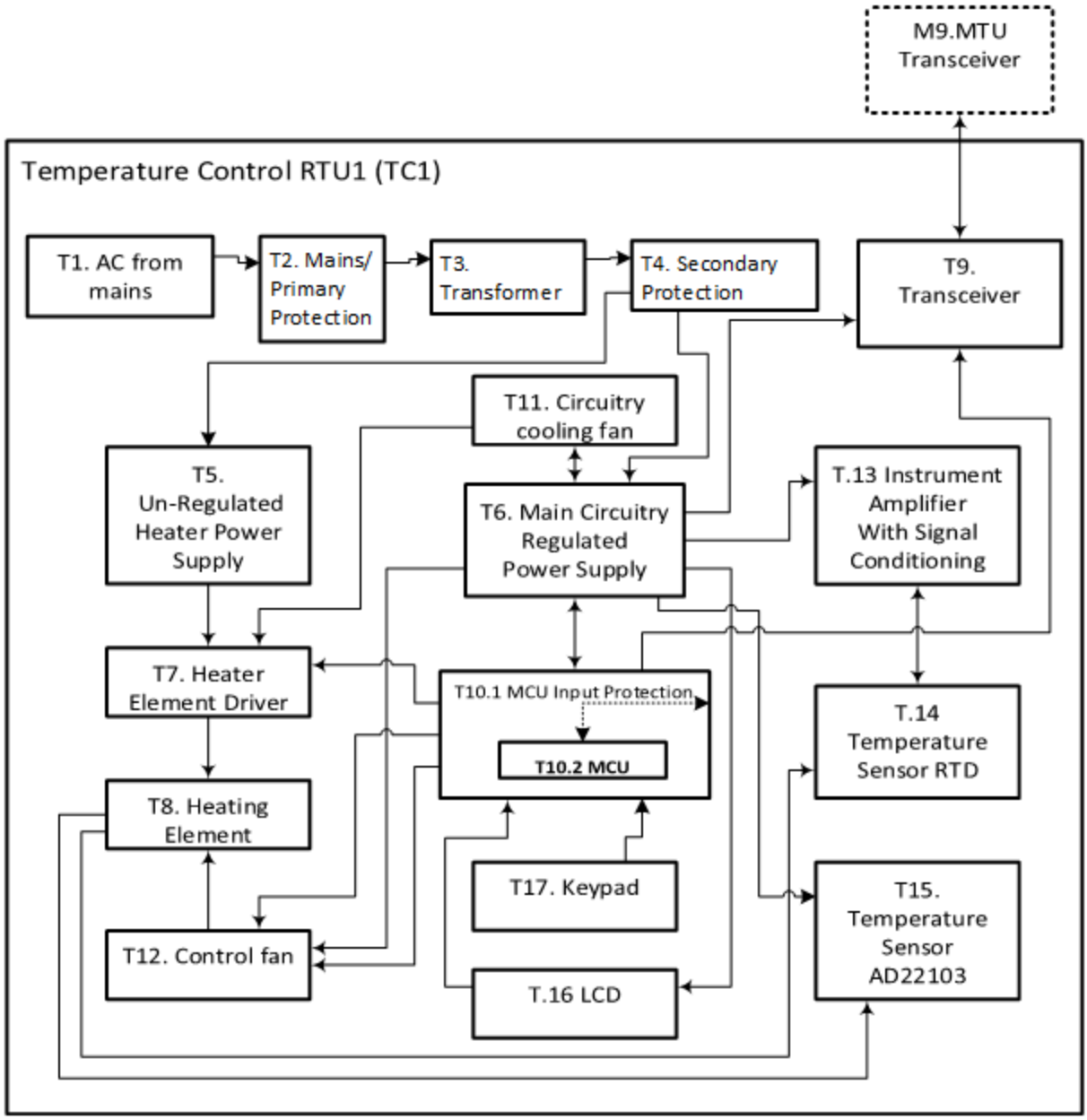


Figure 6 TC1 Function Block Diagram

Table 6 TC1 Function Block Descriptions

Functional Block Name	**Block Functions**
T1. AC from Mains	1. The power cord will plug into a socket for $120V_{rms}$ @ 60Hz to power the MTU circuitry.
T2. Primary/Mains Protection	1. For circuitry protection and safety. 2. 3.15 amp Slo-Blo fuse in case of shorts. 3. Thermal fuse and metal oxide varistor to handle voltage surges. 4. SPDT switch for on/off.

T3. Transformer	- Signal Primary / Dual Secondary
	- Convert $120V_{rms}$ @60Hz (2.5Amp) to:
	  - Winding 2A: $+20V_{rms}$ C.T @60Hz.
	  - Winding 2B: $+43V_{rms}$ C.T @60Hz.

T4. Secondary Protection	1. 1 amp Slo-Blo fuses for both sides of secondary. 2. Shunt series resistor and capacitor across secondary.
T5. Un-regulated Power Supply	1. High voltage diode rectifier bridge 2.Provides + 38 V DC full-wave rectified output 3. High voltage smoothing capacitor 9400uF. (2.15 Volts Ripple output)
T6. Main Circuitry. Regulated power supply	1. Adjustable regulator for +10.75V DC. 2. Adjustable regulator for -10.75V DC. 3. Adjustable regulator for +6.4V DC. 4. Fixed regulator for +3.3V DC. 1. Diode full-wave rectifier circuit. 2. Smoothing capacitors for positive and negative voltages. >1Vp-p Ripple

T7.Heater Driver	1. PNP and NPN Current mirror configuration for *PWM* control
	  - 0 - 3 mA Driving current
	  - BJT current limiting/shunt circuit IC heater max: = 3 A

T8.Heating Element	1. MJ1000 Power transistor driving +38Vcc @3A max
	  - 8-OHM series resistor
	  - 0.33- OHM Shunt resistor (Shunt at IC=3A)
	  - 0t to 3mA Base current from Driver
	  - Thermostat shut off. Opens at 100 Degrees C
	2. Large heat sink
	  - Connected with control fan

T9. Master Transceiver	1.*XBee*-PRO S3B *transceiver* for wireless communications.
T10.1 input protection	1. -Schottky diodes across MCU 3.3V and ground. -Resistors to limit current into MCU.

Table 6 TC1 Function Block Descriptions (continued)

Functional Block Name	**Block Functions**
T10.2 TC1 MCU	1. Calculates: Temperature using ADC on two inputs for RTD and AD22103. 2. Measures fan speed from pulse generator on processor timer capture input. 3. *PWM* controls for heater driver and control fan. 4. Transmits Data to MTU transceiver via T9. 5. Receives temperature setpoint from Keypad T16.
T11. Circuitry cooling fan	1. 12V Dc Fan for board level cooling requirements. - run on +6.4 V regulator for low air flow.
T12. Control fan	1. 12V DC fan for heater control - *PWM* controlled from MCU - 12 volt supply @200mA - Built in Tachometer output and *PWM* control for Brushless AC motor.
T13. Instrument Amplifier W/ signal conditioning	1. 1 mA constant current circuit for RTD. 2. Fully differential Instrumentation Amplifier. - With built in *Span and Zero* functionality Av = 17.8dB Vref = -6v 3. Second Order Sallen-Key lowpass filter, Fc = 15Hz @ -3dB
T14. RTD Temp. Sensor	1. RTD to detect temperature in the heater output air flow into enclosure. 0 to 150 Degrees Celsius range
T15. AD22103	1. Integrated circuit temperature sensor for heat sink temperature monitoring. - Overheat shut off protection
T16. Master LCD	1. Displays: - Temperature Set point - Temperature measurements: RTD and AD22103 - Control fan RPM
T17. Keypad	1. Used to enter temperature set point into MCU.

Table 7 TC1 Function Block Interconnections and Signal Description

Source Block	Destination Block	Signals	Wires/ Terminals	Protocol
T1. AC from Mains	T2. Mains/Primary Protection	AC $120V_{rms}$ @60Hz	power cord to screw terminals	N/A
T2. Mains/Primary Protection	T3. Transformer (primary winding 1)	AC $120V_{rms}$ @60Hz w/ protection	two wire to screw terminals	N/A
T3.1 Transformer - Secondary Winding 2A - $20V_{rms}$ C.T	T4.1 Secondary Protection	AC $20V_{rms}$ @60Hz	three wires to screw terminals (1 to ground)	N/A
T3.2 Transformer - Secondary Winding 2B - $43V_{rms}$ C.T	T4.2 Secondary Protection	AC $20V_{rms}$ @60Hz	three wires to screw terminals (1 to ground)	N/A
T4. Secondary Protection	T6. Main Circuitry. Regulated power supply	AC $20V_{rms}$ @60Hz w/ protection	PCB traces	N/A
T6. Main Circuitry. regulated power supply	T13. Instrument Amplifier W/ Signal conditioning	Op amp VCC/VEE: +/- $10.75V_{DC}$ Current Source Ref: $+3.3V_{DC}$	PCB traces	N/A
T6. Main Circuitry. regulated power supply	T10. Master MCU	Power source: $+6.4V_{DC}$	PCB traces	N/A
T6. Main Circuitry. regulated power supply	T16. Master LCD	Power source: $+3.0V_{DC}$	PCB traces to header pin connector	N/A
T6. Main Circuitry. regulated power supply	T9. Master Transceiver	Power source: $+3.3V_{DC}$	PCB traces to header to wire connector	N/A
T6. Main Circuitry. regulated power supply	T12. Control Fan	Power source: $+12V_{DC}$	PCB traces to header to wire connector	N/A
T6. Main Circuitry. regulated power supply	T15. Temp Sensor AD22103	Vref: $+3.3V_{DC}$	PCB traces to header to wire connector	N/A
T6. Main Circuitry. regulated power supply	T11. Circuitry cooling fan	Power source: $+6.4V_{DC}$	PCB traces to header to wire connector	N/A

Table 7 TC1 Function Block Interconnections and Signal Description (continued)

Source Block	**Destination Block**	**Signals**	**Wires/ Terminals**	**Protocol**

T5. Instrument Amplifier W/ signal conditioning	T10. MCU With input protection	0.25V_{DC} to 3.3 V_{DC} filtered signal	PCB traces	N/A
T4. Secondary Protection	T5.Un-Regulated Heater Power Supply	AC 43 V_{rms} @60Hz w/ protection	Wires and connectors	N/A
T6. Main Circuitry. regulated power supply	T7. Heater element Driver	Vcc: 6.4V_{DC}	PCB traces	N/A
T5.Un-Regulated Heater Power Supply	T8. Heater Element	+38 V_{DC}	PCB traces To screw terminal to wires	N/A
T7. Heater element Driver	T8. Heating Element	+0 to 3mA Driving Current to base pin of MJ1000	PCB traces to header to wire connector	N/A
T10.1. Input Protection	T10.2. TC1 MCU	0.25V_{DC} to 3.25V_{DC} filtered signal w/ protection	PCB traces	N/A
T10.2 TC1 MCU	T10.1. Input Protection	+3.3V from MCU for protection	PCB traces	N/A
T13. Instrument Amplifier W/ signal conditioning - Current Source	T14. RTD Temp. Sensor	Constant: 1mA	PCB traces to header to wire connector	N/A
T14. RTD Temp. Sensor	T10. [illegible]	[illegible]	[illegible]ire twisted [illegible]tor to header to [illegible]aces	N/A
T10. TC1 MCU	T7. Heater element driver	*PWM*: 0-3.3V	PCB Traces	N/A
T10. TC1 MCU	T12. Control fan	*PWM*: 0-3.3V	PCB traces to header to wire connector	N/A
T10. TC1 MCU	T12. TC1 LCD	Parallel digital signal	header to wire connectors	N/A
T17. Keypad	T10. TC1 MCU	Parallel digital signal	header to wire connectors	N/A

Table 1.4. TC1 Function Block Interconnections and Signal Description

Table 7 TC1 Function Block Interconnections and Signal Description (continued)

Source Block	Destination Block	Signals	Wires/ Terminals	Protocol

T15. Temp Sensor AD22103	T10. TC1 MCU	Analog Signal 0 to +3.3V_{DC}	Wire to header to PCB traces	N/A
T12. Control Fan	T10. TC1 MCU	Tachometer Pulse	Wire to header connector	N/A
T10. TC1 MCU	T9. TC1 Transceiver	Modulated wireless communication	wireless @2.4GHz	*XBee* protocol
T9. TC1 Transceiver	M9. Master Transceiver	Modulated wireless communication	wireless @2.4GHz	*XBee* protocol

6.3. SCADA Security System Unit 1 (SS1) Specification

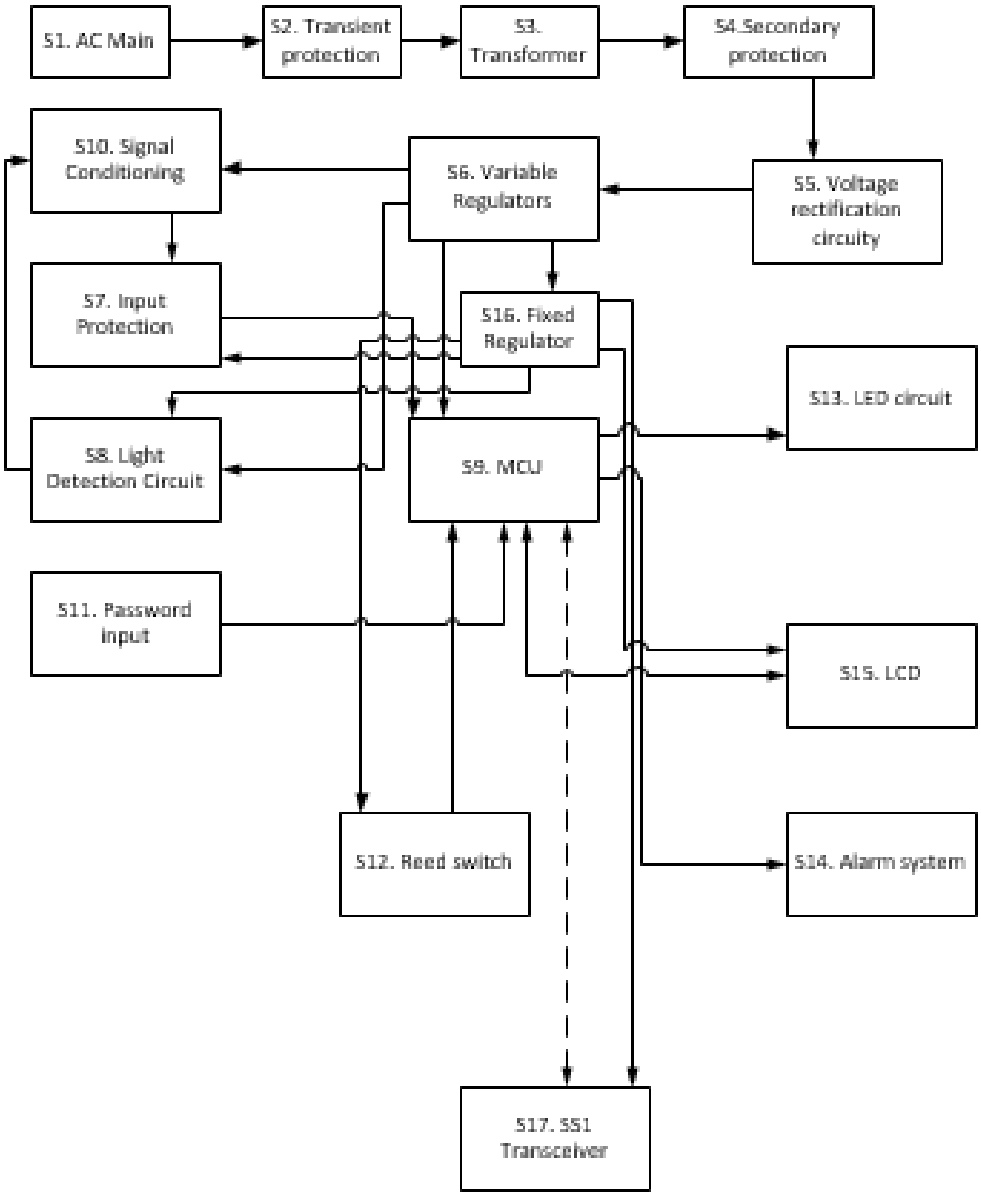


Figure 7 SS1 Function Block Diagram

Table 8 SS1 Function Block Descriptions

Functional Block Name	**Block Functions**

S1. Mains	1. Provide 120 VRMS @ 60Hz through a 3-wire power cord connected to the wall outlet.
S2. Transient Protection	1. Protection circuitry to account for current surges. 2. A fuse and a varistor for voltage surges.
S3. Transformer	1. Convert voltage from mains to 20 Vrms.
S4. Secondary Protection	1. Two protection fuses in series with transformer for current surges.
S5. Voltage Rectification circuit	1. Convert ac voltage to dc voltage.
S6. Regulators(Variable)	1. Provide voltage levels to power the MCU along with light detection components.
S7. Input protection	1. Protect inputs of MCU for voltage spikes and transients
S8. Light Detection Circuit	1. Provide 1 mA current for LDR. 2. Measure voltage across parallel network.
S9. MCU	1. Configured in slave mode to communicate with master MTU 2. Accepts user password entry via keypad and determines authorization of entry 3. Sound a buzzer and turn on LED(simultaneously) to indicate unauthorized entry. 4. Determines brightness level from LDR circuit 5. Update MTU Unit if: - Access attempted - Entry authorized/unauthorized - Lights(on/off)

Table 8 SS1 Function Block Descriptions (continued)

Functional Block Name	**Block Functions**

S10. Signal Conditioning	1. Unity gain Butterworth filter to filter out signal frequencies above 120 Hz 2. *Span and Zero* circuit to convert output voltages of LDR circuit to acceptable voltage range for MCU inputs (0-3.3V)
S11. Password input	1. Keypad that is used for password entry
S12. Reed switch	1. Magnetic contacts that determine if the user attempts entry. 2. Entry without authorization will prompt MCU to sound buzzer and turn on LCD.
S13. LED circuit	1. Turns on in conjunction with a buzzer to indicate unauthorized entry.
S14. Alarm	1. Indicate unauthorized entry (buzzer).
S15. LCD	Displays: 1. User prompt to enter password for access 2. Entered password correct or not
S16. Fixed Regulator	1. Supplies power to run SS1 *transceiver* 2. Supplies power to turn on LCD 3. Acts as input to the Reed Switch 4. Provides contrast to the LCD

Table 9 SS1 Function Block Interconnections and Signal Description

Source Block	Destination Block	Signals	Wires/ Terminals	Protocol
S1. AC Main	S2. Transient protection	120Vrms	Power cord to screw terminal	N/A
S2. Transient protection	S3. Transformer	120Vrms	two wire to screw terminals	N/A
S3. Transformer	S4. Secondary protection	20 Vrms	three wire to screw terminal centre to ground	N/A
S4. Secondary protection	S5. Voltage rectification circuitry	20 Vrms	PCB traces	N/A
S5. Voltage rectification circuitry	S6. Variable Regulators	+/- 14Vdc rectified	PCB traces	N/A
S6. Variable Regulators	S9. MCU	+6Vdc	PCB traces	N/A
S6. Variable Regulators	S16. Fixed Regulator	+6Vdc	PCB traces	
S6. Variable Regulators	S8. Light Detection circuit	+/- 12.7Vdc ,	PCB traces	N/A
S6. Variable Regulators	S10. Signal Conditioning	+/-12.7Vdc	PCB traces	N/A
S16. Fixed Regulator	S8. Light Detection Circuit	+3.3Vdc	PCB traces	N/A
S8. Light Detection circuit	S10. Signal Conditioning		PCB traces	N/A
S10. Signal Conditioning	S7. Input Protection	$0.25V_{DC}$ to $3..3V_{DC}$	PCB traces	N/A
S7. Input Protection	S9. MCU	$0.25V_{DC}$ to $3.3V_{DC}$	PCB traces	N/A

S9. MCU	S14. Alarm system	3.3V	PCB traces	N/A
S11. Password Input	S9. MCU	Parallel digital signal	header to wire connector	N/A
S9. MCU	S13. LED circuit	3.3V	PCB traces	N/A
S9. MCU	S15 LCD	3.3Vdc	PCB traces to header pin connector	N/A
S9. MCU	S17. SS1 Transceiver	Modulated wireless communication	Wireless @ 2.4 GHz	

7. SCADA HUMAN-MACHINE INTERFACE (HMI)

The HMI of the *SCADA* system is divided among the workstation operator as well as the RTU technician. At the MTU, the operator will run the Graphical user interface (GUI) on the workstation in order to monitor the data acquired from the RTU and controls for TC1 set temperature and SS1 password. There is an LCD located on the MTU to display the temperature inside the MTU enclosure.

At the RTU there is the security system unit where the technician will require a password entry in order to access the RTU. The password entry process will require a keypad and an LCD. The TC1 allows the technician to set the temperature locally, requiring another keypad and LCD. The TC1 display has a menu structure that has the options of displaying fan revolutions per minute (RPM), *PWM* levels and temperature level of both sensors.

7.1. MTU OPERATOR HMI

The HMI for the MTU operator is the GUI seen in Fig.9 and the LCD seen in Fig.8.

Figure 8 MTU LCD

The LCD at the MTU outputs what the calculated RTD resistance is and the calculated temperature is of the MTU enclosure.

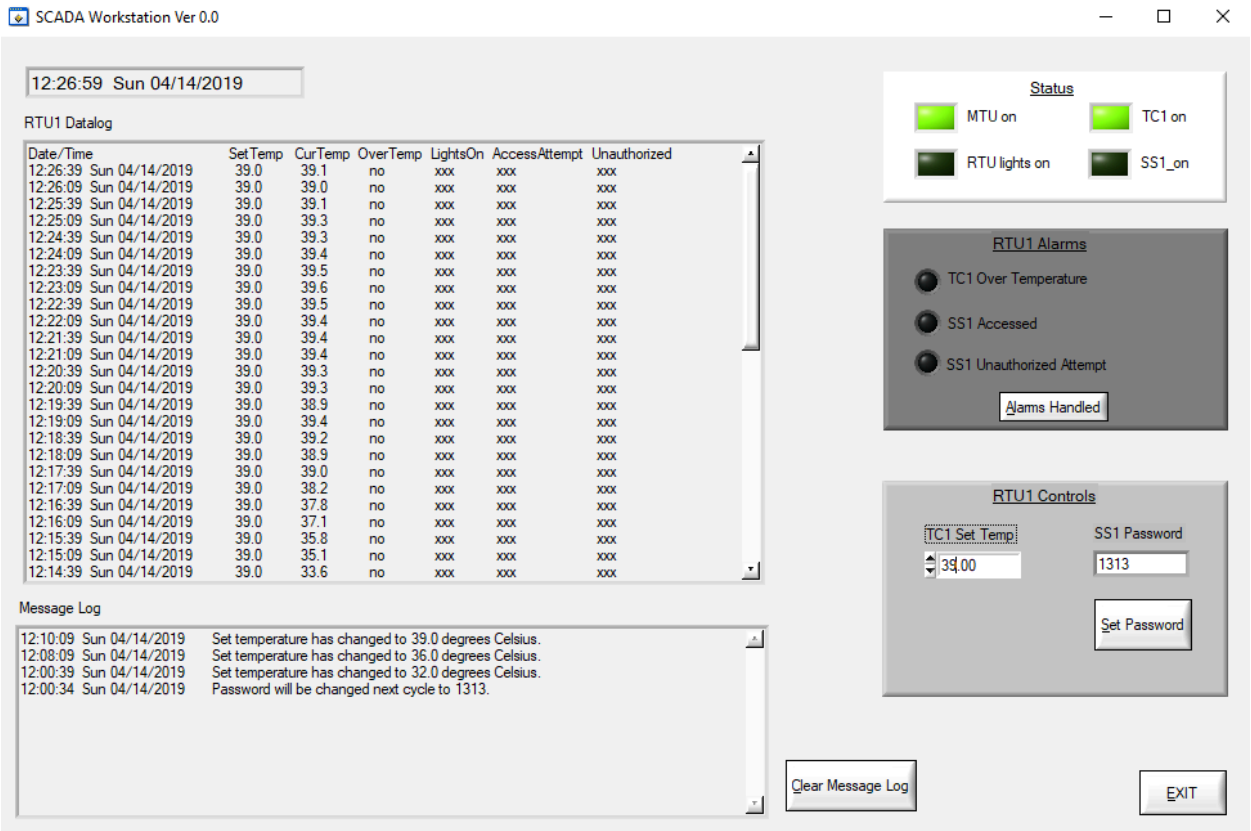


Figure 9 SCADA System GUI

The datalog for the RTU is formatted to show the set temperature, the current temperature as well as the RTU statuses. The message log communicates to the operator based on the performed operations of the GUI and RTU. There are green LED indicators for on/off statuses as well as red LED indicators for alarms. The Alarms Handled button allows the operator to clear the alarms when they have been handled. The Set Password button is used when a password is entered in above it. The program only sends passwords from 0000 to 9999 forward to the RTU. The EXIT button allows the operator to exit the session and close down program safely.

7.2. RTU TC1 TECHNICIAN HMI

The TC1 RTU uses the keypad to perform all of the local functions and the LCD to display the current temperature as well as perform multiple other features as described in fig.10.
The keypad will be used to:

- Enter new set temperature
- Display menu screen (pressing '*' will display menu using an interrupt)

- Select a menu item:
    - 1. Display fan RPM
    - 2. Enter new set point
    - 3. View *PWM* levels of the fan and heater for 2 seconds

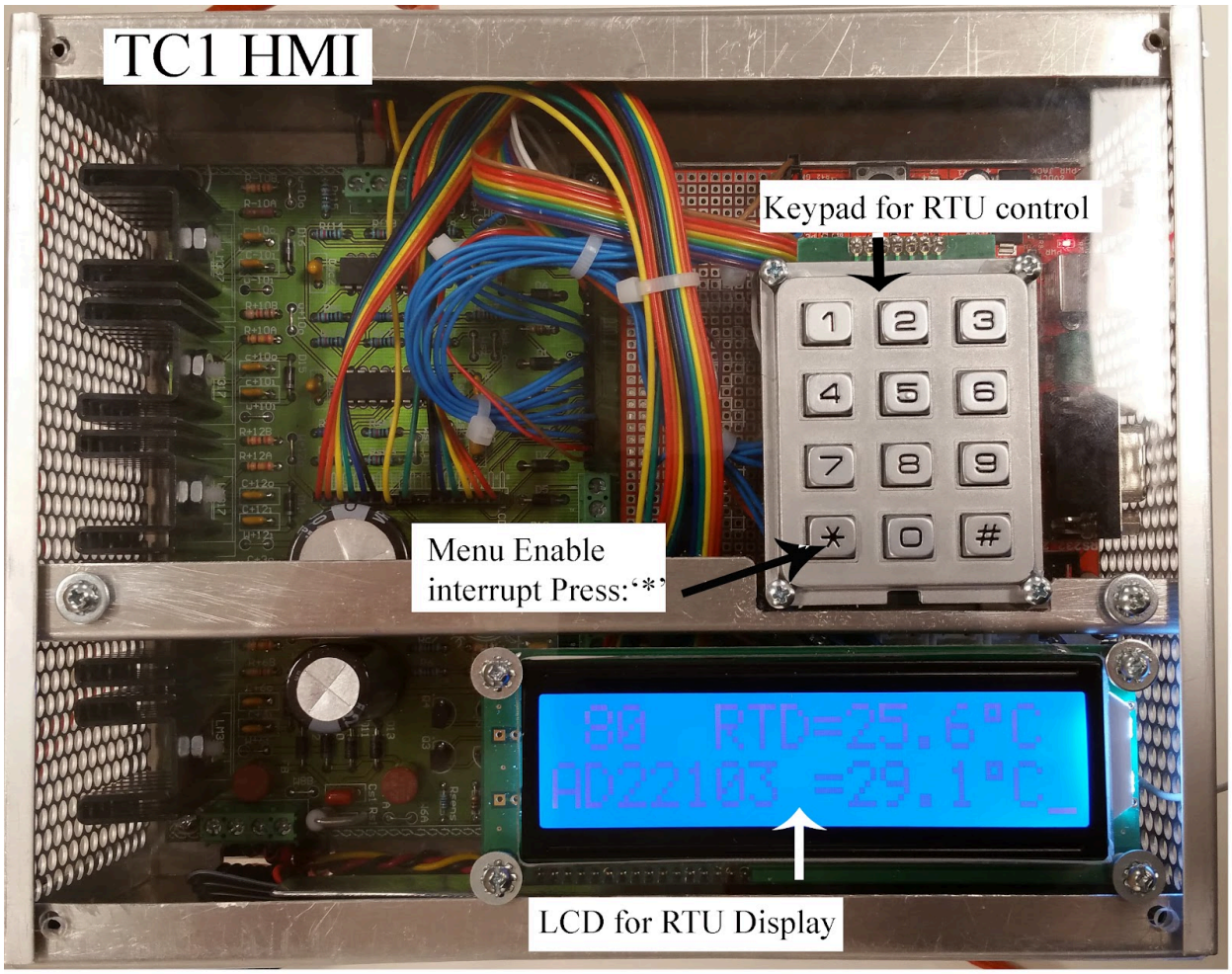


Figure 10 TC1 HMI

Display 1
Temperature Display main screen.
With next adjustment downcounter
This is the Default display screen during normal operation

Display 2
Menu structure
With interrupt trigger. Press the '*' key on the keypad to access this menu.

Display 3
Menu Item: 1 - Fan Rpm Display
When displaying menu Structure pressing '1' will display fan RMP speed, Pressing '*' will exit this function.

Display 4
Menu item: 2 - New Temperature Entry
When displaying menu Structure, Pressing '2' will display this notification screen

Display 5
Menu item: 2 - Enter new value
When in "Display 4" begin entering new temperature, this screen will appear with instructions and the new value.

Display 6
Menu item: 3 - Display current PWM Setting
When displaying menu structure pressing '3' will display values stored in memory used to set PWM levels. Display lasts 2

Display 7
Tx-Complete notification - With setpoint
This Screen will display every time a transmission is recieved and completed

Figure 11 TC1 RTU LCD Display Structure

7.3. RTU SS1 TECHNICIAN HMI

The SS1 RTU uses the keypad to allow user to enter password for access and the LCD to:

- Prompt user to enter password
- Notify user if password input is correct or not

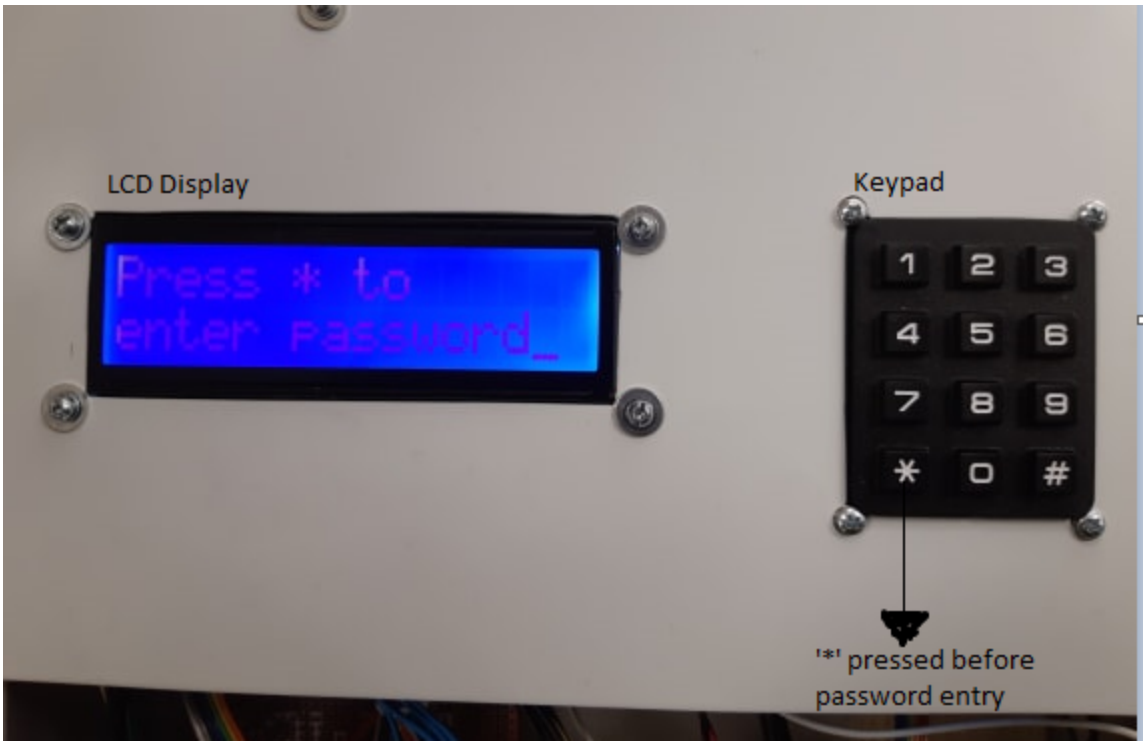


Figure 12 SS1 Front Panel

SS1 LCD Display structure

Display 1: Default state during normal operation

Display 2: User prompted to enter password

Display 3: User entered correct password

Display 4: User entered incorrect password

Figure 13 SS1 Panel Options

8. ELECTRONICS SYSTEM DESIGN

The Electronics hardware design is discussed in full in this section and all circuit schematics are provided in Appendices A,B and C for reference, this section will contain the following:

For the MTU, the electronics system design is broken down into:

- Power supply transformer.
- Power supply smoothing capacitors and fuses.
- Voltage regulator power dissipation.
- RTD temperature sensor.
- Temperature sensor signal conditioning circuitry.

For TC1, the electronics system design is broken down into:

- Power Supply
- Heating Element
- Heating element driver
- Instrumentation Amplifier (in-amp) and RTD circuitry
    - Current Source
    - In-amp with span-zero
    - *Sallen–Key* Low pass filter.
- Microprocessor and peripherals
    - LCD
    - Keypad
    - AD22103 Temperature Sensor

For SS1, the electronics system design is broken down into:

- Power supply transformer.
- Power supply smoothing capacitors and fuses.
- Voltage regulator power dissipation
- Light Dependant Resistor (LDR), light detection sensor
- Light detection signal conditioning circuitry
- Alarm system(LED and Buzzer)
- Password entry
- Door sensor

8.1. MTU SYSTEM DESIGN

The MTU electronics are based around the RTD temperature sensor and the power supply required for the operation of the RTD, the RTD signal conditioning circuitry, the MCU, the *XBee transceiver* and the LCD.

8.1.1. MTU Power Supply Transformer

The power supply transformer will need to be chosen based on voltage and current requirements in Table 10**.**

Table 10 MTU Voltage Requirements and Current Consumption

Component	**Voltage Requirement**	**Current Consumption**
LM324 op-amps	+/-11.6 V	6 mA
XBee transceiver	3.3 V	215 mA (transmitting) 29 mA (idle)
LCD	3.3 V	43 mA
ARM Cortex M4	6 V	87 mA (all peripherals enabled) 60 ma (estimated in MTU) 40 mA (all peripherals disabled)

The LM337/LM317 adjustable regulators have a voltage drop-out around 1.5 Volts. The voltage drop due to the rectifier diodes are approximately 1.4 Volts. A peak-to-peak voltage over A transformer with a minimum of 20 Volts center tapped and 1A should be chosen in order to safely supply 351mA when at maximum current consumption while being able to provide 14.2 V_{pp} on each side of the secondary.

8.1.2. MTU Power Fuses and Smoothing Capacitors

Fuse considerations will be done based on maximum current consumption. The secondary fuses will need to be slo-blo with an Amperage rating well over 500mA, resulting in 1A slo-blo fuses being chosen. The turns ratio of the transformer is 1:6. This means that there will be $1/6^{th}$ the amount of current on the mains transformer closed loop in comparison to the secondary side. A 500mA slo-blo fuse will handle the 83.3mA without any issue. In the case of any short circuit or current surge on the MTU circuitry, the secondary fuse will blow first.

The current consumption of ARM Cortex in this application should be around 60 mA due to the amount of peripherals that are being used. The *XBee transceiver* will be idling in between transmission resulting in idle current consumption state being dominant. Therefore, the average current consumption of the circuitry will be 138mA.The calculation for ripple voltage is given by [:

$$V_{pp} = \frac{I}{(2 * f) * C}$$

Where: V_{PP} is the peak to peak ripple voltage (V)
f is the frequency of the AC power (Hz)
C is the capacitance of the smoothing capacitor (F)

With an average current consumption of 135mA on the positive voltage side of the full wave rectifier circuit, a 2200uF capacitor will be chosen. The ripple voltage peak-to-peak will be $256mV_{pp}$ resulting in a 128 mV drop due to the voltage ripple.
The negative voltage side of the full wave rectifier circuit has 3mA being consumed at any given time. A 1000uF capacitor results in voltage peak-to-peak of $12.5mV_{PP}$ with a voltage drop of 6.25mV at any given time due to the ripple voltage.

8.1.3. MTU Voltage Regulator Power Dissipation

The voltage regulator power dissipation calculations will be on the LM337 and LM317 adjustable regulators, for the +-11.6V, as well as the L7806 and LM1117 (3.3V) fixed regulators. The LM1117 regulator has a recommended maximum input voltage of 15 Volts. This means that the LM1117 regulator will be connected to the 6V regulator and that any current seen by the LM1117 will also be seen by the L7806.

Table 11 Regulator Power Calculations

Regulator	**Voltage drop**	**Current through**	**Power dissipated**
LM337 (-11.6V)	1.1 V	3 mA	3.3mW
LM317 (+11.6V)	1.1 V	3 mA	3.3mW
L7806 (+6V)	6.7 V	132 mA	884mW
LM1117 (+3.3V)	2.7 V	72 mA	194mW

The voltage seen on the input of the LM337, LM317, and L7806 is an magnitude 12.7 Volts (14.1V from power supply – 1.4V from rectifier diodes). The voltage input of the LM1117 regulator is 6V from the L7806. The voltage drop is given by V_{in}-V_{out} and the current through is given by the current consumption of the components that are being supplied by the regulator.

8.1.4. MTU Temperature Sensor

The 100Ω RTD temperature sensor requires a sensing current between 0.3mA and 1mA in order to avoid self-heating. A non-inverting amplifier, seen in Fig.14. can be used with the RTD as the feedback resistor to create a constant current where I_i =3.3V/3.3kΩ=1mA=I_f. The 3.3V supply from the Olimex development board will be used as it includes decoupling capacitors for noise reduction.
It is important to note that the RTD measuring circuitry will need to have a high input impedance and needs to be fully differential. The compensation resistor, R_c, is equal to R_i in parallel to RTD. At any given temperature, RTD will be between 100Ω and 138Ω and is the dominating resistor in the calculation for R_c, so a value of 100Ω is chosen for R_c.

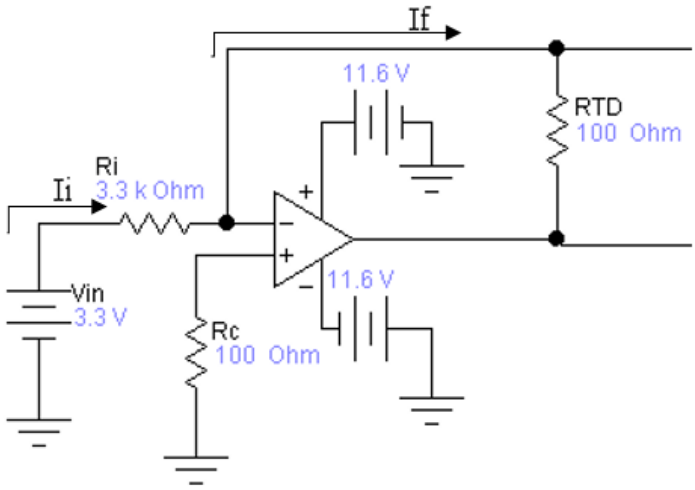


Figure 14 RTD 1mA Circuit

8.1.5. MTU Temperature Sensor Signal Conditioning Circuitry

The temperature sensor signal conditioning circuitry is composed of a:

- Instrumentation amplifier (*in-amp)*
- Low pass filter
- *Span and zero*

The in-amp is fully differential and has a high input impedance so it can be used to amplify the voltage signal seen across the RTD. The low-pass filter is used to reduce the amount of noise that is on the RTD voltage signal. The *span and zero* circuit is used to map the voltage signals of the RTD at temperatures between 0°C-100°C to 0V-3.3V in order to maximize the resolution of ADC used.

8.1.5.1 MTU Instrumentation Amplifier (in-amp)

The *in-amp* in Fig.15 has the voltage gain equation of:

$$A_v = \frac{V_{out}}{V_2 - V_1} = \left(1 + \frac{2R_1}{R_{gain}}\right)\frac{R_3}{R_2}$$

The input from the RTD will be between 100mV and 139mV with temperatures of 0°C and 100°C for 1mA **[2]**. The *in-amp* will be used to provide a voltage gain of 20 to spread the voltage signal bounds outwards before a *span and zero* circuit is applied.

R_3, R_2, R_1 and R_{gain} have been chosen as nominal resistor values with the help of the voltage gain equation of the in-amp. The *in-amp* will out voltages between -2V and -2.78V based on temperatures between 0°C and 100°C.

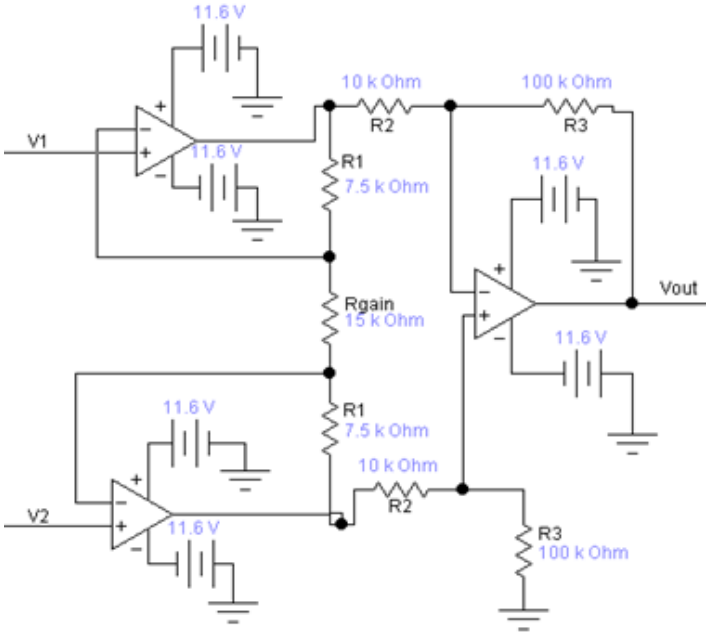
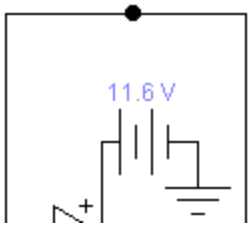


Figure 15 Instrumentation Amplifier(MTU)

8.1.5.2. MTU Low Pass Filter



A first order low pass filter seen in Fig.16 will be used to reduce the noise from the RTD voltage signal. The corner frequency is found through the equation **[3]**:

$$f_c = \frac{1}{2\pi RC}$$

Common nominal values of 10kΩ and 10μF were chosen for R and C. The result is a first order low pass filter with a corner frequency of 1.59Hz.

Figure 16 Low Pass Filter (MTU)

8.1.5.3. MTU Span and Zero

The *span and zero* circuit in Fig.17 works off of the theory of summing amplifiers **[4]**. The currents created from V_{dc} and V_{in} are summed and go through R_f creating a desired output voltage, V_{out}. The goal of this *span and zero* circuit is to map the input voltages from -2V and -2.78V to output voltages of 0V and 3.3V for the ADC. The desired output span is 3.3V from an input span of 0.78V. A voltage gain for the span is:

$$\frac{3.3V}{0.78V} = 4.23 = \frac{R_f}{R_1}$$

This is accomplished by setting R_f to 42.2kΩ and R_1 to 10kΩ.
The desired zero is 0V. From the span calculations, the zero of V_{out} is 8.44V from the span circuitry. To go from a zero of 8.44V to 0V, the V_{dc} voltage must provide -8.44V for the summing amplifier. This is done through setting V_{dc} to 2 volts and R_2 to 10kΩ. R_u and R_l in the *span and zero* circuit set up the required V_{dc} voltage of +2V.

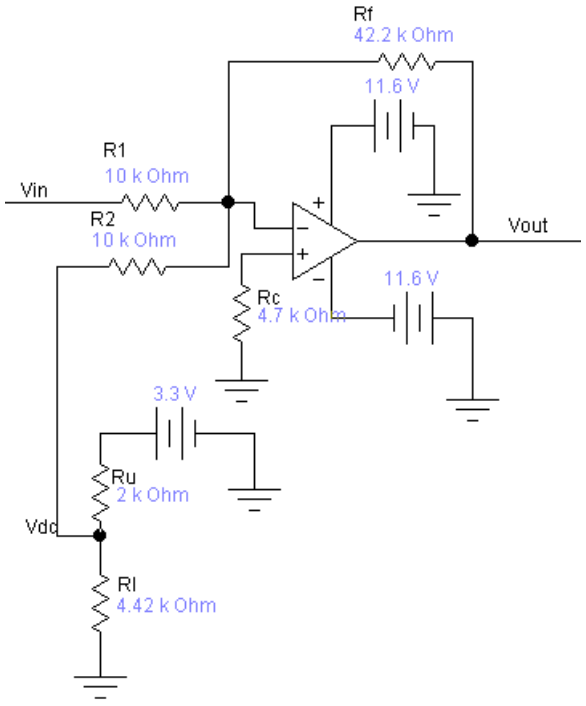


Figure 17 Span and Zero (MTU)

8.1.6. MTU PCB Design

The MTU PCB design is in Fig.18.

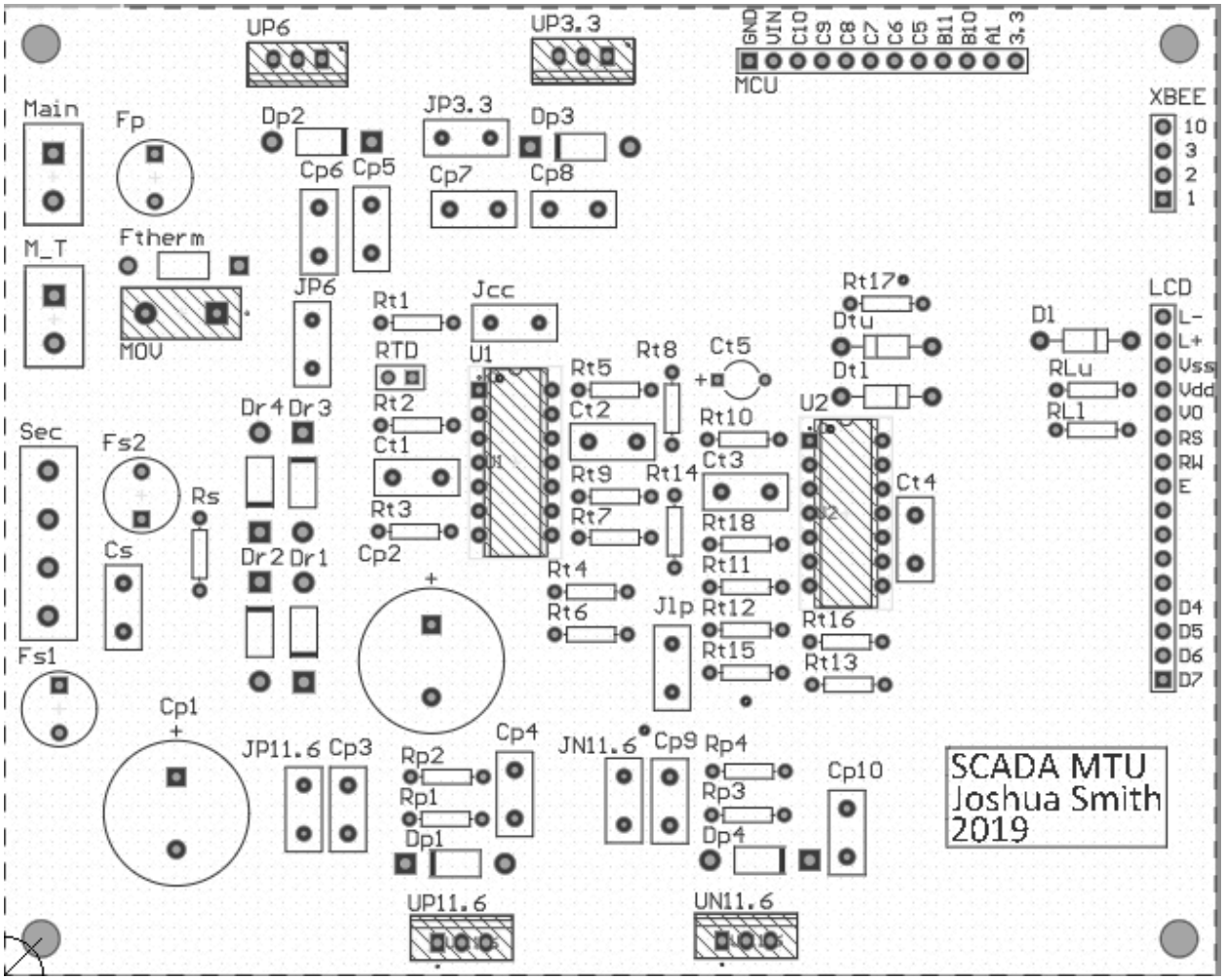


Figure 18 MTU PCB Design

The screw terminals relating to the transformer and mains are located on the left hand side of the PCB. The four heat sinks for the power supply regulators extrude off of the PCB at the top and bottom. Headers are used to link the MCU with the LCD and *XBee transceiver*.

8.2. TC1 RTU SYSTEM DESIGN

The first step in the temperature controller system was to do research into similar systems in order to determine the most practical approach. The most basic form of the temperature controller system would consist of a heating element which can be any device that dissipates energy such as a resistor, transistor or coil, and variable power source. A system of this form would be very simple and inexpensive, consisting of only a power supply and a dissipater. This would be an open-loop control system and not sufficiently complex enough for the purpose of this project. In order to demonstrate a broad range of techniques a closed-loop system will be designed including; a variable power supply, a variable cooling fan, two temperature sensors (one analog and one Integrated circuit), and a feedback loop using the Microprocessor. The LCD will be used for display and the *XBee* wireless transmitter will be used to connect to MTU.

8.2.1 TC1 POWER SUPPLY

The first system that will be designed is the power supply and it will be restricted by two elements that were acquired at no cost.

- The power transistor used to drive the heater (MJ1000)
    - **VCEO Max:** 60V
    - **IC Max:** 10A
    - **Power rating:** 90W
- The Transformer - salvaged from old audio equipment. (see pictorial)
    - Single Primary
        - 120VAC max.
        - 175W max.
        - 2.5 Amp max.
    - Triple Secondary
    - Used as shown in the pictorial. Red pins on actual transformer drawing, indicate pins used.
    - The 20 V c.t will be used to power T6. Main circuitry regulated power supply
    - The 55.5 V c.t will be used to power T5. Un-regulated heater power supply.

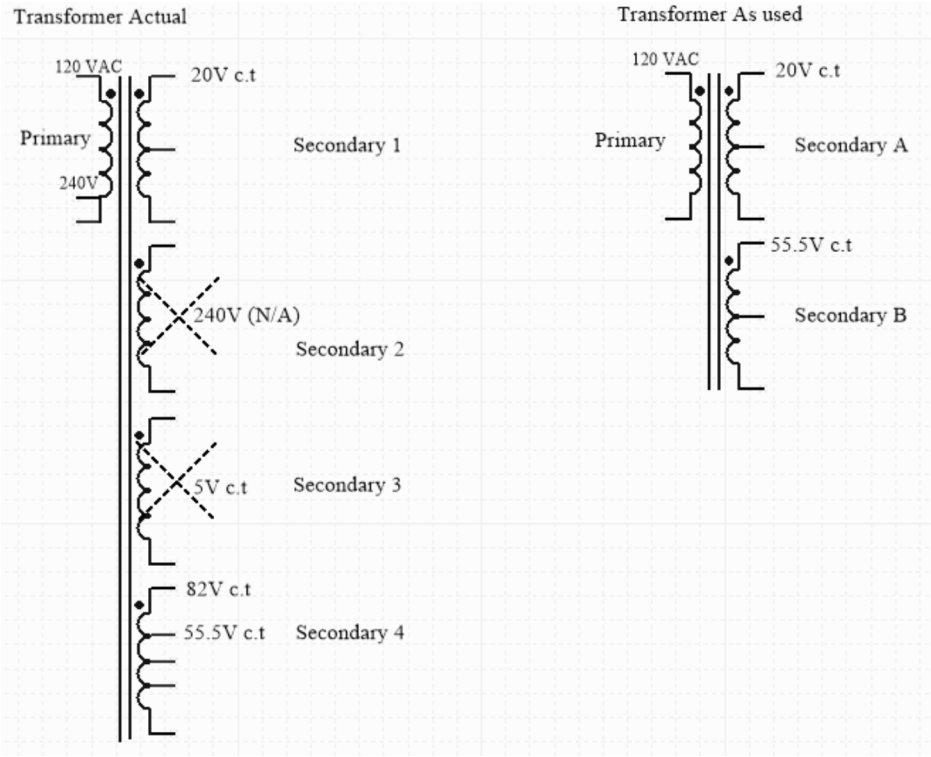


Figure 19 TC1 Transformer Schematic

Using the transformer in the above configuration the following power supplies were designed. The design was based on the Unregulated power supply found in “The art of electronics Third Edition" **[5]** And then combined with concepts from class to implement the regulated power supply used in the Main circuitry power supply.

8.2.1.1. Primary Protection

Consisting of:

- Delta General purpose 05DBAG5 Line filter to ensure consistent performance and accuracy of sensors.
- Double-pole / double-throw power switch.
- A 5 Amp slow blow fuse
- 130VAC Metal oxide varistor (MOV) to protect from power transients and spikes from mains,
    - The MOV must be connected in series with a thermal cut-off device (TCD) placed in close proximity to the MOV in order to protect the circuit in the case that the MOV is exposed to sustained current levels above its maximum rating where the thermal release can cause fire and smoke hazards. As required by UL1449 for Surge protective devices (SPD) and AC line applications **[6]**

8.2.1.2. Unregulated heater power supply

The power supply will provide 38 Volts at 3 Amps, since dual polarity is not required and voltage does not need to be regulated in order to heat the dissipater since it will work just as well if the voltage is a little unstable. An unregulated power supply and a simple two diode center tapped full-wave rectifier can be used. **[7]**

Converting 55 Volts c.t to peak power that can be used in the DC circuit the following equation is used.

$$PeakVoltage = Vrms * \sqrt{2} = \frac{55Vct}{2} * 1.414 = 38.89V$$

The peak voltage that the DC signal will reach is 38.89 however this will be lower when ripple voltage is considered.

8.2.1.3. Main Circuitry Power Supply

Consisting of:

- Secondary protection
    - 1 A Slow blow fuse (little fuse brand)
    - Shunting elements
        - 0.1uF Capacitor and 100 Ohm Resistor (½ W)
- Full-wave rectifier diode bridge
- Dual polarity outputs
- Linear voltage regulators
    - +12 V (Fan driver)
    - +10.75 V (op-amp VCC)
    - +6.4 V (MCU Vin and Current Mirror VCC)
        - System Cooling fan
    - +3.3V (*XBee* Vin and Current Source Vref)
    - -10.75 V (op-amp (VEE)

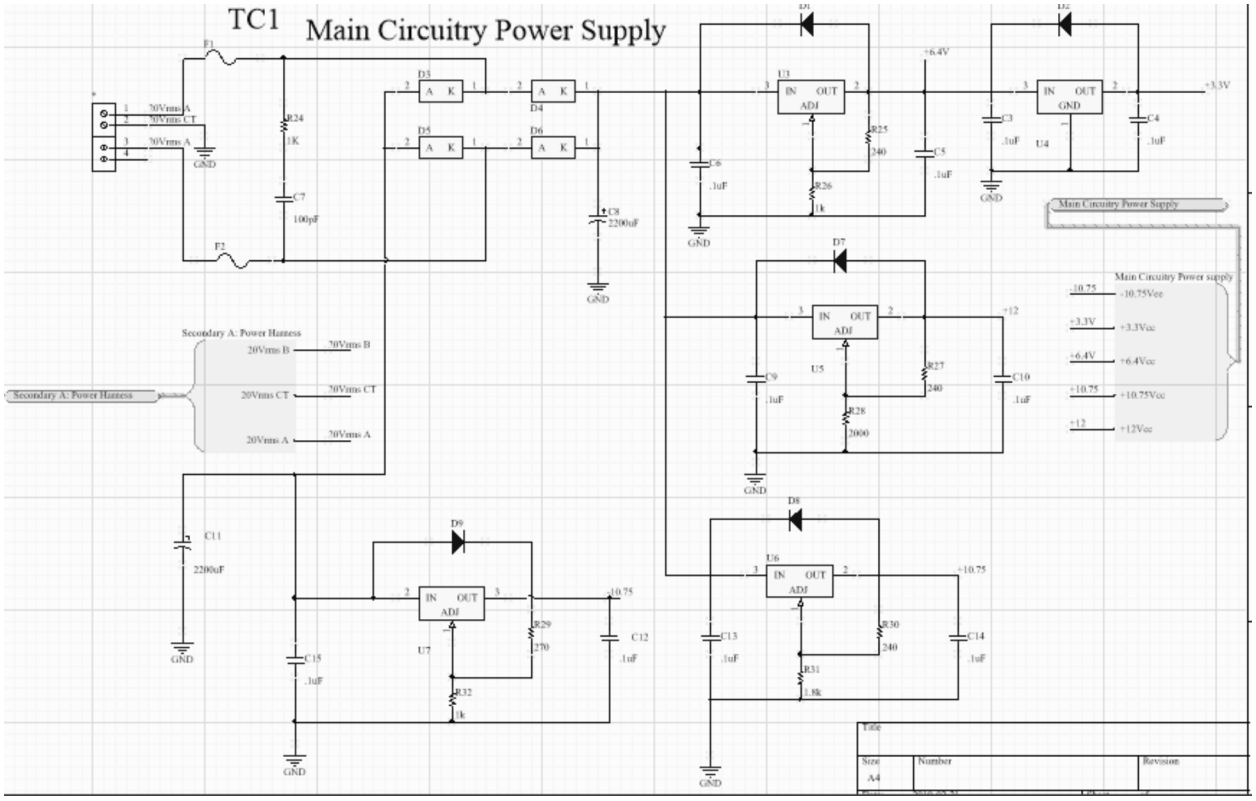


Figure 20 TC1 Main Circuitry Power Supply

8.2.2. HEATING ELEMENT

The initial design plan was to use the MJ1000 power transistor connected to a heat sink as the primary dissipative element, however there are restrictions that apply as well as additional components added for safety and stability. Due to the maximum current rating of the transformer, 3 Amps maximum was chosen for the driving current within the heating element circuit.

Using the power rating curve of the Mj1000 an ideal VCE was chosen for operating at IC = 3A.

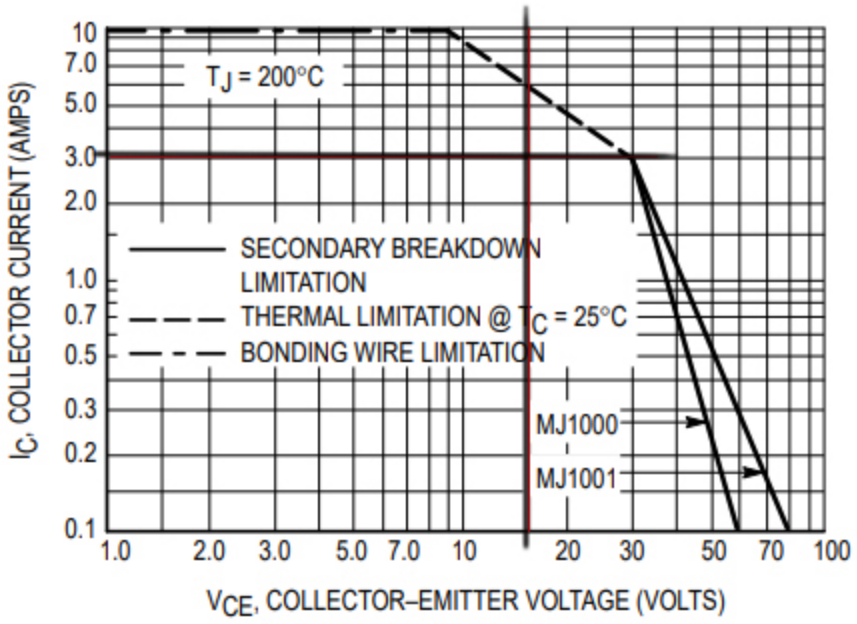


Figure 5. DC Safe Operating Area

Figure 21 From data sheet MJ1000

8.2.2.1. Series Resistor

If a series resistor is added the following operating region (marked with dark lines on the graph) can be designed. Using an 8 Ohm power resistor, and calculating for the Collector Emitter voltage of the transistor using a collector current of 3 Amps and a supply voltage of 38 volts.

$$IC = 3A$$
$$Vcc = 38V$$
$$VR = 8 * 3 = 24v$$
$$VCE = Vcc - VR = 38 - 24 = 14V$$

8.2.2.2. Thermal Cutoff device

A 100 degree C thermal cutoff device was added in series with Vcc and attached to the heat sink for safety purposes.

8.2.2.3. Shunting Circuit

As an added safety measure, a base current shunting circuit was added in order to limit the collector current to 3 Amps.

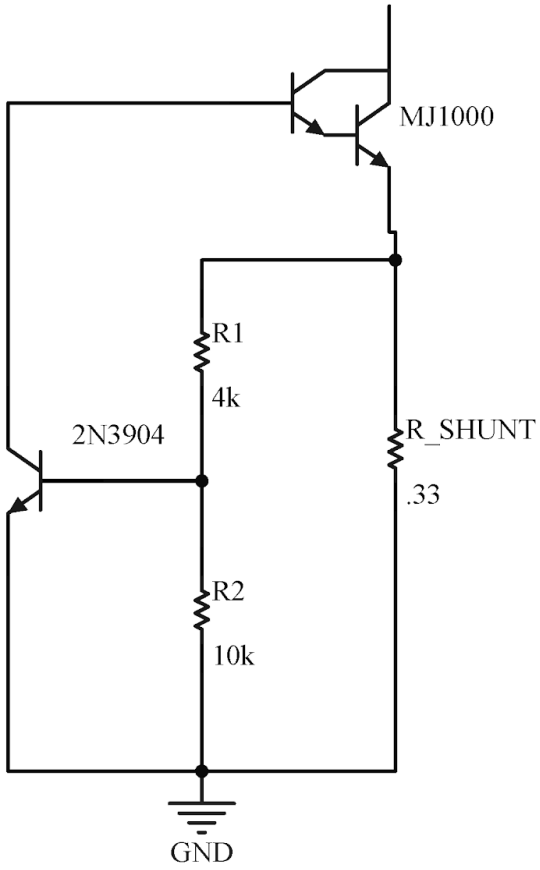


Figure 22 MJ1000 shunting circuit

The 2N3904 must begin to conduct when 3A is flowing through the collector of MJ1000, this is achieved by providing at least 0.65 volts across R2 when 3 Amps are flowing through R_SHUNT. 10K is selected for R2 due to convenience of calculations and availability.
As well as having a larger value of R2 prevents current draw.

Calculations are as follows:

$$I_C = 3A;$$

$$V_{R\_shunt} = R_{shunt} * I_C = .33 * 3 = 0.99V$$

If there is 0.99V across the shunt resistor when 3 amps flow and approximately 0.7 volts are required across R2. Using the voltage divider to determine R1 resistance and the known values of Vx and R2 where, Vx=0.7V; R2=10k;

$$R1 = \frac{R2 * (V_{in} - Vx)}{Vx} = \frac{10k * (0.99 - 0.7)}{0.7} = 4142.86\Omega$$

R1 is chosen at 4000 Ohms and the result is VR2 = 0.707V @ IC = 3 A by reversing the above equation, and the shunting effect will act before 3 A of collector current since the VBE breakdown voltage is nearer to 6.5V typical.

8.2.3. HEATING ELEMENT DRIVER

The heating element requires a driving circuit in order to use *Pulse Width modulation (PWM)* from the microprocessor. A current mirror driving circuit was chosen and is shown in the following image.

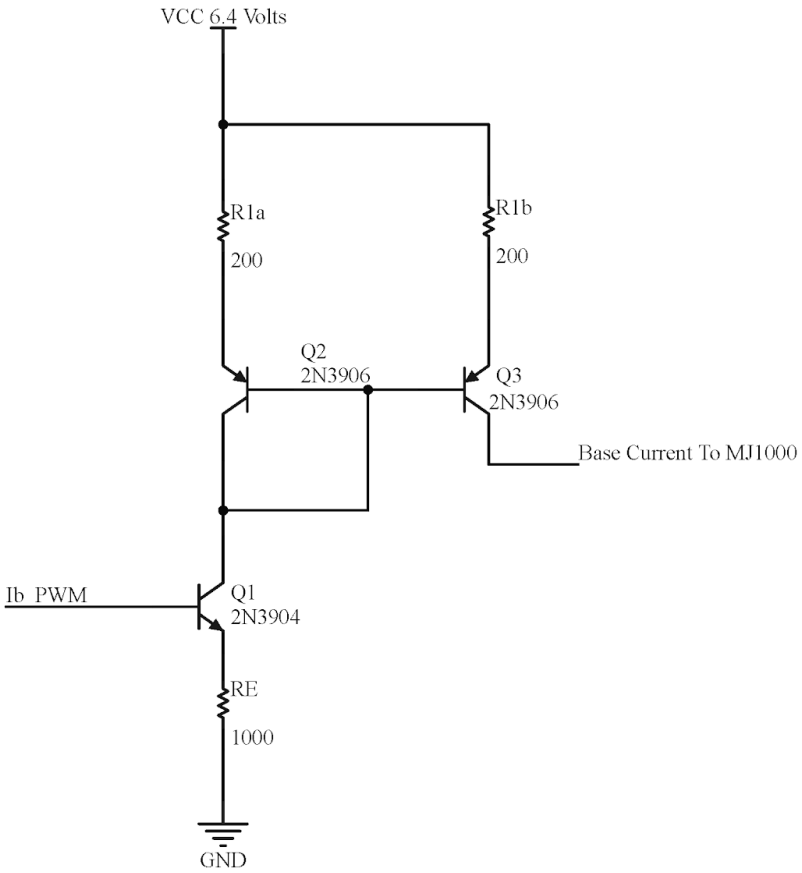


Figure 23 Current Mirror PWM Driving Circuit for Heater

In the above circuit the current driven by Q1 will be matched by Q3 Into the base of the MJ1000 Using the data collected in section **10.3.2 characterizing the heating element** the effective current gain(hfe) at 3 A for the heating element with the shunting circuit included is approximately 1000 therefore the MJ1000 base current(Ib) must be as follows.

In order to drive 3mA at max *PWM* of 3.3V, using the value for the emitter resistor in Fig.23, and the current gain from the characterization section for the MJ1000, The Required base current and the actual base current provided by the current mirror can be compared.

$$I_{b\_Required} = \frac{I_C}{hfe} = \frac{3}{1000} = 3mA$$

$$I_{b\_Actual} = \frac{Vp - V_{BE}}{R_E} = \frac{3.3 - 0.7}{1000} = 2.6mA$$

2.6mA will work since the current gain is unstable due to the heating of the transistor and since 3A is the maximum collector current, it is acceptable to operate below that level.

It must be noted that in order for *PWM* to be successful the current mirror Q2 and Q3 can not be driven into saturation. Using the circuit parameters to determine the voltage-current characteristics.

$$I_{C\_max} = \frac{Vcc}{R1a + RE} = \frac{6.4}{1200} = 5.33mA$$

and VCEO = 6.4V
And the operating region is between IC = 0 A and IC = 2.6mA

It can be seen from the graph below that the operating region will maintain the IC well below saturation since the operating region is in the middle of the V-I characteristic curve it does not approach the saturation region at IC=5.33mA where the switching speed of the transistor is too slow for *PWM* operations..

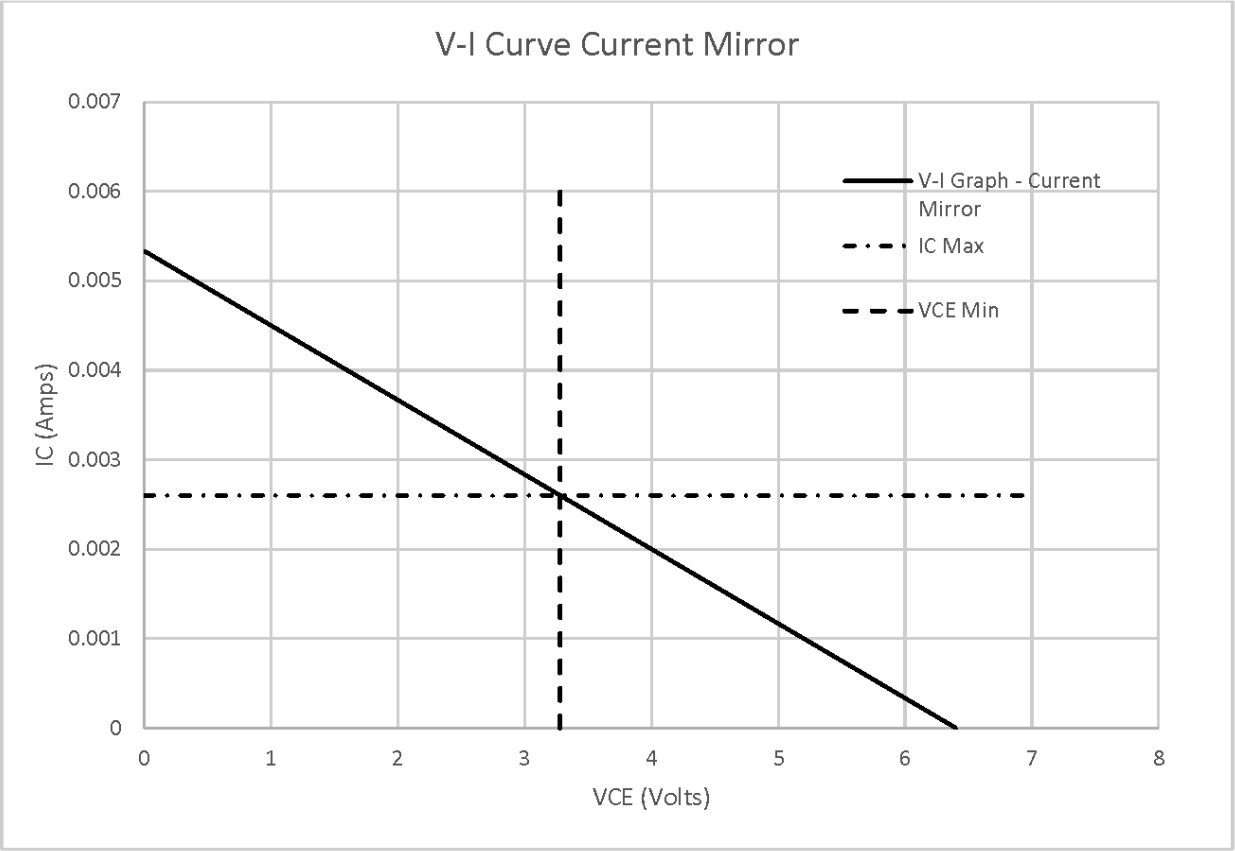


Figure 24 Theoretical Voltage-Current Graph displaying the Operating Region

8.2.4. IN-AMP and RTD CIRCUITRY

The following section will cover the design of circuitry required to interface the resistive thermal device (RTD) as the primary temperature sensor used. The requirements as set out in the elicitation process define the use of an ‘raw‘ analog sensor. A 100 Ohm 0 to 150 Degree Celsius RTD was chosen. The circuit used is a combination of popular RTD implementation methods. A current source will be used to supply a constant current through the RTD. As the RTD resistance changes the voltage across it will also change, this voltage is measured by a fully differential *in-amp* that provides zero loading effect. This is a well known 4 wire measurement technique and will be discussed in further detail in the following sections. The *in-amp* has *span and zero* functions built in and a *Sallen–Key* low-pass filter is attached at the output prior to the ADC pin of the MCU.

8.2.4.1. Current Source

In order to have accurate temperature measurements a precision constant current source will be used to create the voltages across the RTD. The current source should be unaffected by the changing load within the necessary range and should be stable at various temperature ranges.

The following circuit is based on a design from Microchip technology inc. **[8]**

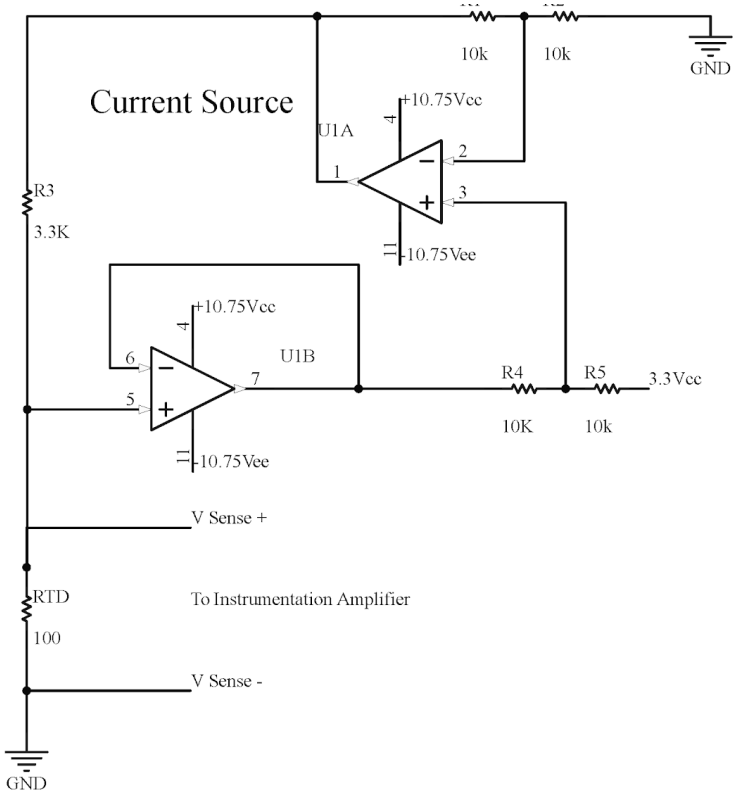


Figure 25 Constant Current Source Schematic

8.2.4.2. Explanation of constant current source

The voltage that is provided at Vcc is the Vref that will be held constant at the output of op-amp UA1, This voltage will be held across R3 and will provide the constant current through the RTD In this case
Vcc = 3.3 V, R3 = 3.3k Ohms
Constant Current = 3.3 / 3.3k = 1mA

1mA is the maximum current recommended by the RTD manufacturer in order to avoid self heating issues. This current value also provides a convenient multiplier for resistance calculations.

8.2.5. Instrumentation Amplifier

The *in-amp* is a fully differential op-amp configuration that provides zero loading effect and adjustable gain, the following configuration will allow for gain and zero adjustments. **[9]**

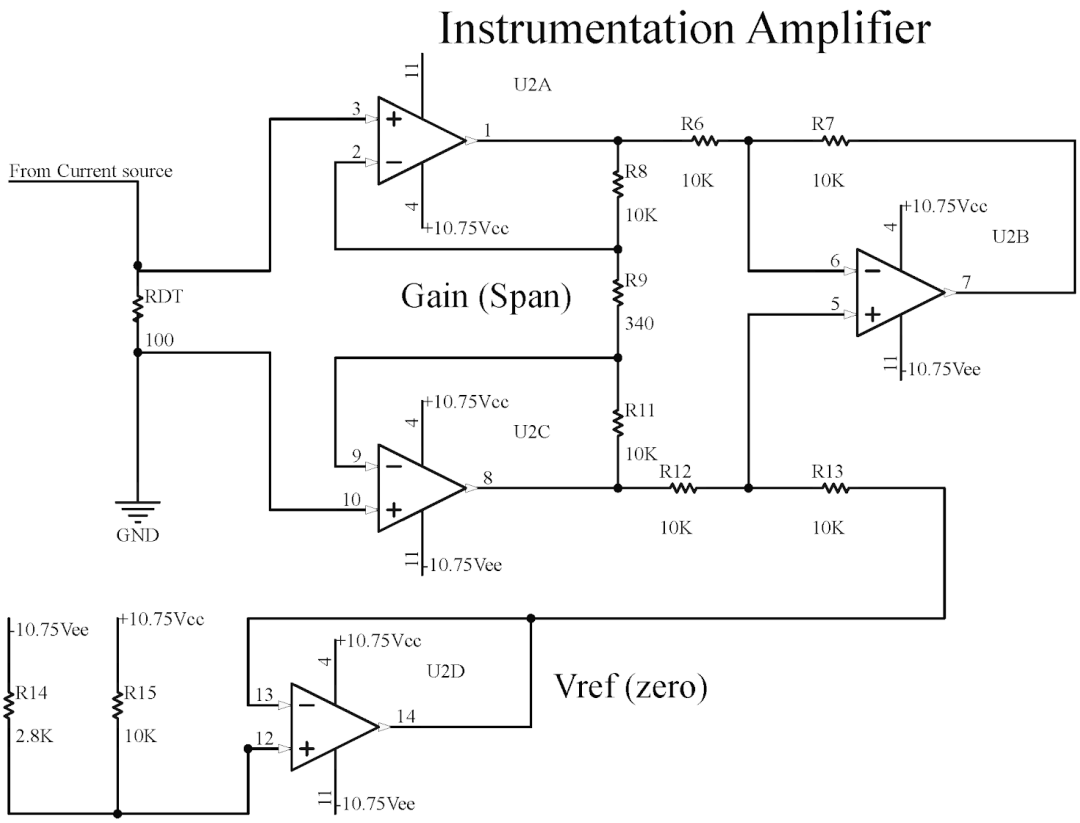


Figure 26 TC1 Instrumentation Amplifier Schematic

8.2.5.1. Span and Zero (Multiplier/gain and offset)

The *in-amp* must transfer the input voltage into an effective output range for use by the ADC of the microprocessor. The output range must be 0v to 3.3v to achieve maximum resolution of the ADC. In order to determine the input voltage range, a potentiometer was used in place of the RTD and a voltmeter was used to determine the full voltage range for the respective resistance range of the RTD.

For the 100 ohm RTD with a temperature range of 0 to 150 degrees celsius the resistance is expected to range from 100 to 150 ohms and thus the 1mA constant current will create a voltage range of 100 mV to 150 mV. Using the voltage ranges for Vin and Vout the *Span and zero* requirements can be determined using the straight line equation.

$$y = mx + b$$

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$ (straight line equation)

$$Span = Slope = m$$

$$m = (V_{out\_max} - V_{in\_min}) / (V_{in\_max} - V_{in\_min}) = (3.3 - 0) / (.150 - .100) = 66$$

$$Zero = Offset = b$$

$$b = Vout - m * Vin = 3.3 - (66 * .150) = -6.6V$$

For convenience these will be rounded down to a gain of 60 and a Vref of -6V. Since the expected range of temperatures within the heating circuit will not reach 150 Degrees the maximum range does not need to be exact.

The Instrumentation amplifier can now be designed for a gain of 60 with a zero intercept of -6V.

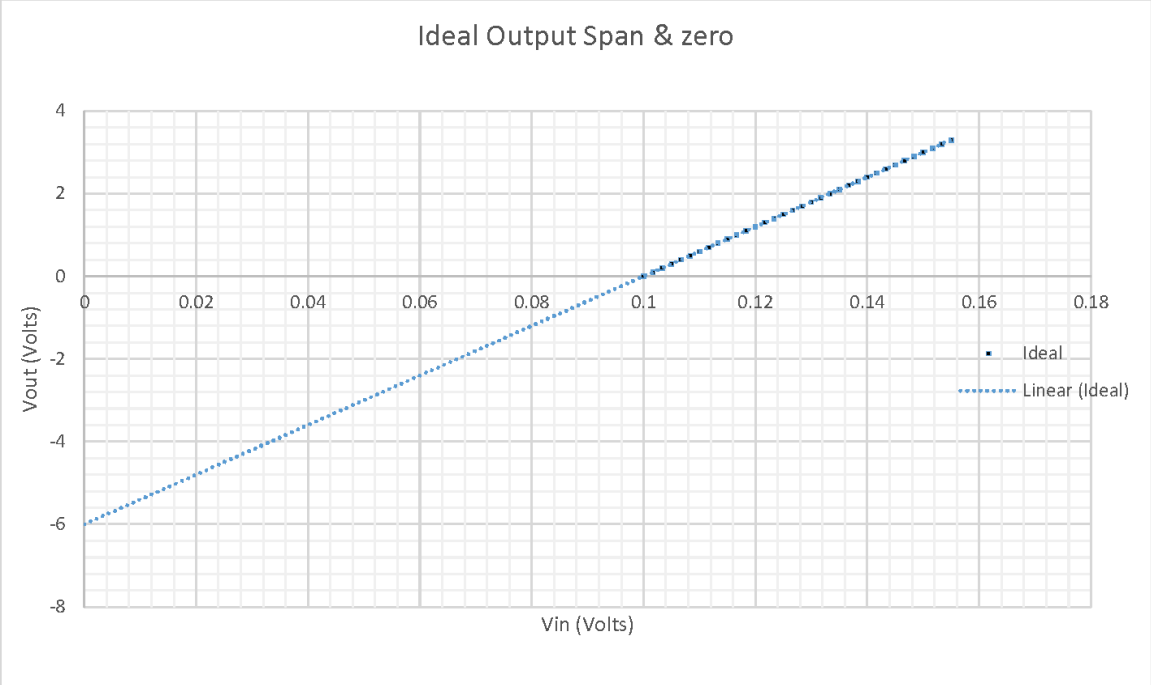


Figure 27 TC1 Instrumentation Theoretical Input to Output Voltage Graph

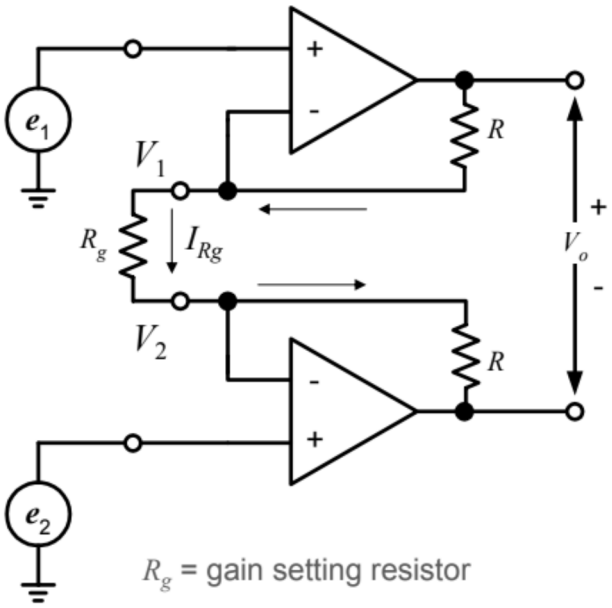


Figure 28 In-Amp First Stage

8.2.5.2. Span (Multiplyer)

In the above circuit, the voltage that is sensed by the non-inverting inputs of the two entry op-amps is then multiplied by the gain determined by the following equation. Assuming that all other resistors are equal then;

$$V_{out} = (e1 - e2) * \left(1 + \frac{2R}{Rg}\right)$$ where e1 and e2 are the voltages present at the non-inverting terminals, pin 10 and pin 3 on the schematic respectively. Rg is called the gain resistor since it determines the multiplication factor and is R9 in the schematics.
Therefore;

$$Gain = 1 + \frac{2R}{Rg} = 1 + \frac{20{,}000}{340} = 59.82 = 17.77dB$$

8.2.5.3. Reference Voltage (zeroing effect)

An additional op-amp configured as a unity gain buffer can be used to provide a zeroing effect as shown in the above figure. The entire circuit as shown has the *transfer function*:

$$V_{out} = (e1 - e2) * \left(1 + \frac{2R}{Rg}\right) + Vref$$

The above calculations and explanation are summarized in the following two images that are found in the “Industrial Electronics Signal Conditioners and Transmission” document **[9]**

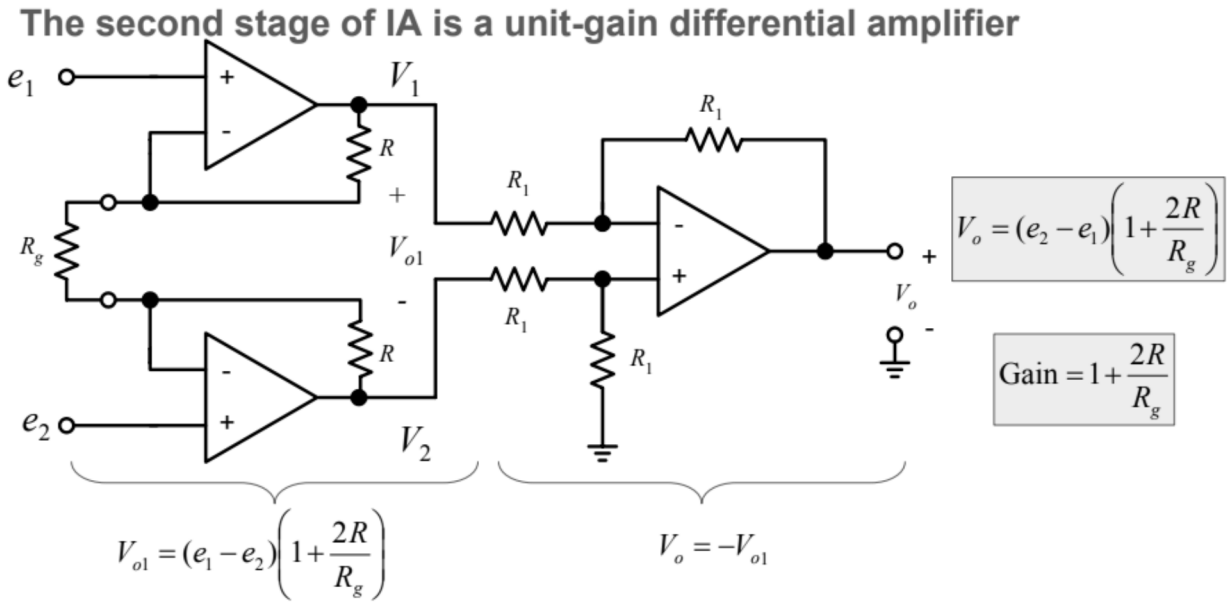


Figure 29 In-amp Second Stage

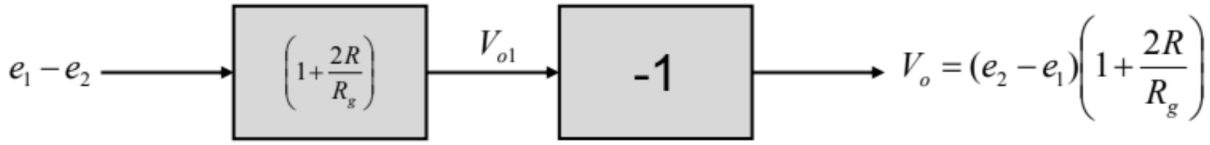


Figure 30 In-amp Transfer Function Block Diagram

8.2.6. *Sallen-Key* Low pass filter.

The signal must be filtered prior to being sampled by the ADC in order to prevent noise and aliasing effects. Temperature measurements do not not need to be taken very frequently, one

sample a second would be adequate. A low pass filter that has a cut off below 60Hz is required in order to eliminate the most likely cause of noise from the mains power supply as well as the 60hz fans running on the system. Following the design used in Microchip technology inc. document. **[8]** which uses a 10 Hz Second order Sallen-key low pass filter was used so 10 Hz was chosen as an acceptable cutoff frequency.

8.2.6.1. Sallen-key Low Pass Filter with Cutoff Frequency of 10Hz

A critically damped frequency response is ideal for sensor signals. In order to achieve a critically damped response an equal elements simplification technique will be used. Using 10k ohm 1% tolerance resistors due to availability the capacitor values must be calculated.

Knowing that: $\omega n = \frac{1}{R \cdot C}$; and $R1 \cdot C1 = \frac{1}{\omega n}$; $Fc = 10Hz$

Therefore $\omega_C = F_C * 2\pi = 10 * 2\pi = 62.83 Rad/s$

Frequency Scaling factor (FSF) = 1.55

$$\omega n = FSF * \omega c = 1.55 * 62.83 = 97.389 Rad/s$$

$$R1 \cdot C1 = \frac{1}{\omega n} = \frac{1}{97.389} = 0.01027 \text{ seconds}$$

$$C1 = \frac{\omega n}{R1} = \frac{0.01027 \text{ s}}{10,000} = 1uF$$

Using R = 10k ohms and C = 1uF the sallen-key was built as shown in the following schematic.

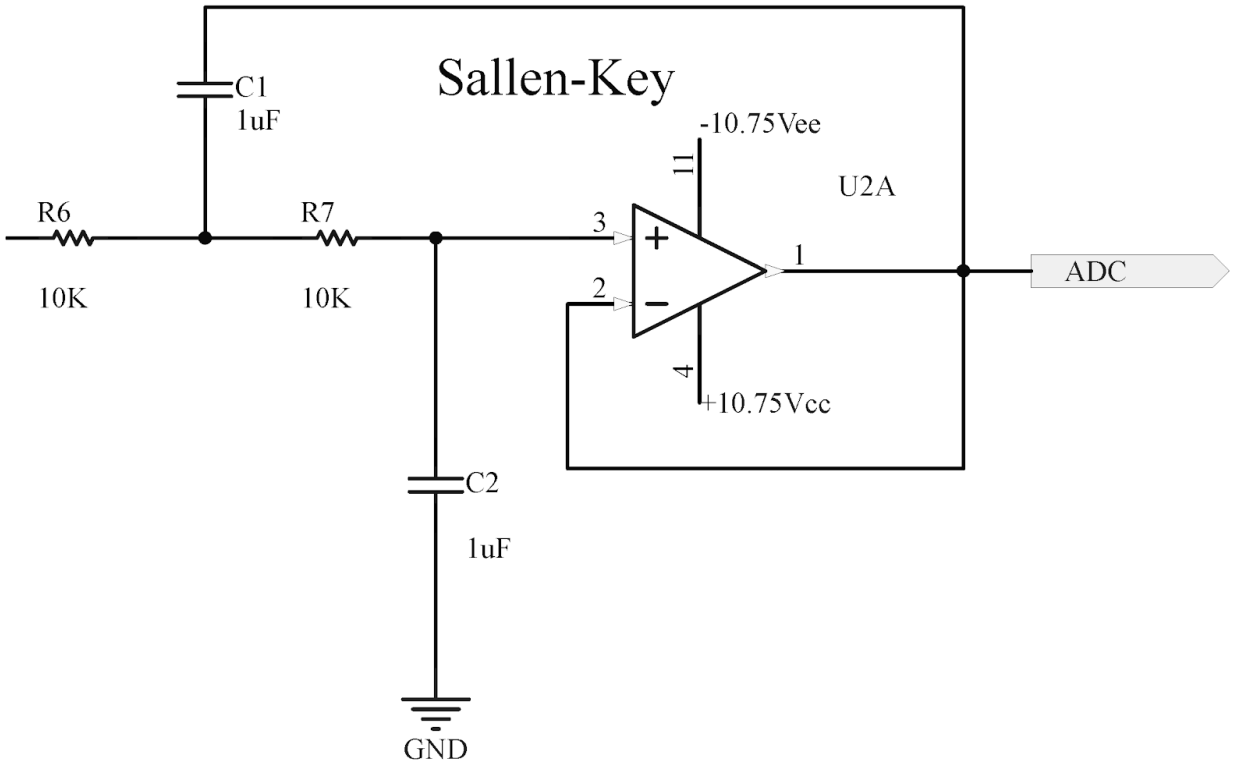


Figure 31 Sallen-Key Low Pass Filter Schematic

8.2.7. Heater Element Physical Design

The physical heater element was built using the components described in the above section. They are summarized as:

- Power Transistor
- Power Resistor
- Shunt Resistor
- Thermal Cut off

Components not mentioned above are also installed on the heat sink and are listed here:

- Heat sink
- Control fan
- AD22103 clamp
- Heat shroud

For a detailed image of the heater structure see section: **11.2 RTU TC1 SYSTEM LAYOUT**

The heat sink was made by cutting an appropriately sized section of aluminum out of a larger heatsink. The material was acquired in the SAIT Electronics Fabrication Laboratory in the materials inventory.

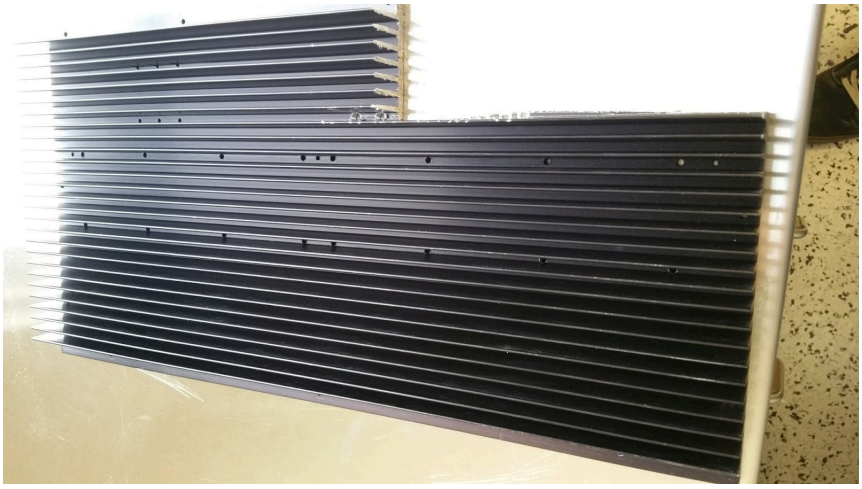


Figure 32 Heatsink Material

The sectioned piece of heat sink was drilled and assembled with the mechanical body that is comprised of:

- Clamp for holding the AD22103 temperature sensor onto the body of the power resistor
- Metal channel to guide the airflow through the fins on the heatsink
- Heat shroud to protect the user from the hot element
- Plastic isolation between heat shroud and element body

These mechanical features are illustrated in the following image

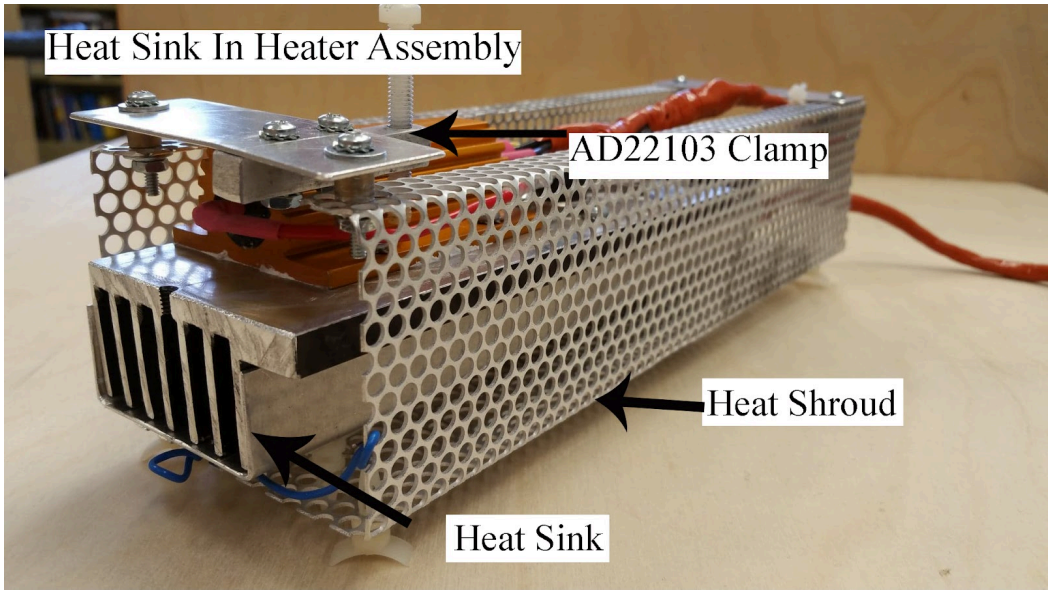


Figure 33 Heatsink Assembled

The components are attached and listed to the top of the heatsink in order to perform the required power dissipation and controls required for the design, these components can be seen in the following image.

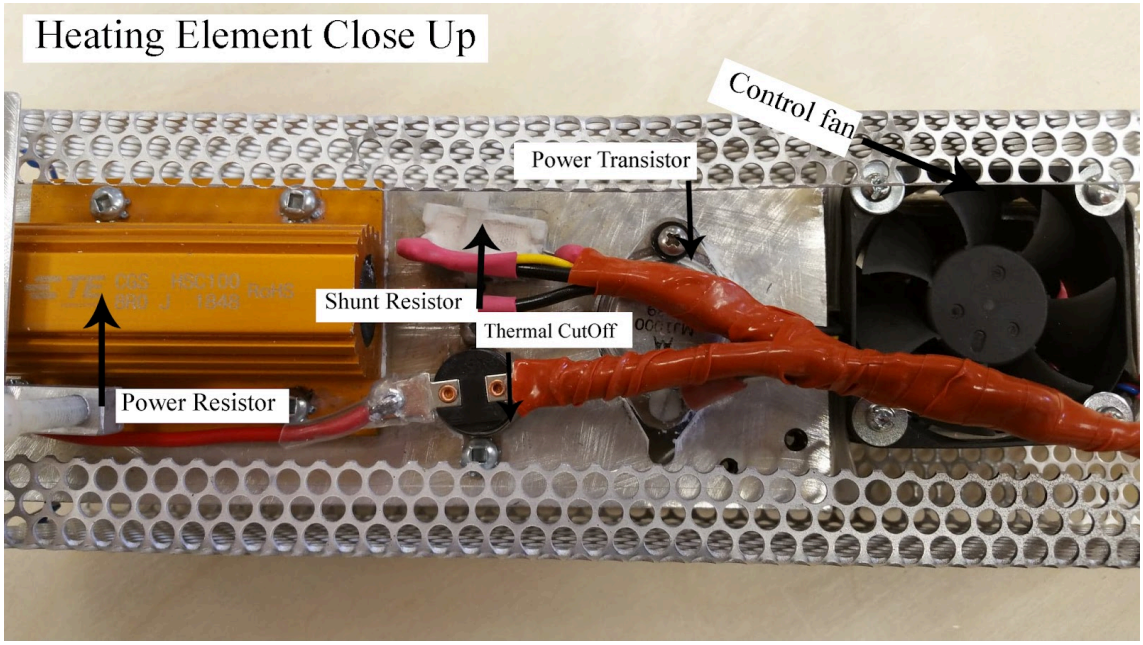


Figure 34 Heater with Attached Components

8.2.8. TC1 PCB DESIGN

Altium Designer is the software that was used to design the PCB, the following images show the PCB layout and the final product received from the manufacturer

Figure 35 TC1 PCB Design Altium Designer

Figure 36 TC1 Assembled PCB

8.3. SS1 RTU System Design

8.3.1 SS1 POWER SUPPLY TRANSFORMER

The power supply transformer will need to be chosen based on voltage and current requirements in Table 12**.**

Table 12 SS1 Voltage Requirements and Current Consumption

Component	**Voltage Requirement**	**Current Consumption**
LM317 op-amps	+/-11.6 V	20 mA
XBee transceiver	3.3 V	215 mA (transmitting) 29 mA (idle)
LCD	3.3 V	43 mA
ARM Cortex M4	6 V	90 mA

The selected fuses for the transformer are selected based upon maximum power dissipation rating. The secondary fuses were selected to be slo-blo with an Amperage rating well over 500mA. For SS1, 1A slo-blo fuses are selected.

8.3.2. SS1 LDR SENSOR

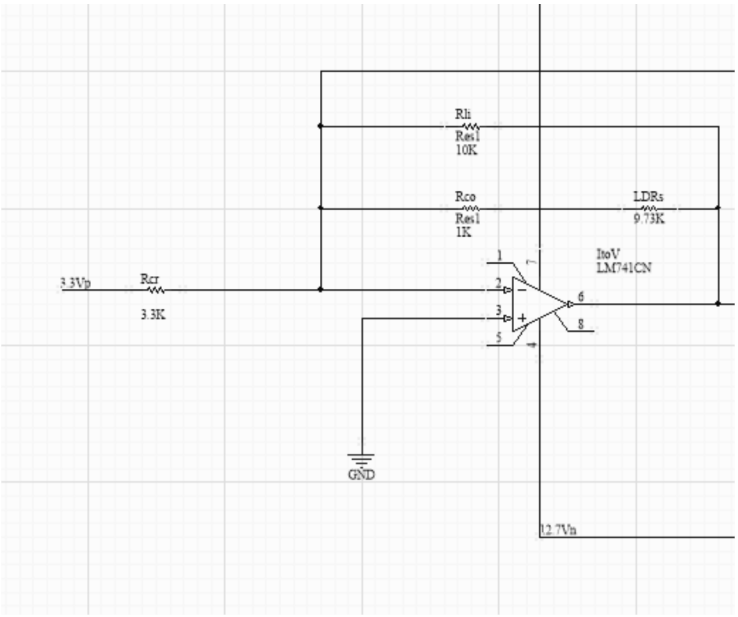


Figure 37 LDR Sensor Schematic

To provide constant current source, a standard inverting op amp configuration using 3.3Vdc and 3.3kΩ to produce 1 mA dc. The output voltage for this stage will be vary according to the resistance of the LDR. The LDR resistance increases as detected light levels decrease. Both upper and lower threshold light levels will be set in program to determine light and dark conditions.

8.3.4. Light Dependant Resistor (LDR) circuit configuration

- The first step for design was to select type of configuration for current source.
- An inverting op amp configuration with steady dc input was selected to produce constant current.
- The following diagram illustrates the basic design of the selected circuit

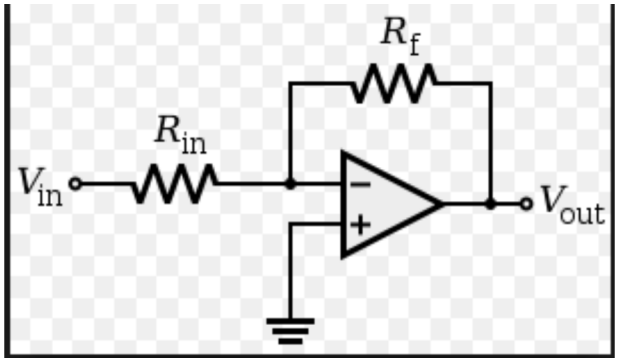


Figure 38 Op-amp Current Source Schematic

Since the current entering the inverting terminal is negligible, Iin will flow through Rf. As the value of Rf changes, Iin remains constant while flowing through Rf. The Rf value of this circuit is R1d // (LDR + Rcof). From the circuit schematic, R1d = 10kΩ, Rcof = 1kΩ, LDR is variable depending on the the detected light levels. The resistance of the LDR used in the design increases as the light intensity decreases.

8.3.5. Differential Amplifier Configuration

To measure output voltage, an inverting differential amp with unity gain will be used. The following diagram displays the configuration that was used in the design

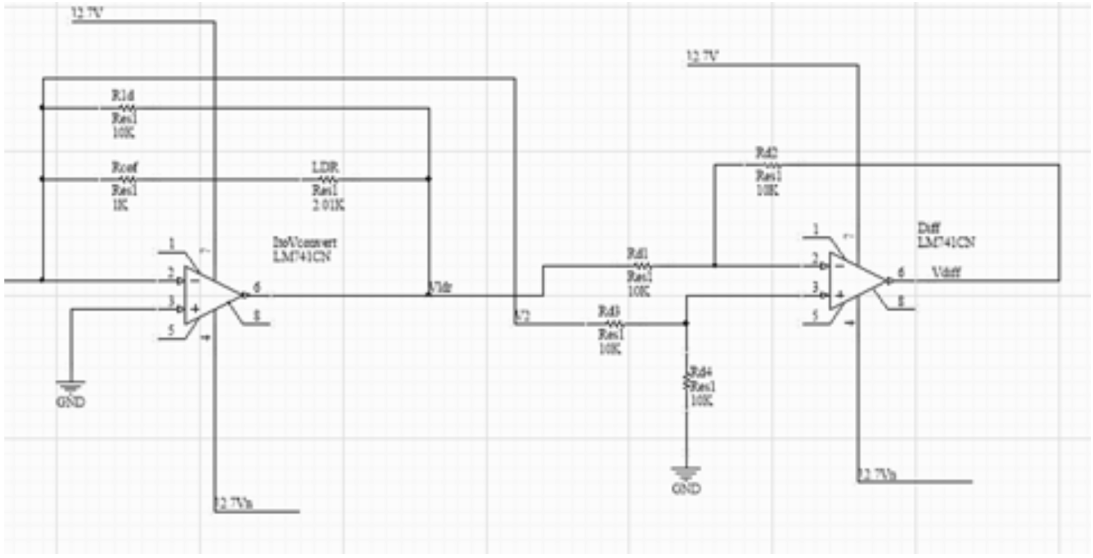


Figure 39 Op-amp Differential Amplifier Schematic

The differential amp measures the difference between the parallel network of the LDR circuit. The output for the differential amp configuration is given by: Since Rd1= Rd2= Rd3= Rd4, the differential amp output equation can be reduced to Viff = -Voldr

8.3.6 Sallen-Key Low-Pass Filter

The next stage in design for the light detection circuit is to filter out external noise, and to reduce the output voltage range to readable values for the MCU. For filtration, a second order unity gain Butterworth Sallen-Key filter is used **[10]**.The following diagram shows the configuration:

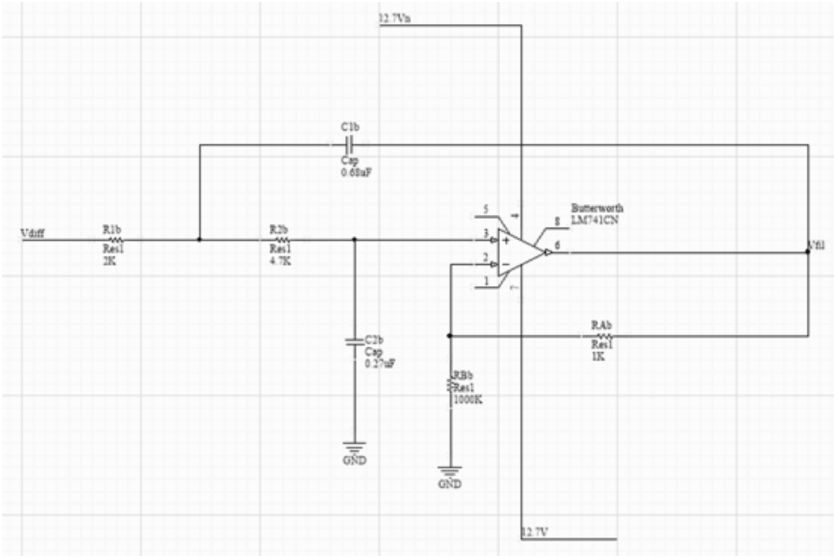


Figure 40 Sallen-Key Low-Pass Filter Schematic

The component values are selected:

- Component ratio α = 0.41
- Damping ratio = 0.586
- FSF = 1
- The 3dB roll off point is selected to be approximately 120 Hz
- K = 1
- The first step for design is to select the value of R_{1b} and C_{1b}
- $R_{1b}C_{1b}$ = = 1.326ms
- R_{1b} is selected to be 2kΩ
- C_{1b} = = 0.663μF
- Since the closest value on Digi key is 0.68 μF; C_{1b} is selected to be that.
- The next step is to determine C_{2b} and R_{2b} using component ratio α
- R_{2b} = = 4.878 kΩ
- R_{1b} is selected to be 4.7 kΩ
- C_{2b} = C_{1b} * 0.41 = 0.28μF; C_{2b} is selected to be 0.27μF
- For K = 1, R_B >>> R_A

- R_A is selected to be 1 kΩ, R_B is selected to be 1 MΩ

8.3.7. Span and Zero

Since the voltage range of the selected circuit exceeds the inputs limits of the MCU, the voltage range must be mapped from 0V to 3.3V. The following diagram will illustrate the configuration in SS1 unit.

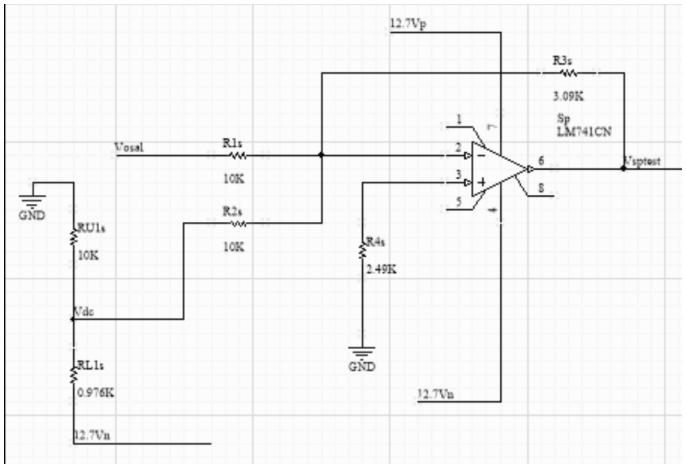


Figure 41 Span and Zero Circuit

8.3.8. SS1 PCB DESIGN

The Altium Designer™ PCB design for SS1 is in Fig.42

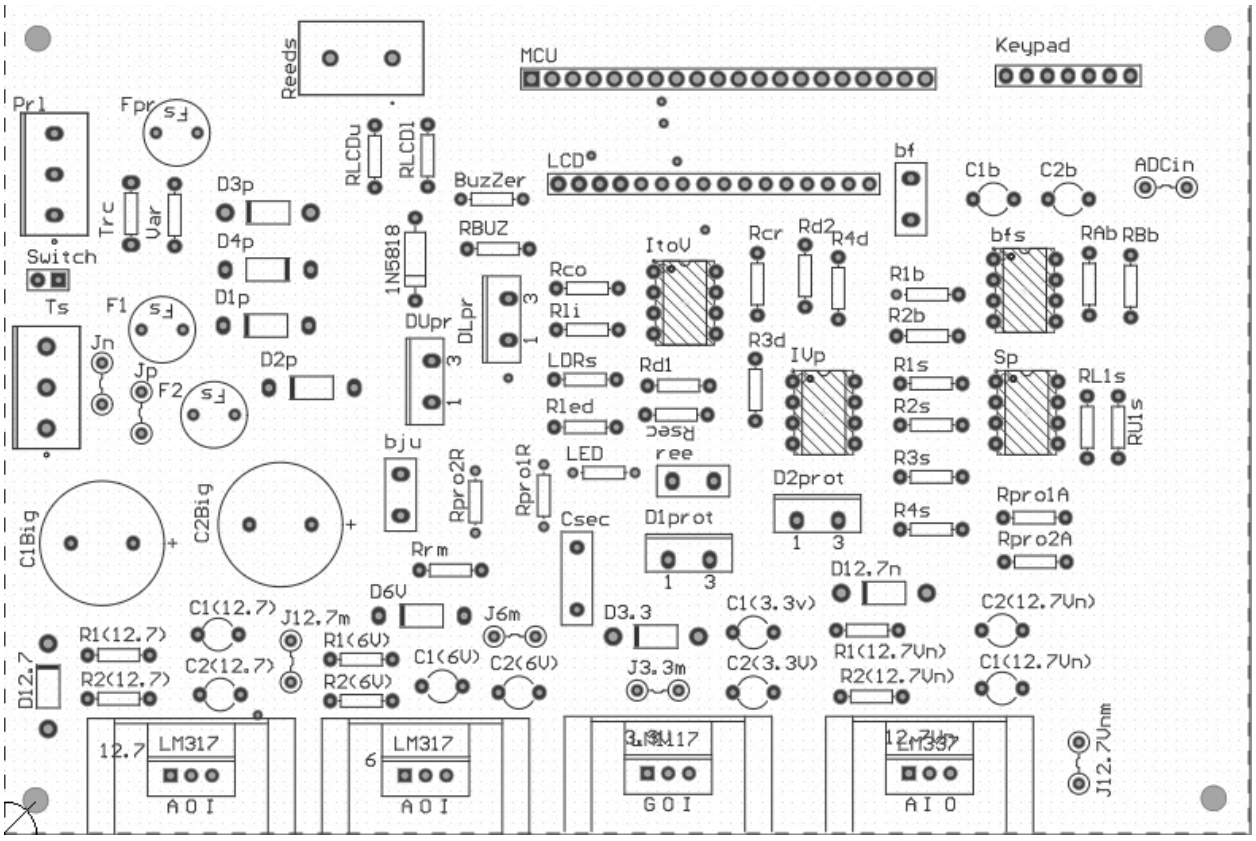


Figure 42 SS1 PCB Design Layout

The power supply circuitry is leftmost on the SS1 PCB with the regulators with heat sinks below. All external connections of the PCB are located on above on the PCB design. All components are through-hole as to make prototype components transferable to the PCB.

9. SOFTWARE DESIGN

The *SCADA* software consists of all the workstation GUI through CVI as well as the software used in all three MCUs. CVI is event driven where certain events on the interface and in the software will cause a section of code to run. The MCU software is interrupt driven. The higher priority interrupt will run it's code until it is completed and then the lower priority interrupt has access to the processor. The MTU, TC1 and SS1 has common functions related to the LCDs and using the*XBee* transceivers.

9.1. GUI SOFTWARE DESIGN

The commented code is available in **Appendix A: SCADA GUI Code**.
The GUI for the *SCADA* system is through National Instruments' LabWindows™ CVI Integrated Development Environment (IDE). The CVI IDE includes built in functions and is an event driven GUI, where the interactive elements create events that run sections of code. An example of this is when the Clear Message Log button is pressed. The button press generates an event to the CVICALLBACK clear_message_log function that runs CVI's ResetTextBox function for the message log text box. The software description will cover functions of the flow diagram seen in Fig.43. as well as supporting functions that are not part of the main processes of the GUI, but are required for the program to run.

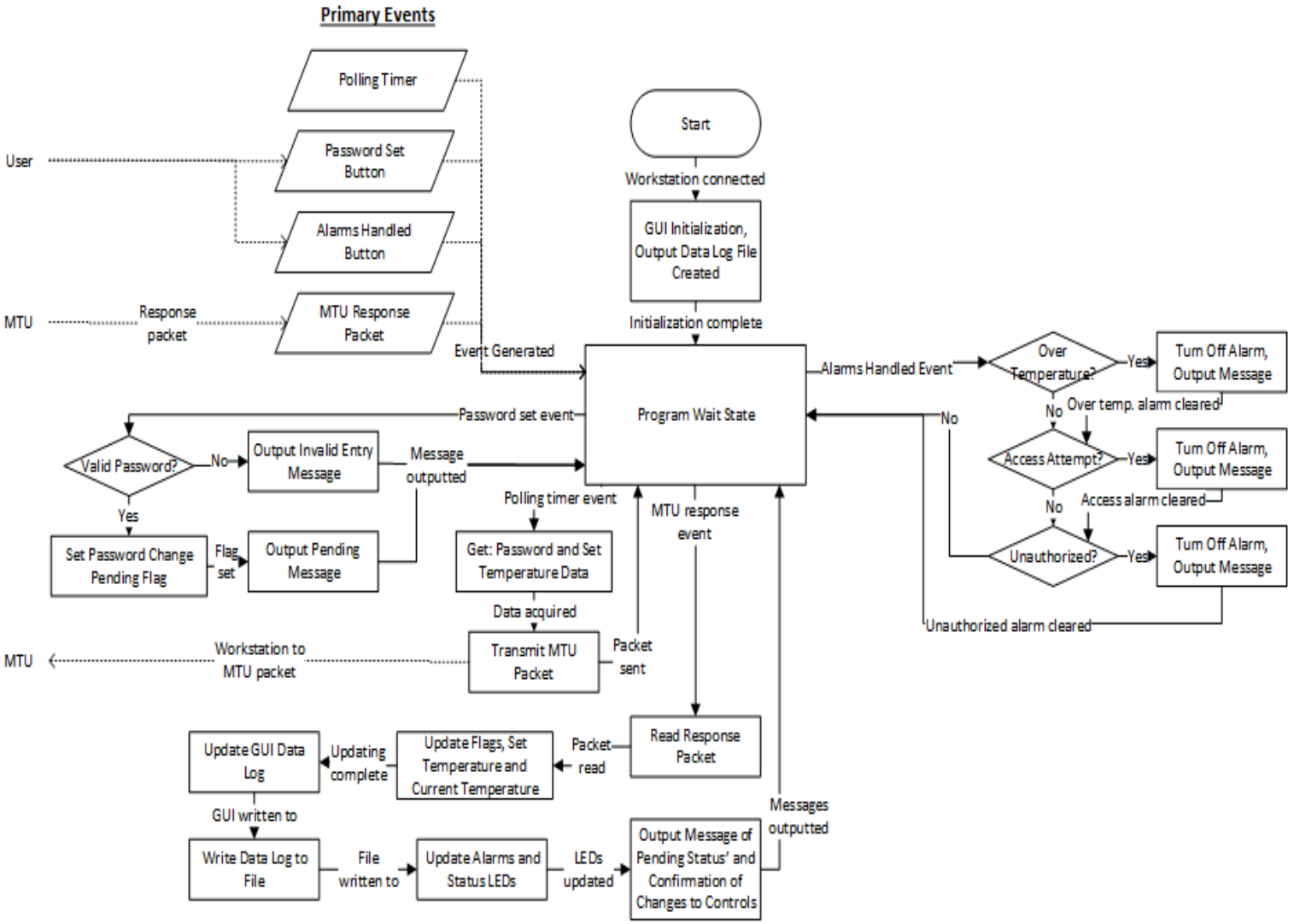


Figure 43 GUI Flowchart

The GUI will run on startup and initialize all indicators, textboxes, and controls as well as the COM port for communication with the MTU and the output data file. The program has a wait state where it awaits for an event. After the event has been generated, a sequence of code runs as a result of the specific event.

Table 13 GUI Function Table

Function Name	Calls	Called by/Event	Use
main	RunUserInterface (CVI)	N/A	- Opens output file. - Initializes COM port 3. - RunUserInterface called to run the GUI.
CVICALLBACK polling_timer	N/A	Event driven from 30 second timer.	- Create transmission string. - Send transmission packet to MTU.
CVICALLBACK alarms_reset	N/A	Event driven from Alarms Handled button.	- Turns off alarm LEDs. - Output to message log text box for each alarm handled.
CVICALLBACK password_set	N/A	Event driven from Set Password button.	- Check for valid set password entry. - Change password of security system on valid entry.
CVICALLBACK clock_timer	N/A	Event driven from 0.1 second timer.	- Updates the GUI clock. - Updates the date/time string for time stamps.
ComCallback	**ProcessReceive** **CreateDatalogMsg** **UpdateLEDs**	Event driven from COM port 3 line feed character.	- Runs functions to handle the received packet from the MTU.
CVICALLBACK clear_message_log	N/A	Event driven from Clear Message Log button.	- Clears the message log text box.
ProcessReceive	N/A	**ComCallback**	- Processes packet from MTU. - Updates flags for statuses and alarms. - Updates set point temperature.
CreateDatalogMsg	**UpdateFlagString**	**ComCallback**	- Creates data log message from MTU packet. - Writes to data log text box. - Writes to output file.
UpdateFlagString	N/A	**CreateDatalogMsg**	- Creates status string for logging.
UpdateLEDs	N/A	**ComCallback**	- Updates status and alarm LEDs.
CVICALLBACK exit_button	N/A	Event driven from Exit button.	- Closes COM port and output file. - Closes GUI.

Events in the GUI

The primary events of the GUI are:
-The initialization of the polling cycle through the poll cycle timer.
-The password set button pressed after a password has been changed.
-The alarms handled button to clear the current alarms of the RTU.
-The callback created from the MTU to workstation response data packet.

Supporting events of the GUI are:
-The clock timer event that updates the current date time of the program.
-The clear message button to clear the message log.
-The exit button that closes the output file and workstation program.

main function (SCADA GUI Initialization)

The *SCADA* GUI runs *main* upon start-up.
The processes included in *main* are:
-Data log file creation.
-Port setup for MTU communication.
-*RunUserInterface* function to start the GUI.
The *main* function finds the current date/time string, creates a unique data log excel file using the date/time string in the file name. After the file is created, the first line is outputted to the file that will hold the header information for the tabulated data log information that will be collected from the MTU.
The communication port is configured through CVI's *OpenComConfig* for COM3, with a baud rate of 9600, no parity bits, 8 data bits and 1 stop bits. This configuration matches the microprocessor USART2 configuration of the MTU and allows for communication between the MTU and workstation. In order to have an event created upon receiving a packet, CVI's *InstallComCallback* function will be used to create an event whenever a linefeed character is received from the MTU.
CVI's *RunUserInterface* function runs the GUI as well as installs the events for the CVICALLBACK functions.

CVICALLBACK polling_timer function (SCADA Polling Cycle)

The polling timer is set for 30 seconds, so that every 30 seconds the workstation will create an event to start the polling cycle of the *SCADA* system.
The processes included in *CVICALLBACK polling_timer* are:
- Get password and set temperature from GUI.
- Create and send packet to MTU.
The password is gathered from the set password process, whereas the set temperature is gathered from the numeric control on the GUI. The numeric control ranges from 25.0°C to 75.0°C as valid inputs. If there has been a change between the previously recorded set temperature and the current set temperature in the numeric, the flag for a pending temperature change is set.

The data packet to the MTU includes the access password, the set temperature, and the flags any changes in password or set temperature. The packet is written to the MTU using CVI's *COMWrt* function.

CVICALLBACK alarms_reset function (Alarms Handled)

The event to run *alarms_reset* is activated upon the button press of Alarms Handled on the GUI screen.
Alarm statuses are saved in the current values of the LEDs of the GUI. This function checks the over temperature alarm, the access attempted alarm and the unauthorized access alarm through the LEDs, clears them, and outputs a message to the message log textbox for each alarm that was cleared.

CVICALLBACK password_set function

The event to run *password_set* is activated upon the button press of Set Password on the GUI screen.
The *password_set* function checks to see if the set password entry is a correct password of between 0000 and 9999. If the set password entry is correct, the current password to the security system is changed at the workstation and will be changed at the security system unit upon the next communication. If the entry was incorrect, then the password is not changed and a message is outputted to the message log text box stating that an invalid set password was entered.

CVICALLBACK clock_timer function (GUI clock)

The event to run *clock_timer* is activated every 0.1 seconds in order to not miss a 1 second change in time due to other processes.
This function changes the clock time on the GUI as well as updates the date/time string of the program which is used for creating a timestamp pair on the data log.

ComCallback function (for MTU response packet)

The event to run *ComCallback* is activated when a linefeed character is received from the MTU. This function reads in the data packet using CVI's *ComRdTerm* and then calls *ProcessReceive*, *CreateDatalogMsg*, and *UpdateLEDs*. The result is that the data packet is read, processed to update values and flags and then sent to the data log on the GUI and excel file.

CVICALLBACK clear_message_log function

The event to run *clear_message_log* is activated upon the button press of Clear Message Log on the GUI.
This function uses CVI's *ResetTextBox* to clear the contents of the message log text box.

ProcessReceive function

The *ProcessReceive* function is called from the *ComCallback* function after the MTU response packet has been read into the MTU_receive string.
This function processes the string to update the flags:
- g_TC1_status_flag - On/off status of Temperature Controller Unit.
- g_TC1_set_change_flag - Flag if the temperature was changed in TC1.
- g_TC1_overtemp_flag - Flag if the heater of TC1 is over temperature.
- g_SS1_status_flag - On/off status of Security System Unit.
- g_SS1_pass_change_flag - Flag if the password was changed in SS1.
- g_SS1_access_attempt_flag - Flag if access was attempted to SS1.
- g_SS1_unauthorized_flag - Flag if the access attempt was unauthorized.
- g_SS1_lights_on_flag - On/off status of the RTU lights.

The current set temperature and previous set temperature are also updated if there was a set temperature processed at the TC1 unit.

CreateDatalogMsg function

The *CreateDatalogMsg* function is called from *ComCallback* function after the MTU response packet has been processed.
This function calls *UpdateFlagString* to create the string to represent the flag status of the processed MTU data for data logging purposes. After all flag strings have been updated, the data log message to GUI is outputted and the excel data log file is updated with the new data.

UpdateFlagString function (data logging purposes)

The *UpdateFlagString* function is called from *CreateDatalogMsg.*
This function takes in the flag status and unit status and outputs a string of either xxx, yes, or no for the over temperature flag, lights on flag, access attempt flag and the unauthorized access flag. The xxx string is used for when the unit status is off. An example of when xxx is outputted is if the TC1 unit is off, then the flag string returned for over temperature will be xxx.

UpdateLEDs function

The *UpdateLEDs* function is called from the *ComCallback* function after the data logging to the GUI and excel file are completed.
This function updates the following status LEDs:

- TC1 on based on g_TC1_status_flag.
- SS1 on based on g_SS1_status_flag.
- RTU lights on based on g_SS1_lights_on_flag.

This function latches the following alarm LEDs:
- TC1 Over Temperature based on g_TC1_overtemp_flag.
- SS1 Accessed based on g_SS1_access_attempt_flag.
- SS1 Unauthorized Attempt based on g_SS1_unauthorized_flag.

***CVICALLBACK exit_button* function**

The event to run *exit_button* is activated upon the button press of EXIT on the GUI screen. This function uses CVI's *CloseCom* to close communications with the MTU. The output data log excel file is then closed and the GUI is closed using CVI's *QuitUserInterface* function. The *exit_button* function is used to ensure a safe exit procedure is done.

9.2. COMMON FUNCTIONS USED IN MCUS

There is a set of common functions to each of the MCUs. This includes:

1. Utility functions such as delay and delay_ms which sets an amount of time to delay.
2. LCD functions:
    a. LCD General Purpose Input-Output (GPIO) pin initialization for 4 pin parallel instructions.
    b. LCD initialization for 4 pin parallel instruction set.
    c. LCD write and command functions.
3. Packet receiving and transmitting functions between MCUs:
    a. USART3 initialization for *Direct Memory Access (DMA)*.
    b. *DMA* streams initialization for receiving and transmitting.
    c. Creating transmission packets.
    d. Processing received packets.

e. *DMA* interrupt handlers for transmission complete and packet received.

9.2.1. Common Utility Functions

Table 14 Common Utility Function Table

Function	**Calls**	**Called by**	**Use**
delay	N/A	All code due to being a utility tool.	- Creates a delay related to the delay parameter.
delay_ms	N/A	All code due to being a utility tool.	**-** Creates a delay in milliseconds related to the delay parameter.

void delay(int a) **function**
The *delay* function takes the parameter ‘int a’ and decrements the value by 1 in a while loop until a=0. The result is that the processor is stuck decrementing a, causing a delay in the program execution.

void delay_ms(uint32_t num_of_ms) **function**
The *delay_ms* function takes the parameter ‘uint32_t num_of_ms**’** and delays 1 millisecond and then decrements the ‘num_of_ms’ parameter by 1. The result is that the processor is stuck decrementing a, causing a ‘num_of_ms’ delay in the program execution.

9.2.2. Common LCD Functions

Table 15 Common LCD Function Table

Function	**Calls**	**Called by**	**Use**
Gpio_init_LCD	N/A	**main**	- Enable GPIOC clock. - Configure GPIO C5,C6,C7,C8,C9 and C10 as outputs.
LCD_init	**LCD_SendCmd** **LCD_SendData** **delay**	**main**	- Wake up LCD. - Clear display. - Set up entry mode. - Create custom ° character.
LCD_SendCmd	**LCD_Pulse** **LCD_PutNibble** **delay**	**LCD_Write_Two_Line** **LCD_Write_One_**	- Sends a command to LCD. Example commands: Display On/Off Control, Clear Display, Return Home.

		Line **LCD_init**	
LCD_SendData	**LCD_Pulse** **LCD_PutNibble**	**LCD_Write_Two_line** **LCD_Write_one_line** **LCD_init**	- Used to send character of data to the LCD.
LCD_Pulse	N/A	**LCD_SendCmd** **LCD_SendData**	- Send a pulse to for LCD execute to execute current command.
LCD_PutNibble	N/A	**LCD_SendCmd** **LCD_SendData**	- Sends a nibble of data across data lines of GPIO C7,C8,C9 and C10.
LCD_Write_Two_line	**LCD_SendCmd** **LCD_SendData** **delay**	In MTU: **Adc1_to_output** In TC1: In SS1:	- Takes two string parameters and writes string one to line one and string two to line two of the LCD.
LCD_Write_one_line	**LCD_SendCmd** **LCD_SendData** **delay**	N/A	- Takes one string parameter and writes it to line one of the LCD.

***void Gpio_init_LCD(void)* function**
This function enables the GPIOC clock and GPIOC pins C5, C6, C7, C8, C9 and C10. Pins C7 to C10 are the 4 data line pins that are used for 4 pin parallel communication with the LCD. Pins C5 and C6 are the register select and enable lines to the LCD that allow data writing operations to the LCD.

***void LCD_init(void)* function**
This function uses the *delay* and *LCD_SendCmd* functions to wake up the LCD and set up the LCD for 4 pin parallel communication. The *LCD_SendData* function is used to create a custom ° symbol on the LCD. The *LCD_init* function is called after the *Gpio_init_LCD* upon startup. After both functions have been called the LCD can have characters written to it using *LCD_Write_Two_line* and *LCD_Write_one_line.*

***void LCD_SendCmd(uint8_t c)* function**
This function clears the register select line (C5) to send a command to the LCD by sending the 4 most significant bits onto the data lines (C7 to C10), pulsing the enable line (C6), and then

sending the least 4 least significant bits with another pulse. The result is that the ‘uint8_t c’ parameter is sent as a command to the LCD and the register select line is set high again.

void LCD_SendData(uint8_t c) **function**
This function sends a character in the ‘uint8_t c’ parameter to be displayed to the LCD. To do this, the 4 most significant bits are sent to the LCD on the data lines (C7 to C10), the enable line is pulsed (C6), and the 4 least significant bits are sent with another pulse. The sent character will then appear on the LCD.

void LCD_Pulse(void) **function**
This function is the GPIO C6 pulse function that is used on the enable line of the LCD. To execute a command the enable line of the LCD needs to receive negative edge. This is done by setting C6 high, delaying and then setting C6 low.

void LCD_PutNibble(uint8_t c) **function**
This function is used to put the 4 least significant bits of parameter ‘uint8_t c’ on the 4 data lines. This is done by setting or clearing the respective GPIO C7 to C10 pins.

void LCD_Write_Two_line(uint8_t *line1, uint8_t *line2) **function**
This function takes the string parameters ‘uint8_t *line1’ and ‘uint8_t *line2’ and writes them to the first and second line of the LCD using the *LCD_SendCmd* and *LCD_SendData* functions.

void LCD_Write_one_line(uint8_t *line1, uint16_t Datatype) **function**
This function takes the string parameter ‘uint8_t *line1’ and writes it to the first line of LCD. The ‘uint16_t Datatype’ parameter is 0 if ℃ is required to be outputted at the end of the first line.

9.2.3. Common Transceiver Functions

Table 16 Common Transceiver Related Function Table

Function	Calls	Called by	Use
USART3_init	N/A	**main**	- Enable GPIOB clock. - Configure GPIO B10 and B11 for USART3. - Enable USART3 clock. - Configure USART3.
DMA_transmit_init	N/A	**main**	- Enable DMA1 clock. - Configure memory to peripheral for DMA1stream3 for transmitting.
DMA_receive_init	N/A	**main**	- Enable DMA1 clock. - Configure peripheral to memory for

			DMA1stream1 for receiving. - Enable DMA1stream1.
DMA_process_rx	N/A	**DMA1_Stream1 _IRQHandler**	- Extract information from received packet. - Determine who the packet is meant for. - Update flags and values if unit is the intended receiver.
DMA_create_tx	N/A	**DMA1_Stream1 _IRQHandler**	- Create packet based on current values and flags. - Enable DMA1stream3 to send packet.
DMA1_Stream3_IRQ Handler	N/A	Interrupt driven based upon transmitted packet complete.	- Clear pending status. - Clear transfer complete flag.
DMA1_Stream1_IRQ Handler	**DMA_process_rx DMA_create_tx**	Interrupt driven based upon packet receive.	- Clear pending status. - Clear transfer complete flag. - Generate response to received packet.

***void USART3_init (void)* function**
This function enables the GPIOB and USART3 clocks. The GPIO B10 and B11 pins are set for alternate function mode to be used for USART3tx and USART3rx. USART3 is set up for 9600 baud rate, no parity bit, no flow control, and 1 stop bit. The DMAT and DMAR lines are set to allow for *DMA* transmit and *DMA* receive respectively.

***void DMA_transmit_init (void)* function**
This function enables the DMA1 clock (if not enabled already). The DMA1_Stream3 memory address is set to the global string of ‘g_send_tx’ and a peripheral address of the USART3 data register. The amount of bytes that will be transmitted is 12 in a direction of memory to peripheral. Since the *DMA* transmit stream requires an interrupt upon transmit complete, the TCIE line needs be set as well as the IRQ enabled in NVIC with a priority of 1.

***void DMA_receive_init (void)* function**
This function enables the DMA1 clock (if not enabled already). The DMA1_Stream1 memory address is set to the global string of ‘g_receive_rx’ and a peripheral address of the USART3 data register.The amount of bytes that will be transmitted is 12 in a direction of peripheral to

memory. Since the *DMA* transmit stream requires an interrupt upon receive complete, the TCIE line needs be set as well as the IRQ enabled in NVIC with a priority of 0.

***void DMA_process_rx (void)* function**
This function checks the start byte (0x77), the unit message address, and the checksum error detection in order to decide whether to process the received packet in in 'g_receive_rx'. The intended recipient for the message processes the received data and updates global flags and values based on the data message. After the message is processed, DMA1_Stream1 is enabled in order to be able to receive the next message.

***void DMA_create_tx (void)* function**
This function creates and sends the transmission packet in 'g_send_tx'. This is done by creating the packet with the start byte (0x77), the unit address, the local data and flags and creating a checksum error detection byte based upon the values in the data bytes.

***void DMA1_Stream3_IRQHandler(void)* interrupt handler**
This interrupt handler clears the pending flag and transfer complete flag after a transmission packet is sent.

***void DMA1_Stream1_IRQHandler(void)* interrupt handler**
This interrupt handler clears the pending flag and transfer complete flag after a packet is received. The *DMA_process_rx* function is used to process the received packet, and the intended recipient of the data message creates a response using *DMA_create_tx*.

9.3. MTU SOFTWARE DESIGN

The commented code is available in **Appendix B: SCADA MTU Code.**
The Integrated Development Environment (IDE) used for the *SCADA* MTU is CrossStudio for ARM 4.2. The microprocessor used is the ARM M4 Cortex on Olimex's STM32-P405 development board. The processes used in the *SCADA* MTU are interrupt driven, where the highest priority interrupt at any given time will be executed first.

MTU Priority Mapping
The interrupts used in the *SCADA* MTU and their mapped priority levels seen in Table 17**.**

Table 17 MTU Interrupt Priority mapping

Interrupt Handler	**Priority**	**Notes**
DMA1_Stream1_IRQHandler	0 (highest)	*DMA* stream for *XBee* packet receiving.
DMA1_Stream6_IRQHandler	0	*DMA* stream for workstation packet receiving.
DMA1_Stream3_IRQHandler	1	*DMA* stream for *XBee* packet transmitting.

DMA1_Stream5_IRQHandler	1	*DMA* stream for workstation packet transmitting.
SysTick_Handler	2	Timeout method for MTU to RTU communications.
ADC_IRQHandler	4	ADC1 is used to calculate MTU temperature and the handler outputs the results to LCD.

The ADC for temperature as well as the LCD output function are a low priority interrupt, resulting in small delay for the communications code which is a high priority.

MTU Flow Diagram

The MTU software can be generalized with the result seen in the flow diagram of Fig.44**.**. The actual processes will allow for the ADC and LCD functions for temperature to run in downtime and wait times of the communication functions.

The MTU software flow diagram in Fig.44**.** relies on communication from the workstation, SS1 and TC1. In the event that the workstation is not connected, no communication takes place. If SS1 or TC1 are not connected, then a timeout will take place when waiting so that the process polling process can proceed.

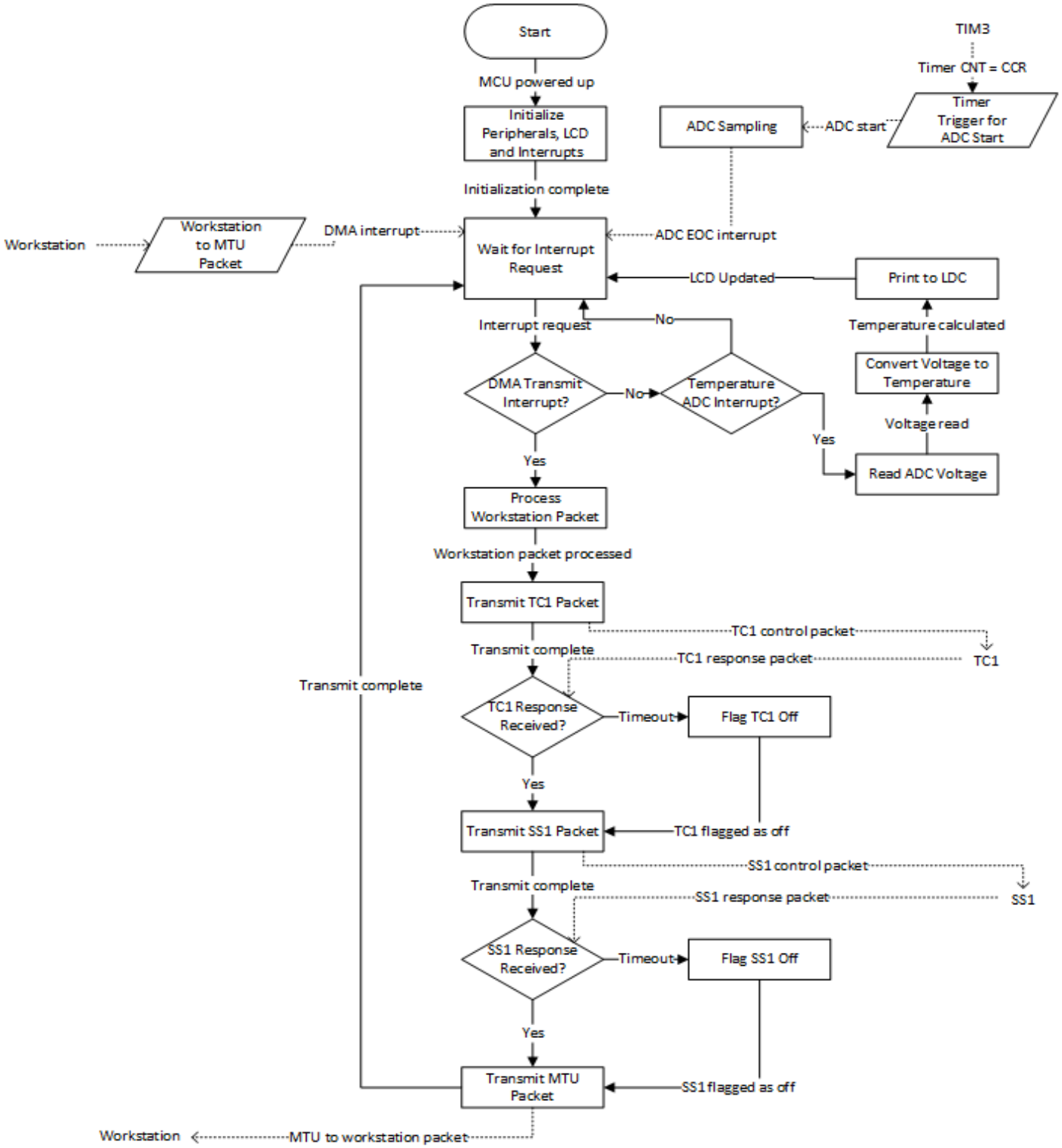


Figure 44 MTU Flow Diagram

Table 18 MTU Function Table

Function	Calls	Called by	Use
main	**timeout_init** **USART2_Init** **workstation_transmit_init** **workstation_receive_init** **Gpio_init_LCD** **LCD_init** **USART3_init** **DMA_transmit_init** **DMA_receive_init** **tim3_init** **ADC1_init**	This function is run on startup.	**-** Call all initialization functions for interrupts and peripherals used in MTU - Hold the while(1) loop for wait state between interrupts.
USART2_Init	N/A	**main**	- Enable GPIOA clock. - Configure GPIO A2 and A3 for USART2. - Enable USART2 clock. - Configure USART2
workstation_transmit_init	N/A	**main**	- Enable DMA1 clock. - Configure memory to peripheral for DMA1stream6 for transmitting.
workstation_receive_init	N/A	**main**	- Enable DMA1 clock. - Configure peripheral to memory for DMA1stream5 for receiving. - Enable DMA1stream5.
workstation_process_rx	**DMA_create_tx**	**DMA1_Stream5_IRQHandler**	- Extract information from received packet. - Update flags and values due to workstation packet. - Enable transmission to TC1.
workstation_create_tx	N/A	**SysTick_Handler** **DMA1_Stream1_IRQHandler** **SysTick_Handler**	- Create packet from current data from RTU. - Enable DMA1stream6 to send packet.

Table 18 MTU Functions (continued)

Function	Calls	Called by	Use
timeout_init	N/A	**main**	- Initializes SysTick to largest timeout reload value.
timeout_start_restart	N/A	**DMA1_Stream5_IRQHandler** **SysTick_Handler**	- Disables SysTick. - Clears SysTick count. - Enables SysTick.
timeout_stop	N/A	**SysTick_Handler**	- Disables SysTick. - Clears SysTick count.
SysTick_Handler	**timeout_stop** **DMA_create_tx** **timeout_start_restart** **workstation_create_tx**	Interrupt driven by SysTick count reaching 0.	- Increase *SCADA* polling state if a communication has timed out. - Create transmission of next poll state through function calls.
DMA1_Stream6_IRQHandler	N/A	Interrupt driven by DMA transmit packet.	- Clear pending status. - Clear transfer complete flag. - Enable DMA1stream5 to allow next workstation receive packet.
DMA1_Stream5_IRQHandler	**workstation_process_rx** **timeout_start_restart**	Interrupt driven by DMA receive packet.	- Clear pending status. - Clear transfer complete flag. - Start *SCADA* polling process.
tim3_init	N/A	**main**	- Enable TIM3 clock. - Configure TIM3 to trigger ADC1 conversion every 1 second.
ADC1_init	N/A	**main**	- Enable GPIOA clock. - Configure GPIO A1 for ADC1. - Configure to start ADC1 for TIM3 trigger event.
ADC_IRQHandler	**adc1_to_output**	Interrupt driven by ADC end of conversion.	- Clear EOC flag. - Call **adc1_to_output** to process ADC conversion.
adc1_to_output	**LCD_Write_Two_line**	**ADC_IRQHandler**	- Use ADC data register value to find ADC voltage. - Calculate RTD temperature. - Call **LCD_Write_Two_line** to up update LCD.

int main(void) function (MTU initialization)

The *main* function runs on start up and initializes all interrupts, peripherals, and the LCD. The processes included in main are:
- Workstation communications initialization for DMAstreams.
- LCD initializations and GPIO initialization for the LCD.
- *Transceiver* communications initialization for DMAstreams.
- ADC initialization for temperature measurements.

The workstation communications run off of USART2 in asynchronous mode with a RS-232 to USB connector from the MTU to the workstation. The functions called for the communication initialization are: *timeout_init*, *USART2_init*, *workstation_transmit_init* and *workstation_receive_init*.

The LCD initialization and GPIO initialization for the LCD processes set up the LCD to be written to using 4 pin parallel communication. The functions called related to the LCD are: *Gpio_init_LCD*, and *LCD_init*.

The *transceiver* communications are the communications between the TC1 and SS1 units. USART3 is used in asynchronous mode with pins going to the *XBee transceiver*. The functions called for the *transceiver* communication initialization are: *USART3_init*, *DMA_transmit_init*, and *DMA_receive_init*.

The ADC initialization for temperature measurements utilizes ADC1 with a software trigger from Timer3 every 1 seconds. The functions called are: *tim3_init*, and *ADC1_init*.
After the initializations are completed, the program enters a while(1) loop, where it waits for interrupts of the ADC and communication DMAstreams for the workstation and *XBee transceiver*.

void USART2_Init (uint32_t interrupt_mask, uint32_t usart_priority) function

This function enables the GPIOA and USART2 clocks. The GPIO A2 and A3 pins are set for alternate function mode to be used for USART2tx and USART2rx. USART2 is set up for 9600 baud rate, no parity bit, no flow control, and 1 stop bit. The DMAT and DMAR lines are set to allow for *DMA* transmit and *DMA* receive respectively.

void workstation_transmit_init (void) function

This function enables the DMA1 clock (if not enabled already). The DMA1_Stream6 memory address is set to the global string of ‘g_workstation_tx’ and a peripheral address of the USART2 data register. The amount of bytes that will be transmitted is 11 in a direction of memory to peripheral. Since the *DMA* transmit stream requires an interrupt upon transmit complete, the TCIE line needs be set as well as the IRQ enabled in NVIC with a priority of 0.

***void workstation_receive_init (void)* function**
This function enables the DMA1 clock (if not enabled already). The DMA1_Stream5 memory address is set to the global string of ‘g_workstation_rx’ and a peripheral address of the USART2 data register.The amount of bytes that will be transmitted is 10 in a direction of peripheral to memory. Since the *DMA* transmit stream requires an interrupt upon receive complete, the TCIE line needs be set as well as the IRQ enabled in NVIC with a priority of 1.

***void workstation_process_rx(void)* function**
This function checks ‘g_receive_rx’ for the start byte (0x77) and processes the master set password and master set temperature flags. The data for set temperature and set password are updated based on the status of the flags. After the workstation data is processed, this function calls *DMA_create_tx* to start the *SCADA* polling cycle.

***void workstation_create_tx(void)* function**
This function creates and sends the MTU to workstation packet in the ‘g_workstation_tx’ string. The TC1 and SS1 on/off statuses as well as data collected from TC1 and SS1 processed packets are all sent in the MTU to workstation packet. After the packet is created, it is sent by enabling the DMA1stream6.

***void timeout_init(void)* function**
This function initializes SysTick with a priority of 2 and the largest reload value to create a timeout after 2 seconds. SysTick is not enabled at the end of the initialization process.

***void timeout_start_restart(void)* function**
This function disables SysTick, clears the current count in SysTick, and then enables SysTick. This function is used when refreshing the timeout count and initially starting the timeout counter.

***void timeout_stop(void)* function**
This function disables SysTick and clears the current count in SysTick. This function is used to stop the timeout counter after the *SCADA* polling cycle is completed.

***void SysTick_Handler(void)* interrupt handler**
This interrupt handler is used in the *SCADA* polling cycle to keep transmission going if TC1 or SS1 are off. The poll state in ‘g_poll_state’ increases so that the polling cycle continues.

***void DMA1_Stream6_IRQHandler(void)* interrupt handler**
This interrupt handler is triggered after the transfer is complete for the MTU to workstation packet. After the packet is sent, the DMA1stream5 is enabled to allow for the next data packet from the workstation

***void DMA1_Stream5_IRQHandler(void)* interrupt handler**
This interrupt handler is triggered after the transfer is complete for the workstation to MTU packet. This handler calls *workstation_process_rx* to process the received packet and starts the timeout sequence with *timeout_start_restart*.

***void tim3_init(void)* function**
This function enables the clock of TIM3 and sets TIM3 for a 1Hz *PWM* signal with a 50% duty cycle. TIM3 is then configured to create a trigger event so that it may be used for triggering ADC1 every 1 second.

***void ADC1_init(void)* function**
This function enables the GPIOA and ADC1 clocks. The GPIO A1 pin is configured for analog mode so that it can be used for ADC1. ADC1 is configured for interrupt from the TIM3 signal's rising edge and enabled in NVIC for an interrupt priority of 4.

***void ADC_IRQHandler(void)* interrupt handler**
This interrupt handler is triggered from the ADC1 end of conversion and clears the status flag. The *adc1_to_output* function processes the data in the ADC data register.

***void adc1_to_output(void)* function (RTD calculations and LCD updating)**
This function takes the value from the ADC data register, converts it to a voltage and uses that voltage to find the RTD resistance. The RTD resistance is used for the calculation of temperature using the quadratic equation. After four conversions, the 4-point moving average DSP mode is enabled to allow for a slower transition of temperature value and reduce the effect of system noise.

The value for temperature and RTD resistance are formatted into strings and outputted to the LCD through the *LCD_Write_Two_line* function.

9.4. TC1 SOFTWARE DESIGN

The commented code is available in **Appendix B: SCADA TC1 Code.**
Some of the processes used in the *SCADA* TC1 are interrupt driven, where the highest priority interrupt at any given time will be executed first.

TC1 Priority Mapping
The interrupts used in the *SCADA* TC1 and their mapped priority levels seen in Table 19**.**

Table 19 TC1 Interrupt Priority mapping

Interrupt Handler	**Priority**	**Notes**
DMA1_Stream1_IRQHandler	0 (highest)	DMA stream for *XBee* packet receiving.
DMA1_Stream6_IRQHandler	0	DMA stream for workstation packet receiving.
DMA1_Stream3_IRQHandler	1	DMA stream for *XBee* packet transmitting.
DMA1_Stream5_IRQHandler	1	DMA stream for workstation packet transmitting.
DMA2_Stream0_IRQHandler	4	ADC1 is used to calculate TC1 temperature on channels 0 and 1
TIM2_IRQHandler	3	TIM2 is used for Input capture used for frequency measurement of fan RPM
EXTI0_IRQHandler	2	Pin B0 is used for keypad menu interrupt

TC1 Flow Diagram

The TC1 software can be generalized with the result seen in the flow diagram of **Fig.9.4.** The process starts when the MCU powers up and then performs the initialization process that configures all GPIO pins, interrupts peripherals, timers and *DMA* requests. Within the while loop the program will test for multiple conditions before performing specific tasks. If at any time during the process an interrupt occurs, it will update the specific flags of that function. If the interrupt is the *transceiver DMA* interrupt request it will perform the *transceiver* routine before returning. The TX complete flag will be tested, this flag is updated in the *transceiver DMA* interrupt routine and determines if the TX complete LCD display will be executed. The Error signal is then calculated based on the difference between measured temperature and setpoint temperature, the error signal is tested to determine the following procedure. If the set point is a new set point the fan and heater will be set to a min max state that provides the fastest rise time. If the error becomes within 0.2 of a degree of the setpoint the set *PWM* function will set the fan and heater to the open loop approximation for that temperature and this is performed only once for a new setpoint. Every time the loop iterates and a new adc conversion is completed the counter will be decremented, this counter value is set to one time constant based on the thermal resistance of the system which is calculated when a new set point is entered. When the counter reaches 0 the adjust *PWM* function will use the error to make *PWM* adjustments in order to match the sensor temperature to the set point. If at any time during the process a *DMA* request is received from the *transceiver* the *transceiver* routine will run before returning to the previous line.

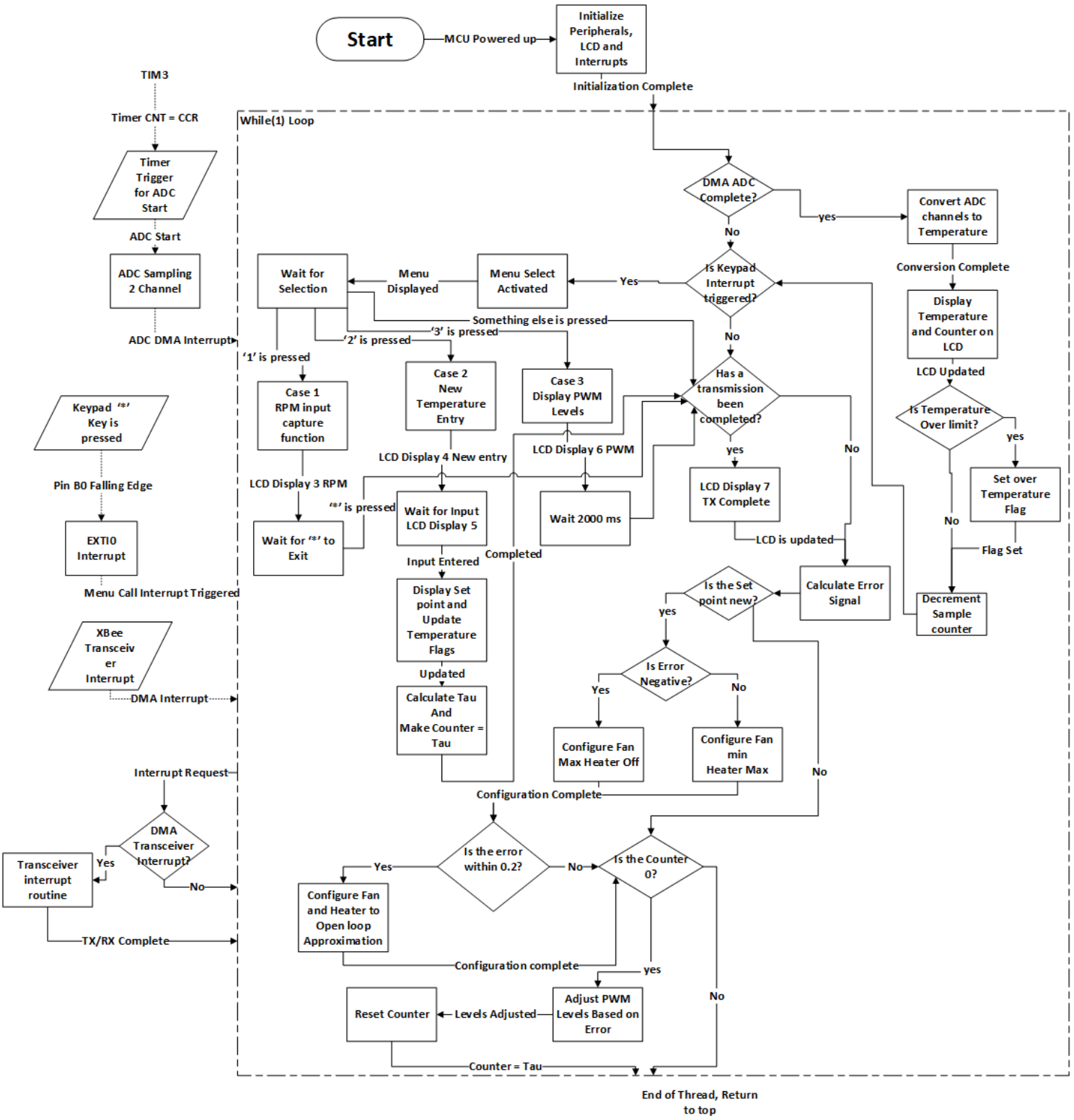


Figure 45 TC1 Flow Diagram

Table 20 TC1 Function Table

Function	Calls	Called by	Use
main	**-Tim3_init** **-ADC1_init** **-DMA_ADC1_init** **-GPIO_EXTI_call_menu** **-PWM_controls_duty_cycle** **-USART3_init** **-DMA_transmit_init** **-DMA_receive_init** **-Gpio_init_LCD()** **-Init** **-Calculate_TimeConstant** **-TC_ADC_2ch** **-LCD_display_cycle1** **-Keypad_main** **-TIM2_frequency_measurement** **-Adjust_PWM_dutycycle** **-delay_ms**	N/A	-performs all initialization before entering loop -The main loop performs all conditions testing and adjustments -the main loop contains the keypad and menu structure -calls the temperature update -calls the LCD update
tim3_init	N/A	**main**	- Enable TIM3 clock. - Configure TIM3 to trigger ADC1 conversion every 1 second.
ADC1_init	N/A	**main**	-Enables GPIO PC0 & PC1 as ADC1 ch0/ch1
DMA_ADC1_init	N/A	**main**	-Configure and enable DMA2_Stream0 to transfer ADC1 Ch0/1
GPIO_EXTI_call_menu	**GPIOx_Read_interrupt** **GPIOx_pin_config**	**main**	-Configures GPIO B0 as falling edge interrupt for keypad

Table 20 TC1 Function Table (continued)

GPIOx_Read_interrupt	N/A	**GPIO_EXTI_call_menu**	-Enables Interrupt for specific pin with configuration parameters
GPIOx_pin_config	N/A	**GPIO_EXTI_call_menu**	-Enables a GPIO pin with configuration parameters
PWM_controls_duty_cycle	**GPIOx_pin_config**	**main**	-configures and enables *PWM* on 2 channels
Keypad_main	**-GPIO_Read_fourpin** **-GPIO_Write_OPENDRAIN** **-keypad_scan** **-Keypad_debounce**	**main**	-keypad function -contains menu structure and sub functions related to keypad operation
GPIO_Read_fourpin	N/A	**Keypad_main**	-Configure 4 pins as inputs with pull-up
GPIO_Write_OPENDRAIN	N/A	**Keypad_main**	-Configure 4 pins as output with Open drain mode
keypad_scan	N/A	**Keypad_main**	-Find the Row or Column that was pressed by the keypad
Keypad_debounce	N/A	**Keypad_main**	-Small delay and condition testing to correct button bouncing
LCD_display_cycle1	**-LCD_Write_one_line** **-LCD_Write_Two_line** **-LCD_SendCmd** **-Delay_ms**	**main**	-Standard function for multiple LCD display modes

Table 20 TC1 Function Table (continued)

Calculate_TimeConstant	N/A	**main**	-Calculate time constant (tau) of the system based on thermal resistance -Used for adjustment counter
TC_ADC_2ch	**RTD_to_temp**	**main**	-Converts ADC value to Temperature
Rtd_to_temp	**TC_ADC_2ch**	N/A	-Converts RTD voltage to temperature
Adjust_PWM_dutycycle	**-Set_PWM_dutycycle**	**main**	-Calculate and set *PWM* levels based on error signal
Set_PWM_dutycycle	N/A	**Adjust_PWM_ dutycycle**	-Calculate and set open loop approximation for *PWM* levels
TIM2_frequency_measure ment	N/A	**main**	-Input capture for fan RPM measurement

void main(void) function

The *main* function runs on start up and initializes all interrupts, peripherals, and the LCD.
The processes included in main are:
- ADC and timer initialization for temperature measurements.
- EXTI_0 interrupt for keypad menu interrupt
-*PWM* timer function initializing *PWM* outputs for the heating element and control fan
- LCD initializations and GPIO initialization for the LCD.
- Transceiver communications initialization for DMAstreams.
- *DMA* stream for ADC conversion

ADC1 channel 0 and 1 are on GPIO C0 and C1 and are run on 1 second intervals based on timer 3 interrupt events, the ADC will execute a two channel conversions and transfer the results via DMA2 stream 0, this handler will activate the *DMA* transmit flag which decides that the voltage to temperature conversion will execute and display sequence will execute.

If the keypad key '*' is pressed EXTI0_IRQHandler will update the B0 button flag to enter the menu display sequence as seen in the flow diagram from which each menu case can be selected.

Menu structure
Case 1: Display control fan RPM speed
- Enter TIM2_frequency_measurement() function
- Display RPM on the LCD until '*' key is pressed on the keypad

Case 2: Enter new Temperature Setpoint
- Enter keypad function to receive new input temperature

Case 3: Display *PWM* levels
- Display *PWM* on LCD for 2 seconds use *PWM* LCD display cycle

Default, Return to main loop if any other key is pressed other than 1, 2 or 3

When a new setpoint is entered the system will optimize for maximum response time by setting the fan maximum and the heater off for temperatures that are set lower than the measured point and setting the fan minimum and heater maximum for temperatures required above the current measured temperature. This condition is tested using the g_set_fan_flag.

The main loop calculates the time constant of the system based on approximated thermal resistance value and theoretically acquired thermal constant value. The adjustment counter is displayed on the LCD along with the current temperature readings and decrements each time a ADC reading is converted (every 1 second).
If the adjustment counter is equal to 0 enter the adjust *PWM* function using the error signal to determine adjustment levels.

If the error becomes within 0.2 degrees of the setpoint before the counter is complete then the adjust *PWM* function will be entered and if it is a new setpoint the adjust *PWM* function will also call the set *PWM* function and set *PWM* function will adjust the system for the open loop approximation based on the calculated system parameters.

***void tim3_init(void)* function**
This function enables the clock of TIM3 and sets TIM3 for a 1Hz *PWM* signal with a 50% duty cycle. TIM3 is then configured to create a trigger event so that it may be used for triggering ADC1 every 1 second.

***void ADC1_init(void)* function**
This function enables the GPIOC and ADC1 clocks. The GPIO C0 and C1 pin are configured for analog mode so that it can be used for ADC1 Channel 0 and channel 1 respectively. ADC1 is configured for interrupt from the TIM3 signal's rising edge and enabled in *DMA* for an interrupt priority of 4.

void DMA_ADC1_init(void) Function
Configure DMA2_Stream0 to transfer ADC1 Channel 0 and channel 1. Using channel 0, 16 bit size, set memory to increment, use a priority level of medium, enable the *DMA* interrupt, circular mode 2 cycles.

void GPIO_EXTI_call_menu(void) Function
This function is used to enable pin B0 as interrupt on falling edge trigger, in order to use keypad key '*' as interrupt in order to activate menu structure. Pin B0 is connected to column 0 of the keypad and is configured as an input with a pull-up resistor. Pin B15 is connected to Row 3 of the keypad and is configured as open drain output. When the '*' key is pressed on the keypad B0 will change from logic level 1 to 0 triggering the interrupt.

void EXTI0_IRQHandler(void) Handler
The EXI0 interrupt handler is used to activate the flag "g_button_press_B0" which is used in condition testing to determine if the menu should be called from the while loop.

void GPIOx_pin_config(GPIO_TypeDef* GPIOx,
uint32_t GPIOx_CLK_EN_bit_mask,
uint32_t GPIOx_pin_number,
uint32_t GPIOx_mode,
uint32_t periph_AF_number,
uint32_t GPIOx_output_type,
uint32_t GPIOx_output_speed,
uint32_t GPIOx_pupd) Function

Takes parameters defined in the function call to enable a GPIO pin of the chosen port and number for the chosen modes of operation as defined in the parameters

void GPIOx_Read_interrupt(GPIO_TypeDef *GPIOx, uint32_t GPIOx_CLK_EN_bit, uint32_t GPIOx_pin,
uint32_t EXTICR_EXTIx_mask, uint32_t EXTICR_EXTIx_PORTx_mask,
uint32_t EXTI_RTSR_TRx_mask, uint32_t EXTI_FTSR_TRx_mask,
uint32_t EXTI_IMR_MRx_mask, uint32_t EXTIx_IRQn, uint32_t GPIOx_priority) Function
Takes parameters defined in the function call to enable a GPIO pin as interrupt using the parameters to determine configuration details.

void PWM_controls_duty_cycle(void) Function
Configures GPIOB Pin 8 and Pin 9 as *PWM* outputs for Fan and heater Respectively

uint8_t Keypad_main(uint8_t keypad_mode) Function
Keypad scanning is a common function and there are multiple sources for the standard implementations methods used, one such method was derived from the *Embedded Systems* textbook.**[11]** used in the microprocessor course at SAIT. The following is a brief description of the method.
This function uses ***GPIO_Read_fourpin*** and ***GPIO_Write_OPENDRAIN*** functions to configure rows and columns as inputs with pull up resistors and open drain outputs in order to determine which key is pressed, the configuration is first set to Columns as inputs, and Rows as open drain, when a button is pressed a row will connect with a column and its pin will be pulled low, this triggers the keypad scan function which will use a bitmask code compare structure to find the column that was pressed, then the pins switch configuration so rows are inputs with pull-up resistors and columns are open drain. The keypad scan function is called again and the row is found. When the row and column are found the key that is pressed can be determined by comparing with a predefined array that represents the physical keypad.

This function contains multiple modes for different functions depending on which mode the function was called in based on the input in the function call.

If the keypad is called in Mode 1:
The keypad will scan for inputs 1, 2, or 3 and will return these values to indicate the menu choice, the value will be returned to the function that called.

If the keypad is called in Mode 2: New keypad entry
The keypad will scan for inputs up to 2 digits for temperature input, the value will be returned to the function that called.

void GPIO_Read_fourpin(GPIO_TypeDef *GPIOx, uint32_t GPIOx_CLK_EN_bit, uint32_t GPIOx_pin1, uint32_t GPIOx_pin2, uint32_t GPIOx_pin3, uint32_t GPIOx_pin4) Function
GPIO read four pin takes inputs from the parameters and pins that are used used in the keypad (columns or rows) function and sets gpio pins as inputs with pull-up resistors

void GPIO_Write_OPENDRAIN(GPIO_TypeDef *GPIOx, uint32_t GPIOx_CLK_EN_bit, uint32_t GPIOx_pin1, uint32_t GPIOx_pin2, uint32_t GPIOx_pin3, uint32_t GPIOx_pin4) Function
Takes the parameters and pins as inputs using the pins for the keypad (columns or rows) and configures them as outputs set as open drain

unsigned char keypad_scan(uint16_t Column_or_row) Function
The keypad scan function uses a bit mask code for each character and compares this bitmask with the GPIO pins associated with the keypad. Each code is tested and the code that matches will return the correct key. The function takes rows or columns as inputs to determine the mode of function.

int Keypad_debounce(void) Function
Using a condition testing and delay sequence a software button debounce corrects the issues of multiple inputs caused by mechanical buttons with springs. The function will be entered when the button is pressed and will not return true unless the button remains in the pressed state for enough time.

void LCD_display_cycle1(uint8_t mode) function
Using the common LCD functions the LCD display cycle will output the desired LCD display based on the mode which it is called.

Mode 1: Displaying Temperature from sensors Real time in main loop 1 second refresh rate
Mode 2: Displaying new set point for 1 second
Mode 3: Displaying *PWM* level Settings for 2 seconds
Mode 4: Displaying Indicator that Transmission is complete and what the Set point is, Display for 1 second

float Calculate_TimeConstant(void) function
Using the current set point the thermal resistance is set to its minimum value based on system parameters, the thermal resistance is determined by the fan speed and airflow. The thermal capacitance was determined experimentally and 1 time constant is (thermal resistance)*(thermal Capacitance) the value for time constants is returned and used as the adjustment counter.

double Rtd_to_temp(double rtd_voltage) function
Uses the quadratic equation and RTD coefficients to calculate the temperature of the RTD from its resistance. Takes RTD voltage from the ADC as an input and returns the temperature

void TC_ADC_2ch(void) function
Called from main on 1 second intervals, uses the results[] array that has the *DMA for* ADC1 Channel 0 and channel 1 values from voltage to temperature for both AD22103 and RTD sensors.

void Set_PWM_dutycycle(void) function
This Function is only called once per new setpoint and is used to calculate and set the *PWM* levels for the open loop estimate based on system parameters. Activates only when the error signal is within 0.2 of a degree.

void Adjust_PWM_dutycycle(float error_signal)
This function takes the input of the error signal and calculates the *PWM* levels required to achieve the desired set point.

void TIM2_frequency_measurement(void)
GPIOA Pin 1 Is configured as Input capture on TIM2_CH2 Used for Tachometer pulse generator input from fan for RPM.

9.5. SS1 SOFTWARE DESIGN

Some of the processes used in the *SCADA* SS1 are interrupt driven. The execution will take place based on order of priority where 1 is the highest configurable priority.

SS1 Priority Mapping
The interrupts used in the *SCADA* SS1 and their priority levels are show in Table 21**.**

Table 21 SS1 Interrupt Priority mapping

Interrupt Handler	**Priority**	**Notes**
DMA1_Stream1_IRQHandler	0 (highest)	DMA stream for *XBee* packet receiving.
DMA1_Stream6_IRQHandler	0	DMA stream for workstation packet receiving.
DMA1_Stream3_IRQHandler	1	DMA stream for XBee packet transmitting.
DMA1_Stream5_IRQHandler	1	DMA stream for workstation packet transmitting.
GPIOx_Read_interrupt	2	GPIOB interrupt to to prompt user for password entry with pin B0
GPIOx_Read_interrupt	3	GPIOA interrupt to monitor status of the door

SS1 Flow Diagram

The SS1 software program flow can be shown flow diagram of Fig.46**.**

The process starts on MCU power up. The program execution begins by initializing and configuring all the desired GPIO pins, interrupts peripherals, timers and DMA requests. The program tests password input as well as monitor status of door along with light sensor.

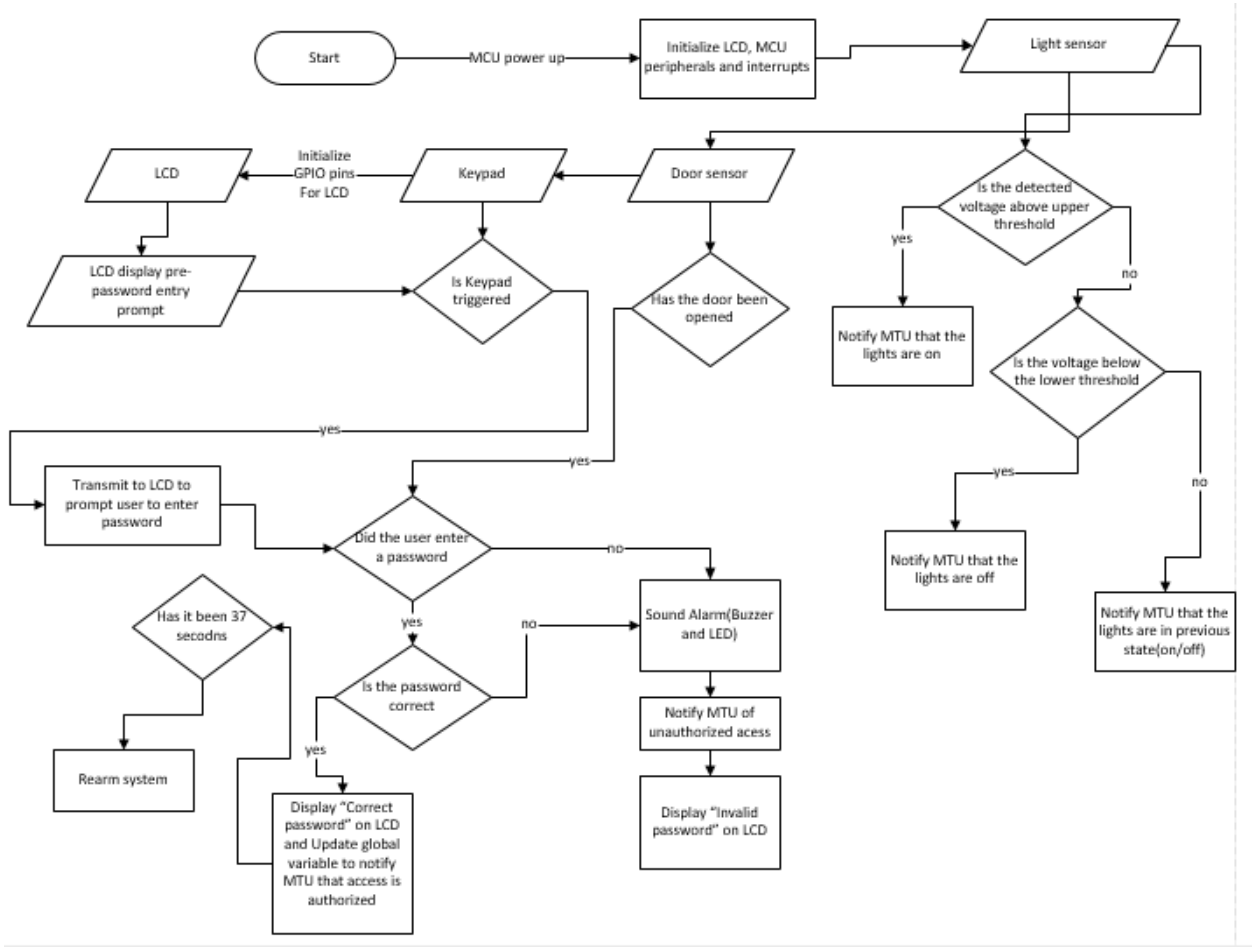


Figure 46 SS1 Flow Diagram

Table 22 SS1 Function Table

Function	Calls	Called by	Use
main	**- LDR_init** **- doorinit** **- Gpio_init_LCD** **- init** **- GPIO_EXTI_call_menu** **- delay_ms** **- LDRoutput** **- doorsensor** **- password** **-**	**N/A**	**-** The main function calls all necessary functions and initialization - Performs test conditions - Updates flags for transfer of data to MTU
LDR_init	**-peripheral_pin_multiplexing** **- ADCx_Init**	**main**	**-** Transfers control to process voltage levels for ADC
peripheral_pin_multiplexing	N/A	**LDR_init**	- Configures pin to alternate function mode
doorinit	N/A	**main**	- Configure GPIO PA8 as general purpose input to monitor door sensor - Configure interrupt for falling edge
GPIO_EXTI_call_menu	**GPIOx_Read_interrupt** **GPIOx_pin_config**	**main**	-Configures GPIO B0 as falling edge interrupt for keypad
GPIOx_Read_interrupt	N/A	**GPIO_EXTI_call_menu**	-Enables Interrupt for specific pin with configuration parameters
GPIO_Write_OPENDRAIN	N/A	**Keypad_main**	-Configure 4 pins as output in Open drain mode

Table 22 SS1 Function Table (continued)

password	**Keypad_main**	**main**	**-** Updates local password from MTU and compares it with user input password
Keypad_main	- **GPIO_Read_fourpin** - **GPIO_Write_OPENDRAIN** **- -keypad_scan** **-Keypad_debounce**	**main**	-keypad function -contains menu structure and sub functions relating to keypad operation
GPIO_Read_fourpin	N/A	**Keypad_main**	-Configure 4 pins as inputs with pull-up
GPIO_Write_OPENDRAIN	N/A	**Keypad_main**	-Configure 4 pins as output in Open drain mode
keypad_scan	N/A	**Keypad_main**	-Determine the Row or Column that was pressed by the keypad
Keypad_debounce	N/A	**Keypad_main**	-Small delay and condition testing to correct button bouncing
LDRoutput	N/A	**main**	- Reads voltage level from ADC and compares it to predetermined values. Updates lights flag
doorsensor	N/A	**main**	Reads status of door

void main (void) function
The *main* function runs after start up and initializes the Keypad, LCD, as well as the MCU peripherals and interrupts. The processes included in main are:
- ADC initialization for reading the LDR input
- EXTI_0 interrupt for keypad menu interrupt
- LCD initializations and GPIO initialization for the LCD.
- Initializing door input

ADC1 channel 1 is on GPIOA channel 1 and processes the analog input based on calculation in word. Based on the written values, the g_lights_flag will be updated to communicate with the MTU if the lights are on/off.
GPIOA PA8 is configured in general purpose input for falling edge trigger to determine

If the keypad key '*' is pressed, the EXTI0_IRQHandler will update the B0 button flag to allow user to input the password. When the user has entered the password, the program compares it with the password predetermined by the master terminal unit. If the password is correct and the external door is opened, the MTU is notified that that authorized access has been attempted. If the door is opened without the correct password or any password at all, the MTU is notified of unauthorized access. Unauthorized access will sound the buzzer as well as turn on the LED. Entering the correct password will cause the alarm system(buzzer and LED) to deactivate. Upon closing of the door after authorized entry, the system will rearm itself after approximately 37 seconds.

void LDR_Init(void) function
This function enables the ADC to read the voltage of the LDR circuit design from the PCB. Pin PA1 is configured as an analog channel.

void doorinit(void) function
This function configures pin PA8 for general purpose input mode. It is configured for falling edge trigger with an interrupt priority level 3.

void GPIO_EXTI_call_menu (void) function
This function is configured identical as in the TC1 software design and is common between it and SS1 software design

void LCD_Write_Two_line (uint8_t *line1, uint8_t *line2) function
This function is used to display a string on the LCD

uint8_t LDRoutput (void) function

This function sets the flags for the detected light levels (on/off). Returns 1 to light_status in main if the lights are on; 0 to light_status in main if the lights are off;

uint8_t doorsensor (void) function

This function sets the flags for the access attempts. Returns 0 to main if the door is open, 1 if the door is closed.

uint8_t password (void)

This function stores the MTU password locally into SS1. This password is then compared with the user input password to determine if the password was correct. The variable ‘attempts’ is incremented each time an element matches. In order for the password to be valid, attempts must increment to 4 with an initial value of 0. Once the door is closed after valid password input the system rearms and attempts = 0.

10. TESTING AND CHARACTERIZATION

10.1. MTU TESTING AND CHARACTERIZATION

Testing and characterization of the *SCADA* system occurs in the prototyping and assembly stages of the project. Each major electronics subsystem is tested to prove theory and allow application into the *SCADA* project.

10.1.1 MTU POWER SUPPLY

The voltages taken from the power supply regulators are in Fig.47. The -11.652V and +11.642V voltages are 10mV of each other in magnitude. These supply voltages are used with the LM324 op amps and are close to equal in magnitude. The +5.933V and +3.3067V voltages confirm that the LM7806 and LM1117 regulators are working as intended.

The oscilloscope screenshots in Fig.48 and Fig.49 show that the smoothing capacitors are working as intended for the power supply. The drop in voltages are due to the current requirements of the system's circuitry.

Figure 47 MTU Power Supply Voltages

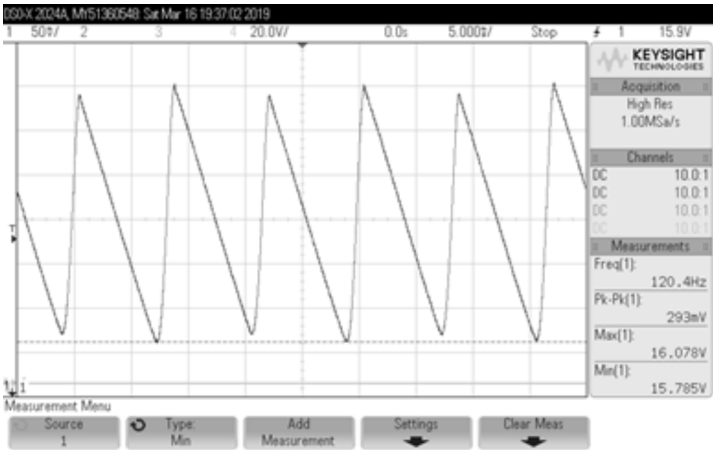


Figure 48 Smoothed +Voltage Capacitor(MTU)

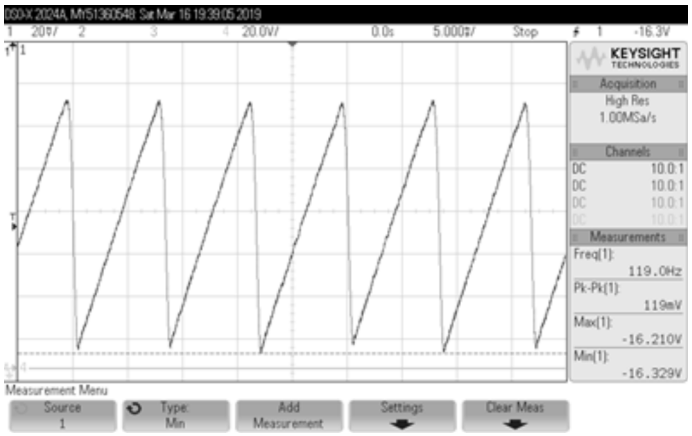


Figure 49 Smoothed -Voltage Capacitor(MTU)

10.1.2. MTU TEMPERATURE SENSOR CIRCUITRY

The temperature sensor circuitry includes the RTD with the 1ma constant current circuit, the instrument amplifier, the low-pass filter and the *span and zero* circuit to map the signal voltages from 0V to 3.3V. The mapping of 0V to 3.3V at temperatures 0℃ to 100℃ is the goal of the temperature sensor signal conditioning circuitry. In Fig.50 To test this, the RTD is switched with a 100Ω pot and a 50Ω pot. The resistance of the RTD goes from 100Ω to 139Ω when transitioning from 0℃ to 100℃.

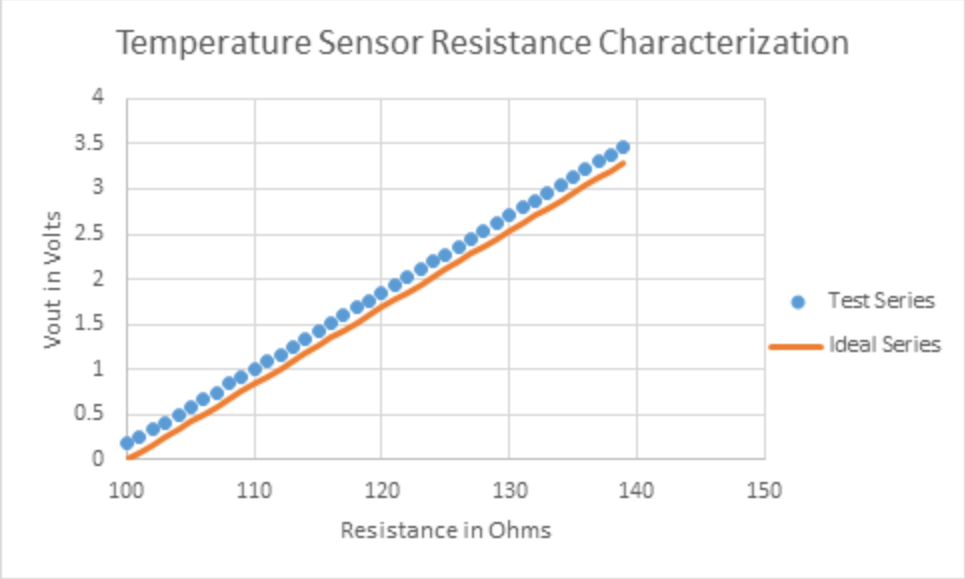


Figure 50 Temperature Sensor Circuit Test

Testing the temperature circuit at various resistances for RTD reveals that there is an intercept issue between the theory and application. This error is related to the 1% tolerance resistors used in the circuitry. To reduce this error, 0.1% resistors can be used instead. To reduce costs, it is more practical to develop an automated test that calibrates the software for the specific circuit components.

10.3 TC1 Characterization and Testing

10.3.1. TC1 Power Supply

10.3.1.1 Ripple Voltage Characterization Heater power supply

In order to determine the ripple voltage the heater was turned on to full load conditions to obtain the following oscilloscope measurements. A measured ripple voltage of 2.15V is considered acceptable due to the simple requirements of the energy dissipater. However two 4700uF capacitors were placed in parallel in order to achieve this level, they are practically speaking the largest capacitor that should be used due to price and size constraints.

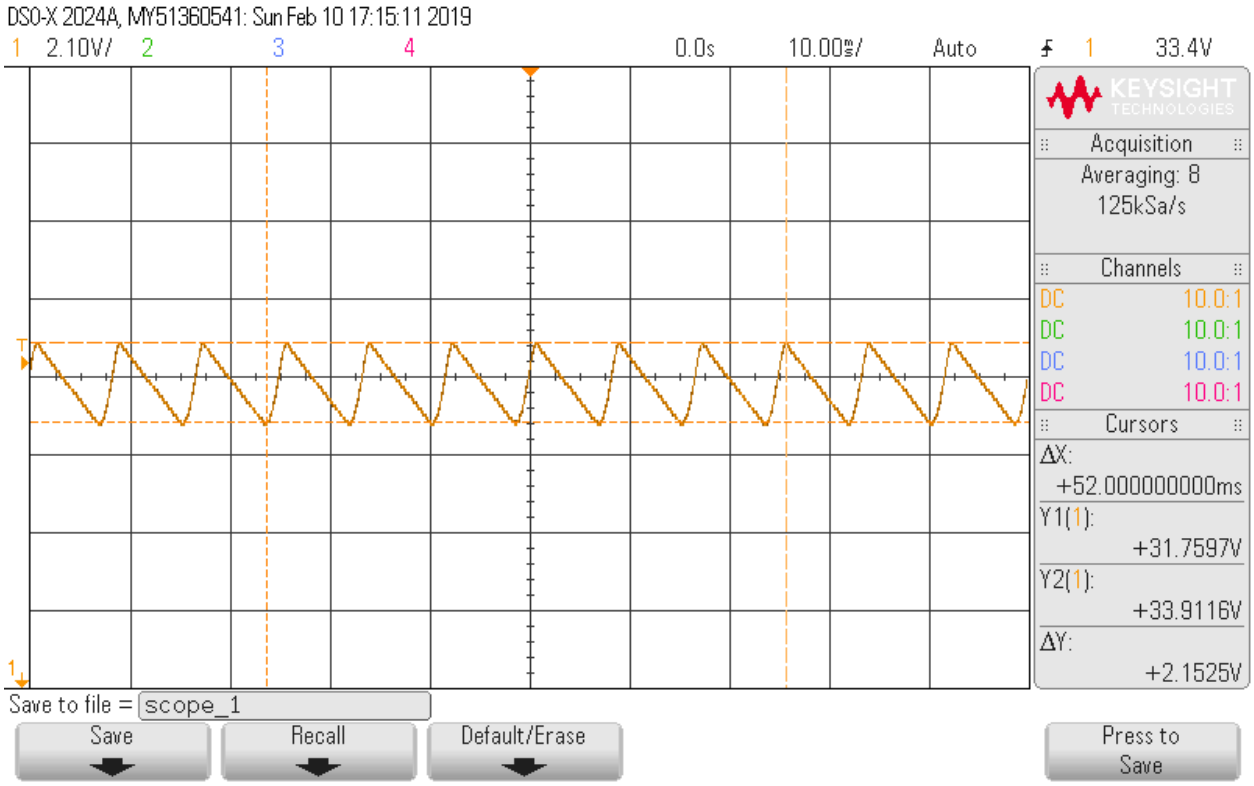


Figure 51 Ripple Voltage Unregulated Power Supply (TC1)

10.3.1.2. Ripple Voltage Characterization Main Circuitry power supply

Under full load conditions the ripple voltage of both polarities was measured.

The negative side ripple voltage was measured approximately 54.25 mV and the positive side at approximately 500 mV which is well within the line regulation rating of the LM317

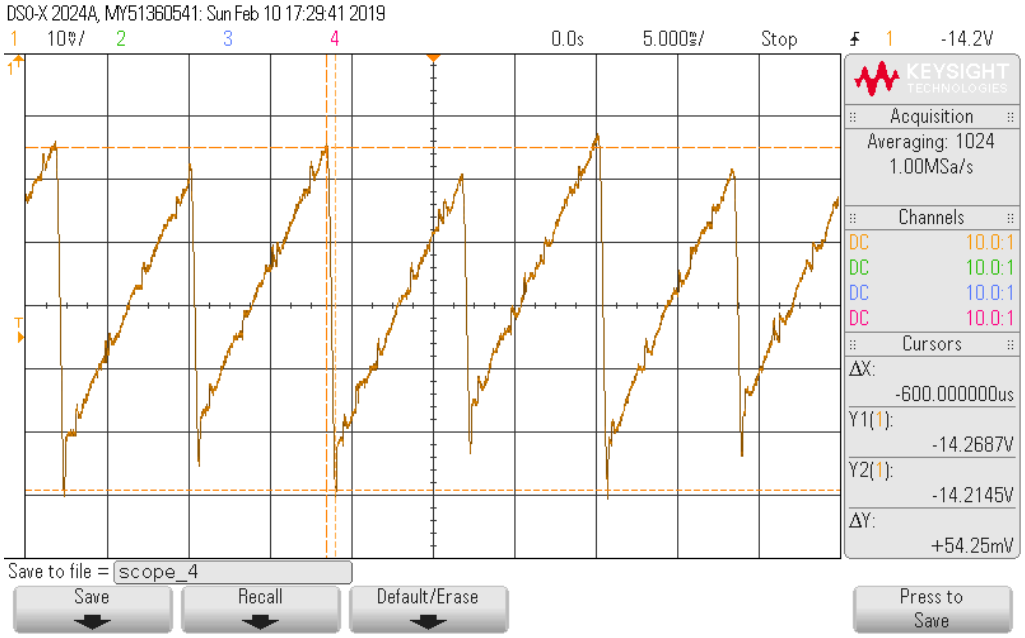


Figure 52 Main Circuitry Power Supply Full-load Ripple Negative side = 54.25 mV

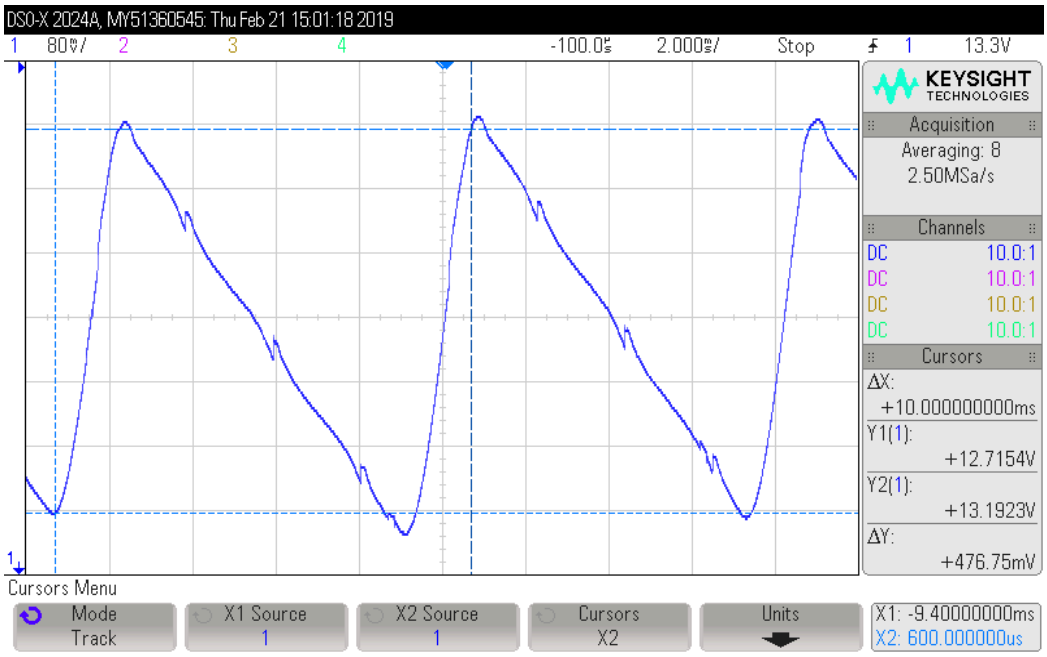


Figure 53 Main Circuitry Power Supply Full-load Ripple Positive side = 476.75 mV

10.3.2. Characterizing the Heating element Voltage

Connecting the heating element alone with no driving circuit as shown in the following image, the circuit was tested both with and without the shunting circuit in order to determine the effects of the shunting circuit.

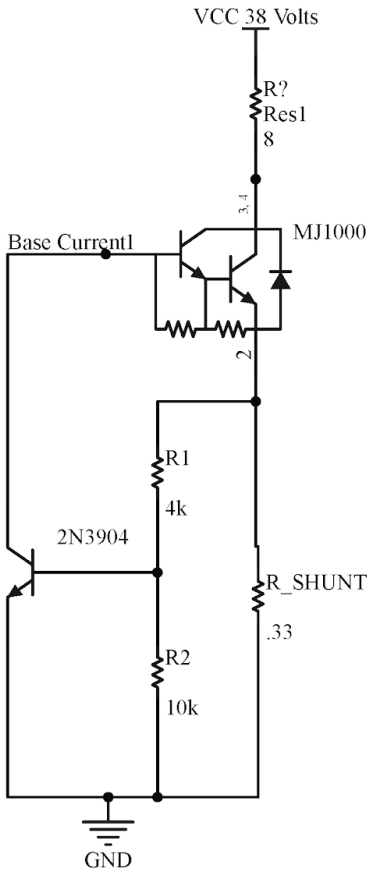


Figure 54 Heating Element with Shunt Circuit Schematic

Using the circuit in Fig.55 with and without the lower shunting circuitry the following graphs were created.

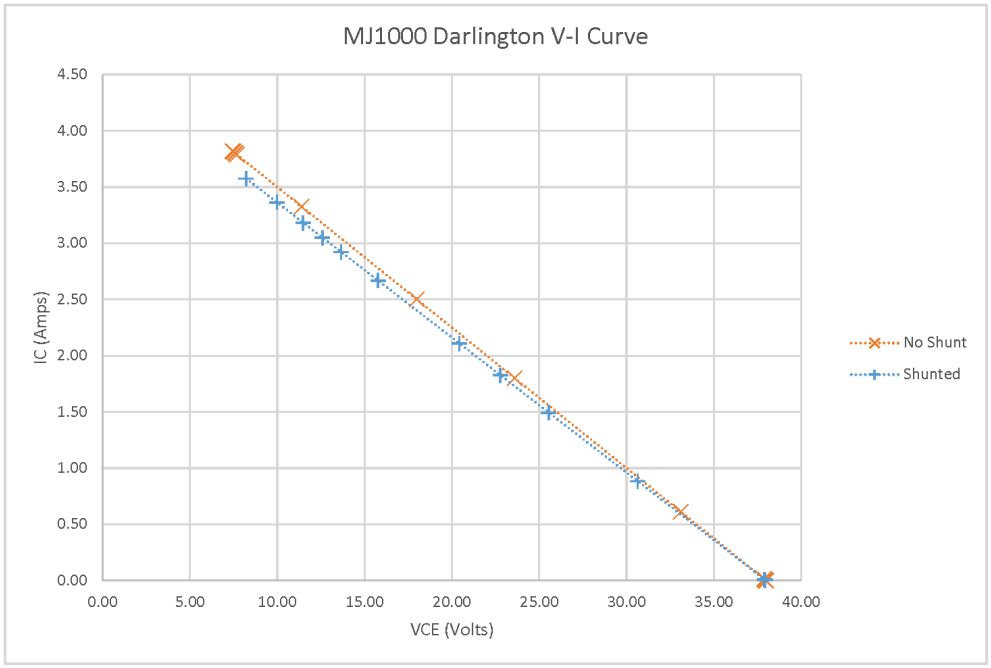


Figure 55 V-I Curve MJ1000 Shunted and Unshunted

From the above graph it can be seen that the unshunted MJ1000 and series resistor reaches saturation at 3.8 A. The shunting circuit can reduce this to IC (Sat) = 3.58 A @ VCE = 8.2 V

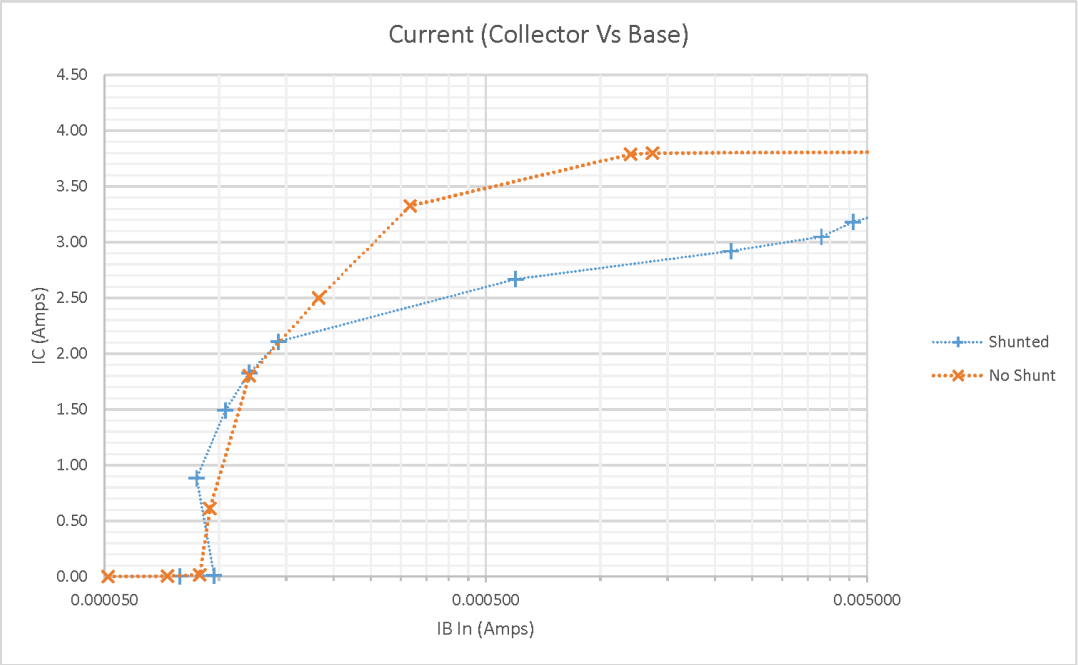


Figure 56 Base to Collector Current Graph Shunted and Unshunted

From the above graph the shunting effect can be seen as the Collector current approaches 3 Amps the gain is reduced and the amount of base current required to drive IC over 3 Amps increases.

10.3.3. In-amp and current source

10.3.3.1. Current Source Testing and characterization

The current source was tested under varying load conditions and the resulting data was used to plot the following graph. The stability of the current allows for 1mA to be considered constant under all possible loading conditions created by the RTD.

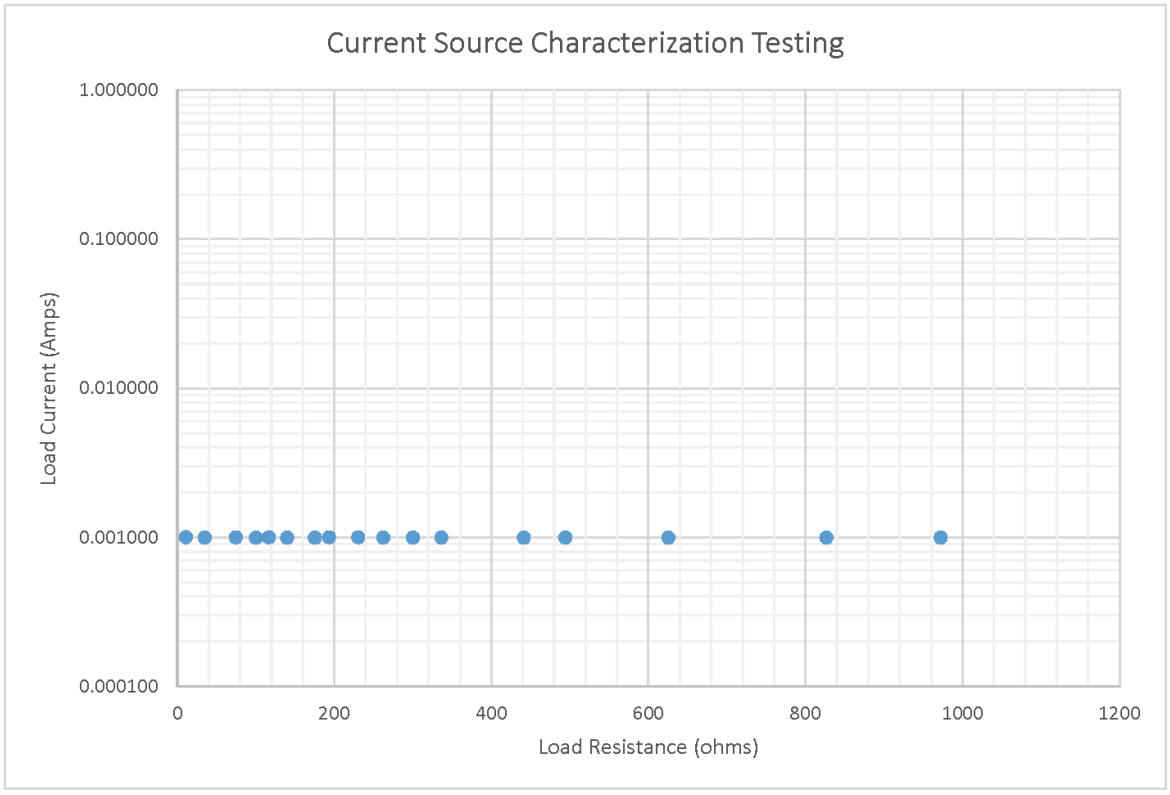


Figure 57 Current Source Load Characteristics

10.3.3.2. Instrumentation amplifier

Using a Vref of -6V as determined previously the circuit was built and tested as shown in the above *in-amp* circuit schematic. And the results are plotted in the following graph.

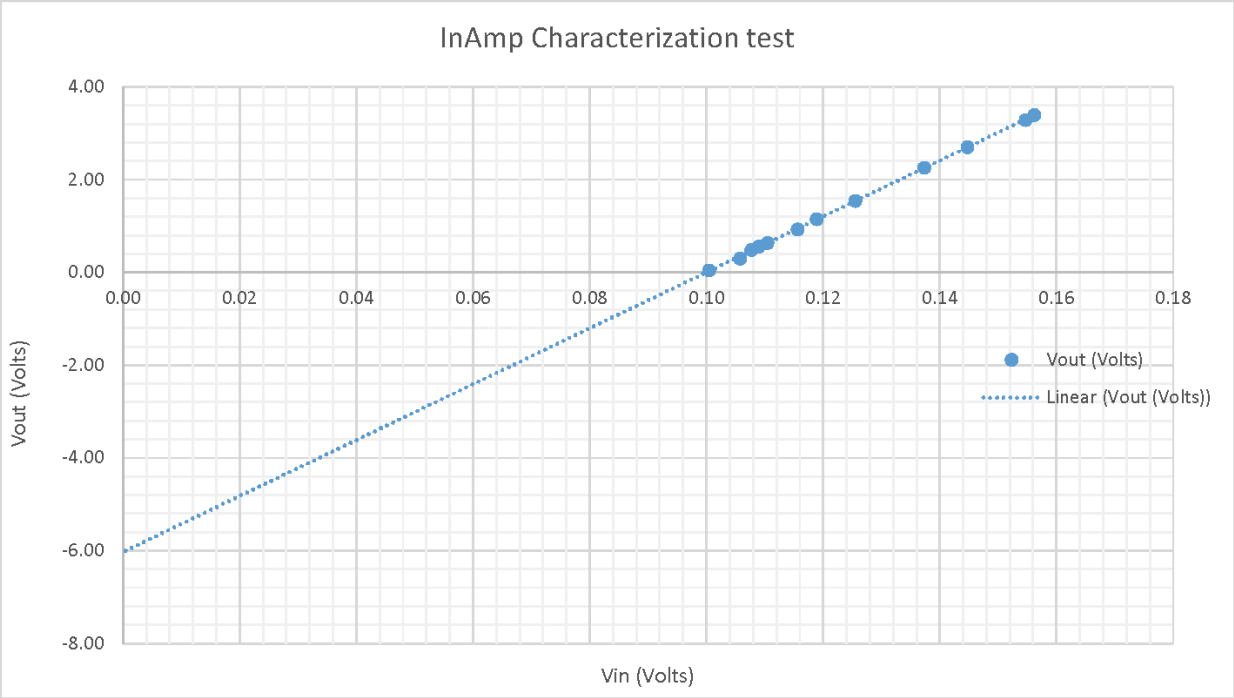


Figure 58 In-amp Characterization Test - Span and Zero Output

The *transfer function* of the *in-amp* has been verified to provide an output that is within the usable range of the ADC for the voltage input range created by the RTD and current source circuit.

10.3.4. Sallen-key filter testing

The Sallen-key filter was built and tested, the following graph displays the actual frequency response of the filter.

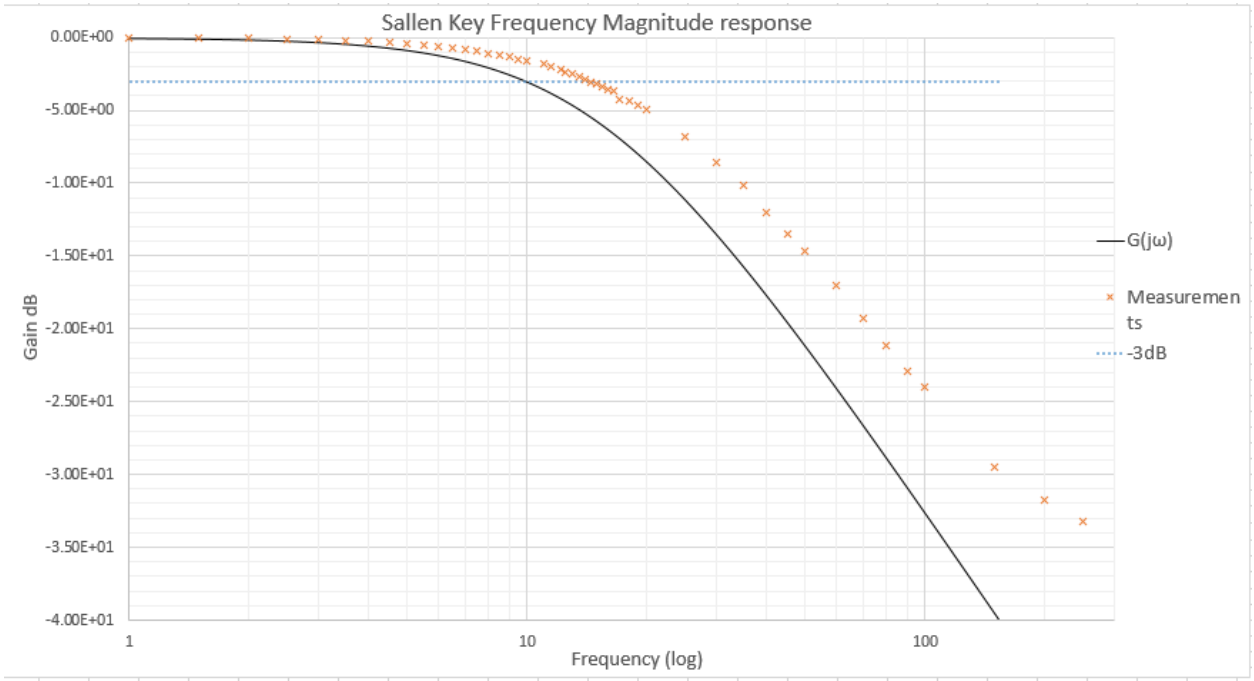


Figure 59 *Sallen–Key* Frequency Magnitude Response

From the experiment the actual cutoff frequency was measured to be 15Hz at the -3dB point, this is acceptable. If more filtering is required software can be used to average the measurements.

10.3.5. Heating Element Thermal Characteristics

The heater consists of the power dissipation elements such as the power transistor and power resistor, the power that is consumed by them can be determined by the voltage and current delivered by the power supply during operation.

$$Power = V * I = 38V * 3A = 114W$$

Thermal systems can be described by the following equation

$$T = R_T q$$

Where, T is temperature in Kelvin or degrees Celsius,
R is the thermal resistance of the system
And q is the heat flow, heat flow is the flow of work energy in Watts and can be equated to the power from the above calculation. Using the equation for the thermal system, the element can

be heated to its maximum temperature at its maximum power dissipation and once the temperature has stabilized the thermal resistance of the system can be calculated by solving for R. The thermal time constant of the system can be determined by finding the time required to reach 62.8 percent of the final temperature, and the time constant is equal to the thermal resistance multiplied by the thermal capacitance therefore the thermal resistance and capacitance can both be determined by performing an experiment that measures incrementally the temperatures of the element from room temperature up to max temperature for what is known as the heating charge and discharge curve.

An experiment was set up in the SAIT Electronics automated testing lab using the National instruments-DAQmx interface with the desktop PC and a thermocouple temperature sensor. The following image is from this experiment and the monitor shows the thermal heating charge discharge curve.

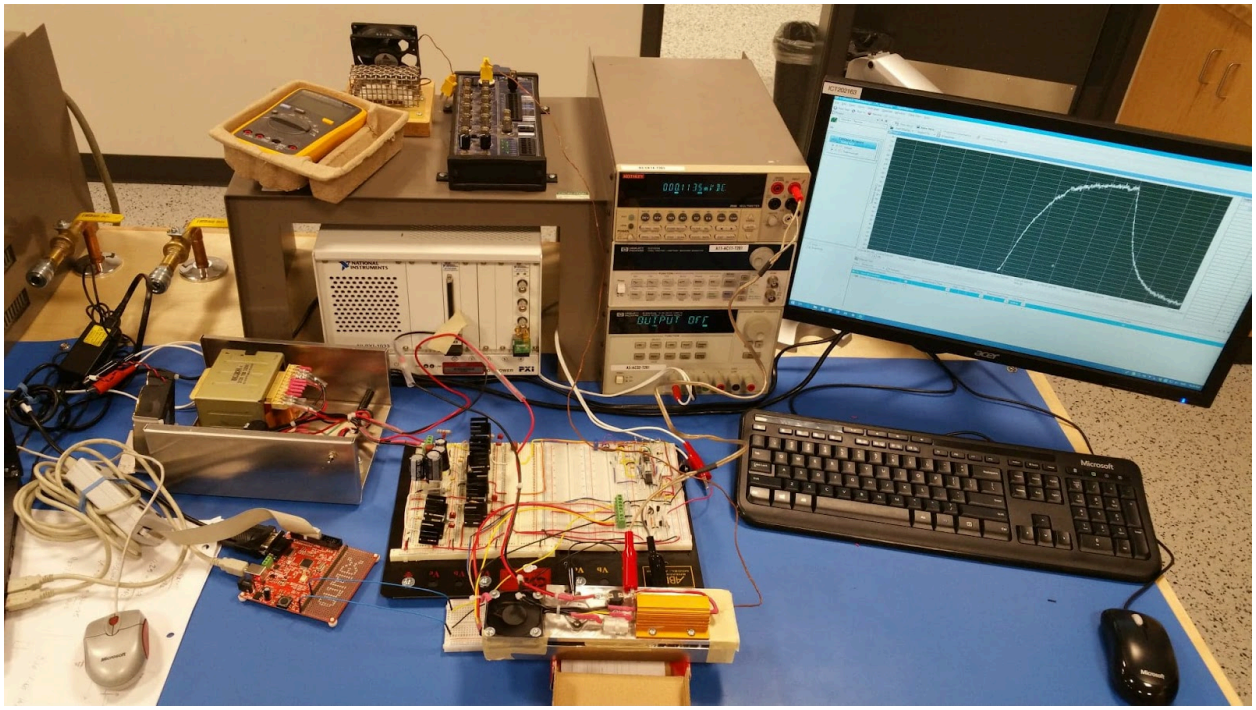


Figure 60 Automated Testing of Thermal Characteristics

The data from the experiment was logged for multiple fan speeds which causes the thermal resistance to change due to more airflow energy being removed from the heatsink at varying rates. The following graph is the compiled data from one of these tests.

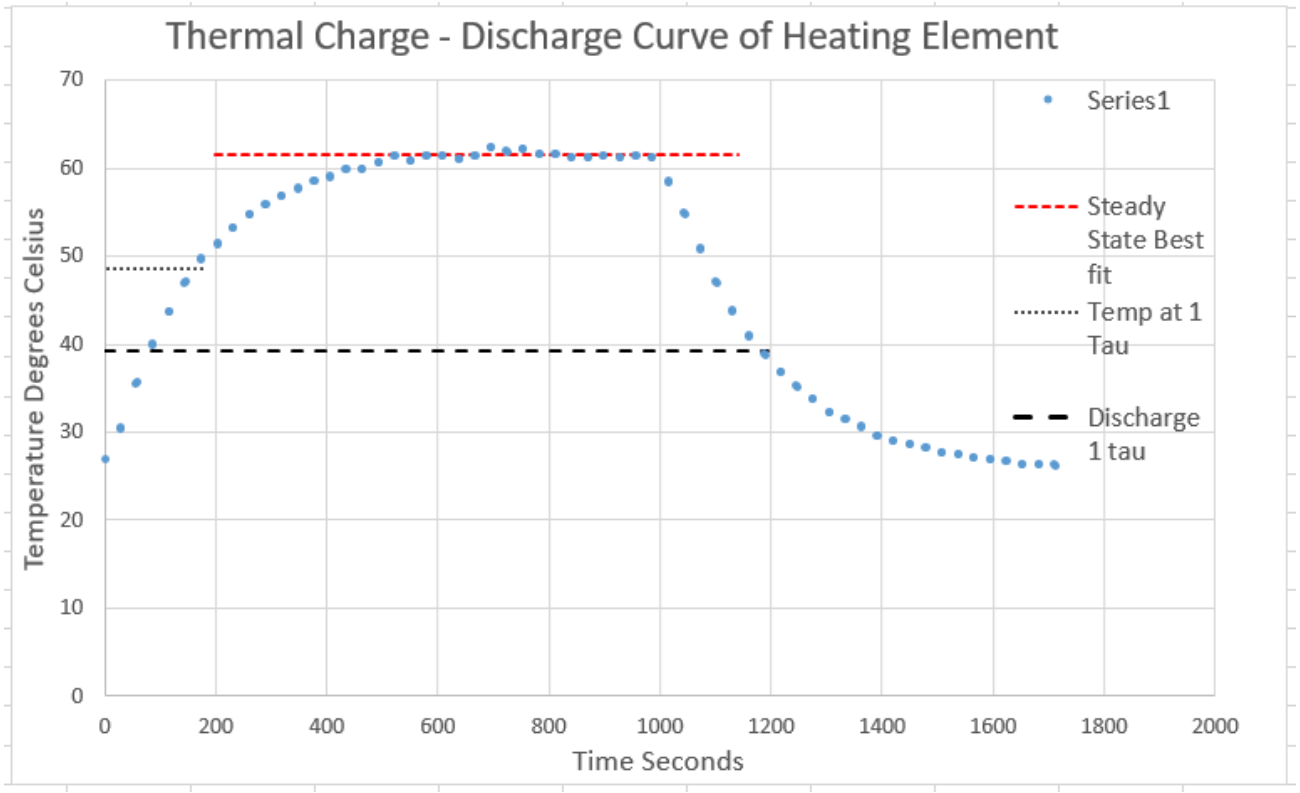


Figure 61 Thermal System Characterization

Using the data collected from the experiments the time constant of the system can be determined by finding the time required to reach 62.8 percent of the maximum temperature in charging and 62.8 percent of the decline from max down to room temperature for the discharge. The thermal capacitance value can be used to determine the estimated response time of the system at any given thermal resistance for different fan speeds which will help with determining the response time of the control system.

$$R_T = \frac{T_{max} - T_{Room}}{q} = \frac{62 - 25}{114} = 0.325$$

The thermal resistance is therefore calculated from the data to be 0.325 degrees celsius per watt.

From the graph the value of 1 time constant is measured at 174 seconds, and the thermal capacitance is calculated by:

$$C = \frac{Tau}{R_T} = \frac{174}{0.325} = 535$$

Using the values from several thermal tests and different fan speeds the average calculated value for thermal capacitance was 660, this is the value that can be used in software to develop

a response time that is sensitive to the thermal time constant of the system by using the thermal resistance multiplied by thermal capacitance conversion to time constants.

10.4. SS1 Characterization and Testing

10.4.1. LDR circuit testing

The output voltage of the LDR circuit at varying LDR resistances was tested to characterize the LDR *span and zero* circuit

Table 23 SS1 LDR Theoretical and experimental Outputs

Theory			
Resitance of LDR(k)	Vodiff	Vosallen	Vospan
9.73	5.18	5.18	1.68
8.78	4.96	4.96	1.76
7.79	4.68	4.68	1.84
6.85	4.4	4.4	1.92
5.01	3.76	3.76	2.12
9.99	5.24	5.24	1.66
20.1	6.78	6.78	1.19
30.1	7.56	7.56	0.946
40	8.04	8.04	0.8
50.1	8.36	8.36	0.7
60.1	8.58	8.58	0.628
70.1	8.76	8.76	0.575
80.4	8.9	8.9	0.532
0.104	0.96	0.96	2.98
0.198	1.07	1.07	2.96
0.296	1.125	1.125	2.93
0.602	1.38	1.38	2.86
1.03	1.69	1.69	2.76
2.01	2.32	2.32	2.57

Resitance of LDR(k)	Vospan
0.104	2.95
0.198	2.93
0.296	2.91
0.602	2.83
1.03	2.73
2.01	2.54
8.74	3.01
10.67	1.79
27.02	1.26
30.3	1.21
55.4	0.963
9.73	1.64
8.78	1.71
7.79	1.79
6.85	1.88
5.01	2.09
9.99	1.63
20.1	1.14
30.1	0.891
40	0.743
50.1	0.64
60.1	0.57
70.1	0.52
80.4	0.474

Graphing the data collected in the above tables results in the following: showing experimental and theoretical values

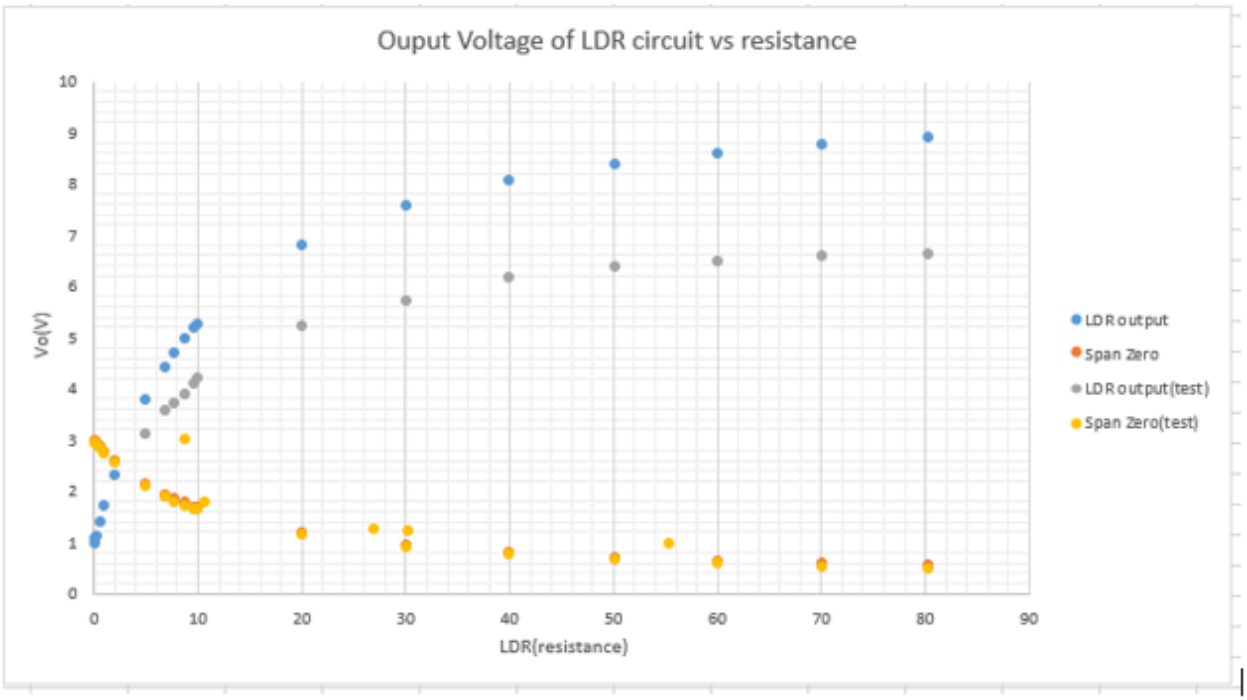


Figure 62 SS1 LDR Span and Zero Output Voltage vs LDR Resistance

11. SCADA SYSTEM LAYOUT

The *SCADA* system is composed of multiple chassis, each with their own PCB and chassis layout.

11.1 MTU System layout

The pictures in Fig.63 and Fig.64 show the external and internal

Figure 63 MTU External

Figure 64 MTU Internal - System Layout

11.2 RTU TC1 System layout

The following images show the complete project chassis layout and final design of the temperature controller unit

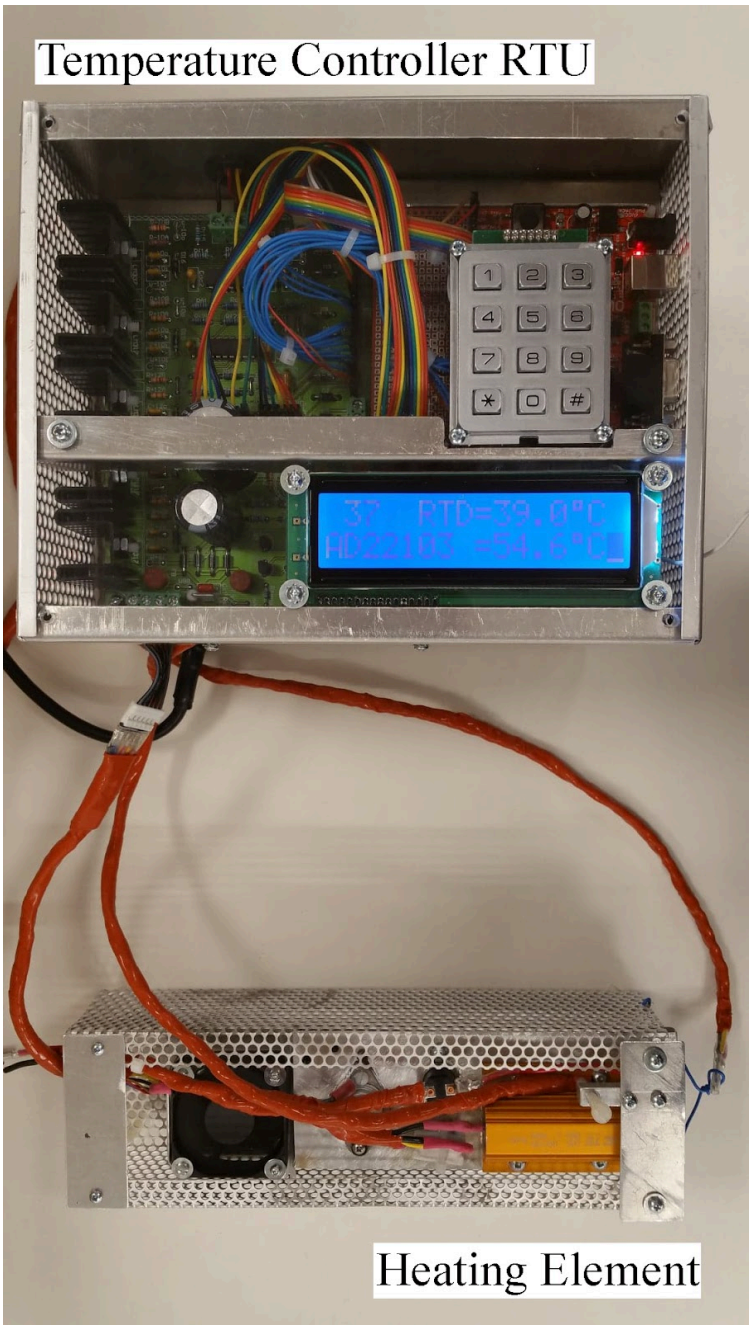


Figure 65 TC1 System Layout

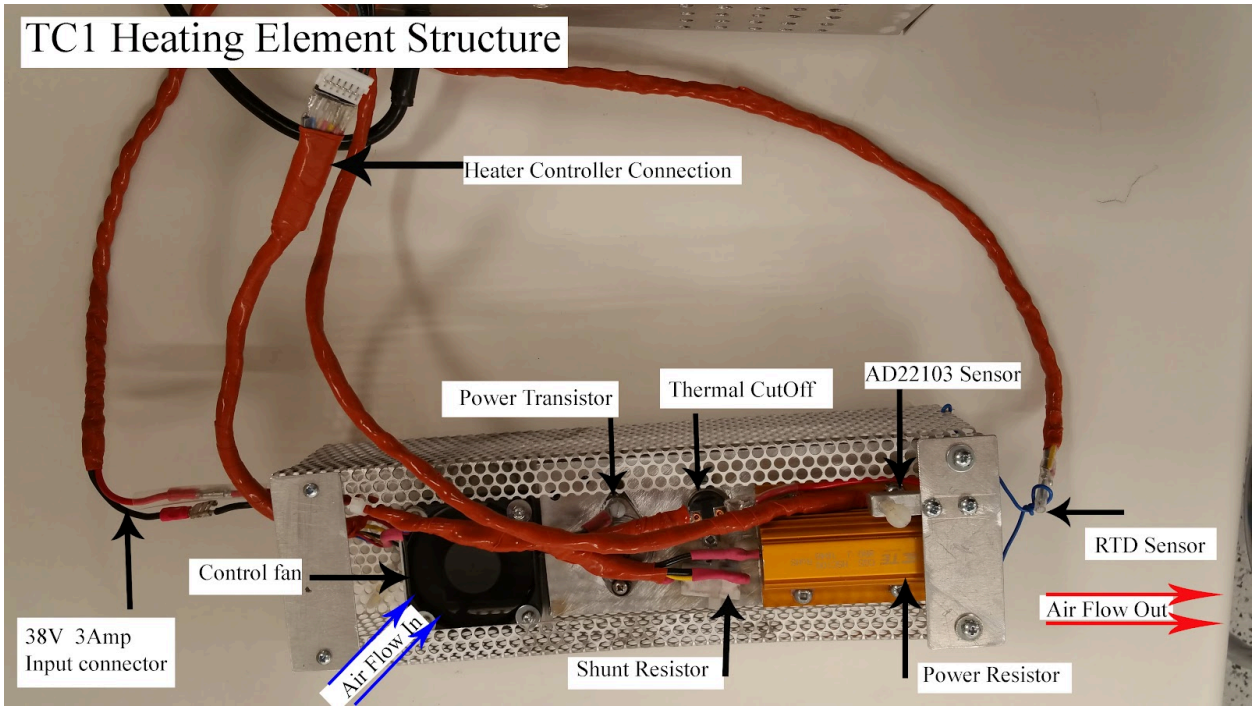


Figure 66 TC1 Heating Element Layout

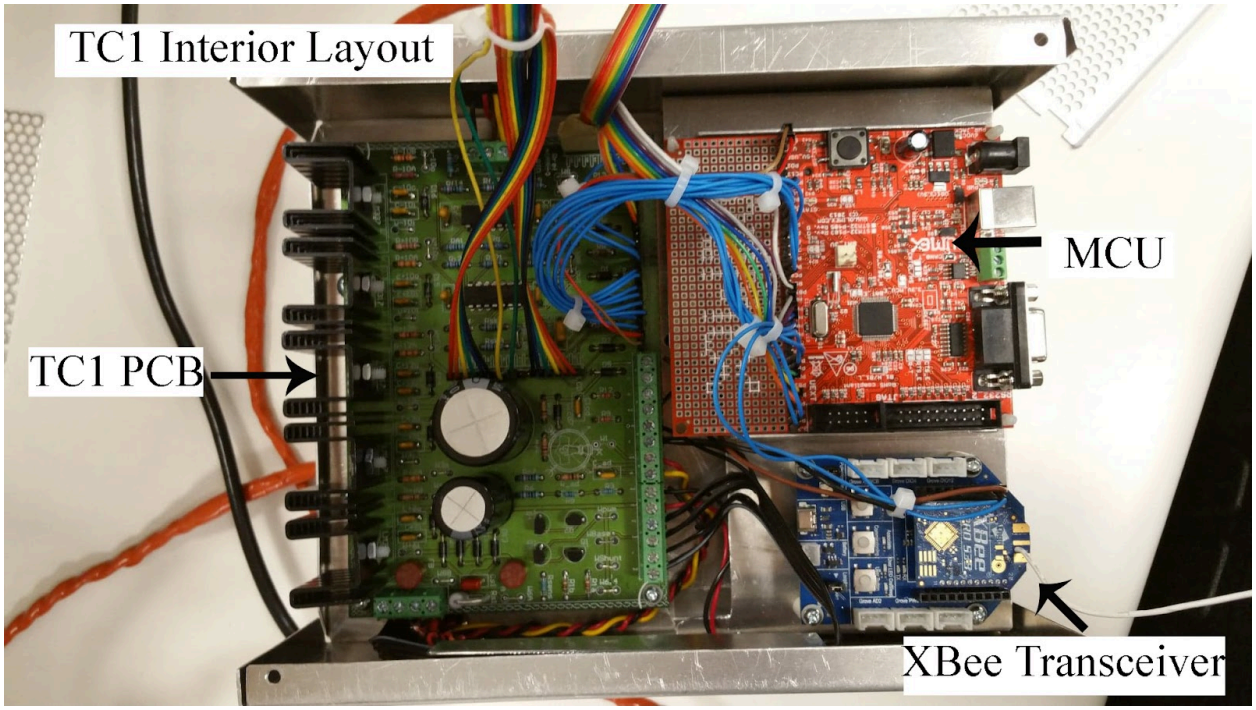


Figure 67 TC1 Interior Layout

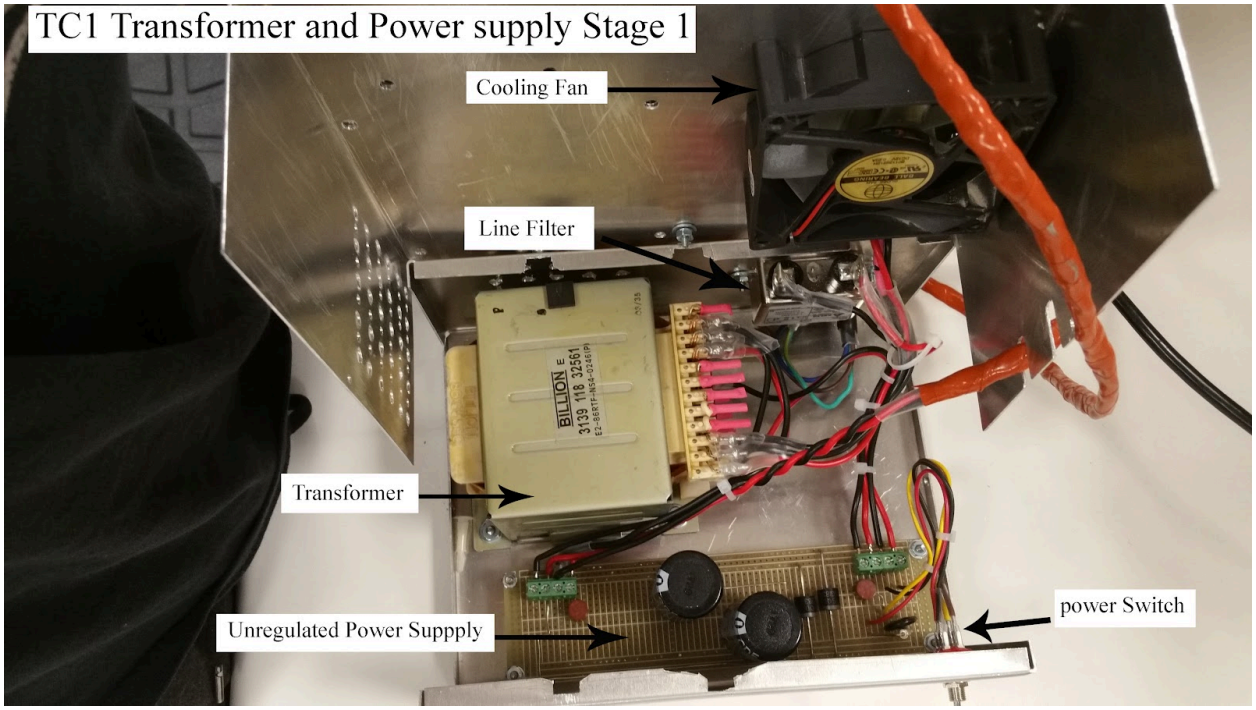


Figure 68 TC1-Power Supply: Stage 1 Layout

11.3 RTU SS1 System layout

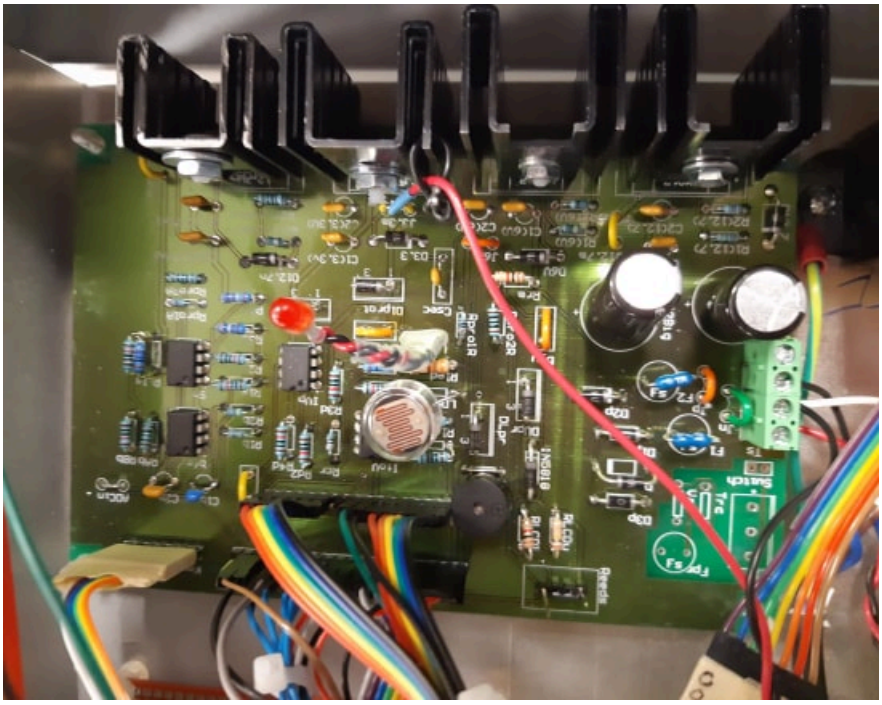


Figure 69 SS1 PCB Layout

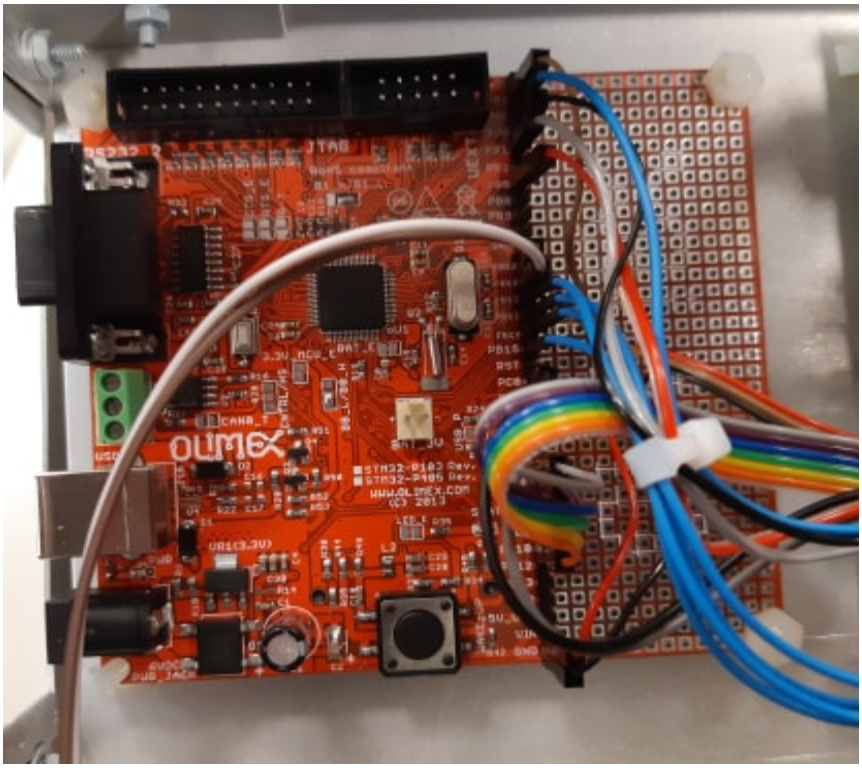


Figure 70 SS1 MCU

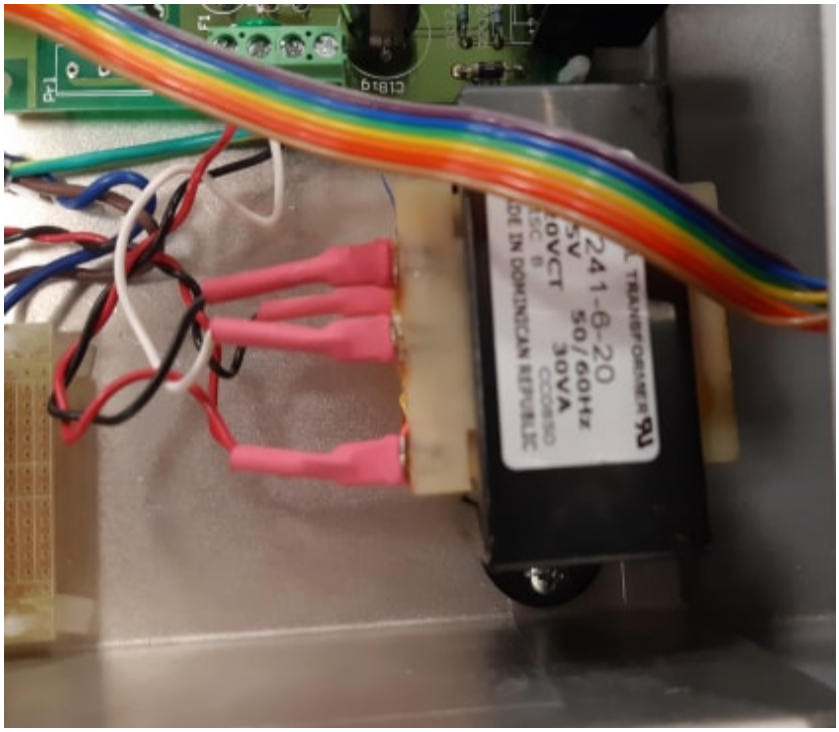


Figure 71 SS1 Transformer

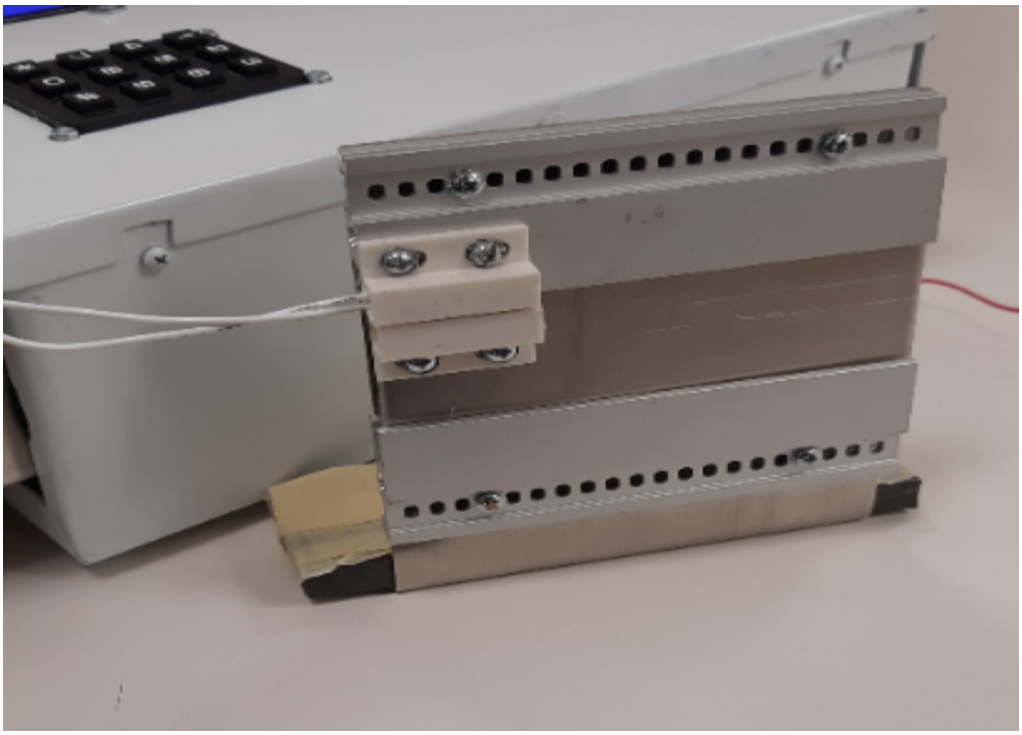
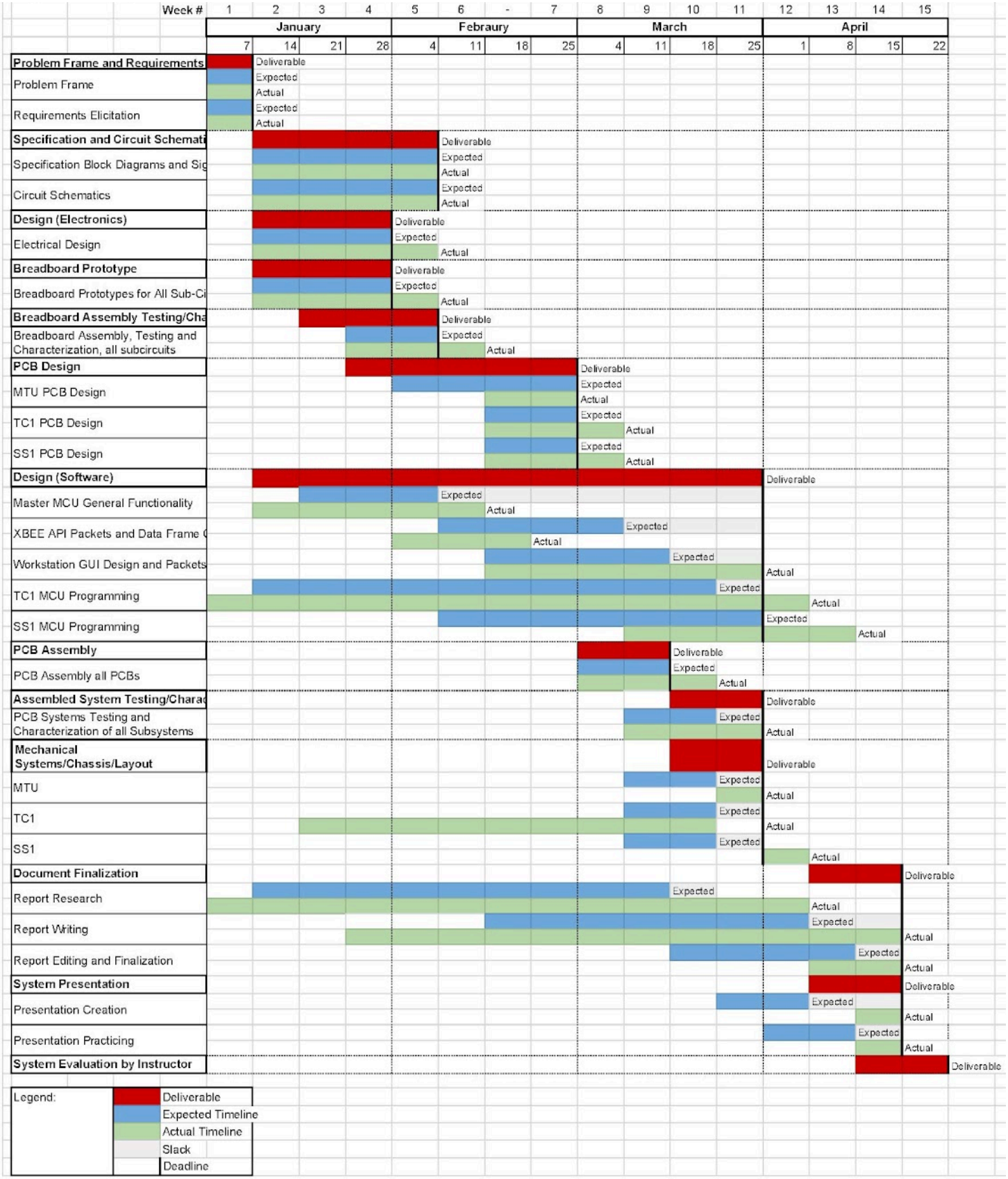


Figure 72 SS1 Door Sensor(white attachment) and Door

12. GANTT CHART



13. REQUIRED RESOURCES

The resources used in this project are both those that are available through SAIT as well as electronic components to be ordered for prototyping and the final solution. The required resources for this project include:

- SAIT room NJ406 laboratory equipment:
    - Keysight InfiniiVision DSOX2024A oscilloscope.
    - BK Precision 1760A DC power supply.
    - BK Precision 5492 5½ digit multimeter.
    - Agilent 33521A arbitrary waveform generator.
- SAIT room NJ405 fabrication equipment:
    - Pexto sheet metal finger brake.
    - Kidder MFG. corner shear.
    - Di-Acro spartan shear.
    - Rockwell bandsaw.
    - Hakko FX-951 soldering station.
- SAIT room NJ405 fabrication materials:
    - Screws, nuts, bolts and washers.
    - Chassis material (sheet metal, plexiglass, metal grates).
- Electronic components (see Appendix H: Parts List).

14.0 COSTS

The costs of the capstone *SCADA* system are broken down into the materials used in prototyping and final product as well as the ENT labour at an rate of $20/hour.

14.1. Materials

The following table is a summarized list of all components that were purchased during the process of the project. The list is comprised of all items that were used in:

- The final assembled products
- Throw away prototypes

Also included are:

- Extra and spare components
- Unneeded and unused components

A complete list of all components and part numbers is available in the appendix H: PARTS LIST

Table 24 Materials List and Total Costs

Materials list			
Component	**Quantity**	**Subtotal**	
Capacitors	64		$46.97
Resistors	259		$44.01
Diodes	128		$30.65
Regulators	43		$65.27
Fuses	19		$16.39
Transformers	2		$29.82
Sensors	5		$29.71
Heatsinks	16		$16.26
LCD	3		$43.30
Screw Terminals	14		$9.81
Switches	6		$14.70
Misc	57		$93.58
Component Count	613.00	**Total**	**$440.46**

Shipping Costs	**Number of orders**	**Total Shipping Costs**
26.68	9	**$240.12**

	Total Project Material Costs	**$680.58**

14.2. Labour

The labour is broken down by the time spent on the major deliverables found in the GANNT chart in section 12 and is placed into Table 25.

Table 25 Labour Spent and Cost Calculation

Deliverable	**Hours Spent**	**Labour Cost @ $20/h**
Problem Frame and Requirements Elicitation	15	$300
Specification and Circuit Schematics	23	$460
Design (Electronics)	46	$920
Breadboard Prototyping General	12	$240
Breadboard Testing and Characterization	15	$300
PCB Design	14	$280
Software Design	41	$820
PCB Assembly	15	$300
Mechanical Systems/Chassis/Layout	51	$1020
Document Finalization	45	$900
System Presentation	13	$260
Totals	**290**	**$5800**

15. CONCLUSION

The *SCADA* system monitors and controls a security system unit and temperature controller unit via *XBee* wireless *transceivers*. The *SCADA* system project has successfully allowed for ENT students to demonstrate essential skills and knowledge gained from their program. The MTU demonstrates knowledge gained in communication systems and CVI programming. The TC1 demonstrates knowledge gained in control systems analysis and *PWM* controls with driver circuitry. The SS1 demonstrates knowledge gained in raw sensor signal conditioning and state machines. All systems demonstrate essential skills in ARM M4 Cortex programming, electronic fabrication, PCB design and lab equipment use.

RECOMMENDATIONS

To replicate and/or further develop the *SCADA* system, the recommendations are:

1. Use a bus communication protocol between *XBee* transceivers in transparent mode for easy communication. The MCU communication to the *XBee* is to be done through UART with *DMA*, whereas communication between the *XBee*s is a bus protocol. The Application Programming Interface (API) mode of the *XBee* can be used for communication, but it caused errors when troubleshooting the wireless communications.
2. Adding additional RTUs is viable through:
    - multiple windows in the CVI GUI.
    - multiple output files.
    - including additional addresses into the MTU software.
    - additional polling states to the *SCADA* cycle in the MTU software.

REFERENCES

[1] D. Bailey and E. Wright, *Practical SCADA for industry*. Oxford: Elsevier, 2006. p.19, pp.46-47

[2] *Temperature handbook,* 2nd ed. Stamford, CT: Omega Engineering, 2000.

[3] R. L. Boylestad, *Introductory circuit analysis*, 13th ed. Boston: Pearson, 2016. pp. 968-971

[4] D. A. Bell, *Fundamentals of electronic devices and circuit*s, 5th ed. Don Mills, Ont.: Oxford University Press, 2008. pp. 562-563

[5] P. Horowitz and W. Hill, *The art of electronics,* 3rd ed. New York: Cambridge University Press, 2018.

[6]"designing with thermally protected TMOV varistors in SPD and ac line application", *Littelfuse.com*, 2019. [Online]. Available: https://www.littelfuse.com/~/media/electronics_technical/application_notes/varistors/littelfuse_designing_with_thermally_protected_tmov_varistors_in_spd_and_ac_line_application_note.pdf. [Accessed: 16- Apr- 2019].

[7]"Unregulated Power Supply", www.electronics-tutorials.ws, 2019. [Online]. Available: https://www.electronics-tutorials.ws/blog/unregulated-power-supply.html. [Accessed: 16- Apr- 2019].

[8] B. Baker, "Precision Temperature-Sensing With RTD Circuits"*, Ww1.microchip.com*, 2019. [Online]. Available: http://ww1.microchip.com/downloads/en/appnotes/00687c.pdf. [Accessed: 16- Apr- 2019].

[9]A. Teeramongkonrasmee, "Industrial Electronics Signal Conditioning and Transmission", *Pioneer.netserv.chula.ac.th*, 2019. [Online]. Available: http://pioneer.netserv.chula.ac.th/~tarporn/487/HandOut/SigCon.pdf. [Accessed: 16- Apr- 2019].

[10] T. L. Floyd, *Electronic devices: conventional current version,* 10th ed. NY, NY: Pearson, 2018. pp.763-764

[11] Y. Zhu, *Embedded systems with ARM Cortex-M microcontrollers in assembly language and C*, 3rd ed. United States of America: E-Man Press LLC, 2018.

GLOSSARY

Digi XBee - Is the brand name of a family of radio modules from Digi International.

Direct memory access (DMA) - Is a feature of computer systems that allows a hardware subsystem to access main system memory independently of the central processing unit (CPU).

Embedded system - Is a controller that has a dedicated function within a larger system, it is embedded as part of a complete device and is controlled by a real-time operating system

Instrumentation Amplifier (in-amp) - Is a fully differential amplifier which utilizes input buffer amplifiers that eliminate the need for input impedance matching and provide zero loading effect. This makes the in-amp ideal for use in measurement and test equipment.

Modbus - Is a serial communications protocol originally published by Modicon

Pulse Width modulation (PWM) - is a method for modulating the average power delivered by an electric signal. The signal is controlled by switching between on and off at a fast rate, effectively chopping it up into discrete parts.

Sallen–Key - Is an electronic filter topology used to implement second-order active filters that is particularly valued for its simplicity.

Span and Zero - Is a linear calibration technique that maps inputs to outputs for a straight line equation. The zero adjustment is used to produce a parallel shift of the input-output curve. The span adjustment is used to change the slope of the input-output curve.

Supervisory Control and Data Acquisition System (SCADA) – is a system of software and hardware elements that allows:

- local or remote control of processes
- The monitoring, gathering and processing of data in real time
- Direct interaction through HMI
- Recording of events into memory logs

Transceiver - A device that can both transmit and receive communications, in particular a combined radio transmitter and receiver.

Transfer function - Is a mathematical function used in electronic or control systems which theoretically models the output of a system for each possible input.