High Precision Temperature Controller

Initial Project Proposal

Group A

Ashley Desiongco
Stacy Glass
Martin Trang
Cara Waterbury

Sponsored by

INFRARED SYSTEMS DEVELOPMENT CORPORATION

Executive Summary

Infrared Systems Development Corporation (ISDC) is an engineering design firm that specializes in the design and construction of infrared test equipment. One of these test systems is a blackbody source with a high precision temperature controller. The current temperature controller being used by Infrared Systems Development is a commercial off the shelf product that specializes in controlling industrial heaters. The company that produces these controllers has made modifications to the hardware and software design to accommodate the needs of ISDC at a high cost and long turnover time. Since there are essential features that are absent and a plethora of unnecessary ones that are present, Infrared Systems Development wants to develop and design a temperature controller that will be able to have the same accuracy and speed as the current COTS product but with the ability to make modifications quickly and easily.

Motivation

Motivation for this project came from one of our group members, Martin Trang. He currently has an internship at Infrared Systems Development located in Orlando, Florida. His manager and CEO of the company, Andy Duran, expressed interest in developing a high precision temperature PID controller at their facility for his company to both utilize and sell. Once Martin heard of this interest, he quickly asked for the opportunity for our senior design group to take on the challenge. Mr. Duran graciously agreed to let us design this controller and offered to sponsor the project financially.

Infrared Systems Development Corporation has many specialties, including: preamplifiers, radiometers, blackbody sources and spectrometers to name a few. Our project will specifically cater to Infrared Systems Development’s specialty in blackbody sources. The company’s blackbody sources have very large temperature ranges, some going as high as 1200 degrees Celsius and have temperature resolutions of 0.1 degrees Celsius. These sources can take up to 80 minutes to fully respond and therefore, need some kind of controller to regulate the source and a monitoring system to alert the user of temperatures approaching the set point. This is our motivation: to design and build a controller to regulate and monitor these blackbody sources.

To supplement the research and design of our PID controller, we also need to give some information on the blackbody sources we will be interfacing with. The internal temperature sensors are embedded inside the cavity of the source and are, depending on the temperature range of the source, type S thermocouples, type T thermocouples or Platinum RTDs. There are two of these temperature sensors, each connected through a thermocouple connector and referenced to an external ice bath (0 degrees Celsius). The readings from both are outputted to

2 pins of the output connector, via thermocouple extension wire, which will be explained in its own section below.

Blackbody Sources

The two sections below, Cavity Blackbodies and Extended Area Blackbodies, describe the different type of blackbodies the temperature controller will need to interface with. The sections -will discuss the general operating parameters including the temperature range, power source, and other characteristics that may affect the design of the temperature controller.

Extended Area Blackbodies - The extended area blackbody sources have a large cavity area versus the cavity blackbodies. Their temperature ranges are lower than compared to the cavity blackbodies. The IR-140 has a 12” by 12” heating area and a temperature range of ambient to 230°C. The IR-160 has a heating area of the 12” by 12” and has a temperature range of ambient to 350°C. The IR-150 has a heating area of 12” by 12” and has a temperature range of ambient to 500°C. The IR-2100 has a heating area of 2” by 2” and a temperature range of -5°C to 145°C. The IR-2101 has a 2” by 2” heating area and has a temperature range from -30°C to 75°C. The IR-2103 has a 3” by 3” heating area and a temperature range from -5°C to 145°C. The IR-2106 has a 6” by 6” heating area and a temperature range from 5°C to 150°C. The IR-140, IR-150, IR-160 heater are heater coils using AC power. The IR-2100, IR-2101, and IR-2103 are thermoelectrically heated and cooled using DC power.

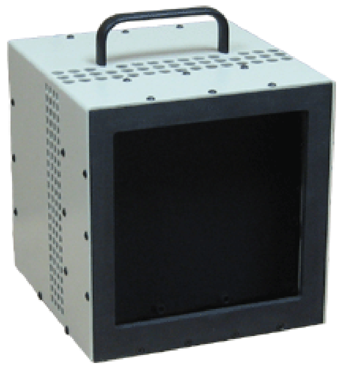


Figure 1: IR-2106 Extended Area Blackbody

Cavity Blackbodies -The cavity blackbodies have a smaller cavity area than the extended area blackbodies. There are seven different models that are being designed by Infrared Systems Development. The IR-508 has a ¼” cavity and a temperature range of 50°C to 1050°C. The IR-519 has a .4” cavity and a temperature range of 50°C to 1200°C. The IR-518 has .4” cavity and 50°C to 1050°C. The IR-704 has a ¼” cavity and a temperature range of 50°C to 1050°C. The IR-564 has a 1” cavity and a temperature 50°C to 1200°C. The IR-563 has a 1.0” cavity and a temperature range of 50°C to 1050°C. The final blackbody is the IR-574 with a 2.25” cavity and a temperature range of 50°C to 1200°C. The

IR-518, IR-519, IR-563, IR-564, and IR-574 use heater wire that are AC powered.

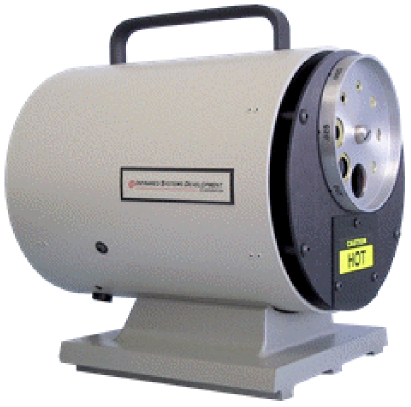


Figure 2: IR-563 Cavity Blackbody

The following table highlights the different blackbody sources that the temperature controller would have to be able to control.

Type	**Model Num.**	**Cavity Size/Heater Area**	**Minimum Temperature**	**Maximum Temperature**	**Thermocouple Type**	**Power Source**
Extended Area	IR-140	12” x 12”	Ambient (25°C)	230°C	Type T/RTD	AC
Extended Area	IR-150	12” x 12”	Ambient (25°C)	500°C	Type T/RTD	AC
Extended Area	IR-160	12” x 12”	Ambient (25°C)	350°C	Type T/RTD	AC
Extended Area	IR-2100	2” x 2”	-5°C	145°C	Type T/RTD	DC
Extended Area	IR-2101	2” x 2”	-30°C	75°C	Type T/RTD	DC
Extended Area	IR-2103	3” x 3”	-5°C	145°C	Type T/RTD	DC
Extended Area	IR-2106	6” x 6”	5°C	150°C	Type T/RTD	DC
Cavity	IR-508	.25”	50°C	1050°C	Type S	AC
Cavity	IR-518	.4”	50°C	1050°C	Type S	AC
Cavity	IR-519	.4”	50°C	1200°C	Type S	AC
Cavity	IR-704	.25”	50°C	1050°C	Type S	AC
Cavity	IR-564	1.0”	50°C	1200°C	Type S	AC
Cavity	IR-563	1.0”	50°C	1050°C	Type S	AC
Cavity	IR-574	2.25”	50°C	1200°C	Type S	AC

Table 1: Table Comparing Different Blackbody Models/Types

Temperature Controller Requirements

The temperature controller that is to replace the commercial off the shelf product currently being used by Infrared Systems Development Corporation will have to satisfy a wide range of requirements. These requirements are broken down into different subsections: General.

General - The temperature controller must be able to control an electronically heated or cooled blackbody source with lower than +/- 0.1°C of error. The controller must be able to be accessed through an intuitive front panel controls as well as remotely through a computer interface. The temperature controller will need to regulate the amount of power needed to heat the blackbody source while monitoring for any disturbances that will alter the actual temperature of the blackbody source that occur during the rise time and stabilization time of the blackbody. A P.I.D type control algorithm should be used to monitor the rise and control the device’s temperature. The controller must be able to be housed within an existing chassis that will also house the power supplies and high current devices. The controller must be able to run through a power on self-test to ensure that all sensors and devices are operating normally. If any abnormal conditions are present, the control should display the failure type and take appropriate action based on the error.

Alarm Outputs - Relay outputs that are set by the alarm control setup parameters must be included to allow users to connect other peripherals that will react to fluctuations between the temperature set point and the actual temperature of the blackbody.

Computer Interface - The temperature controller must contain a serial port, IEEE-488, USB, and Ethernet interface. These interfaces need to include a simple command set that will allow the user to read the actual temperature, modify the temperature set point, report alarm conditions, as well as other maintenance tasks.

Display - One key parameter that must be shown at all times is the actual blackbody temperature. This value must be readily visible from at least ten feet away from the controller. The temperature must be show two digits of precision at all temperature values. A variable time constant needs to be implemented to smooth the value and reduce the display flicker. Beyond the actual temperature, the controller must be able to display the current set point, alarm conditions, percent power, chopper speed, and other system parameters.

Temperature Sensors - Analog temperature sensors, like thermocouples and resistance temperature detectors, will be used to determine the actual blackbody temperature. The temperature controller needs to contain the appropriate circuitry to convert the voltage and/or the resistance readings from the temperature sensors into digital values that can be used in the control algorithm.

Our temperature controller must be able to interface with Type “S” thermocouples and RTD devices.

User Interface - In order to provide a clean, intuitive interface, the temperature controller needs to use a minimal number of dials and keys. The user should be able to scroll/select different device options and parameters. Operating the controller needs to be intuitive and require little instruction for proper use.

Microcontroller Research

Overview - The microcontroller will function as the brains behind the High Precision Temperature Controller. It will be responsible for interpreting the temperature of the blackbody source and take appropriate action to maintain an accurate temperature output. In addition, the microcontroller must handle user driven interrupts, display updates, and other tasks.
The following subsections will evaluate the various subcomponents found within the microcontroller. If applicable, a comparison between integrated components and external components will be included. When interfacing with external devices like a displays and buttons, the corresponding subsections will discuss how the microcontroller will interface with the peripheral. It will not include the research and design process of the devices.

Requirements - The microcontroller will need to satisfy a variety of performance and longevity criteria. Regarding performance, the microcontroller must be able to implement a PID algorithm. It needs to be able to quickly and accurately calculate and adjust the three main parameters. In addition, the microcontroller will need to ensure zero crossing when handling AC power. Zero crossing will ensure that no electrical interference occurs when power is being switched. The microcontroller needs to be able to handle a variety of computer communication interfaces, including USB, RS232, and IEEE-438, and user interactions, including displays and buttons. Also, it must be able to understand and command other peripheral devices like a chopper controller. The microcontroller must be able to perform complex In terms of longevity, the product lifecycle for the microcontroller and all its associated external components must be at least ten years following the completion of the design in December 2012. All the components must have a mean failure rate of at least ten years.

Package Type

There are three main package types that are available for all the microcontroller candidates that we are considering: SPDIP, SOIC, and QFN. All packages should be limited to 44 pins.

Skinny Dual In-line Package (SPDIP) - SPDIP is a package that have through hole leads that can be soldered to the board or inserted using a socket. The socket approach will allow for easy replacement if issues arise with the microcontroller. One of the benefits of SPDIP is the ease of prototyping because of the ease of mounting. However, SPDIP has a significantly larger footprint and is not significantly easier to mass produce than other electronic packages. Thus, we will not consider SPDIP for our microcontroller.

Small-Outline Integrated Circuit Package (SOICP) - SOICP is a surface mounted package. The footprint of these is around thirty to fifty percent smaller compared the DIP. Also, they have larger contacts than the QFNP which makes it easier to correct any errors during PCB production. However, this package is larger than QFNP and since we need to minimize the board space when possible, we will not consider SOIC for our microcontroller.

Quad-Flat No Lead Package (QFNP) - QFN is another surface mounted package. The footprint is even smaller than SOICP which is extremely advantageous for us. One of the disadvantages of having a small footprint is the difficulty of correcting any errors during PCB production. Because of the lack of clearance from the contact pads and the board, it is extremely difficult to correct any errors that exist on the board. Thus, we will not use the QFN package.

Plastic Lead Chip Carrier Package (PLCCP) - PLCC is a surface mounted package with “J”-leads. This allows the chip to be soldered directly to the board or the use of PLCC sockets. While the footprint of PLCC is larger than QFN, the ability to quickly and easily change malfunctioning microcontrollers is extremely beneficial. In addition, we can place the microcontroller in another circuit to program before placing it in the final PCB. This will allow us to save the I/O lines that we would have to use for in-circuit programming. Thus, we will prefer the PLCC package over any other for our microcontroller.

Memory

SRAM - There are three main functions in the temperature controller that are RAM intensive. First, storing the proportional band, integral band, and derivative band will take place in the RAM. These parameters will need to be updated constantly and do not need be saved if and when power is removed from the controller. Next is the implementation of data smoothing with respect to the display temperature. The particular algorithm will be discussed in the software design portion, but the RAM will temporarily hold the values for the display temperature until the averaging takes place. The final main task is the calculating of the electromagnetic energy radiated by the blackbody source using Planck’s Law. The energy curves defined by Planck’s Law do not linearly vary with temperature. This means that we will have to recalculate the curves when the temperature varies and calculate the area given a wavelength range specified by the user. There are two main approaches that are discussed in the software design portion. Since it would not be feasible for us to store a table of Planck’s Law values, we use RAM to help calculate the value according to Planck’s Law.

This will allow us to have higher precision and free up EEROM space for more important parts. We will try to limit the number of reads/writes that are made to the EEPROM by utilizing the SRAM to hold common values that the user will typically manipulate. If we do not have adequate amount of SRAM on the microcontroller, we will use the Microchip 23A640 with 64 Kbyte of SRAM. 23A640 will use SPI to communicate from the microcontroller and the chip. To read values from the 23A640, we will pull the chip select line low, and transmit the READ instruction (x03) followed by the 16-bit address sent to the SDI line which will be followed with the data at that address, if we reading one byte. If we want to read multiple bytes, this can be done without having multiple read cycles by clearing the data from the SDO line and waiting for another clock cycle. The address is automatically incremented and the byte from the following address is returned. For writing, we are capable of writing single bytes and multiple bytes sequentially. We start by sending the WRITE instruction (x02) followed by the 16-bit address then the eight bits of data. If we want to write multiple bytes, we will follow the first eight bits of data with subsequent eight bit packets. There is no limit to the number of eight bit packets that we can write.

EEPROM - The temperature controller has several functions that heavily utilize the EEPROM. These features are expected to present after power is terminated from the temperature controller. The first major item that will be stored on the EEPROM is the user interface. This includes all of the menus found within the temperature controller and the rules and conventions that will translate the text into readable graphics. Since the user interface will allow the user to correctly operate the temperature controller, extensive space will be allocated to this feature. The next part captures the various parameters of the temperature control subsystem. This includes the offset points and values, PID algorithm, digital filtering parameters, and temperature lookup tables. We will also need to store the protocol required to communicate with any peripheral device developed by Infrared Systems Development. It will dictate how our master temperature controller will command the slave peripheral device and how to interpret the responses that the slave sends back to the controller. Another feature to be stored in the EEPROM is the custom scripts that the user will uploaded into the temperature controller. The scripts will contain commands that will set a certain set point at given time for a given duration. More specifics will be discussed in the software design portion. Since these features will require a significant amount of EEPROM, we will have to choose a microcontroller that has plenty of onboard EEPROM. We will also need to look into interfacing with external memory if the microcontroller does not have any onboard for program and/or data. If we need to implement an external EEPROM, we will use the Microchip 25AA640A. This features 64 Kbit of memory (8192 x 8 bit). In terms of reading/writing, the same convention for the 25AA640A as it was for the 23A640. Since we are using a 16-bit microcontroller and the 25AA640A returns a single byte, we will need to read from the memory twice. Since EEPROM has a limited number of read/write cycles before failure; we will need to optimize the code to avoid excessive reading/writing. This is discussed in the software design portion.

Communication

Ethernet - If our microcontroller does not provide native Ethernet support, we will use the Microchip ENC28J60 Ethernet controller with SPI interface. This controller is IEEE 802.3™ compatible with support for 10/100/1000Base-T Networks. To interface this controller with our microcontroller, we will use four-wire SPI to send and receive data and an interrupt pin to notify us that there a particular packet has been received. The ENC28J60 requires a 25 MHz parallel cut crystal with 15 pF capacitors tied to ground. We will also need two Ethernet transformers that meet IEEE 802.3 isolation requirements. We will need to ensure that all power supply pins are connected to the same power source and all share a common ground.

IEEE-488 - IEEE-488 is an 8-bit parallel communication bus often referred to as GPIB. To implement this interface in our controller, we will utilize ICS Electronics 4806 GPIB to Serial Interface board. This adds a GPIB interface to our temperature controller. It includes SCI commands necessary to customize the 4806, including defining the GPIB address. The board also features a serial-to-serial data path, so an additional DB9 connector does not have to be present to interface with this device. The serial output will be RS-232 which we'll connect to a MAX232, if we do not have native support, so our microcontroller can obtain the information.

RS-232 - For two way transmission, RS-232 Interface requires a minimum of three lines, Receive, Transmit, and Ground Pins. The RTS/CTS lines are not necessary because software flow control will be used instead of hardware flow control. Software flow control, using standard XOFF and XON, reduces the complexity of the microcontroller design in favor of an easy software implementation. When connecting the RS-232 interface to the microcontroller, we will connect each RX and TX lines to two 2:1 multiplexors. This will save two pins when implementing the three different communication protocols. The outputs of the multiplexors will be tied to the RX/TX lines found on the microcontroller. If the controller needs to support IEEE-488, then the RS-232's RX and TX lines will be tied to the ICS GPIB to Serial board. The board has an output RX/TX line that will then be routed to the 2:1 Multiplexors, if we only have a single UART line in our microcontroller. If our microcontroller does not natively support RS-232, we will need a device that can modify the voltages so that it can be interpreted by our microcontroller since RS232 has voltage ranges of +/- 3V to +/- 15V, we will need a device that can modify the voltages so they can be interpreted by our microcontroller, which operates in the 0 – 5V range. Using a Maxim MAX232, it will convert Logic 1 from [-15V, -3V] to 5V and Logic 0 from [3V, 15V] to 0V. So we will tie the RX and TX from the DB9 connector to the MAX232 which will connect with the 2:1 Multiplexors mentioned above.

USB - If our board does not provide native support for USB, we will utilize the Microchip MCP2200. By using this chip, we will avoid the difficulties associated with implementing a standard USB class like HID or Mass storage on our microcontroller or creating our own USB driver on the particular computer. MCP2200 uses standard Windows drivers for Virtual Com Port and translates USB protocol to UART protocol. An associated .inf file will need to be loaded for the computer to recognize the microcontroller. This file is provided by Microchip. The MCP2200 has both hardware control lines has active low. Since we will have software handshaking, we will tie both of these lines to ground, telling the MCP2200 that transmission can freely occur. Since the 16-bit microcontrollers we are considering do not have a fast enough internal clock, we will be using a 15 MHz external clock for the timing of the MCP2200. To power the device, the +5V from the USB connector will be tied to V_{DD} with capacitors to satisfy the inrush current limit and lower power suspend mode. Prior to use, the desired baud rate will be assigned to the MCP2200. After doing so, we will use RX/TX lines from the MPC2200 connect to our microcontroller to read/write value from/to it. Since we are not going to use hardware handshaking, we will need to ensure that values are read before they are overridden and lost.

Input

A/D Convertor - Most microcontrollers have onboard ADC that has 10 to 12 bits of resolution. For our temperature controller, we will need 16 bits of resolution. There are techniques, like oversampling, available that will allow us to achieve our required 16 bits of resolution which will be discussed in detail in the corresponding ADC section.. These techniques require extremely low noise environments, which we cannot guarantee. Thus, we will use a 16 bit ADC. More thorough research and evaluation of potential candidates can be found in the temperature sensing sub section.. To interface a 16 bit ADC, like the AD7715, with our microcontroller, we need an external clock and serial input/output lines. The inputs required for actual conversion are discussed in the ADC section**.**

User Inputs - User interactions with the temperature controller will be governed by a series of buttons or a touch interface. If physical buttons are used, each button will be mapped to a single line on the microcontroller using a pull-down resistor to ensure that a signal is near zero volts when the button is not being pushed. A classic four button interface will be used (^, V, <, >). All buttons will connect with a pull down resistor to ensure that logic LOW is seen when the button is not pressed. If a touch interface is used, a touch screen controller must be included. Specific discussion on the implementation is discussed in the Display section. To interface the touch screen controller to the microcontroller, we will use a four wire touch screen controller that supports SPI.

Power - The microcontroller operates around the 5V range. Circuitry needs to be designed that will allow a 110VAC/220VAC to power the microcontroller without any issue.

Output

Power - All AC powered black body sources must have zero cross sensing at the AC relay in order to lower the electrical interference generated when switching at a nonzero point. The research and design of the zero cross sensing circuit can be found in the power section**.** The circuit returns a value of Logic High. Since we need the relay to react quickly to the crossing point, it will be tied directly to an AND gate. These logic gates can output up to 7Vdc, which will **(not)** be enough to drive the solid state relays. If the voltage is not adequate, an amplifier will be used to raise the DC voltage to adequate levels.

Display - A more in-depth research and design of the display can be found **here**. To interface the display with the microcontroller, we can use 16 bit parallel data bus or SPI. Using the a parallel bus would allow us to quickly refresh the screen at the cost of more complex circuitry and consuming more I/O ports on the microcontroller. The three/four wire SPI will be slower than the parallel approach, but we would only require three to four pins, which would be used for other SPI peripherals. Since we do not need to quickly update the screen, we will use four wire SPI to interface the display with the microcontroller.

Difference between four and three wire SPI = four wire allows you select between registers and data.

Timing - The overall timing of the microcontroller can be done using an internal oscillator or an external crystal. The chief benefits of using an external crystal would be higher clock speeds and accuracy. Since the performance of our temperature controller will be bounded by the black body source, the addition of a faster, an external crystal with higher frequency will not help. When applicable, peripherals that require an external clock will use the internal clock on the microcontroller. The MCP2200 requires a 15 MHz clock, which exceeds the limits of the internal oscillator. Instead we will use an external crystal to interface with the clock.

Microcontroller Candidates

TI MSP430F449

The MSP430F449 is a 16-bit microcontroller with an 8 MHz clock frequency and 48 general input/output lines. It comes in a 100-pin LQFP. While this package is not the desired PLCC, it has exposed “gull-wing” leads that will allow us to make any necessary repairs to the solder connections. Since the MCP2200 requires a 20 MHz external clock, we will need to supply an external crystal since the 8 MHz clock from the MSP430 will not suffice. It has 60 Kbytes of onboard Flash memory and 2 Kbytes of onboard SRAM. This will suffice our program memory and data memory requirements as well as our SRAM requirements. Thus, we would not need to have any external EEPROM or SRAM. TI's supplies an IDE, Code Composer Studio that will hasten the time to complete the software portion of our temperature controller. In addition to assembly, we can use C and C++ to program the MSP430. This will allow us to quickly implement Planck's Law and other math intensive functions. While the MSP430 does not offer native USB support, it does have two USART lines that can be used to handle USB (via the MCP2200), RS232 (via the MAX232), and IEEE 488 (). This microcontroller has a 12-bit ADC that uses successive approximation. While the conversion methodology is what we want, we need 16 bits of resolution in order to get two degrees of precision from our readings. We do not want to use oversampling to achieve this resolution, so we will use an external ADC. Since we have two USART lines, we will use the remaining line as SPI to communicate with the ADC as well as other peripherals. We will need to communicate with at most four different peripherals, the analog to digital convertor, display, and two slave devices. Instead of daisy chaining the SDI/SDO lines and have a common CE between all peripherals, we will use four independent peripherals by having common SDI/SDO lines and using a 4:1 multiplexor to control the CE lines. One key feature present on the MSP430F449 is LCD driver capable of controlling up to 160 LCD segments. This will give us greater flexibility in choosing a display interface because the display does not need to have an integrated controller. This lowers the cost of the display and will allow us to use generic LCD displays. There are four lines available that can be used as PWM lines that are capable of driving relays to control the heater power. We will need at least one for the cavity blackbody heat sources and two for the extended area blackbodies.

Microchip PIC24HJ128GP204

The PIC24HJ128GP204 is a 16-bit microcontroller with 7.37 MHz internal clock and 35 I/O pins. It comes in a 44-pin TQFP and 44-pin QFN package. The only available Microchip microcontrollers that supported PLCC were the 8 bit varieties. The Thin Quad Flat Package has gullwing leads that will allow us to make any necessary repairs to the solder connections. Since the PIC24H does not natively support USB, we will have to use the MCP2200 to interface with it, which has a 20 MHz oscillator requirement, which exceeds our external clock. Thus, we will use a 20 MHz crystal. It has 128 Kbytes of program memory which will be able to hold all of the present and future development needs of the temperature controller. It also has 8 Kbytes of SRAM. This extremely important because we need to handle complexity of implementing Planck's Law. In addition, having a significant amount of SRAM will allow us to lower the number of read/write cycles on the EEPROM, prolonging the lifetime of our components. The PIC24HJ128GP204 does not have any data EEPROM onboard, so we will use an external 64 Kbyte EEPROM that communicates via SPI. Microchip offers an IDE, MPLAB, that will allow us to quickly and easily program the microcontroller. MPLAB has a built in C compiler that is compatible with any PIC24 microcontroller. This will allow us to utilize the same code with another PIC24H microcontroller if the PIC24HJ128GP204 goes out of production.

Microchip PIC24FJ256DA206

The PIC24FJ256DA206 is a 16-bit microcontroller with an 8 MHz internal oscillator and 29 general I/O pins. It is available in a 64-pin TQFP/QFN package. The Thin Quad Flat Package has gullwing leads that will allow us to make any necessary repairs to the solder connections. With the QFN, the footprint is extremely small, but it any repairs are difficult because the contacts are found underneath the package and are not exposed. This microcontroller has onboard USB 2.0 support. Thus, we will not have to use the MCP2200 to provide a USB interface to the end user. It has 256 Kbytes of program memory, which will be more than enough to encompass our current and future development needs. It has 24 Kbytes of SRAM which will allow us to dynamically calculate electromagnetic energy radiated by the blackbody source, instead of resorting to using lookup tables. We will also be able to lessen the load we put on the EEPROM by loading the values from the EEPROM and manipulating through the SRAM. It does not have any onboard data EEPROM so we will use 64 Kbyte EEPROM that communicates via SPI. Some special features offered by the PIC24FJ256DA206 are USB 2.0 support and a graphic controller capable of supporting 24-bit TFT displays with 640x480 resolution. Just like the PIC24HJ128GP204, we will use MPLAB to program the PIC24FJ256DA206, which allow us to program any PIC24F microcontroller in Microchip's

Freescale MC68HC16Z1CAG16

The Freescale MC68HCZ1 is a 16-bit microcontroller with 16 MHz internal clock and sixteen general input/output pins. The only package that it comes in is the 144 pin QFP. While this package is not the PLCC, it has exposed "gull-wing" leads that will allow us to make any necessary repairs to the solder connections. The internal clock will not be sufficient to drive the MCP2200, which requires a 20 MHz clock. Thus we will use an external crystal to provide the timing for the MCP2200. It can address up to 1 Mbyte of program memory and 1 Mbyte of data memory. The MC68HCZ1 does not have any onboard EEPROM, but has an SPI line that can interface the external memory. We will be using 64 Kbytes of EEPROM, with the amount of data and program memory to be allocated at a later date. It has 1 Kbyte of SRAM. This will adequate to implement Planck's Law. The Freescale CodeWarrior Development Suite is not compatible with the MC68HC16Z1CAG16, so all programming must be done at the assembly level. The M68MMDS1632 Development System will allow us to quickly test and evaluate our code with real-time bus analysis and debugging. Freescale does have a rich support community that has open-sourced several programs that can help us tackle some of the more challenging programming problems including implementing Planck's Law, which would involve a fifth order polynomial as well as integration, and display interfacing. The Freescale microcontroller has five 16-bit registers (three index registers and two accumulator registers) that will allow us to handle any math or computational tasks need. In terms of interfacing with computer peripherals, it does not have native USB support, but has two UART lines so we would not need to use a 2:1 multiplexor for serial communication between the USB, RS232, and IEEE-488. Since we will need to interface with other integrated circuits like EEPROM and other devices like a chopper controller, we will need to use Freescale's single SPI channel. Since we only have one channel, we will tie the SDO and SDI lines together and use a 4:1 MUX to control the CE pins on the SPI lines of the slave devices. It has a successive approximation analog-to-digital convertor that supports up to ten bits of resolution. While the MC68HC16Z1CAG16 has the correct type of ADC, we need sixteen bits of resolution in order for our temperature ranges to be between 30.0°C and 1400.0°C and we do not want to oversample to get sixteen bits of resolution. Thus, having a ten bit successive approximation ADC is a nonfactor for this microcontroller. Since this microcontroller does not have any built it graphics drivers, we will need to design the protocol that will drive the display. We are going to interface with displays that support SPI, so no additional hardware will be required to communicate between this microcontroller and the display. If a four-wire touch screen is not used, we will require a set of four direction buttons (^, V, <, >) to serve as the user controls for the microcontroller.

Comparison

The table below summarizes the features that are relevant to our temperature controller.

Candidate	**Package**	**Number of Pins**	**Number of I/O Pins**	**SRAM (Kbyte)**	**Program Memory (Kbyte)**	**Data Memory (Kbyte)**
MSP430F449	LQFP	100	48	2	60 (Shared)	60 (Shared)
PIC24HJ128GP204	TQFP	44	35	8	128	None
PIC24FJ256DA206	TQFP	64	29	24	256	None
MC68HC16Z1CAG16	QFP	144	16	1	None	None

Candidate	**UART Lines**	**SPI Lines**	**I^2C Lines**	**PWM Lines**
MSP430F449	2 (Shared)	2 (Shared)	0	1
PIC24HJ128GP204	2	2	1	4
PIC24FJ256DA206	4	3	2	9
MC68HC16Z1CAG16	2	1	0	3

Candidate	USB Support	RS-232 Support	Graphics Support	Ethernet Support	Cost @ Qty 1
MSP430F449	None	None	Yes 160 Segment LCD Driver	None	$10.67
PIC24HJ128GP204	None	None	None	None	$5.43
PIC24FJ256DA206	Yes	Yes	Yes 24-Bit 640x480 TFT	None	$7.03
MC68HC16Z1CAG16	None	None	None	None	$11.37

Table 2: Table Comparing Features of Various Microcontrollers

The above table summarizes the main features available from our different candidates. Our main point of comparison is the amount of onboard memory, both SRAM and EEPROM, and the peripheral support on the microcontroller. We remain flexible with the package sizes, but we do prefer PLCC, then TQFP, then QFN. We also want to minimize the number of pins on the microcontroller. In terms of memory, the PIC24FJ256DA206 has the most, with respect to program memory and SRAM. This allows us to be extremely flexible in our software design and will allow us to implement some complex functionality in our temperature controller. One down side of the PIC24FJ256DA206 is the lack of data memory, which will force us to have external EEPROM. It also has the highest number of SPI, I^2C, and UART lines. This allows us to avoid using multiplexers and will let us have dedicated communication lines for some of our peripherals. Since the PIC24FJ256DA206 offers more peripheral support in terms of USB support, RS-232 support, and display support, we would avoid using the MAX232, MCP2200, and a display controller to make our different peripherals work with our microcontroller. This reduces the complexity of our circuit, size of our board, and the price of a temperature controller. For all of these reasons, we will use PIC24FJ256DA206 as our microcontroller for our temperature controller.

Microcontroller Design

Interfacing USB with PIC24FJ256DA206 - Since the PIC24FJ256DA206 has native support for USB, we will not require the MCP2200 to convert USB to UART. The microcontroller will be functioning in device mode. The effective capacitance from the V_{USB} cannot exceed 10 µF, so we chose 2.2 µF. The majority of setup will be handled with software and will be discussed in a later section.

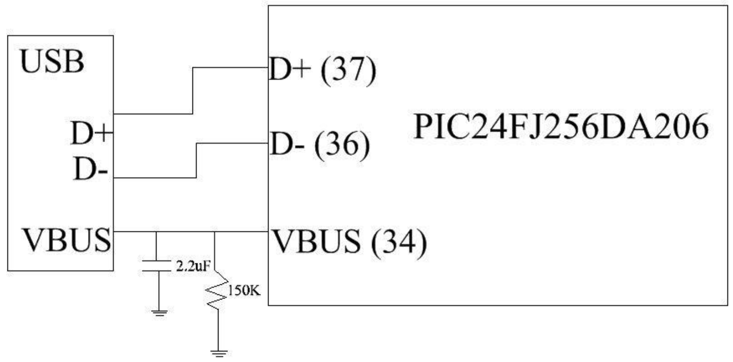


Figure 3: Interfacing USB with PIC24FJ256DA206

Interfacing RS-232 with PIC24FJ256DA206 - Since RS-232 uses -3V to -15V for Mark (1) and 3V to 15V for Space (0), we will have to use another IC to transform these signals to useable values. We will use MAXIM's MAX232 to do this. We use two of the general I/O pins on the PIC24FJ256DA206 to serve as the RX and TX lines. Additional circuitries on the MAX232 that include some external capacitors are not shown to simplify the schematic. Configuring the I/O pins will be discussed in a later section.

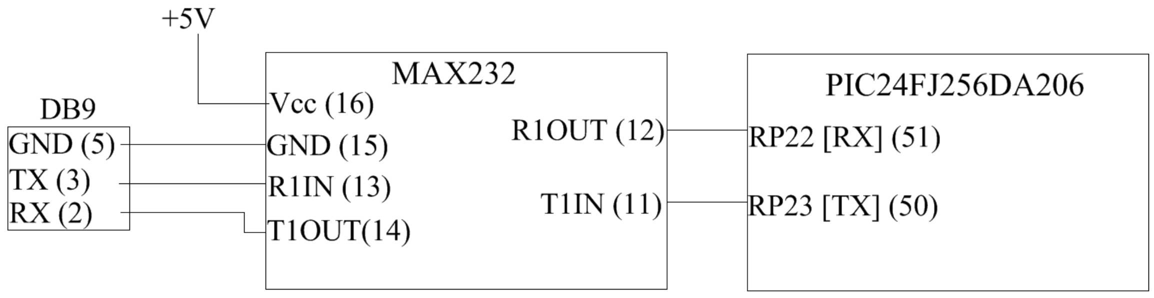


Figure 4: Interfacing RS-232 with PIC24FJ256DA206

Interfacing IEEE-488 with PIC24FJ256DA206 - Due to the complexity, we will be using ICS 4806 to convert IEE-488 to RS-232. Since the board also supports RS-232, we will use both of these connections in our temperature controller. The board will handle switching from the IEEE-488 and RS-232. The output of the board is RS-232 which needs to be formatted using the MAX232.

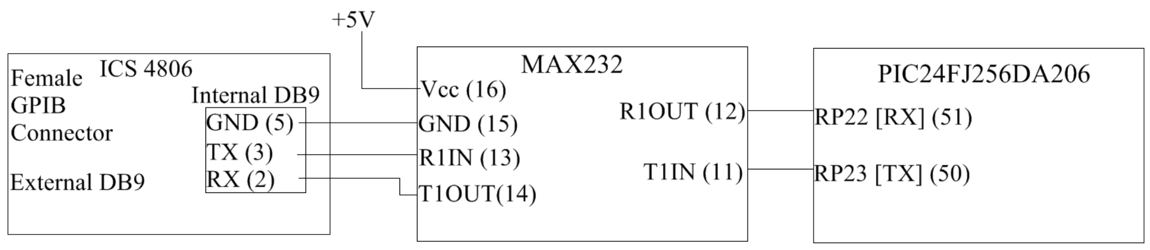


Figure 5: Interfacing IEEE-488 with PIC24FJ256DA206

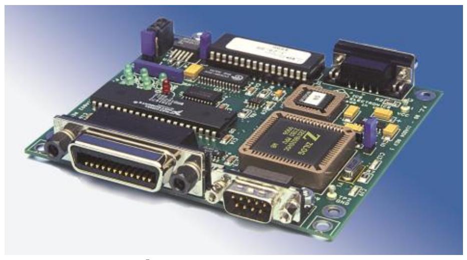


Figure 6: ISC 4806

Interfacing Ethernet with PIC24FJ256DA206 - Since the PIC24FJ256DA206 does not natively support Ethernet, will use Microchip's ENC28J60 to allow us to use SPI to send and receive data through the ENC28J60. Since we only are only using one of the three available SPI on our microcontroller, we will use independent slave devices with a shared slave select line. So all of our devices will have common SO, SI, and SCK lines and we'll use a 1:8 multiplexor to ensure that we only enable a single slave device at a time. The red text means that it is an inverted value, like CS is CS NOT. Additional circuitry, like the 25 MHz crystal, is not shown to simplify the circuitry. The Ethernet transformer is two 1:1 current transformers.

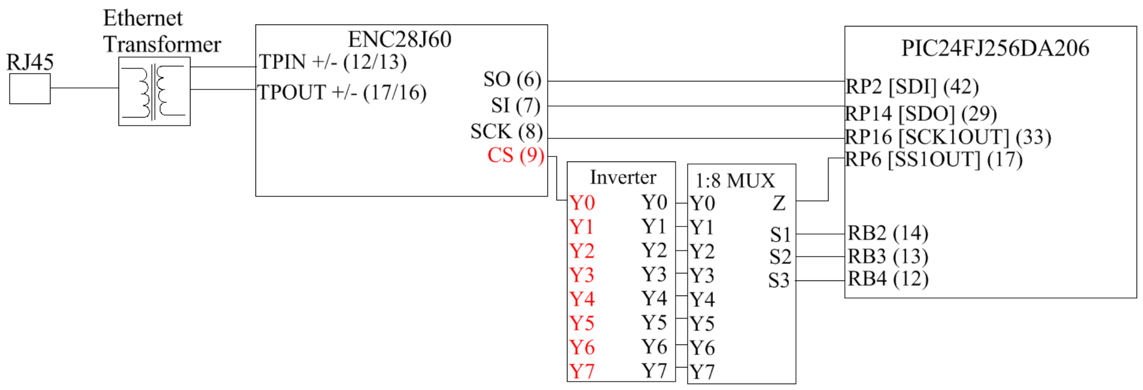


Figure 7: Interfacing Ethernet with PIC24FJ256DA206

Interfacing EEPROM with PIC24FJ256DA206 - Since the PIC24FJ256DA206 does not have any onboard data memory, we will have to use an external source. We will use Microchip's 25AA640A 64 Kbyte EEPROM. The 25AA640A uses SPI to read and write data to the EEPROM. It has an average of one million write/erase cycles before failure. The circuitry above does not show some features like HOLD and WP because the scarcity of available pins. If we determine a need for these pins, we will set them up at a later date. Since we're using +5V for V_{CC} we are able to have a maximum clock of 5 MHz.

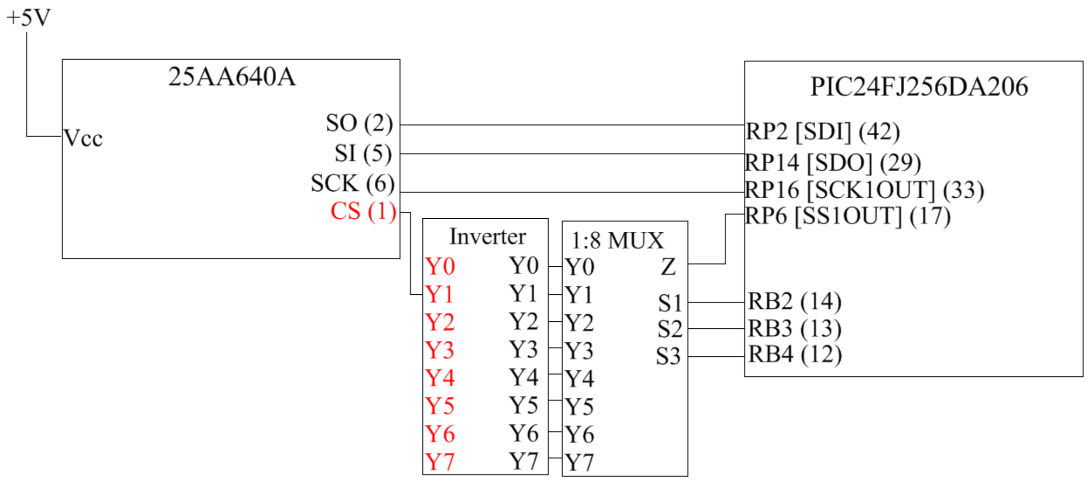


Figure 8: Interfacing 25AA640A with PIC24FJ256DA

Interfacing Four-Wire Resistive Touch Screen Controller with the PIC24FJ256DA206 - Since the PIC24FJ256DA206 does not have an onboard four wire resistive touch controller, we will have to use an external controller. The particular controller will be discussed in the display section, but the controller will communicate using SPI.

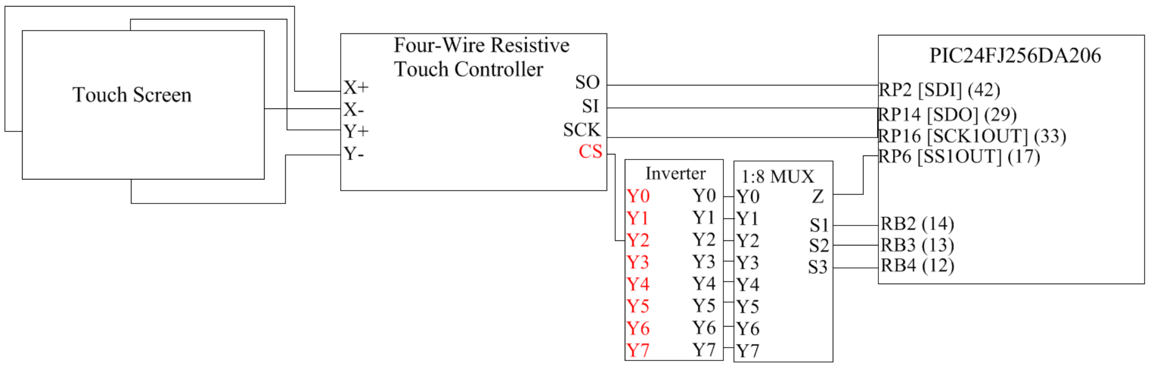


Figure 9: Interfacing Four-Wire Resistive Touch Screen Controller with the PIC24FJ256DA206

Interfacing Analog to Digital Convertors with the PIC24FJ256DA206 - Since the PIC24FJ256DA206 only has a 10-bit ADC and we do not want to use oversampling to achieve higher resolution, we will use an external 16-bit ADC. The particular part will be discussed in another section, but the device will support SPI.

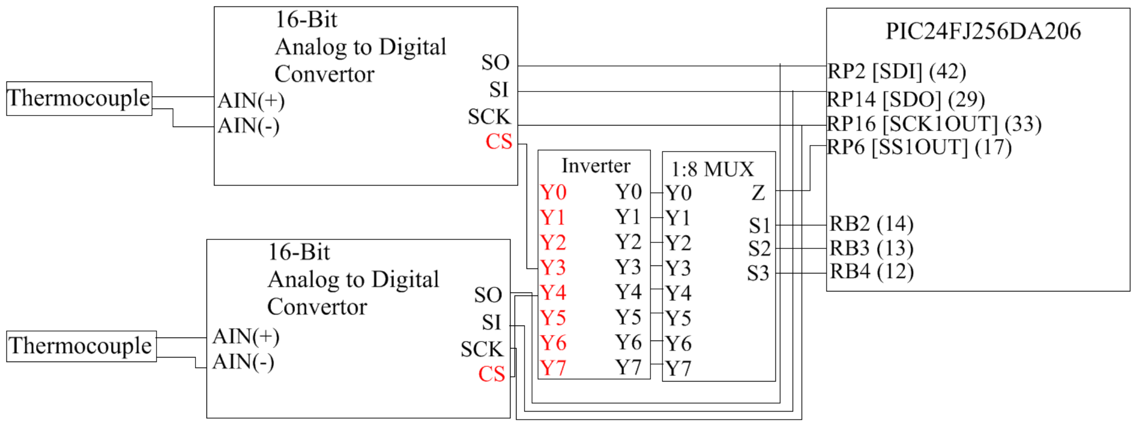


Figure 10: Interfacing Analog to Digital Convertors with the PIC24FJ256DA206

Interfacing a Slave Device with the PIC24FJ256DA206 - Another requirement of our temperature controller is the ability to control another peripheral device like a chopper controller. By using SPI to communicate, it allows for easy communication between the two devices. Additional circuitry will be added when a more definite diagram of the device is determined.

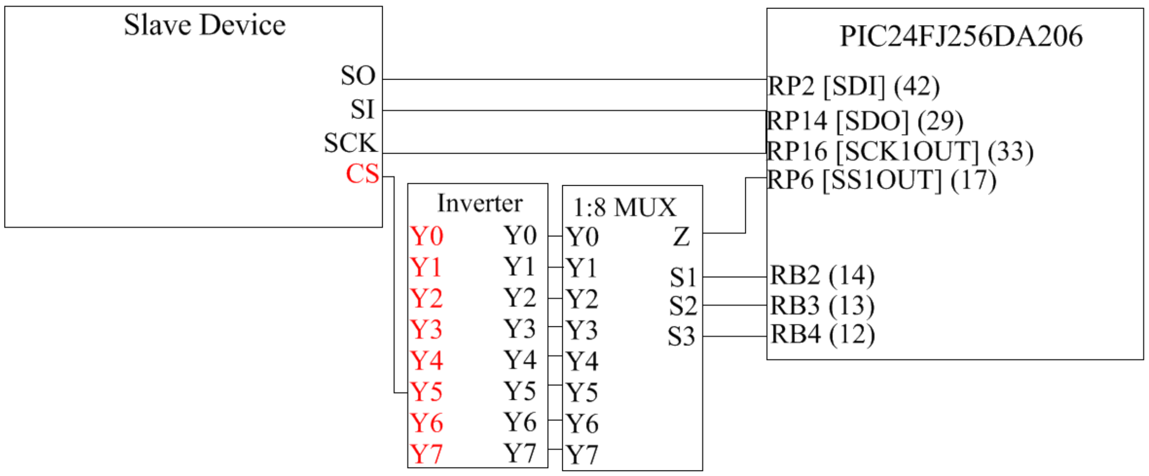


Figure 11: Interfacing a Slave Device with the PIC24FJ256DA206

Interfacing a Solid State Relay with the PIC24FJ256DA206 - To power the blackbody source, we will be using AC or DC power. To regulate this power, we will be using a solid state relay. By attaching the control lines of the relay to a pulse width modulate output, we will be able to precisely control the amount of power sent to the heater. We will require two solid state relays because one of the blackbody sources is thermoelectrically cooled and heated. This means that we will need to positive voltage will provide heat, while a negative voltage will cool the heater. The PWM outputs are OC1 and OC2. The other pins that are used are the error pins associated with the PWM lines. It is required to implement these pins in order to provide correct functionality.

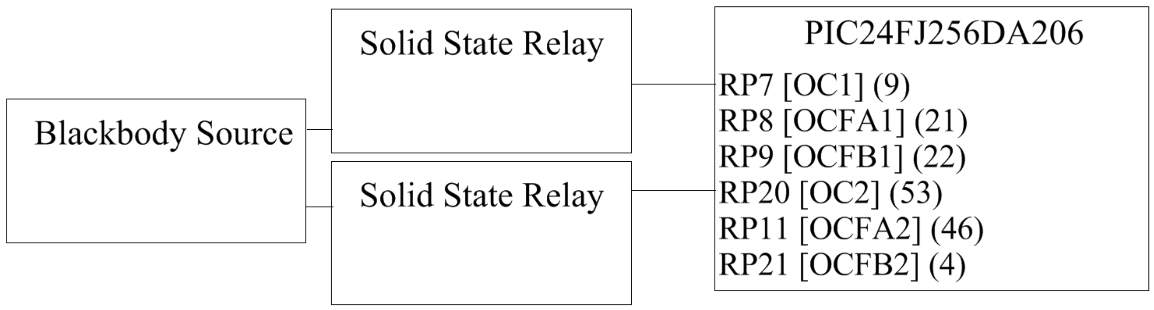


Figure 12: Interfacing a Solid State Relay with the PIC24FJ256DA206

Interfacing Alarm Outputs with the PIC24FJ256DA206 - Another requirement of our temperature controller is the presence of two separate alarms. These alarms will be triggered based off of user conditions, which include deviation of temperature and set point temperature. Specific software functionality of the alarms will be discussed in the software section. In terms of hardware, Alarm 1 and Alarm 2 possess a switch and indication LED. When an alarm goes high, an external switch will close and the LED will illuminate. The switch will allow users to set up their own circuitry and is capable of providing 3.3V.

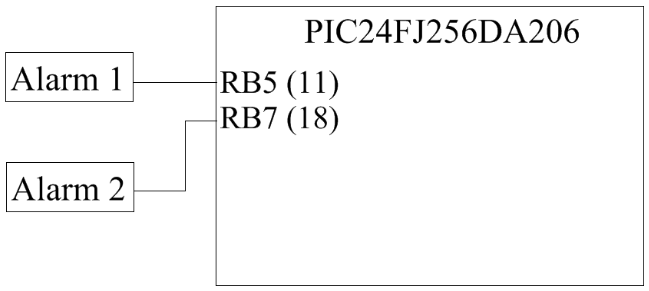


Figure 13: Interfacing Alarm Outputs with the PIC24FJ256DA206

Interfacing 24-Bit Display with the PIC24FJ256DA206 - The PIC24FJ256DA206 has the capability of driving 16-bit, 18-bit and 24-bit displays, but it only has 16 pins (RGB 5:6:5). Since we are going to be using a 24-bit TFT display, we will need some modifications to allow our 16-bit RGB 5:6:5 display controller to drive a 24-bit RGB 8:8:8 display. For the red (GD11-GD15) and blue bits (GD0-GD4), the upper three MSBs will serve as the upper three MSBs and the lower three MSBs for their 8-bit equivalent. For the green bits (GD5-GD10), we will use the upper two MSBs to serve as the upper two MSBs and the lower two MSBs for its 8-bit equivalent. The table below summarizes these rules.

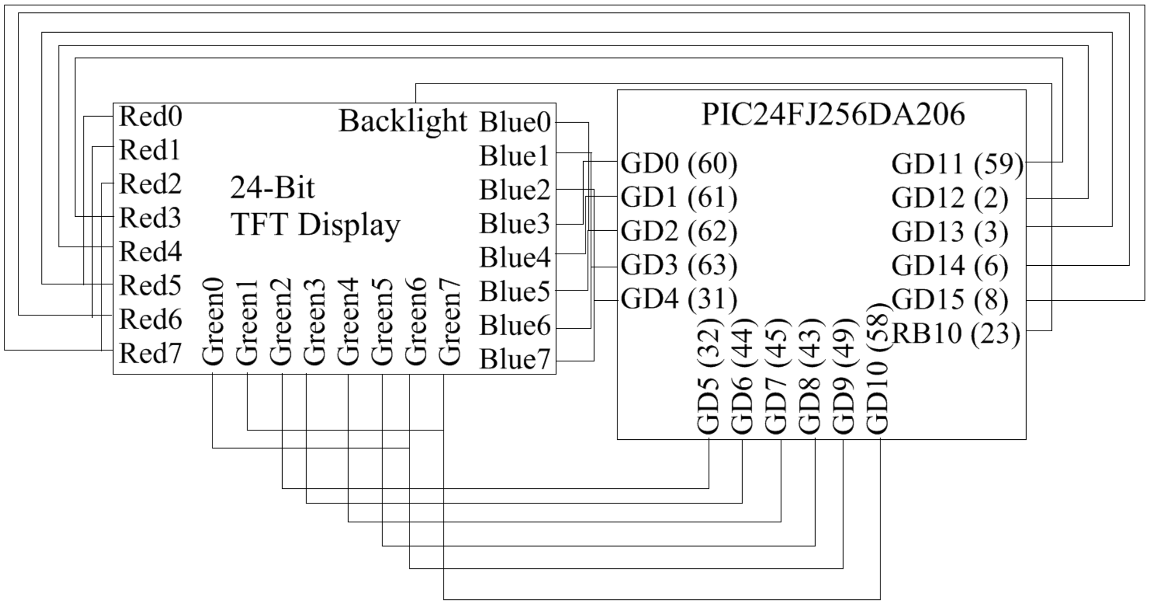


Figure 14: Interfacing 24-Bit Display with the PIC24FJ256DA206

5:6:5 RGB							8:8:8 RGB							
MSB					**LSB**		**MSB**							**LSB**
R4	R3	R2	R1	R0			R4	R3	R2	R1	R0	R4	R3	R2
G5	G4	G3	G2	G1	G0		G5	G4	G3	G2	G1	G0	G5	G4
B4	B3	B2	B1	B0			B4	B3	B2	B1	B0	B4	B3	B2

Table 3: Using 5:6:5 RGB with 8:8:8 RGB Interface

Microcontroller Pin-Out Table

Below are the pins on the PIC24FJ256DA206 that we are going to use for our temperature controller. It shows the pin number, pin description as described by Microchip, and our description of the pin. It describes the functionality that we have enabled in the microcontroller.

Pin #	Microchip Label	Description
2	GD12 (Red 1)	Used as Red 4 for our 24-bit Display
3	GD13 (Red 2)	Used as Red 0 and Red 5 for our 24-bit Display
4	RPI21 (OCFB2)	PWM2 Fault 1
6	GD14 (Red 3)	Used as Red 1 and Red 6
7	MCLR	Master Clear – Resets the Device. Low = Reset
8	GD14 (Red 4)	Used as Red 2 and Red 7
9, 25, 41	V_{SS}	Ground Reference for Logic I/O
10,26,38	V_{DD}	Positive supply for Logic I/O
11	RB5	Alarm Output 1
12	RB4	Mux Select 3 (S3)
13	RB3	Mux Select 2 (S2)
14	RB2	Mux Select 1 (S1)

15	PGEC1	In-Circuit Debugger Clock (Needed to Program)
16	PGED1	In-Circuit Debugger Data (Needed to Program)
17	RPI6 (SSOUT [SPI])	Used as the Slave Select line output for SPI
18	RB7	Alarm Output 2
21	RPI8 (OCFA1)	PWM1 Fault 0
22	RPI9 (OCFB2)	PWM1 Fault 1
23	TMS	JTAG Test Mode Select Input
24	TDO	JTAG Test Data Output
27	TCK	JTAG Test Clock Input
28	TDI	JTAG Test Data Input
29	RPI14 (SDO [SPI])	Serial Data Out for SPI (Master -> Slave)
31	GD4 (Blue 4)	Used as Blue 2 and Blue 7
32	GD5 (Green 0)	Used as Green 2
33	RPI16 (SCLK [SPI])	Clock used in SPI
34	V_{BUS}	USB Voltage needed for Device Mode
35	V_{USB}	USB Voltage needed for Device Mode
36	D-	USB Internal Transceiver
37	D+	USB Internal Receiver
39	RC12	Backlight for Display
42	RPI2 (SDI [SPI])	Serial Data Input (Slave -> Master)
43	GD8 (Green 3)	Used as Green 5
44	GD6 (Green 1)	Used as Green 3
45	GD7 (Green 2)	Used as Green 4
46	RPI11 (OCFA2)	PWM2 Fault 0
49	GD9 (Green 4)	Used as Green 0 and Green 6
50	RPI23 (U1TX)	Used as UART Transmit Line
51	RPI22 (U1RX)	Used as UART Receive Line
52	RPI25 (OC1)	Used as the output of PWM Line 1 (into SS Relay)
53	RPI20 (OC2)	Used as the output of PWM Line 2 (into SS Relay)
58	GD10 (Green 5)	Used as Green 1 and Green 7
59	GD11 (Red 0)	Used as Red 3
60	GD0 (Blue 0)	Used as Blue 3
61	GD1 (Blue 1)	Used as Blue 4
62	GD2 (Blue 2)	Used as Blue 0 and Blue 5
63	GD3 (Blue 3)	Used as Blue 6 and Blue 1

Table 4: Pinout Table for PIC24FJ256DA206

Future Microcontroller Development

Based off of the pin-out table shown in the previous section, there are a several versatile pins still available, which give us room for future development. In particular, there are four general digital I/O pins available and three remappable peripheral pins. The three remappable pins give us the ability to implement an additional UART or SPI interface (3-wire SPI). Unfortunately, we cannot implement I^{2}C because the clock and data lines are occupied by the graphics controller. If we have to implement I^{2}C, we could use the slave device and then send the information via SPI. Below is table showing the available pins and the function then contain. The four general I/O pins could be used for a 1:16 multiplexor for the SPI/UART line which could allow us to communicate with more peripherals like additional SRAM and EEPROM modules. In terms of more software features, we have the space in SRAM, data memory, and program memory to implement more critical features we deem important down the road. Below is a table of the available general purpose I/O and remappable peripheral I/O.

Pin #	Microchip Label	Description
5	RPI26	Remappable Peripheral (I/O)
30	RPI29	Remappable Peripheral (I/O)
40	RC15	Digital I/O
47	RC13	Digital I/O
48	RPI37	Remappable Peripheral (Input Only)
54	RD6	Digital I/O
55	RD7	Digital I/O

Table 5: Extra Pins on the PIC24FJ256DA206

Software Design

This portion discusses the methodology that will be used to allow us to implement the various features of the temperature controller.

Temperature Control Software

Temperature Processing

In order for our controller to do any data monitoring, processing or controlling we have to be able to translate our voltage readings into usable, meaningful data. The microcontroller will have access to digital voltages sent from all the thermocouples and reference sensors, whether this be type T or type S thermocouple or the cold junction RTD. The controller and other peripherals will need to know how to operating depending on these temperature readings. If a black body source is approaching its set point, the processor will need to send a signal to the DC relay, which will in turn trigger the one of the two alarms. This process is just one example of many processes that rely on the temperature data to trigger its response. If the data is not taken in and understood properly, we could risk damaging the heater and the controller itself.

There will be several functions programmed into the microcontroller specifically made to handle these voltages. Separate functions will need to be written for each type of temperature sensor because of their unique responses to temperature changes. We will also be dealing with double precision arithmetic; therefore, we need to allocate enough space in the PIC's memory to accommodate this coding. Also considering memory space, we will need to store the look up tables for type T and type S thermocouples and also the look up for the cold junction RTD. Most of these values are also double precision and the tables each have 10,000 plus entries. These tables will be stored in EEPROM and therefore also need a command from the controller to fetch this data from the external device.

For the purposes of this paper, we will discuss different pseudo-code functions that will be generic enough to accommodate both thermocouple types. We will then discuss another function to realize the RTD input. The following pseudo-code aims to take a user input from the touch panel display and associate this variable with a sensor type. From this declaration, the controller will know whether we are dealing with voltages from thermocouples or resistance changes from the RTD, and can go to the appropriate look up tables.

```
char sensor_type = getuser_input ()        //Fetches user input from
                                           //touch panel
                                           //display
if (sensor_type == T)                      //Type T thermocouple
Tvolt = get_T_reading ()
Ttemp = goto_Ttable ()
Sends temperature to PID loop

if (sensor_type == S)                      //Type S thermocouple
Svolt = get_S_reading ()
Stemp = goto_Stable ()
Sends temperature to PID loop

if (sensor_type == C)                      //Cold junction RTD
Cvolt = get_C_reading ()
CJvolt = translate_C2T (Cvolt)             //Need to translate RTD
                                           //temp to TC voltage
Sends voltage to be subtracted from Xvolt
Ctemp = goto_Ctable ()
Sends ambient temperature to PID loop
```

The 'get_X_reading' function will get the data from the output of the analog to digital converter and subtract any experimentally measured offset from this value (this will be hardcoded). It will also subtract the cold junction voltage for cold junction compensation. This function also takes into consideration a delay for the charge injection, caused by the multiplexers, to dissipate before taking the reading. All values in the following functions are double precision.

```
double get_X_reading ()

Sends command to decoder to enable via CS appropriate ADC
Delays with no operation to allow for charge injection dissipation

VIN = goto_pin()                              //Input from ADC
double reading_noffset
reading_noffset = VIN – OFFSET - CJvolt       //This offset will be
                                              //predetermined
                                              //through experimental
                                              //measurement
RETURN reading_noffset                        //Gives PID loop a
                                              //clean voltage
```

The ‘translate_C2T ()’ function will contain the majority of our arithmetic operations. The need for this function comes from the fact that the voltage input given from the RTD amplifier, is a function of the RTD resistance. This RTD resistance will need to be translated into a thermocouple voltage via both the RTD and thermocouple look up tables. This data needs to be a thermocouple voltage in order for it to be a useful reference to the measurements
.

```
double translate_CJT (Cvolt)
RTD_RES = 1000 * [(Cvolt / Vref) – 1]         //Solves for RTD
                                              //resistance from known
                                              //R1 and Vref values
Ctemp = goto_Ctable ()
CJvolt = goto_Xtable (Ctemp)                  //Gives us cold junction
                                              //voltage depending
                                              //on ‘X’ thermocouple
                                              //type
```

The ‘goto_Xtable’ function will implement a binary search algorithm on the appropriate look up table to find the corresponding temperature values. We’ve chosen a binary search algorithm because of its fast and reliable runtime. This function will also need to interpolate between voltage values, as there is almost a certain chance that the 5 decimal place, precise value will not correspond to a value on the table. The ‘goto_pin’ function grabs the data from the analog to digital converter and returns the digital voltage value.

Other Software

This section deals with the software not related to the temperature control portion of our temperature controller. This section deals
Locking Out Non-Administrators - Since there are several parameters that can significantly shorten the lifespan of the blackbody heat sources like the maximum

power limit and the maximum temperature limit, we will need to ensure that the typical user does not have the ability to modify certain parameters. These parameters will be hidden from the user and will only be accessible to the administrators at Infrared Systems Development. To hide these parameters, we will be using two separate menu layouts for regular users and administrators. To access the administrator page, we will prompt the user to enter an ASCII formatted password. The length can be four characters or larger. Once the correct password is entered, the administrator will be able to modify the sensitive parameters that are protected from the user. To implement this, we will compare the entered value from the user with the current password that is being stored in the PIC24FJ256DA20. If the password matches, the program will update the screen with the new menu options. Otherwise, the screen will not update and the program will return to the user only state.

Powering the Blackbody Source - In order to precisely power the blackbody heat source without burning on the heater core, we will use a solid state AC relay for the cavity blackbodies, whose maximum temperature is 1050°C and 1200°C and two solid state DC relays for the extended area blackbodies, which reach maximum temperatures of 300°C and minimum temperatures of -10°C. To control these relays, we will be using the PWM lines that are available on the PIC24FJ256DA206. Before utilizing these pins, we need to configure them for our purposes. To do this, we need to first configure Timer1 which we will use to serve as the clock for the PWM line. We will write 8000_{Hex} to the T1CON Register. This will enable Timer1 and configure Timer1 to use the internal oscillator of 8 MHz with a prescale of 1:1. After Timer1 is configured, we will write $1F_{Hex}$ to the OC1CON2.SYNCSEL bits and 0 to OC1CON2.TRIG. This will set the synchronization to OC1. We will write 4 to OC1CON1.OCTSEL to use the Timer1 clock. We will also write 0 to OCSIDL, ENFLT2, OCFLT2, OCFLT1, and OCFLT0 and 1 to ENFLT1, ENFLT0. We will then write the PWM period to PR1 and the duty cycle to OC1R register and OC1RS register. There are a variety of PWM period and duty cycles that we can use on the PIC24HJ128GP204, which are listed below.

PWM Frequency	7.6 Hz	61 Hz	122 Hz	977 Hz	3.9 kHz	31.3 kHz	125 kHz
Duty Cycle Resolution	16	16	15	12	10	7	5
Duty Cycle Step Size (us)	2.007 7	.250144	.250144	.249888	.250401	.249601	.25

Table 6: PWM Period and Duty Cycle

By minimizing the duty cycle step size, we are able to more accurately and precisely control the amount of power that we send to the heater. This will minimize the time required for the blackbody to reach stability. Using a PWM Frequency of 3.9 kHz (PWM period is 256.4 us) with seven bits of resolution, we are able to achieve a duty cycle step size of .249601 us. These statistics are relevant for the DC powered blackbodies, like the extended area blackbodies, since the AC powered blackbodies would be relegated to a PWM period of 33.33

milliseconds. Since most DC relays operate in the millisecond range, our performance will be bounded by the performance of the relay. Once the AC and DC relays are chosen, modifications will be made to the PR1, OC1R, and OC1RS registers to reflect the performance of the specific relay. The following equation can be used to calculate the value to write to PR1 register.

$PR1 = \frac{PWM\ Period}{T_{CY} * Prescale\ Value} - 1$, where T_{CY} is the period of the syncing clock, which for our case is Timer1.

In order to avoid burning out heater, we will need to limit the amount of power that we will send to the heater. To do this, we will need to change the lower the duty cycle or extended the PWM period. Once we determine the minimum response time of the AC relays, we will calculate the PR1 using the equation above and multiply this value by 1000. This will allow us to capture 1000 periods of the 60 Hz sine wave and classify it as a single period for our purposes. Then, we will set the duty cycle from 0 to 100%, with intervals of .1%. This will allow us to use 0 to 1000 periods, with increments of 1 period. By setting the duty cycle to .1%, we are including just one period of the sine wave, limiting the power that is sent to the heater. The same approach can be applied for the DC relays and will allow us to control DC power at .1% intervals. The administrators at Infrared Systems Development will be able to set the maximum percent power which will then be translated to a duty cycle value that will be written to OC1R and OC1RS. The formula for calculating the value of OC1RS and OC1R is shown below, where the minimum response time refers to the minimum response time of the relay. .

$$OC1R = OC1RS = \left(1000 - 1000\left(\frac{100 - Desired\ Percentage}{100}\right)\right) Minimum\ Response\ Time$$

Smoothing the Display - Due to the characteristics of the blackbody heat source and thermocouples, the perceived temperature tends to fluctuate when viewing it one digit of precision. To reduce this flickering, our temperature controller must be able to smooth the data by extending the time constant. To make the concept more intuitive, we will implement the time constant by determine the number of desired samples before updating the screen. By increasing the number of samples we are extend the time constant which allows to smooth the display temperature. Our program will maintain three different values to implement this smoothing. The first parameter is the number of samples before updating. The second parameter is the current sum of the temperatures. The final parameter is the number of our current sample, which ranges from 1 to number of samples. Once the final parameter and first parameter equal, the program will take the average temperature and update the display.

Calculating Planck’s Law - Another feature that we need to implement in our temperature controller is the calculation of the blackbody radiations using Planck’s Law. Since the user will want to determine the radiation given a certain wavelength interval, we will be using Planck's law with respect to wavelength. The equation is listed below.

$$B_\lambda(T) = \frac{2hc^2}{\lambda^5}\frac{1}{e^{\frac{hc}{\lambda k_B T}} - 1}$$

To calculate the blackbody radiation, we will integrate the curve that is defined with the equation above. Since C does not have an integral library, we need to use alternative methods to accurately calculate the integral. Since the B(T) is parabolic, we can accurately approximate the integral using Simpson's 1/3 rule. Alternative methods like the trapezoidal rule would return higher errors. Since the accuracy and complexity of Simpson's 1/3 rule increases with more intervals, we will need to determine the point where the additional accuracy is not worth the additional memory requirements. Once the user defines the upper and lower wavelength bounds, our temperature controller will continue to refresh the blackbody radiation as the temperature changes.

Communicating with Peripherals

This section discusses how we are going to send data to and receive data from the PIC24FJ256DA206 to the various peripherals that we are communicating with.

Preventing Haphazard Writes from Peripheral to Microcontroller - To ensure that the users (non-administrators) do not haphazardly modify the contents of extremely sensitive registers, like the maximum power and maximum temperature, we need to have a byte(s) that are sent to the PIC24FJ256DA20 before the actual value is sent. This section deals with the computer communication peripherals that reside on our temperature controller. This byte(s) will tell the microcontroller that the sender is an administrator and will be allowed to modify the contents of the register. If this "unlock" byte(s) is not sent, then the PIC24FJ256DA206 will not modify the contents of the register. To determine if the register is user write protected, we will organize registers into groups of eight, numbered 1 through 8. Register 0, that precedes the group, will indicate if a particular register is user write protected, with 1 indicating that the register is user write protected and 0 indicating that the register is not user write protected. For example, If Register 0 is 80_{Hex}, then Register 8 is write protected and the Registers 1 through 7 are not write protected. If the register is not user write protected, we would not need to send the "unlock" byte(s) prior to sending the register and new value. The PIC24FJ256DA20 will simply perform the desired write action. Another issue that our microcontroller must handle is writing to a read only register. To tell the PIC24FJ256DA20 which registers are read only, we will use the same grouping that is being used to determine if the register is user write protected. The register preceding the grouping and the user write protected register (Register 0) will be used to determine if the read only. A 1 will indicate that the register is read only, and a 0 will indicate that the register can be read to and written to.

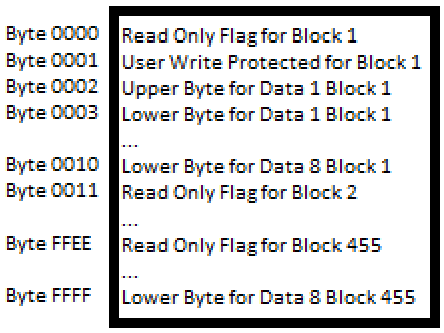
Communicating with 25AA640A - Microchip's 25AA640A is a 64 Kbyte (8192x8 bit) EEPROM. To read information from the 25AA640A, we will need to start by setting up the SPI1STAT register and SPI1CON1 register. To do this, we need to first write 0 to SPI1STAT.SPIEN in order to allow us to set SPI1CON1. After doing so, we will write $07F7_{Hex}$ to SPI1CON1 and 0000_{Hex} to SPI1CON2. After doing so, we will write 1 to SPISTAT.SPIEN and we are ready to read values from the 25AA640A. We will start by writing 03_{Hex} to the SPI1BUF register. Writing to this register will place the value onto the transmit buffer (SDO). We will then write the starting address that we want to read. After those values are clocked in, the 25AA640A will respond with the 8 bit value that exists at the address. We will then wait for an additional 8 clock cycles to get the second 8 bit word. To write information we will have to use the same setup procedure as reading. After the CON1, CON2, and STAT registers are set, we will then write 06_{Hex} to SPI1BUF, then set CE to high. This will enable the 25AA640A to be written to. Then, we will set CE to low then write 02_{Hex} to SPIBUF, then the 16-bit address, and the word that we want to write to the address. Since the 25AA640A is 8 bits wide, we will be writing to the address and the following address. As mentioned in the Microcontroller Design section, we will be forming data into groups of eight, with two bytes that indicate the read only status and user write protected status of the data. So we will need to use a unique addressing scheme that will ensure that we are able to determine the location of the correct read only status and user write protected status register that is associated with the correct group. We will use the following formulas to determine the block that a particular register belongs to and the location of the read only status register and the user write protect status register. Block numbers will go from 1 to 455.

$$Group\ Number = Floor\left(\frac{Address}{18}\right)$$
$$Read\ Only\ Status\ Address = (Block\ Number - 1) * 18$$
$$User\ Write\ Protect\ Status\ Address = Read\ Only\ Status\ Address + 1$$

When we want to write information, we will need to first check the Read Only Status to see if we are allowed to perform the write action. After that, we will look at the User Write Protect Status register and determine if the appropriate “unlock” byte(s) is required and if it has been sent. If both tests pass, then we are able to write to the particular memory address and we will proceed with writing the value to the address. If we are reading values from the 25AA640A, we will not have to look at the Read Only Status register or the User Write Protect Status register. To utilize the 25AA640A with our 16-bit microcontroller, we will have to shift the eight bytes into its correct position. To do this, we will multiply the first eight bytes by 1 and then the second byte by 255 to shift it into the upper byte.

Organizing the Data EEPROM - Since the PIC24FJ256DA206 does not have any onboard data EEPROM, we will be using Microchip's 25AA640A 64 Kbyte

EEPROM. The 25AA640A is an 8K by 8 bit EEPROM. It will capture important parameters that are critical to the success of the temperature controller, like the set point, offset points, and offset values. We will organize the data into groups of eight. Since we are using a 16-bit microcontroller, one unit of data with respect to the PIC24FJ256DA206 will extend over two memory cells. Each group will have two bytes that are reserved as flags that indicate user-write protect and read only memory. Thus, a single group of eight will occupy eighteen bytes of memory. This will give us the ability to have 455 groups within a single 25AA640A. The image below is a summary on the organization of our data EEPROM.



Byte 0000	Read Only Flag for Block 1
Byte 0001	User Write Protected for Block 1
Byte 0002	Upper Byte for Data 1 Block 1
Byte 0003	Lower Byte for Data 1 Block 1
	...
Byte 0010	Lower Byte for Data 8 Block 1
Byte 0011	Read Only Flag for Block 2
	...
Byte FFEE	Read Only Flag for Block 455
	...
Byte FFFF	Lower Byte for Data 8 Block 455

Figure 15: Data EEPROM Organization for 25AA640A

Communicating with USB - To communicate with the USB interface on the PIC24FJ256DA206, we will start by setting the device into Device Mode. We start by resetting the PPBRST bit, disabling all USB interrupts, clearing all interrupt flags, set the USBEN, OTGEN, EPRXEN, and EPHSHK, USBPWR to high. If we want to send data, we will create a data buffer and add the data that we want to send. We will the set up the status register with the desired toggle and the byte count of the data buffer and the starting address on the address buffer. Once the IN token is received, all of the data on the buffer is sent. A similar process is doing to receive data, but instead of actual bytes we are going to write the number of expected bytes.

To determine if USB on the PIC24FJ256DA206 is operating correct, we will have a predetermined string that is sent to the computer. Once the computer receives this string, it will know that it is a test call and will echo the string back to the microcontroller. If the communication times out or the return string does not match the string that was sent, then we will know that the USB module is malfunctioning. An appropriate error message will appear notifying the user that the USB is not working.

Communicating with Ethernet - To communicate with the ENC28J60 using SPI, we need to start by setting up some of the parameters of the interface. It only supports SPI mode 0,0 , SCK must be Idle in a low state, selectable clock polarity is not supported, data is sent to it on the SI line on the rising edge of SCK, data is sent on the falling edge of SCK and the CS must remain low during the entire operation. On the PIC24FJ256DA206, we will have the opposite; since SI on a slave device is SDO on the master and SO on a slave device is SDI on the master. So we want the data to be clocked out on the rising edge and data to be clocked in on the failing edge. To do this, we will set SPI1STAT.SPIEN to low and then write $06BF_{Hex}$ and 0000_{Hex} to SPI1CON1 and SPI1CON2, respectively. After setting up the SPICON1 and SPI1CON2, we will set SPI1STAT.SPIEN to high to enable the SPI line. Since the ENC28J60 stores information on the read buffer memory, we will use RBM Opcode to obtain this information. To do this we will write $3A_{Hex}$ to the SPI1BUF. The following bytes on SPIBUF will be the data residing on the buffer. To write information to the buffer memory, we will have to ensure that the SPI1CON1 and SPI1CON2 are set like we had to read the buffer memory. We will then write $7A_{Hex}$ to the SPI1BUF. After writing that, the next N bytes will be written to the buffer memory.

To test if the Ethernet port is operating correctly, we will write a certain string to the buffer memory. The ENC28J60 will transmit the data to the host computer. The computer will recognize that the string is a test string and will echo it back to the PIC24FJ256DA206. We will then check to see if the same string is received. If the process times out or the string is not the same, we will know that the Ethernet controller is malfunctioning. We will notify the user that the ENC28J60 is not working.

Communicating with RS-232/IEEE-488 - Since we will be converting the IEEE-488 to RS-232, our temperature controller needs to handle the RS-232 signals. As mentioned in the microcontroller design portion, we will be using the MAX232 to convert the RS-232 signals so we can handle them in the PIC24FJ256DA206. Since we are not going to use the CTS and RTS lines, we will need to ensure that we configure the UART lines to reflect that we are only going to be using three lines. To configure the UART to our desired settings, we will write 8890_{Hex} to the U1MODE register. This will enable the UART pins, turn off flow control, use the U1TX and U2RX I/O pins, enable the U1TX and U1RX pins, disable Loopback mode, sett Idle state to 0, set the data format to 8-bit, no parity, two stop bits, and have low speed baud rate. Since our transmission is going to be around 19200 baud, we do not need to have high speed transmission. If we want to enable high speed transmission, we would have to write 8898_{Hex} to the U1MODE register. To set the baud rate, we need write a certain value to the U1BRG register. There are different values depending on if we enable high speed or low speed transmission. We are using the PIC24FJ256DA206's internal oscillator of 8 MHz to calculate the baud rate.

	BRGH = 0 (Low Speed)		**BRGH = 1 (High Speed)**
Baud Rate	**U1BRG Value**		**U1BRG Value**
110	$4544_{Decimal}$		$18181_{Decimal}$
300	$1666_{Decimal}$		$6666_{Decimal}$
1200	$416_{Decimal}$		$1666_{Decimal}$
2400	$207_{Decimal}$		$832_{Decimal}$
9600	$51_{Decimal}$		$207_{Decimal}$

19.2K	$25_{Decimal}$		$103_{Decimal}$
38.4K	$12_{Decimal}$		$51_{Decimal}$
56K	$8_{Decimal}$		$35_{Decimal}$
115K	Not Possible		$16_{Decimal}$

Table 7: Setting up the UART Control Lines

As shown above, enabling the BRGH bit and permitting high speed mode will allow us to achieve a baud rate of 115K, which is not attainable through the low speed mode (setting BRGH = 0).

To test if the IEEE-488/RS-232 are operating correctly, we will send certain string to the computer that will notify the computer that we are requesting it to echo the same message back. After the PIC24FJ256DA206 sends out the string, we will look at the response from the computer. If we timeout or we do not receive the same string that was sent, then we will know that the IEE-488/RS-232 is not operating correctly. An appropriate message will be displayed if this error is found.

Communicating with the Display - To communicate with the NHD-4.3-480272MF-ATXI#-T-1, will be using a 24-bit parallel RGB (8:8:8) interface. As mentioned in the microcontroller section, will be using GB0-GB15 (RGB 5:6:5) with some modifications. We will also need to a clock input of at least 5 MHz and a hardware sync pulse that goes low then high for the duration of the data transmission. To do this, we will need to start by setting the SPI1CLK to transmit at 8 MHz. To do this we will need to start by setting SPI1STAT.SPIEN to 0. This will allow us to configure the SPI1CON1 and SPI1CON2. We will send 00001XXXXXX11111 to SPI1CON1, with Xs are used for Don't Cares. SPI1CON2 will be written with 0000_{Hex}. After doing so, we will set SPI1STAT.SPIEN to high. In SPICON1, we use SDO as a general I/O port. This will be plugged into the HSYNC pin on the New Haven Display. To begin writing the values in the R, G, and B bus, we need to have HSYNC go high then low. We will start by setting SDO pin to low, then setting the SDO pin to high. This will tell the driver to read in the data on the R, G, and B data buses. To control the backlight, we will be using the PWM1 line. Modifying the duty cycle will allow us to control the brightness of the backlight. We will allow the user to modify the duty cycle by writing the value to the OC1RS and OC1R registers. We will need to wait a clock cycle for the changes to take effect.

Communicating with the Touch Controller - To communicate with the Texas Instruments TSC2046, we will be using SPI. On the TSC2046, data is clocked in on the rising edge and data is clocked out at the trailing edge. So we will need to set the SPICON1.CKE = 1 and SPICON1.CKP = 0. After doing that, we are ready to transmit to the TSC2046. To start, we need to send a control byte. Bits 6-4 determine what value we want to read from the display. Assuming that we are in differential reference mode, SER/DFR NOT (Bit 2) = 0, bits 6-4 equals 001 for Y position, 011 for Z_1 position, 100 for Z_2 position, and 101 for X position. We need to set Bit 3 to 0 in order to allow 12-bit resolution. Bits 1-0 will be set to 11 which will allow the device, reference, and ADC to be powered on. After passing in the control byte, the TSC2046 will respond with the value that we are requesting. Since we will need the X position, Y position, Z_1 position, and Z_2 position to determine the actual location of the press, we will allocate four 16 bit registers to store the values until we are ready to calculate them. We will need to send four control bytes in order to get all of our desired information. The table below summarizes the four different commands that we will be sending to the TSC2046.

Control Bytes	**Value Returned**
10010011	Y position, with 12 bits of resolution
10110011	Z_1 position, with 12 bits of resolution
11000011	Z_2 position, with 12 bits of resolution
11010011	X position, with 12 bits of resolution

Table 8: Control Bytes to Interface with the TSC2046

Communicating with the Slave Device - Another requirement for our temperature controller is the ability to communicate with an additional peripheral device that is developed by Infrared Systems Development. Specifically, we will need to control and display information from a chopper controller. This includes the current chopper speed and the ability to modify the chopper speed from our temperature controller. The specific command structure to communicate with the peripherals will be determined at a later date.

Determining Abnormal Operation

Our temperature controller must be able to determine if there are any malfunctioning sensors and devices when we power on the device and when we are using the device. In particular, we must be wary of malfunctioning temperature sensors and malfunctioning heating/cooling unit.

Malfunctioning Temperature Sensors - When the temperature sensors on our blackbody source are malfunctioning, we will not be able to determine the actual temperature and may take actions that are extremely harmful to the blackbody. For example, the blackbody may be at 1200.0°C but we are reading a value -30.0°C. Our temperature controller will send the maximum power to correct this error, which may burn out the heater inside the blackbody. When the sensors do malfunction, the temperature controller will read values at the two extremes (the minimum readable temperature (<<0.0°C) and the maximum readable temperature (>>1200.0°C). Since we know this, we can monitor the temperature readings and alert the user that the temperature sensors are malfunctioning. After displaying the error message, the temperature controller needs to disable the output power to avoid any damage to the heater. We will store the two threshold values within our data EEPROM and will modify the values depending on the blackbody source that is associated with the temperature controller.

Malfunctioning Heating/Cooling Unit - When the heating unit within a blackbody source malfunctions, we will not be able to maintain the current temperature set point and the temperature will either rise or fall to ambient over time, depending on the current blackbody temperature. Since the blackbody begins to deviate and the error term increases, the temperature controller will respond by increasing the power that is being sent to the heater/cooler. This additional power does not result in the reduction of the error term. Instead, the error term increases over time. Since we know the characteristics of this error, we will be able to detect it and take appropriate actions. To detect a malfunctioning heating/cooling unit, we will observe the response of the heater when the power being sent to the heater approaches the maximum value. We will use the temperature readings that are going to smooth out the display temperature to determine if the blackbody is responding appropriately to the stimulus. If after an appropriate time/sampling window and the error term between the set point temperature and blackbody temperature does not improve, then we can assume that the heating/cooling unit has malfunctioned. Our temperature controller will display an error message conveying this information. Also, it will turn off the power that is being sent to the blackbody. These time/sampling window will be stored within the Data EEPROM and can be modified depending on the characteristics of the blackbody source.

Minimizing the Write/Erase Cycles on Data EEPROM - Since the data EEPROM has a minimum number of write/erase cycles before the unit fails, we will have to take preventative measures to minimize unnecessary write and erases. To do this, we will start by loading the values in the EEPROM into the SRAM on startup. Once the values are loaded into the SRAM, we will be modifying the values in SRAM instead of the ones in the EEPROM. This will allow us to maintain the same capabilities when using the SRAM as when we use the EEPROM. Since we are using SPI to communicate with the 25AA640A, we are going to have lower read/write speeds then communicating with the SRAM that is onboard the PIC24FJ256DA206. Since we are modifying all of the values within the SRAM, any changes are going to be lost once the microcontroller powers down. We will need to determine a methodology that will allow us to preserve these changes while minimizing the unnecessary read and write cycles. One approach will be prompting the user to save the changes made to a particular value. These values will be essential the operation of the temperature controller and must be preserved if the temperature is lost. The current set point and offset values are examples of these essential parameters. To implement this in the PIC24FJ256DA206, we could have a byte reserved for the 8 byte data group that will indicate if the value is essential and must be saved to the EEPROM. This will give us flexibility because we would be able to remove and/or add this feature to other important values. We could also hardcode this distinction in our program memory. This will save space on our data EEPROM, but we would not be able to modify these values unless we re-burn the microcontroller. In a typical day, the user may modify the set point, alarm parameters, and other essential values around two to three dozen times; the external EEPROM has a mean time before failure at around 76 years, assuming that we are able to write/erase one million times. This method will not allow us to save all of the values on the SRAM like the current PID parameters, so our performance may be reduced. Another approach to lower the number of write/erase cycles is only write values to the EEPROM before power is shut off from the microcontroller. When the user turns off the temperature controller, power will be maintained to the microcontroller so it can write the values to the EEPROM. We will be using one of the general I/O ports that are available on the PIC24FJ256DA206 to determine if and when power is shut off from the temperature controller. Once this flag is set, we know that power has been shut off and we will have to write the values that are residing on the SRAM onto EEPROM. This will allow to save all of the values from the SRAM onto the EEPROM since we will be making the save once and only once per power cycle. If we assume the user will turn off the controller each day, the 25AA640A will be able to last 2740/(number of values to write) years. If we want to maintain the same lifecycle as the first method, we will be able to write twenty different values to the EEPROM which is significantly more than what is allowed by the first method. We would have to implement additional circuitry that will ensure that power is being maintained to the microcontroller when power is unintentionally shut off. A third approach would be to use an additional EEPROM chip that will be responsible of storing the values that the user decides to save. This method will be similar to the first one, but instead of constantly writing and erasing from the main data EEPROM, we will be doing that to the second one. On startup, we will write the values that are currently being stored on the secondary EEPROM on the primary one. This will effectively lower the stress we place on the main EEPROM. This method will be able to increase the longevity of the main data EEPROM, at the cost of the secondary EEPROM. Once the secondary one exceeds its write/erase life cycle, we will switch to use the primary EEPROM in the method descried in the first method.

Calibration

Overview

Due to the sensitivity of the temperature sensing equipment to external noise, with 6 µV to 12µV contributing to a .1°C difference between two temperatures, a set of offsets will be used to mitigate the variability with our temperature sensor subsystem. These offsets must be calculated by comparing the display temperature with the actual temperature of the blackbody source. The calibration process will define the meaning of Offsets and describe the calibration process.

Offsets

Offsets are the realized difference between the displayed temperature and the actual temperature. The actual temperature will be determined using an external thermocouple and other equipment. This measurement equipment will be calibrated on an annual basis, using standards outlined by Infrared Systems Development.

The temperature controller must regulate the temperature of the blackbody source between 25.0°C to 1400.0°C. If each displayed temperature had an associated offset, the controller would have to store 13,750 different points. This would result in accurate readings for the entire temperature range, but would be extremely cumbersome to calculate all the offsets and would take an excessive amount of memory within our microcontroller. Thus, we must determine an offset count that will allow for high measurement accuracy while being plausible in its hardware implementation and calibration process. Since most blackbodies operate close to its upper temperature limit, the temperature readings in this range must be accurate and well within the required specifications. Thus, a majority of the offset points will be concentrated towards the upper end of the temperature range. A sufficient number of points must exist in the lower range to prevent large discrepancies at lower temperatures. A curve fitting technique will be used to determine the offset value for points in between the particular offset points, so an additional point must exist below the minimum allowable temperature and above the maximum allowable temperature. This will allow the curve fitting algorithm to work at all temperature ranges. The offset point should be placed 1°C away from the desired temperature. This will prevent the temperature at a common set point, like 800.0°C, from constantly switching between two different curves.

The temperature controller will have seventeen offset points. Depending on the type of blackbody, the positions of the offset points will change. For cavity blackbodies, which have a high upper limit, ten points will be evenly distributed from 800.0°C to the upper limit of the blackbody. The remaining points will be dispersed as follows. The remaining points will be at 10.0°C, 51.0°C, 101.1°C, 201.0°C, 401.0°C, and 601.0°C , and the upper temperature limit plus 1°C. For extended area blackbodies, which have a low upper limit, the seventeen points will be evenly distributed between the upper and lower limits, with two points falling outside the range.

Because we are not including an offset an every temperature reading, we must calculate the offset at a given temperature by interpolating between two given offset points. Since we have several offset points in between the temperature range of the blackbody source, we can simply use linearly interpolate the offset value that is found between the two closest offset values. The other offset points will not have a weighting when calculating the particular offset value. The values of the offsets will fall between 100.0°C and 100.0°C. The offset points, temperatures that are linked with the offset value, can vary depending on the blackbody source. The formula that will be used to calculate the offset values is given by:

$$O_{New} = \frac{O_{Upper} - O_{Lower}}{T_{Upper} - T_{Lower}} T_{New} + O_{Old}$$

O_{New}=Offset Value at T_{New}, O_{Upper}=Upper Offset Value at T_{Upper}, O_{Lower}=Lower Offset Value at T_{Lower}, T_{New}=Current Temperature, T_{Upper}=Temperature of Upper Offset, T_{Lower}=Temperature of Lower Offset

Calibration Process

The calibration process will tune the offsets that are found in the temperature controller in order to remove the variability in the temperature measurement equipment. To calibrate the temperature controller, we need to compare the displayed temperature with the actual temperature of the blackbody. To measure the actual temperature, an external thermocouple is read by a multimeter. This thermocouple is set in parallel with the internal thermocouple that is used by the temperature controller, and any difference in mV readings should be insignificant. In order to obtain useful data from the external thermocouple, one end must be connected with a probe that is inserted into a 0°C. The millivolt reading will now show the difference between the two temperature sources, the blackbody and the 0°C ice bath. Since we know the temperature of the ice bath is 0°C, the reading will correspond to the temperature of blackbody. The calibration of the ice bath probe is handled by Infrared Systems Development.

The process begins by setting the set point of the temperature controller to one degree lower than the corresponding offset value. For example, when calibrating the 51.0°C offset point, the set point will be set to 50.0°C. The blackbody must reach stability before any changes are made to the particular offset point. Stability is defined as the blackbody's temperature changing at most by a predefined threshold during a sampling window, with the average temperature in a sampling window equal to the set point and the highest and lowest value obtained in the sampling window differing at most by a predefined range. The sampling window, magnitude of the highest and lowest temperature reading, and the number of temperature fluctuations, will be defined by the technician that is calibrating the temperature. Typical values for the parameters are a sampling window of thirty seconds, temperature difference of .1°C, and one allowable temperature change. Once stability has been reached, the offset value can be altered so that the display temperature will match the external temperature reading. After modifying the offset value, the blackbody needs to regain stability. When that occurs, the display temperature and external temperature should match. If they do not, the offset value is altered again, and the steps are repeated until the two temperatures match. When changing offset points or values, the controller must check that no invalid entries are made. This includes going out of the allowable temperature ranges for either the offset points or offset values.

Also, it needs to check that two offset points do not conflict. This means that two offset points cannot have the same temperature and a lower offset point cannot exceed the higher offset point. If the second case occurs, the next offset point will be raised to the lower offset point plus .1°C. This process continues until all offset values have been determined.

Calibration needs to be able to be done manually and automatically. The manual portion will be handled by an employee of Infrared Systems Development. No other user may have access to the offset points and values. Thus, these features must be locked out, until the locks are cleared by an employee at Infrared Systems. To lockout the controller, we will be using an alphanumeric password that is defined by Infrared Systems. Also, the offset values must be readily accessible and easily modified when the locks have been removed. The controller must be able to be calibrated by an existing calibration tool developed by Infrared Systems Development. This tool was written in LabView and utilizes the serial port to modify the offset values. In order for this to be possible, the controller must interpret serial commands that will change the set point and modify the offset values accordingly. Only employees and Infrared Systems may have access to these commands to avoid other users for tampering with the calibration parameters.

PID

Overview

As previously stated we are working on making a High Precision Temperature Controller; this section will cover the control process this device will use to normalize the temperature. There are a lot of ways to go about this route, but for our specific needs we have decided to use a closed-loop controller that uses, and is named after, the Proportional, Integral, and Derivative (PID) algorithm.

Our goal is to regulate temperature so that the oscillations diminish quickly as a constant temperature is reached. A common example to explain this phenomenon is with the temperature of water. When preparing a bath you do not want the water to get too hot or too cold. Initially you turn up the heat, but come back and realize it is too hot so the nozzle is then set to cold. This process continues back and forth, hot and cold. While switching back and forth would work to reach the ideal temperature, the drastic changes cause more time to elapse before settling upon the desired warmth because it takes time for the cold to adjust to the hot already there and vice versa. The role of the PID algorithm is to eliminate some of those oscillations and achieve the necessary temperature in a more efficient manner. Instead of jumping to such extreme temperature ranges the algorithm transforms the hot and cold ranges to be smaller and smaller distances from the desired temperature, which effectively means the error is gradually eliminated.

Requirements

While this control algorithm alone is commonly used for its efficiency, adaptations of it can lead to better performance. For the specific instance of increased effectiveness, we are going to create our controller such that algorithm runs in consecutive loops. Through this it is meant that we plan on running two cycles within the closed-loop control system in hopes that each time more and more noise can be eliminated. In order for the control loop to work correctly it is important that specific parameters are set correctly, if not the system can become increasingly instable. Furthermore, since this algorithm is being implemented for a mechanical device it needs to account for certain adaptations to avoid problems, such as termination of the loop within the dead band.
To make sure this algorithm will work for our design each stage is broken down such that the step by step process can be viewed graphically. As the process progresses some of the aforementioned requirements will be implemented to see their impact on the controller. It is believed that these new adaptations of the algorithm will help the controller run a bit more smoothly. Through step by step testing we can make sure the algorithm is implemented correctly, and have a graphical display of what is occurring in the device.

Breakdown

As visible in Figure 16, the input and the present temperature are put together before entering in the individual PID sections. Once the three steps of the PID are found, they are summed together and moved into the plant. The plant is comprised of the devices within the controller such as a multiplexer, amplifier, AD converters and sensors. The signals of these devices are used to create the output transfer function and the transfer function within the plant. Unfortunately without the knowledge or the ability to physically test each of these devices it is rather difficult to devise their individual transfer function which leaves us currently devoid of information to implement into the paper at this time.

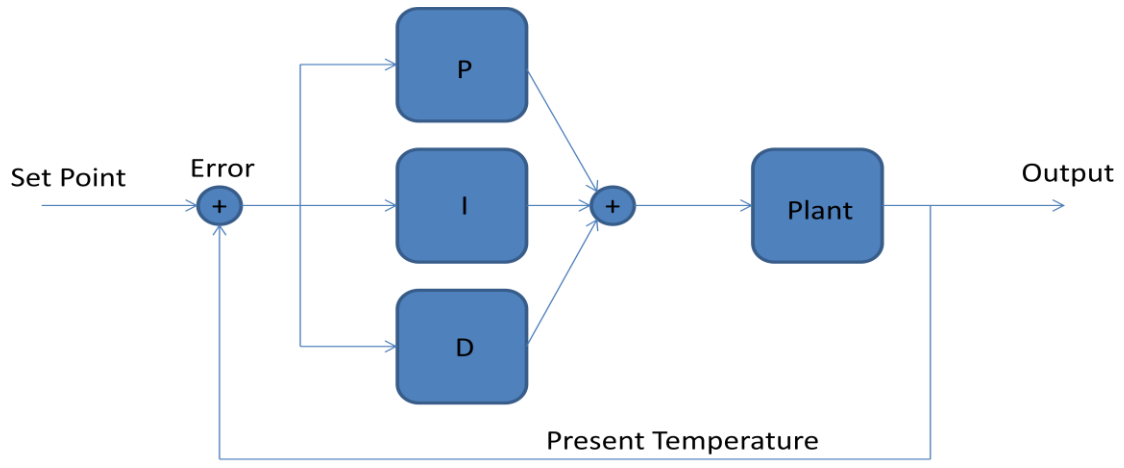


Figure 16: A basic block diagram of PID Controller

Fortunately, there is alternative solution to ameliorate this issue but to still provide some useful and applicable data for the purpose of this paper. The client has given us some data of temperature ranges and their settings along with some set points that are currently used. From here we can back track to find the gain of the current controller which may help us later on once the transfer function is known. With that being said, the old controller and the one we are making are not the same so these values will most likely change through implementation once the actual function transfer function for the plant is found. Meanwhile a generic transfer function will be used to show the step by step implications of tuning and interaction by having the parts added together from P, to PI, and finally to PID. This too will leave us with an idea what to expect when we get our final transfer function.

Error

Before starting with the individual components, it is important to understand the center that this whole controller revolves around, the error. To no surprise very rarely do things work out exactly as planned the first time around, often it takes some work to get there. Equation 1 dictates how the error within the system is determined:

$$e(t) = SP - PT \quad (1)$$

In which $e(t)$ is the error in respect to the instantaneous time, SP is the Set Point which is the desired temperature for the controller. The set point is placed by the user before running the device, when testing the PID we will use 100 degrees Celsius and 1000 degrees Celsius for our two example set points. The other variable PT is the Present Temperature the device is actually at as sensed through a type S thermocouple. With this equation, we can find the difference to determine just how far off the temperature is and so that the controller can assist in fixing it accordingly. Another form of error calculations is sometimes used in the instance of multiple loops such as those in parallel, serial or the Cascade model itself. This practice and procedure will be mentioned in depth later on when the different versions of PID are discussed.

Determining the Gain

Another few variables that will appear throughout the next three sections outlining the PID algorithm are the gain of each term. While each part will have its own respective gain, the gain of one is dependent on the gain of another. The gains combined are described as tuning parameters. In order to perform any tests and show any examples it is first important to determine what the gain is for each individual part. This way, when the variable is mentioned it is clear where the constant we are using for it came from. First we must look at the transfer function of the PID controller in its simplest forms in terms of its parameters.

Where, K_P is the proportional gain, K_I is the integral gain, and K_D is the derivative gain as seen in Equation 2.

$$C(s) = K_P + \frac{K_I}{s} + K_D s = \frac{K_D s^2 + K_P s + K_I}{s} = K_P\left(\frac{T_I T_D s^2 + T_I s + 1}{T_I s}\right) \quad (2)$$

In respect to understanding their overall impact, we will resort to them in their $K_P, K_I,$ or K_D terms. This information provides the steps we will take in determining the gains once our transfer function is found. Each of these gains has their own respective impacts upon the system's attributes such as rise time, overshoot, settling time, and the steady state error. In Table 9 this idea is broken down further to see what exactly each happens if either of these attributes is altered and thus helps in gaining an understanding of how they are determined. As seen, the proportional gain has minimal impact on the settling time, just as the derivative gain has minimal impact on the rise time and the steady-state error.

Gain	Rise Time	Overshoot	Settling Time	S-S Error
$\boldsymbol{K_P}$	Decrease	Increase	Minimal	Decrease
$\boldsymbol{K_I}$	Decrease	Increase	Increase	Eliminate
$\boldsymbol{K_D}$	Minimal	Decrease	Decrease	Minimal

Table 9: The gain's impact on the system

It is already known that one major goal for a PID controller is to diminish if not eliminate error, thus from looking at the table it is important to see which gain variable can help in that plight. When evaluating what impact K_P might have, once again reference Table 9 and look at that row. It is clear that the influence it has on overshoot and the settling time will not help all that much; however it does decrease the rise time which is the main aspect it will be used for. The reason it is not used for the steady state error, is that while it decreases the error, K_I can eliminate it completely. As a result, K_I will be far more useful if it is used for the steady-state error than K_P. The last gain we are looking at is that of the derivative. Since we have already covered the rise time and the steady-state error, this leaves the overshoot and the settling time, both of which can conveniently be decreased through using K_D.

Now that is known what each variable does and their influence on the system the process of how we will set them once we can derive the transfer function of the plant can begin. To initiate tuning the controller all of the gains will be set to zero, since it is known that as K_P increases the over shoot increases it will then be slowly incremented in respect to the plant's transfer function until the output starts to overshoot. From here K_D can be increased until the overshoot begins to diminish. Finally K_I is then adjusted until the error is eliminated, and thus equals zero.

The above mentioned method of trial and error through manual tests is a great way to assure stability, yet there is another way that is a lot faster in computation but does not always prove to be as stable. This method for tuning begins in a similar fashion, but has a far more direct path in determining the other gains once the proportional gain is found, this method is known as the Ziegler-Nichols method. The first step, as before, is the have the proportional gain set to zero; however, this time the gain is increased until it reaches its optimum point which is a new variable K_U that is then plugged into two equations to form K_I and K_D. As a result of so much weight relaying on this ultimate gain, if it is off then both of the other parameters are off as well. If the parameters are set incorrectly the system will remain unstable, and the outputs will provide a plethora of oscillations that will eventually lead to an incorrect divergence if any at all. Should this occur, it is nearly impossible to have your controller correctly set since the output produced is incorrect. Thus for our efforts we can use this method to get us approximate gains, and then plug them into the manual method to see how the work and adjust as need be before the final implementation of the device. This way we are not grasping at straws trying to adjust the parameters, but have a guideline that gives us an idea of where to start.

In the last part of Equation 2 some new variables are introduced, these are T_I the integral time constant, and $T_D$the derivative time constant. The former is derived from the proportional gain divided by the integral gain, and the latter is formed from the derivative gain divided by the proportional gain. Once we can find the proper transfer function of our plant, these values will be determined for a series of sets. This way if the user has the device at a specific temperature range, the tuning parameters are automatically adjusted accordingly. For our device there will be five different sets of temperature ranges, each with their own respective set coefficients. In Table 10 five examples are given based off the clients' previous sets. While these values will not be the same in our device since the overall PID hybrid used will be different, these temperature ranges for the five sets, which will consist of the data in the first four rows, will remain the same. Table 10 demonstrates how different, or similar, the various variables are despite the change in temperature and another aspect of our device that relays on these gains along with displayed in the last two rows how the settings will result in the gains for the Type A equation. Through the user interface on the touch screen the client will be able to choose any one of the five sets so that that the gain for that temperature range is fixed before and adjusted more specifically to that range before beginning the control loop.

Temp Range in degrees C	0-150	150-390	390-690	690-890	890-Max
K_P (in Degress C)	17.8	17.8	17.8	17.8	17.8
T_I/min	0.3	0.3	0.3	0.3	0.3
T_D (min)	1	1	1	0.8	0.6
K_I	59.33	59.33	59.33	59.33	59.33
K_D	17.8	17.8	17.8	14.24	14.24

Table 10: Five sets of Data Sets Available through the User Interface

Even if the exact transfer function were known and a theoretical understanding of what should happen, without some sort of reference there is no true way to say if the results of tuning the device or what is happening is actually correct. Thus for the purpose of this paper and to gain an understanding of what our device will actually be doing in each stage, a general transfer function will be used for reference and examples. Once the overall device is built, and the transfer function is known we will know how to test each stage through MATLAB simulation and what to expect through the PID process. The equation for our general example is in Equation 3:

$$\frac{X(s)}{F(s)} = \frac{1}{s^2+10s\_20} \quad (3)$$

While the most common Type A form of the PID algorithm is displayed in Equation 4, there are a few other types that we considered using that will be mentioned later on. Here you can see where the various gain inputs are placed in relation to the proportional, integral, and derivative functions taking place. Within the next few sections this equation will be broken down and explained term by term. To implement each step, the equations and data will be executed through MATLAB with the use of the transfer function in Equation 3 to provide us some visual graphics.

$$u(t) = P + I + D = K_P e(t) + K_I \int_0^t e(T)dT + K_D \frac{de(t)}{dt} \quad (4)$$

After it is broken down, a few adaptations and other things to watch out for will also be discussed in depth. These too will use the same values so that any modifications to the basic equation can stand out. Hopefully this can provide a deeper understanding to the algorithm itself, along with our implementation of it along with making it clearer what to expect from our final design.

Proportional

The first part of the PID algorithm is the proportional step. As may soon become evident, each element works exactly as their names depicts, and are highly dependent on the error. The resulting output from this step is a value proportional to the error as shown in Equation 5:

$P = K_P * e(t)$ (5)

Once again $e(t)$ represents the error in respect to time; however, here the error is multiplied by the proportional gain K_P. Since the error is constantly changing, and is the initial part of the equation that is getting finer tuned, it is safe to say that this step is really only making short term corrections. As depicted in Figure 2 and anticipated from Table 1, the proportional gain's influence leaves the system with quite a bit of overshoot. While the signal eventually plateaus through the rest of the PID process it is believed that such oscillations can be obliterated.

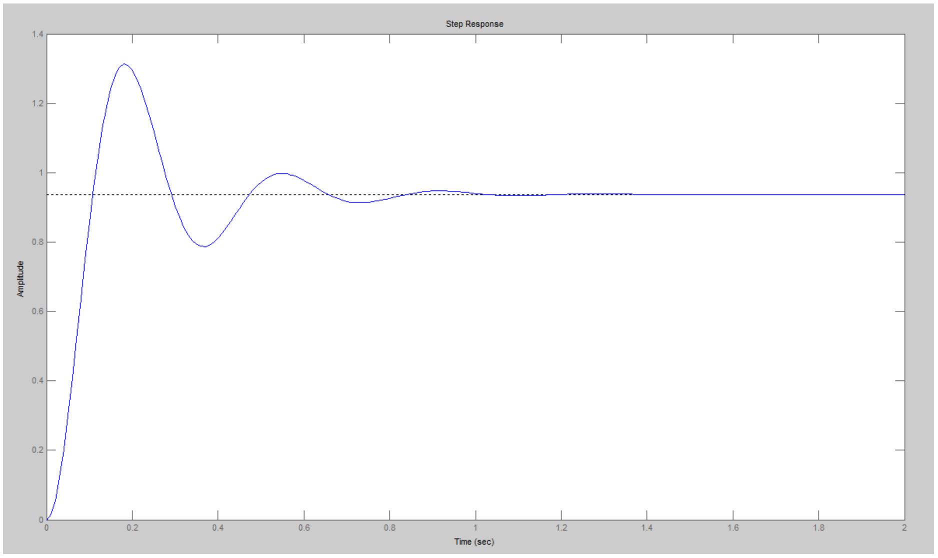


Figure 17: The Proportional response from the general transfer function

Integral

Next we have the integration component, which as the name implies consists of using integration in relation to the time parameters, the gain, and most importantly, the error as seen in Equation 6. This step is known as a manual reset for the equation, and should this manual reset be replaced with an automatic reset, the system may notice some lag. In the instance of lag another variable will be added to the final transfer function to account for any dead time. Seeing as our device is sensing through a sensor before going through an amplifier, the chances of there being a time-delay between when the initial step is taken, and when the output is received is great. This is why in our final transfer function; we must include a parameter to account for dead time.

$I = K_I \int_0^t e(T)dT$ (6)

In Equation 6, $e(T)$ remains the error in respect to time that is then integrated from when the device is started up until time t; however, here the integral of the error is multiplied by the integral gain K_I. This step eliminates the steady-state error and thus assists in moving closer to the desired set point. As mentioned in Table 1, the integral gain causes the over shoot to increase. This is a result of the integral being taken of all the errors, not just the present error, but the past errors as well. On the bright side because it is accounting for past and present errors, it assists in adding some long-term precision.

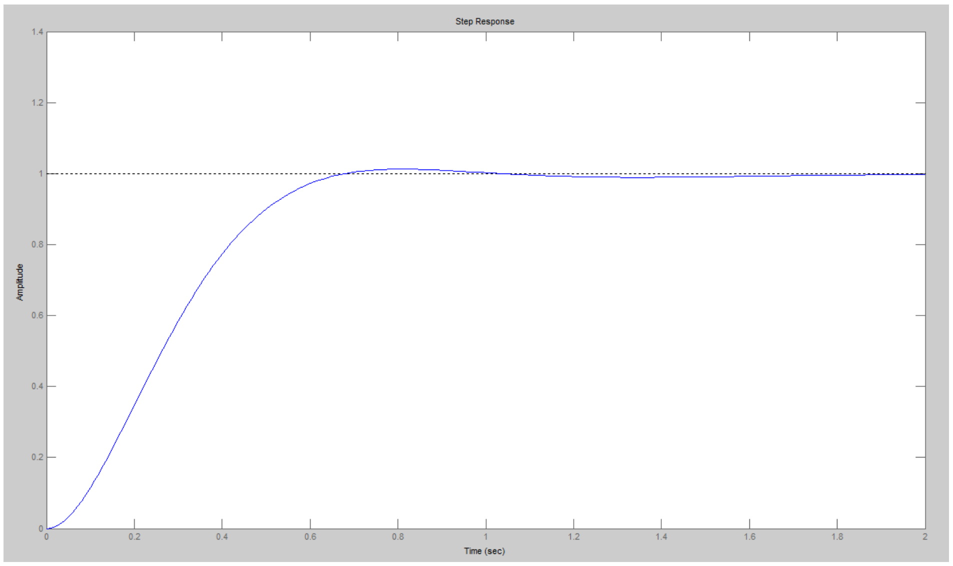


Figure 18: The new signal after influences of the Integral are implemented

As evident in Figure 3, the addition of the integration factor drastically reduced the oscillations that previously existed through properly eliminating the steady-state error. Yet under closer examination along the dotted line, it is evident that the signal is still not exact. At the end of the day our device needs to have accuracy in the hundredths, thus even the one minor oscillation is far too much and needs to be corrected. While in some cases people leave the PI alone and end it here, our device cannot wait that long for a settling time to be reached and needs to be even more exact and thus we continue to work toward the overall PID equation.

Derivative

Finally is the derivative portion of the algorithm, here is where things can get a bit messy, and by messy we mean noisy. In the process of trying to find our rate of change so that the error does not grow exponentially this step often has the issue of introducing noise.

$$D = K_D \frac{de(t)}{dt} \quad (7)$$

For Equation 7, the derivative of $e(t)$ is taken in respect to time and is multiplied by the derivative gain K_D. As many know, the derivative assists in finding the slope, or the rate of change, of a curve. This instance is no different in that in this instance the derivative is found through finding the slope of the error in respect to time. While it is not visible when looking at Equation 7, it will later become more visible that unlike the integral function, that uses all of the error, the derivative function only uses the current error and the previous error. As anticipated and evident in Figure 19, through this process the derivative gain assists in predicting the next term used and decreases the settling time and overshoot of the signal.

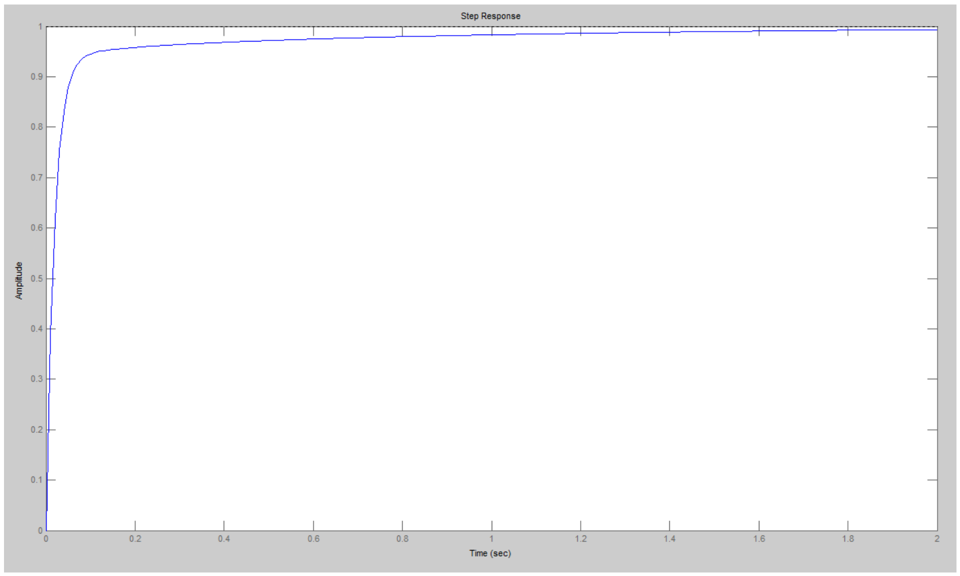


Figure 19: The final signal after going through all three PID processes

Variation

The PID formula takes on many forms, and while the general principle is the same the impact of the error to each portion of the equation often changes. Three common variations are Type A, Type B, and Type C. Meanwhile, other variations of the process include a cascade method or PID with dead band parameters. These latter methods are less differentiated by error, and more so in their form of sampling and overall process.

The first three bring discussed are Type A, Type B, and Type C. Type A, otherwise known as the Textbook PID, and the form used in the above discussion, has the error term impacting all three portions of the equation. This can cause some issues because derivative term is highly dependent on the previous set point and its impact on the error. As a result, if the set point is changed throughout the process this will have a negative impact on the controllers output. In attempts to fix this, the set point is removed from the derivative term and this new form is known as Type B. This formula is much like the textbook version, except that the derivative is taken solely of the present temperature value. So instead of the derivative using the previous error and the current error, it uses the previous present temperature and the current present temperature. From here it is considered where else using the set point could cause the controller to become less efficient.   Since the initial gain of the system with regards to this paper is being set through a step function the ideal temperature, the proportional gain already relies heavily on the set point. It was then decided to, and in many cases recommended to use the new variation known as Type C. Similar to Type B, the set point is taken away from the derivative term, and unlike Type B it is also taken away from the proportional term. Through this type the only term that still relays on the error is the integral.

After talking to our client it has been determined for the purposes of this project the proportional band needs to know of the error in order for it to determine a linear power. This alone eliminates Type C from contention. Secondly, when discussing the derivative portion of PID it was mention that it used as a predictive measure in that it linearly calculates for future error through the use of present and past error. This process seems a little void if it were only calculated based off of the previous present temperature and the current present temperature, for while it would predict the next present temperature it would not account for the error. If the slope of the error is too great in order to look out for over shoot, and thus making Type B not a useful variation for our purposes. This leaves us with the traditional Type A format and design of the PID equation as the basis of our controller in which any other loops or adaptations will be worked around.

When thinking of a cascade, a person might think of a cascading waterfall or something along the lines of a domino effect in which the result of one process leads to that of another. This theory remains true even in relation to PID. The cascade method uses two loops of PID in which one set point is merely the output of the previous loop. The error is still found in the same way as before, by taking the difference of the present temperature from the set point however each loop has their own independent sensor that reads them this present temperature. The outer loop often referred to as the master or the primary loop has a sensor close to the device that is being tested. In our case, this means that the sensor is located near the thermocouple that is reading the direct temperature from the blackbody source or any other device that is actually running the test. This outer loop should have a period at least four times larger than that of the secondary loop and thus should also have an extremely large C. Meanwhile the sensor in the inner loop also known as the slave or the secondary loop is placed near the controlling energy source. This is done so that outer sensor can determine an accurate temperature of the unit, while inner sensor assists in patrolling the amount of energy that is getting built up. At this stage the settings to make the device get hotter or colder are controlled. Some necessary requirements when setting this process up is that the secondary loop is infinitely faster than the primary loop, and also be able to have influence over the primary loop. These two loops work together in attempt to find a more precise process with less over shoot and the hopeful early elimination of dead time which assists in making the stabilization process much smoother. When attempting to tune this device it is important to start the process as you would a normal run, first begin with the inner loop and set those PID parameters before moving to tuning the PID parameters of the outer loop. This process is used in the high hopes that some of the problems with noise or disturbances can be eliminated or at least reduced in the beginning.

Along the lines of Cascade is feed forward and feedback. While a feedback loop would take our outputs of the system and directly input them into the loop again, the feed forward would take data from disturbances before they actually reach the system. It is almost as if it is solving the systems problems before the controller knows it has any to begin with. To include this implies another sensor and requires knowledge of the plant. Where the above sensors in the Cascade model measure the current temperature and the pent up energy, this sensor would measure the changes in the disturbances. Thus, while the Cascade model has two feedback loops measuring two different aspects of the device, feed forward measures the disturbances and the feedback through one specific feedback loop. There are two forms of feed forward, that of set point feed forward and that of disturbance feed forward, the latter is the typically seen in purposes similar to our own in which the ambient temperature is measured and adjusted accordingly. In order to use this method the system has to me causal. Through determining any disturbances before they actually make it into the system, they microprocessor can attempt to fix, adjust, or avoid them. While this is desirable for our device, the Cascade method would be more specific and efficient since those two loops can be clearly defined.

One other setting similar in theory to dead time that is important to watch out for and that is more commonly used in mechanical devices, such as ours, is the Dead Band. This is typically used in a mechanical device because it used in turning off the output. How this works, is that there exists an area that is essentially dead, in that once within this range nothing happens. In PID, this area is located closely to the set point. This way once the algorithm has gotten to a specific distance from the set point it stops producing an output otherwise once it reaches the set point there could be further complications of the system turning on/off. Should an instance such as this occur, the alarms and relays within our controller will go off to alert the user of the situation so that they can choose the next path they would want to take accordingly.

Our System

Within our device we have chosen a few different ways to go about implementing PID control into the closed-loop. First there a few things that we still may or know not know from the get go. First off, we know that we want to have about five individual sets that each have their own proportional gain, integral time constant and proportional time constant already set and determined. We also know that none of these final values can be computed without the transfer function of the plant, and that only once that is found can the manual method of tuning begin. Secondly, we know that we want to use the basic Type A design of the PID formula for implementation of our controller so that each variable is dependent on the error and not just the present temperature. This then uses the values already saved within the five sets, and plugging them into the basic PID equation. As will be discussed further in the paper, this whole process will be programmed in C and taking place within the microcontroller, who will be using type S thermo couples for the sensors, and converting the data through a multiplexer.

As far as extras and actual changes to the design go we know we plan on having multiple loops. The outer two will be that of the Cascade control process where the first loop sensor will read the present temperature to get it close to the set point, and the second loop will watch out for power. Within these loops it is also important to be looking out for dead band. As a result of the desire to have our device within the hundredths in terms of precision, it is extremely necessary to use caution so that the device will signal a stop in the loop when the dead band area is reached. When the dead band is reached the device will either turn off or an alarm will go off to inform the user. An alarm will also go off in the instance that the temperature has shot too high or too low. Hopefully the PID algorithm will work properly so that this issue does not occur.

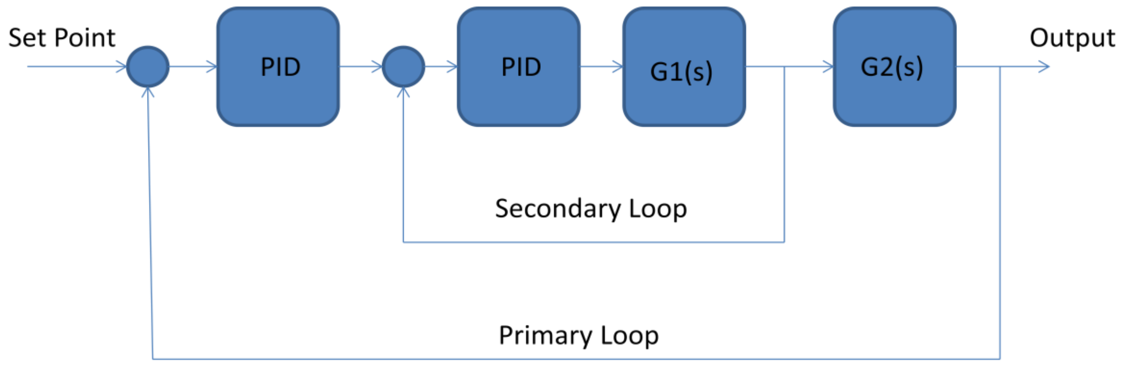


Figure 20: Diagram of Control Process

In the above Figure 5, a rough diagram of the process is given in more depth based solely on the Cascade method. The input for our system is the set point, otherwise known as the ideal temperature we want, this value will be set by the user. This set point will go into a summing junction, along with present temperature as found by the primary controller to determine the error. The output of which will be used as the set point in determining the error of the secondary controller. After the signal makes it through both loops, it goes through the amplification process. What comes out of the amplification process goes into the multiplexer and from the multiplexer back to the A/D converter where the present temperature will once again be calculated and sent to compute the error once more. So on this closed loop will continue, repeating the algorithm every second.

Pseudo Code

As previously mentioned, the overall programming of the PID algorithm and any of the adaptations we make to it will be done in C. The data from the devices will also be read into microprocessor where the actual code will be implemented through the physical control loop. An example of the pseudo code or the steps take in writing the overall code for the Cascade PID algorithm for this controller is seen below in Figure 6. Once the hardware is put together and the system can be tested, we can go about computing the transfer function while including variables for the dead time and a command to terminate the loop should the output reach the dead band.

```
Kp is proportinal gain, Kd is the derivitave gain, Ki is integral gain, and t is the timestep

prev_error = 0.0
Iterm = 0.0

while pid is running
{
    CalculatePID(CalculatePID(setpoint, measurement), measurement);
}
CalculatePID(setpoint, measurement)
{
    error = setpoint - measurement
    Iterm = Iterm + error*t
    Dterm = (error - prev_error)/t

    prev_error = error

    return Kp*error + Ki*Iterm + Kd*Dterm;
```

Figure 21: Pseudopodia for Cascade PID Algorithm

Within the pseudo code, the variables are initialized and the previous error is set to zero. From here the first, outer loop of the PID is run as it takes in the measurements, the output of this first loop is then used to calculate the error of the second loop which is then too run through the PID algorithm. That output is then sent out of that loop and back out to the plant. When testing the PID Algorithm we can simply follow the steps described throughout this section to tune the parameters with the use of the transfer function our plant before we implement the Cascade upon our High Precision Temperature Controller.

Temperature Sensing Subsystem

Overview

The temperature sensing system for our High Precision Temperature Controller for Infrared Systems Development must be able to accurately read temperatures from black body heat sources and send this data to a microcontroller for the information to be processed. This temperature information will then be used to maintain or readjust the temperature and the loop will keep accepting feedback from this temperature sensing system. The system must constantly update the information in order to monitor the temperature in real time. The data captured from the our sensing devices must be accurate within 0.1 degrees Celsius and have the lowest noise achievable for reliable analog to digital conversions.

The temperature sensing subsystem can be broken up into four main sections: temperature sensing devices, amplification, multiplexing and analog to digital conversions. For the temperature sensing devices section, we will consider different types of commercially available sensors and compare their advantages and disadvantages. The devices discussed will include resistance temperature devices and thermocouples. For the amplification section, we will compare different candidates for the amplification process and determine what sort of gain will be needed. Several methods may be used in our final design and the integration of different types of filters, amplifiers and other electronic devices will be considered in this section. Several multiplexers will be required for our design; 2 will be needed for data selection from our temperature sensors to our analog to digital converters, and 1 will be needed for our microcontroller to select which peripheral it needs to communicate with. In the analog to digital converter section, we will discuss the theory of analog to digital conversions and the different methods used in modern day applications. This section may also include attributes of the microcontroller choices we are considering. After processing by the microcontroller, the output of the PID algorithm will need to go back out to the heat source. This will be done via a digital-to-analog converter and then a decoder to select control of the fan or heater.

To complete our discussion on the temperature sensing subsystem, we will need to consider the possible configurations of the combined components mentioned above. Each reasonable candidate from our sensing, amplifying, multiplexing and analog to digital converters will be theoretically matched up with other candidates and compared in that particular setup. Product longevity will also be a factor in our design since the finished controller will eventually become a product that Infrared Systems Development will manufacture and sell. The block diagram below shows the relationship between components in the temperature sensing subsystem.

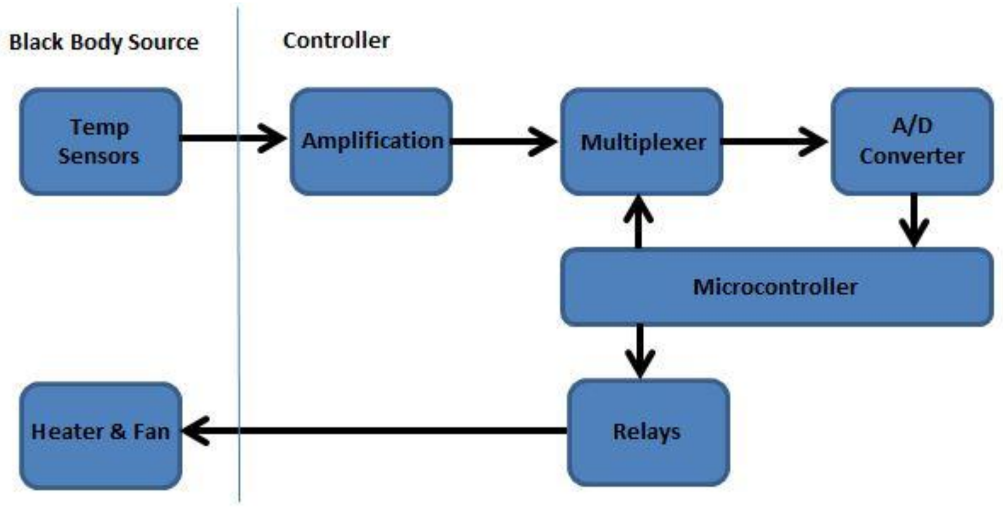


Figure 22: Temperature Sensing Block Diagram

Requirements

The temperature sensing system must be accurate within +/-0.1 degrees Celsius. The input temperature range for the device will be -30 to 1400 degrees Celsius and the ambient temperature will range from 0 to 40 degrees Celsius. Temperature sensing devices must be remotely located from the controller itself but still send real time data back to the microcontroller. The signal amplifiers must produce enough voltage to be recognized by the analog to digital converter but still be accurate to its original state. The multiplexers and decoders must have enough inputs to accommodate all sensors along with reference readings.

Temperature Sensing Devices

Overview

There are two temperature sensing devices currently embedded in Infrared Systems Development's black body sources. Both devices are of the same time type, whether that be a type S or type T thermocouple. Type S thermocouples are used for sources with upper temperature outputs of above 500 degrees Celsius and type T are used for temperature readings under 500 degrees Celsius. We are considering adding 2 more thermocouples to the sources in different positions than the already existing thermocouples. The logic behind this idea is that if we average the measurements between all 4 thermocouples, we might account for the thermal gradients inside the heat source and gain more reliable data. We will need to gather data from ISDC's black body sources to determine the gradient inside of their cavity and extended area sources and see if this added measurement will be beneficial. Although this is a change in design to the sources themselves, we will still need to account for these improvements in our controller design.

Thermocouples

Thermocouples are the only choice we really have for temperatures readings over 850 degrees Celsius, however we will have to compensate for their non-linear responses and vulnerability to electrical noise. There are many different types of thermocouples, with various temperature ranges and accuracies. Type S thermocouples are the only devices that will accurately read our upper temperature limits for the extremely high temperature sources without becoming damaged. Infrared Systems Development constructs their own thermocouples in house which they use in their sources; therefore, we will be using proprietary thermocouple tables to determine the corresponding voltages for every tenth of a degree. Shown below is a plot of the temperature (in degrees Celsius) and output voltage (in milli Volts) for the specific type S thermocouple embedded in ISDC's high temperature cavity sources. As you can see from the graph, the response of this thermocouple is non-linear but comes close to linearity in the upper temperature range of 1100 to 1400 degrees Celsius.

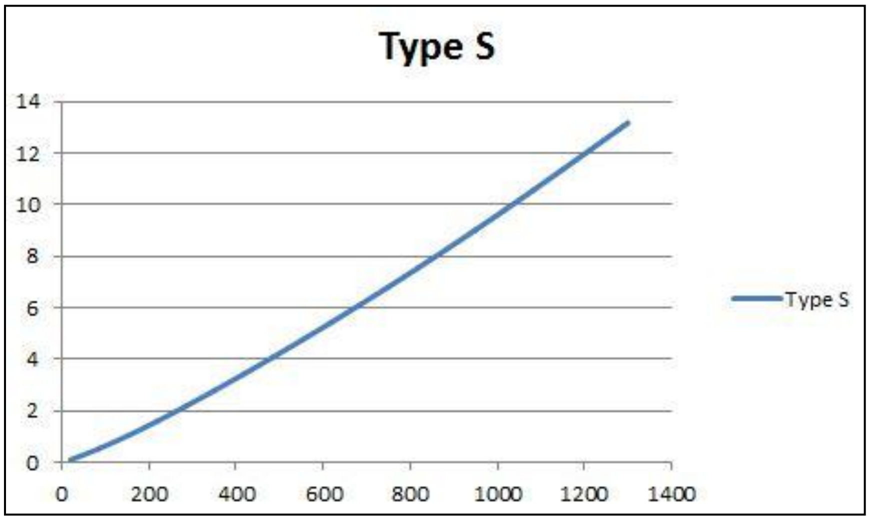


Figure 23: Temperature vs Voltage Plot of Type S Thermocouple

We must be careful to eliminate any sources or error and noise from our sensing system. There are 2 major sources of measurement error in thermocouple wire: de-calibration and unwanted virtual junctions caused by wire insulators. To prevent de-calibration of the thermocouple wire, we must be cautious to not cause a steep temperature gradient in the wire and also take care in handling our wire.

Thermocouple Extension Wire

Since the heat source and the controller will be remotely located from each other, we will need to use thermocouple extension wire to connect the two. This will also benefit us in the fact that our measurement wires will need to be bundled in a wire conduit, of which small thermocouple wire cannot withstand. In order to reduce magnetically induced noise, we will seek out extension wire that comes in a twisted pair configuration. We will need 2 different types of thermocouple extension wire for type S and type T thermocouples. For type S, we will use extension wire made of Copper and a thermocouple alloy #11. These two wires come in a twisted pair and can be insulated with a variety of materials. For type T, we will use extension wire made of Copper and Constantan. Again, these two wires come in a twisted pair and can be insulated with a variety of materials. These metals and alloys are specifically chosen to match each thermocouple type in order to prevent creating a thermal junction where they meet; therefore, we will not need to monitor and compensate for the temperature of that junction. As mentioned previously, we need to be aware of decalibration of the wire by preventing too steep a temperature gradient; which should not be a problem in our system. Also, we need to take into consideration virtual junctions created by insulation. The gauge of these wires can range from 16 to 24 AWG.

Cold Junction Compensation
Cold junction compensation is needed whenever a thermocouple is used to measure temperature. This is because most times, the thermocouple wire is connected to leads of a voltmeter, which is usually made of a different material thus creating another thermal junction. Since we have been careful to find extension wire and connector pins that are the same material of the thermocouple wire, we have not created any thermal junctions. However, we will still need to measure the ambient temperature and subtract this temperature out of the reading from the thermocouple; this is considered cold junction compensation. To monitor the cold junction of our thermocouples, we will need a high precision RTD attached to our cold junction. The output of our RTD will be a change in resistance due to the change in temperature. This change in resistance can be correlated to a temperature value, much like the change in temperature can be related to a voltage for a thermocouple. This resistance-temperature relationship can be found from looking at an RTD lookup table. From here, we will need to look up this temperature in the appropriate thermocouple table. After this is successfully done, we need to correlate this temperature to its correct voltage output needed for the compensation of this cold junction. This can all be realized within the microcontroller, since the EEPROM will be storing all of our required thermocouple and RTD lookup tables. The final value of cold junction compensation voltage can be subtracted from our readings once all measurements have been sent to the microcontroller, and the true temperature can be found. The method of cold junction compensation can be summarized in the following flow chart.

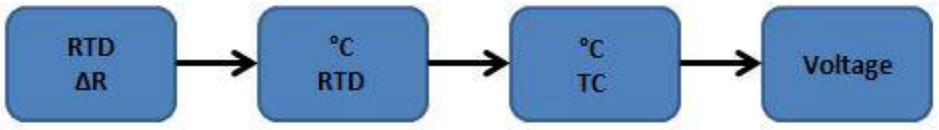


Figure 24: Cold Junction Compensation

Calibration

Infrared Systems Development uses a 0 degree Celsius ice bath to calibrate their cold junctions. To maintain the reliability of our cold junction compensation, we will monitor the change in resistance of our cold junction mounted RTD. Any large, sudden changes in resistance will be considered an error and the data measured during that time period will be discarded. This will also prompt the user that there may be an issue with the device.

Resistive Temperature Detectors

Since our cold junction compensation helps determine the accuracy of our whole system, it would be wise to seek the most accurate measuring device to detect this ambient temperature. Resistive temperature detectors are considered the most accurate and stable temperature sensors because they are predictable and virtually immune to noise.[2] RTDs made out of Platinum are the most expensive devices, but are also the most accurate. We will need to mount this RTD to our controller, to measure the ambient temperature around the readings. Omega offers a variety of mounting options for Platinum RTDs including cement on and bolt on configurations.

Connectors

Because thermocouple wire is sensitive to what material it encounters, we need to find pins and connectors that will not create a thermal junction. We will also need to make sure that the pins for the male connectors are made for wire gauges from 16 to 24 AWG. We will need to find a connector with a minimum of 8 pins. Since the heater and fan from the source will need to be controlled depending on the output of our PID algorithm, they will also need to be connected through this piece; therefore, we will need the following pins:

- Fan power (+)
- Fan power (-)
- Fan ground
- Heater power (+)
- Heater power (-)
- Heater ground
- Thermocouple 1
- Thermocouple 2

The male connector will be attached to a rubber conduit holding the bundle of wires from the source to the controller. A female connector will be mounted to the chassis of the controller and will therefore need to be flanged style with mounting holes. Omega offers many types of thermocouple connectors for both type S and type T. They also offer gold plated pins for those connections not needing to correspond with the thermocouple wires, which we would use for the heater, fan and ground.

Amplification

Since the amplitude of the data we are collecting will be in the magnitude of milli Volts, we will be needing some sort of amplification circuit for our analog to digital conversion to be reliable, and for the data to be differentiated from noise by the multiplexers. Because our data is so sensitive, we need to be extra cautious about voltage drift when selecting our amplifiers. Assuming that the temperature of our amplifier will be no greater than 40 degrees Celsius (this is a huge over-estimate) and ambient temperature will be around 25 degrees Celsius, the maximum voltage drift we are willing to afford will be 0.5 micro Volts per degree Celsius. This would produce an induced offset of 7.5 micro Volts which will be our maximum noise via voltage drift. Therefore, we will limit our voltage drift to no more than 0.5 micro Volts per degree Celsius when considering our amplifier candidates. The input offset voltage will also be considered when choosing an amplifier to minimize the amount of error in our amplifying circuit.

We must also be cautious to keep our output in the range of 0 to 5 Volts if we wish our multiplexer to operate on a single +5 Volt supply. From this, we must determine the closed loop gain desired for our amplifier. If we decide to let our maximum output be 4.5 Volts, we must divide by our maximum temperature voltage as given by our thermocouple or RTD tables.

Amplifier Design

For each type of sensor, we will use the same model amplifier in order to keep the offset voltage and drift the same for each measurement. This way, after they are fed through the whole system and are ready to be used by the microcontroller, these noise values will cancel each other out.

Type S Thermocouple

The maximum temperature voltage for the type S thermocouple already embedded in the high temperature heater is 13.17098 milli Volts; therefore, this would give us a closed loop gain of 4.5V divided by 13.17098 milli Volts which gives us a 341.66 maximum gain. We can now choose resistor values for Rd to equal 300 Ohms and Rf to be 10.2 kilo-Ohms, both of which are commercially available.

Type T Thermocouple

The maximum temperature voltage for the type T thermocouple already embedded in the low temperature heater is 20.87 milli Volts; therefore, this would give us a closed loop gain of 4.5V divided by 20.87 milli Volts which gives us a 215.62 maximum gain. We can now choose resistor values for Rd to equal 300 Ohms and Rf to be 64.4 kilo-Ohms, both of which are commercially available.

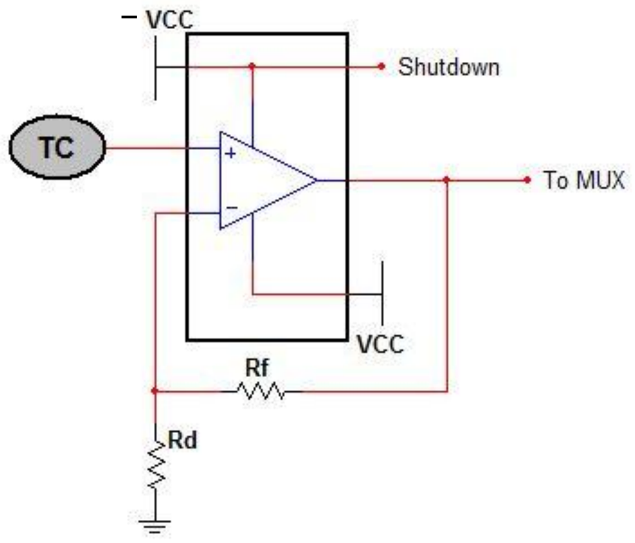


Figure 25: Thermocouple Amplifier Design

Platinum RTD

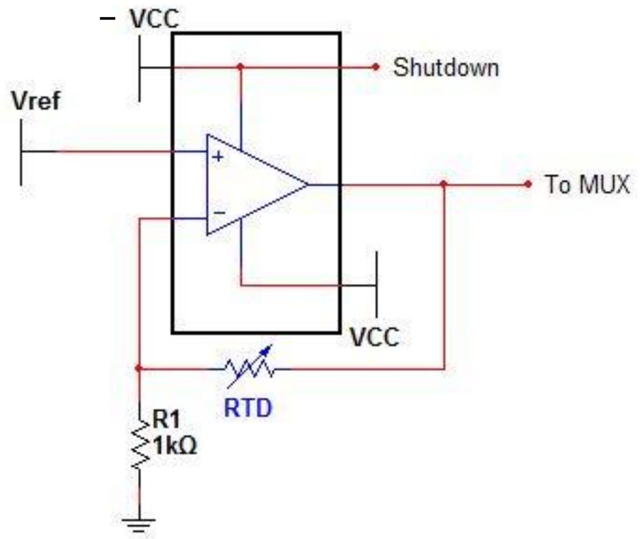


Figure 26: RTD Amplifier Design

The amplifier design for the RTD is a little more complex than the ones previously mentioned. Because of the nature of an RTD, direct voltage cannot be measured and its temperature correlation is a function of its resistance. Therefore, we will use the RTD as the feedback resistor in our circuit. We will also need a reference voltage source going into our non-inverting terminal, of which we will also design based on our desired output. We know the output of our RTD amplifier will be

$$V_{out} = V_{ref}[1 + \frac{R_{RTD}}{R}]$$

and we know our desired maximum value; which is 4.5V. We can find the maximum RTD resistance via a platinum RTD lookup table; which is 269.35 Ohms. If we choose our resistor to have a value of 10 kilo-Ohms, we can then calculate our needed Vref.

$$V_{ref} = \frac{4.5\,V}{[1 + \frac{269.35\,\Omega}{10\,k\Omega}]} = 4.38\,V$$

Since the exact value of 4.38 Volts is not commercially available, we will seek out a 4.5 Volt reference voltage. This will still keep our output value below 5 Volts. We will discuss reference voltages in another section of this paper.

Candidates

OPA211 – Texas Instruments
This precision operational amplifier is rated for low voltage noise at 1.1 nano Volts per square root Hz at 1 kHz. Although our signal is very close to a DC signal, any noise introduced to our data could alter our data severely. The datasheet for this op amp also boasts that this device is perfect for driving high-precision 16-bit ADCs, which is what we will be eventually interfacing with. The voltage drift for this device is rated at 0.35 micro Volts per degrees Celsius which would correspond to an induced noise of 5.25 micro Volts. This device also offers a shutdown feature which protects the device from an overage in supply voltage. It will shut off in between the rail – 3V and the rail – .35V; therefore, we will need to account for this variance in supply voltage when we consider our power design. The input voltage offset is rated at +/-30 micro Volts typically but has a maximum rating of +/-125 micro Volts. The maximum open loop voltage gain is 130 dB.

TLC2652 – Texas Instruments
This precision amplifier is chopper stabilized as well as having an extremely low offset voltage and voltage drift. The benefits of a chopper stabilized amplifier are that offset and drift compensation is done for us inside the op amp package and that noise is filtered out internally. The chopper stabilization technique is preferred for high precision DC circuits and is ideal for our temperature sensing application. However, with this chopper circuitry, two external capacitors will be needed for operation. This op amp is ideal because of its super low voltage drift of 0.003 micro Volts per degrees Celsius and offset voltage of 1 micro Volt. The datasheet gives the large-signal differential voltage amplification which is more accurate than the open loop gain information because it is measured with a load. Its large-signal differential voltage amplification is 150 dB.

LTC2050 – Linear Technology
This amplifier claims to be a zero-drift op amp. The input offset voltage is +/-3 micro Volts maximum and the drift is 0.030 micro Volts per degrees Celsius, but does not use the chopper stabilization technique. This amplifier also has a shutdown feature to protect the device from an overshoot in supply voltage. This feature will be useful to us because if a user accidentally changes a setting and ruins the op amp, he/she will have to send the controller to Infrared Systems Development for maintenance. The shutdown option will circumvent problems like this.

Part	Manufacturer	Power Supply	Voltage Drift	Offset Voltage	Price
OPA211	Texas Instruments	Single, 5 V	0.35 uV/°C	125 uV (max)	$5.15
TLC2652*	Texas Instruments	Dual, +/- 5 V	0.003 uV/°C	1 uV (max)	$5.85
LTC2050	Linear Technology	Single, 5 V	0.030 uV/°C	3 uV (max)	$2.88

Table 11: Amplifier Candidates

Because of its super low drift and offset ratings, we will choose the TLC2652 by Texas Instruments. The chopper stabilization technique implemented by this device is also desirable. We will need external capacitors for this device.

Multiplexing

We will be using 2 multiplexers and 1 decoder in our subsystem. The multiplexers will be used before the inputs to our 2 analog to digital converters. One multiplexer will switch in between sensing devices for the temperature sensing ADC, the other will switch between the cold junction RTD and its reference voltage for the cold junction ADC. The decoder will be used to select which SPI peripherals the microcontroller will need to access. The first multiplexer will need to take in inputs of possibly 4 thermocouples. The device will make the switch between the 4 thermocouples in order to average their readings in the microcontroller. The decoder will be used to save pins on the microcontroller, and be able to send a high signal to any of the chip enable lines for all the peripheral devices. For sake of convenience and what is commercially available, we will consider four-to-one multiplexer, a two-to-one multiplexer, and a one-to-eight decoder. The output of the multiplexer will go through some other electronic devices to be determined below and ultimately to an analog to digital converter to be digitized and sent to the microcontroller. Some specifications that need to be discussed before further selection can be made are: input offset, voltage supply and charge injection.

Input offset will be an important specification when considering the DC accuracy of our multiplexers and decoders. We will need to find the amount of voltage offset and compare it to our input signal to see how much error will be induced.

Input voltage offset has contributing factors of the device’s on-state resistance, drain on-state current leakage, input bias current and source resistance. Our source resistance is unknown, therefore we will estimate the input offset assuming a constant source resistance. Since our input signal will be no greater than 4.5 Volts, according to our amplification circuit, we will not consider any devices with an offset greater than 0.5 micro-Volts.

The voltage supply is a flexible specification for us since we will have control of the power design. For convenience and power consumption ratings however, let us seek out components that allow us to use a single power supply of 5 Volts.
The charge injection for this device will play an important part in deciding what other electronic devices may be needed for this circuit. If the charge injection is very large, we may use a filtering device to filter out any spike in voltage that the switching process will incur. If the charge injection is relatively small, it might be more difficult to spot in our data; therefore, we may use a delay device to time our analog to digital converter to only turn on once this charge injection voltage spike has dissipated and only true data is flowing through. Since we are dealing with relatively small signals, noise ratings are crucial for reliable data and will play a big part in picking a component. We will make an effort to choose a multiplexer and decoder with the smallest channel-to-channel crosstalk. Since our measurements will be closer to a DC signal than AC, we will not heavily consider the time delay properties of these devices.

Temperature Sensor Multiplexer

The temperature sensor multiplexer will need to be a 4:1 circuit; each input represents a thermocouple inside the source.

ADG704 – Analog Devices
This device is a 4:1 CMOS analog multiplexer that comes in a small 10 pin, micro SOIC package. It can be powered by a single supply from 1.8 to 5.5 Volts, of which we will use the 5 Volt option. At a 5 Volt supply, it is rated to have a rail-to-rail operation. Its on-state resistance is typically 2.5 Ohms and its on-state current leakage is typically 0.01 nano-Amps. Its on-state resistance stays fairly constant over the entire range of analog inputs, with a variation of 0.75 Ohms typically. The typical charge injection is 3 pico-Coulombs and its channel-to-channel crosstalk is -82 dB typically.

MAX4632 – Maxim Integrated Products
This device is a 4:1 CMOS analog multiplexer that comes in a tiny 10 pin, TDFN package. It is capable of running off a single, 5 Volt supply and has a typical on-state resistance of 2.5 Ohms. This on-state resistance stays fairly constant over the entire range of analog inputs, with a variation of 0.75 Ohms typically. The on-state current leakage is typically 0.01 nano-Amps and the charge injection is 2 pico-Coulombs. Its crosstalk is -52 dB. Besides a slight advantage in charge

injection and a disadvantage in crosstalk, these two candidates are virtually the same.

Part	Manufacturer	On-State Resistance	Current Leakage	Voltage Supply	Charge Injection	Crosstalk	Price
ADG704	Analog Devices	2.5 Ω	0.01 nA	5 V	3 pC	-83 dB	$2.84
MAX4632	Maxim Integrated Products	2.5 Ω	0.01 nA	5 V	2 pC	-52 dB	$3.03

Table 12: Temp Sensing Multiplexer Candidates

Because of its lower cost, crosstalk and desired package type, we will choose the ADG704 by Analog Devices.

Cold Junction Multiplexer

The cold junction multiplexer will need to be a 2:1 circuit; one input will be the RTD amplification circuit output and the other input will be the reference voltage.

74LVC1G53 – NXP Semiconductors
This device is a Si-gate CMOS, analog multiplexer/decoder that comes in a small 8 pin, TSSOP package. It has an on-state resistance of 6.2 Ohms, a large unwanted current leakage of 0.1 micro-Amps and a charge injection of 7.5 pico-Coulombs. The datasheet recommends the device be supplied by a 5.5 Volt supply. The channel-to-channel crosstalk rating is not specified.

AD8180 – Analog Devices
This multiplexer can be supplied by dual supply of +/- 5 Volts and comes in an 8 pin, SOIC package. Its crosstalk is rated at -80 dB and has a rating of total harmonic distortion of -78 dBc, typically. The datasheet specifies and input voltage offset of 1 milli-Volt typically but can have a maximum of 12 milli-Volts, which is huge. This device seems to be commercialized for fast switching speed and not necessarily high DC accuracy.

Part	Manufacturer	On-State Ω	Current Leakage	Voltage Supply	Charge Injection	X-talk	Price
74LVC1G53	NXP Semiconductors	6.2 Ω	0.1 μA	5.5 V	7.5 pC	Unknown	$0.48
AD8180	Analog Devices	2.2 MΩ	1 μA	+/- 5 V	Unknown	-80 dB	$3.84

Table 13: Cold Junction Multiplexer Candidates

Because of its low current leakage, known and acceptable charge injection and ability to be powered by a single supply, we will choose the 74LVC1G53 by NXP Semiconductors.

Microcontroller Peripheral Decoder

The microcontroller peripheral decoder will need to be a 1:8 circuit; the peripherals controlled will be the following: Ethernet controller, EEPROM, touch screen driver, two ADCs, and a slave device. This device will be connected to the peripheral's $\overline{CS}$ pins and will therefore need an 8-bit inverter in between. There will be 2 extra outputs.

HEF4051B – NXP Semiconductors
This device contains bidirectional analog switches, allowing it to function as both a multiplexer and decoder. Its supply voltage will need to be dual but is flexible from the range of 3 to 15 Volts. Its on-state resistance is rated typically at 350 Ohms at a +/-5 Volt dual supply but is rated at maximum to be 2500 Ohms. This part has a leakage current of 0.1 micro-Amps. It has a -50 dB crosstalk between switches but does not specify its charge injection. This device comes in either a 16 pin, SSOP package or 16 pin, DIP package.

SN74CBT3251 – Texas Instruments
This FET demultiplexer device has its best ratings at a voltage supply of 4.5 Volts, therefore we will discuss specifications at this operating voltage. Its maximum on-state resistance is at 7 Ohms. The typical current leakage of this device is 1 micro-Amp. The datasheet does not specify the device's charge injection or channel crosstalk. This device comes in a 16 pin, SSOP package.

CBT3251 – NXP Semiconductors
This FET demultiplexer has a flexible range of supply voltages, but will allow us to use 5 Volts. Typically, at a 5 Volt supply, the on-state resistance is 5 Ohms. The typical current leakage for this part is 1 micro-Amp. The datasheet does not specify the device's charge injection or channel crosstalk. This device comes in a 16 pin, SSOP package.

Part	Manufacturer	On-State Resistance	Current Leakage	Voltage Supply	Crosstalk	Price
HEF4051B	NXP Semiconductors	350 Ω	0.1 μA	5 V	-50 dB	$0.55
SN74CBT3251	Texas Instruments	7 Ω	1 μA	4.5 V	Unknown	$0.74
CBT3251	NXP Semiconductors	5 Ω	1 μA	5 V	Unknown	$0.60

Table 14: Microcontroller Peripheral Decoder Candidates

Because of its low on-state resistance and current leakage, we will choose the CBT3251 by NXP Semiconductors.

8-bit Inverter

CD74ACT14 – Texas Instruments
This device is being considered by it is an existing part in a similar technology, the Watlow Process Controller, of which Infrared Systems Development currently uses. This 8-bit inverter is needed to invert the high signal sent from the microcontroller to the $\overline{CS}$ pins on all the SPI peripherals. This device contains 6 independent inverters, each having its own input threshold levels. They are all Schmitt triggered circuits; supplied by a 5.5 Volt maximum input, but we will discuss its ratings at a 5 Volt supply. With this supply voltage and at ambient temperature, the positive threshold is 2 Volts maximum and the negative threshold is 1.3 Volts minimum. This part comes in a 14 pin, SOIC or PDIP package.

Charge Injection Problem

Since we are dealing with analog data, it is very crucial to not introduce any noise into our circuit. Charge injection is the biggest problem with multiplexers for our project. The voltage spike that is produced when the multiplexer switches inputs can be interpreted as a jump in temperature in our data if not properly taken care of. There are three potential solutions to this issue which we are going to pursue: designing a filtering circuit, developing a timing schema to turn on and off the analog to digital converter at specific times or to configure the converter's reading capability in a manner that will ignore the charge injection. The latter will be discussed in the analog to digital converter section.

To develop a filtering circuit would be a little complex. First, we would need to use a chopper to turn our nearly DC signal into an AC signal of known frequency. Then, we would have to derive the frequency at which the maximum charge injection noise occurs. Next, we would need to search for a low pass filter that meets our filtering criteria and implement it in a way that fits both our multiplexer and analog to digital converter. To test the circuit, we could use an oscilloscope to view data from our temperature sensors in the frequency domain. Plotting with a secondary probe, we could also view the output from the multiplexer or decoder in the frequency domain and pay close attention at the times when we switch inputs on the multiplexer. This approach seems like it could fault in many aspects of design and testing and will only be considered as a last resort option.
To design a timing schema would also be slightly difficult, however; this would at least allow for a simpler approach to the circuitry. When we send a signal to the multiplexer and decoder to switch inputs, we could send a signal at the same time to the analog to digital converter to temporarily turn off. The graph below shows an example of what our timing schema would look like, where the spikes in the data are the induced charge injection.

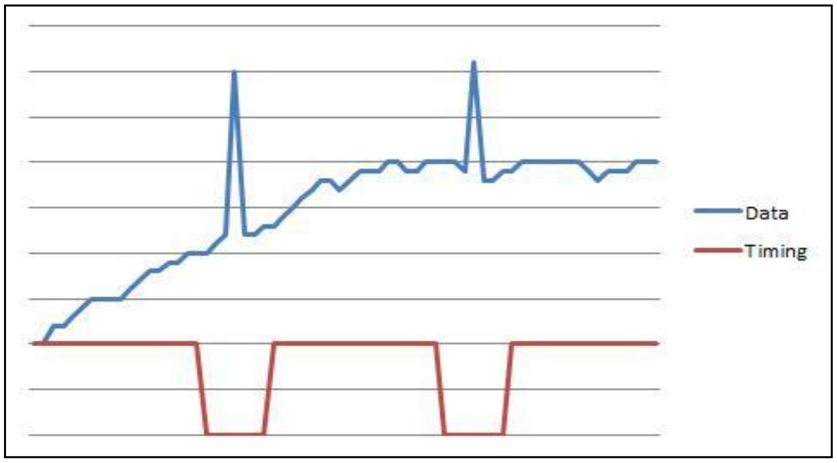


Figure 27: Possible Charge Injection Timing Schematic

To implement this, we could disable the chip enable line to the analog to digital converter. The last option would be to take multiple readings from the analog to digital converter and disregard the appropriate reading containing the charge injection. We will discuss this option when we talk about the converter itself.

Analog to Digital Converters

Our choice in analog to digital converter is crucial for the reliability of our data. Number or bits, sampling rate, DC accuracy and signal-to-noise ratio will be the main criteria for us to select an ADC. The internal voltage reference, if appropriate, will also be examined and a more precise external reference will be considered if necessary. A few different methods of conversion will be considered including flash ADC, successive approximation and sigma delta ADC. SNR and DC accuracy will be the most critical specifications in our discussion. We must also consider the data interface type and make sure that it is as universal as possible, to be flexible with our choice of microcontroller. Our interface choice will most likely be SPI because of its ease of use. The other known limitation is that the converter cannot be any less than 14 bits resolution. Since our temperature range is from -30 to 1400 degrees Celsius, that would give us 14,310 discrete measurement points to the accuracy of 0.1 degrees; thus requiring at least 14 bits or resolution. Although 14 bits is our minimum, we will be interfacing with a 16 bit microcontroller so for ease of use, a 16 bit ADC will be preferred. We will use two analog to digital converters: 1 for the cold junction voltage and reference voltage reading, and 1 for the thermocouple or source RTD inputs. The input for the present reading will be selected via a multiplexer. We will need to look for converters with bipolar inputs, since our temperature range goes into negative values.

Flash ADC

Flash analog to digital conversions are very fast, with multiple parallel comparators. To realize an N bit converter, 2^(N-1) comparators would need to be used in the converter's architecture. The maximum number of bits most flash ADC devices are sold in is usually 8. Since our project will require higher resolution than 8 bits, we will not consider any flash converters for our controller.

Successive Approximation

Successive approximation converters will be great candidates for our project because of their high accuracy and nearly linear responses. This high accuracy comes from a sample and hold internal circuit which holds the current value of the analog signal until the conversion is complete. Also contributing to this accuracy, is the notion that each sample is compared to the output of a DAC which has been set by the previous sample's compared value. The successive approximation register can be large, thus meeting our criteria of a minimum 14 bits of resolution. The cost of this high accuracy is conversion time, since the time is based off the number of times the sample is compared and stored in the register; which is the number of bits. When considering a successive approximation ADC, we need to take into account the pipeline delay associated with these converters and the internal DAC. The internal DAC specifications will be thoroughly researched before selecting a candidate of this type.

Sigma Delta ADC

A sigma delta analog to digital converter has 2 main parts of its design: an analog modulator and digital filter. The analog modulator uses a summing integrator, comparator, D flip-flop and a 1 bit DAC as feedback to the summing integrator. Depending on the amount of summing integrators, depends on the order of the converter; higher order systems have more precision. The sigma delta ADC uses an oversampling technique dependent on the clock rate of the internal D flip-flop. This oversampling rate is also a factor of precision. In the digital filter, there is a low pass filter that helps increase SNR. Because of the oversampling technique, all the quantization error is bound to the upper frequencies; which can be easily filtered out. The digital filter also reduces the D flip-flop's clock rate so that the output is the sampling rate of the ADC.

Candidates

AD7715 – Analog Devices
This device is being considered because it is a part of a similar existing technology, the Watlow Process Controller, of which Infrared Systems Development currently uses. The AD7715 is a 16-bit, sigma delta ADC specifically made for low frequency measurement applications, which is perfect for our DC inputs. This converter also is compatible with low level signals at the input because it contains its own programmable gain amplifier, with programmable gains of 1, 2, 32 and 128. The sampling rate of this device depends on the gain chosen and the clock rate. This device has on chip registers, allowing for external control of key parameters such as input gain and signal polarity. The chip has 16 pins and comes in PDIP and SOIC_W (wide) packages.

The internal gain amplifier of this device may actually eliminate our need for an external amplification system completely, thus simplifying our circuitry by many parts. According to our calculations in the amplification section above, we will need to a unity gain. The output noise associated with this converter depends on the master clock rate we choose and the adjustable front end gain. Depending on these selection criteria, the noise can range from 0.9 micro-Volts to 550 micro-Volts. The datasheet does not specify a voltage drift amount from the amplifier, but it does note that it is low because of chopper stabilization, which we determined in the amplifier section above to be preferred.

AD7641 – Analog Devices
This device is relatively new to Analog Device’s product line, therefore making it a great candidate for product longevity. It is a successive approximation type converter although one of its features it boasts is a “no pipeline” delay. It also has a high typical throughput rate of 1.5 mega samples per second and an 18 bit resolution. It has a decent signal-to-noise ratio of 95.5 dB and its internal DAC is a charge redistribution converter. The internal reference voltage is set a 2.048 Volts, and has a temperature coefficient of 10ppm/°C. This reference is too small to compensate for our maximum 4.5 Volt input, so we will need an external reference. Since we will most likely be using a 16 bit microcontroller, we will need to compensate for our 18 bit resolution somehow. These compensation methods are discussed in the input section of the microcontroller research. This device has 48 pins, most of which we will not use; especially the parallel interface pins which we will have to ground. This layout will make for complicated PCB circuitry later on. This chip also offers different interfacing modes for 18, 16, and 8 bit parallel outputs and one mode for serial output. In our PCB design, we will need to hardcode the mode select pins to only use the serial interface. The chip comes in an LFQP package. This device seems a little “over-kill” for our application; therefore, if the budget becomes an issue, we will not consider this device.

AD7684 – Analog Devices
This device is a successive approximation ADC with 16 bits of resolution and a high throughput rate of 100 kilo-samples per second. With a supply and reference at 5 Volts, the SNR plus distortion is a decent 91 dB. Its internal reference has a temperature coefficient of 0.3 ppm/°C. The device has no internal reference, therefore we will need to choose our own. The internal DAC is a charge redistribution converter, which is preferred. It comes in a small, 8 pin MSOP package and has an SPI interface.

ADS8326 – Texas Instruments
This ADC is the successive approximation type with 16 bits of resolution. It has a small offset error of 1 mV and a high throughput rate of 250 kilo-samples per second. It has a decent SNR of 91.5 dB and a SNR plus distortion at 91 dB. The device requires an external reference and clock, which may be preferred so we can set the accuracy of the voltage reference. This device comes in a small 8 pin, MSOP package and has an SPI interface.

Part	Manufacturer	ADC Type	# of bits	Throughput Rate	Voltage Supply	Price
AD7715	Analog Devices	Sigma Delta	16	Depends	3 V or 5 V	$12.68
AD7641	Analog Devices	Successive Approximation	18	1.5 Msps	2.5 V	$51.90
AD7684	Analog Devices	Successive Approximation	16	100 ksps	2.7 ~ 5.5 V	$13.83
ADS8326	Texas Instruments	Successive Approximation	16	250 ksps	2.7 ~ 5.5 V	$8.91

Table 15: ADC Candidates

Because of its low offset error, high SNR and high throughput rate, we will choose the ADS8326 by Texas Instruments as our analog to digital converter. It also meets our standard requirements of 16 bit resolution and the ability to be powered by a 5 Volt supply.

Voltage Reference

We will need 3 voltage references in our temperature sensing subsystem: 1 for our RTD amplification system and 2 for our analog to digital converters. When choosing a voltage reference, we need to be cautious of 2 major specifications: the temperature coefficient and initial accuracy. The temperature coefficient can be calculated by

$$T_C = \frac{(e - A)}{\Delta T}$$

where e is the error percent, A is the initial accuracy at 25 degrees Celsius and ΔT is the extreme operating temperature different from 25 degrees Celsius. From this equation, we can select a device with a known initial accuracy and determine an appropriate temperature coefficient depending on our operating temperature range and error allowance.

RTD Amplification Reference

The precision of this voltage reference is not as critical as the references for the ADCs; however, we still want to minimize the error if possible. Let’s assume that our maximum operating temperature will be 40 degrees Celsius (again, an over-estimate); this is 15 degrees away from ambient temperature. The error percent we are willing to afford will be 0.2%; therefore, our allowable temperature coefficient will depend on our initial accuracy by

$$T_C \leq \frac{1}{75} - \frac{A}{15}$$

So we shall consider components where the specs fall within allowable ranges of this equation. We had previously decided in our amplifier design section that we wanted a V_{ref} of 4.5 Volts.

ADC External Reference

In order to maintain the accuracy of our analog to digital converters, we need an extremely high precision voltage reference. The performance of this voltage reference must surpass the ADC’s internal reference performance in order to justify consuming more power on an external reference. Like the RTD amplification reference, we will look at references that have a low error percentage and have a tolerable temperature coefficient and initial accuracy. Since we have narrowed our choices of converters to a 16 bit successive approximation converter, we can calculate the amount of voltage per resolution bit given from a chosen reference voltage. In order to allow the input of our ADC to match the maximum amplifier output we chose of 4.5 Volts, we need to seek a standard voltage reference of 5 Volts, this would give us

$$\frac{5\,V}{2^{16}} = 76.3\ \mu V$$

for each bit of resolution. Because the accuracy of the analog to digital conversion is one of the most crucial components of our system, we will expect an error of no more than 0.5LSB.

Candidates

REF5045 – Texas Instruments

This bandgap voltage reference fits our allowable range for temperature drift and initial accuracy: its temperature coefficient is 3 ppm/°C and its initial accuracy is 0.05%, both maximum values. These specifications give us an error percent of

0.055%, which is way below our allowance. This 8 pin package is available in a MSOP footprint. From V_{out} to ground, there must be an output capacitor ranging between 1 and 50 micro-Farands for the output to be stable. A bypass capacitor is recommended by the datasheet between V_{in} and ground. The power supply required for this reference is between 4.7 and 18 Volts. We will be able to determine whether or not our controller can supply the maximum voltage in the power design section of the paper.

MAX6161 – Maxim Integrated Products
This device comes in an SO 8 pin package, although there are only 3 usable pins: V_{in}, V_{out} and ground. The maximum temperature coefficient is 5 ppm/°C and the initial accuracy is 2 milli-Volts maximum, which equals 0.044%. This gives us an error percentage of 0.049%, even better than the previous candidate. This datasheet also recommends a bypass capacitor on the input but does not require an output capacitance for stability. The absence of this output capacitor could help us save board space. This device can operate on a supply voltage range of 4.7 to 12.6 Volts, also giving it another advantage over the previous candidate.

Part	Manufacturer	Supply Voltage	Error Percent	Price
REF5045	Texas Instruments	4.7 ~ 18 V	0.055%	$4.46
MAX6161	Maxim Integrated Products	4.7 ~ 12.6 V	0.049%	$3.24

Table 16: Voltage Reference Candidates

Because of our budget allowance given by Texas Instruments, we will choose the REF5045 as our voltage reference device.

Solid State Relays

For our project, we have decided to use solid state relays versus generic mechanical relays because of many advantages. These advantages include but are not limited to: little switching noise, no bouncing, increased lifetime and a much smaller package. There will be 3 relays used in our design; 2 to control power to the heater and 1 to control inputs to the alarms. The heater will need an AC relay for the cavity sources and a DC relay for the extended area sources; each where we can adjust the power level and signal duration. The alarm relay will need a DC relay with an output of about 5 Volts, 1 Amp and an input of 3 Volts or less. The inputs of the relays will come from the microcontroller.

Heater Relays

DC – Extended Area Sources
The heaters for Infrared Systems Development's extended area sources are DC powered. This relay will need to control the power level and signal duration going into the heater. For the coupling of our signals, we will use an optical coupling system. We will use an LED to stimulate a photo-sensitive diode as our coupling

device and can choose from a MOSFET, thyristor or controller rectifier to switch the load. When we deal with DC signals, we will also use MOSFETs and a TRIAC to conduct the load current to the load.

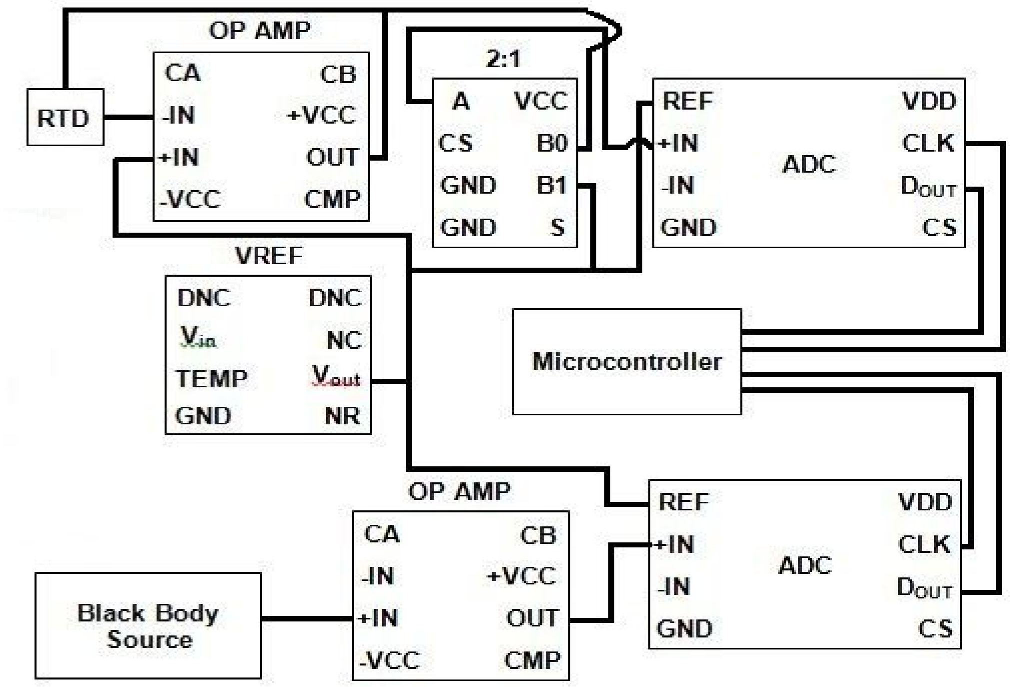
AC – Cavity Sources
The heaters for Infrared Systems Development’s cavity sources are AC powered. One of the requirements we need to meet for our device is that we need to have zero cross sensing at the AC relay, which will ensure that we turn off the power to the heater when the voltage crosses from positive to negative. This will ensure that we eliminate or lower the electromagnetic interference that would be generated when switching at a nonzero point. There are two different ways we could go about doing this. We can either design a logic circuit to implement the zero cross sensing or we can find a chip that will interface with the microcontroller. The issue with designing a logic circuit is that it will take up more space and cost more because we would have to buy more components. Some AC relays come with a zero crossing detector integrated into it, which would be a good option because it will cut cost and use less space.

Alarm Relays

We also need DC relays for the alarm outputs. If the relay is closed it will allow the user to add external circuitry that will be triggered once the alarm goes high. If the relay is open it will deactivate the circuit. We will need to have two sets of normally-open and normally-closed relay contacts, so we will use double pole single throw (DPST) relays. A DPST relay is a relay that has two pairs of terminals actuated by a single coil. We will need two of these relays, one DPST (2 Form A) and one DPST (2 Form B). 2 Form A refers to two normally-open contacts which connect the circuit when the relay is activated, and 2 Form B refers to two normally-closed contacts which disconnect the circuit when the relay is activated. By using one of each of these relays we will meet our requirements of having two sets of normally-open and normally-closed contacts. Since we are using a 5 volt input voltage we will use a relay with a 5VDC coil voltage. To minimize cost and the amount of parts, we want a relay that has a diode in it, the diode is used to keep the voltage on the coil leads from going negative. Since the microcontroller can’t produce enough current to drive the relay we need to use a FET to drive our relays, there are other ways to drive the relays but this is a good option because it doesn’t take up as much space as a transistor would. Since our display will include the alarm status lights we will not need to integrate a separate LED into this design. The alarm status lights will be controlled by our microcontroller. TE Connectivity makes many different DPST relays in both Form A and Form B. Until we know for sure exactly how much current we need we will consider relays with a current rating of 2A, which should be enough for our alarm outputs.

Temperature Sensing Subsystem Design

Shown in the image below, is our temperature sensing subsystem design. As you can see from the schematic, the inputs to our system will come from Infrared Systems Development's source (with the exception of the cold junction signal). The output of our system will also end at the source to control the heater and the fan. The analyzing of this data will all be done within the microcontroller, which will also command most of the electronic devices in this subsystem.



This schematic is assuming all grounds and voltage supplies are connected. The CS lines will all be controlled by the microcontroller which depends on a 1:8 decoder and 8-bit inverter to select the devices. Any address lines will also be controlled by the processor and the NC and DNC pins will not be connected.

Printed Circuit Board Fabrication

The final design for our temperature controller is shown in the final design section below. As anyone can see by our block diagram, our design contains many ICs and in all variety of package types. We will need a versatile, but relatively compact board to maintain a small chassis size for our controller. Since no one in our group has lengthy experience in designing PCB layouts, the ease of CADing software use and understanding will play a big role in the decision of our PCB manufacturer. Our PCB will have a maximum of 2 layers for the testing phases of our design. When the controller is ready to be produced and sold, the option for more than 2 layers could be considered. We want our PCB manufactured at the highest and latest fabrication standard of IPC 6012 Class III. Our ideal board size would be about 4 by 4 inches, with a thickness of around 0.062 inches. We would like our finished copper weight to be the standard 1 ounce since we are not distributing large amounts of current through our board. If we have room in the budget, a white silkscreen and green solder mask on our board would add a professional touch and create more isolation between traces. Our board also has to be easily mounted to the controller chassis. The following components will take space on our PCB:

Device	Pins	Package Type
Amplifier(s)	8	SO
4:1 MUX	10	SOIC
2:1 MUX	8	TSSOP
1:8 Decoder	16	SSOP
8-bit inverter	14	SOIC
ADC(s)	8	MSOP
Reference(s)	8	MSOP
Ethernet Controller	28	SOIC
IEEE – 488	16	SOIC
EEPROM	8	SOIC
Touch Controller	16	TSSOP

Table 17: PCB Components

Candidates

ExpressPCB

ExpressPCB offers free PCB layout software downloadable from their website. We've already downloaded their PCB layout software and have experimented with their user interface. The click and drag approach implemented by their program is very intuitive and quick to use. Their software package also comes with a schematic designing program which you can link to their PCB layout program; linking the two programs will highlight the pins that need to be connected in the PCB layout according to the schematic. The software even comes with an instant quote that will give us the exact manufacturing price and estimated time of shipping. Orders under 500 boards (which will include our 2 board order) will be shipped the following business day; therefore, we should not have to wait long after we order our board to get the prototype. This company is preferred over the other candidates because of the easy integration between schematics, PCB layouts and quotes. The cost and time analysis for this manufacturer is shown in the table below.

EAGLE PCB

EAGLE PCB does not offer free software, but is highly recommended for the paid professional version of their PCB layout software. Our sponsor, Infrared Systems Development, has a license for this software, should we choose to use it for our design. If we choose to purchase the software, a "light" version is available for an economical $69.00. EAGLE PCB does not fabricate their own PCBs, they only provide the CAD tools needed for layout design and recommend partner vendors for actual fabrication. Sunstone Circuits are one of EAGLE PCB's fabrication partners and will accept any generic Gerber file for processing.

Sunstone Circuits
Sunstone Circuits offers free PCB layout software downloadable from their website, called PCB123. Their user interface is similar to that of Express PCB, where they have a user friendly GUI. They also offer a schematic designing program which can be linked to the PCB layout; whenever an update is made to the schematic, the PCB layout automatically adjusts and vice versa. The company boasts a large parts library of over 500,000 new parts and also has a unique feature that allows users to search Digikey in real time. The software outputs both Gerber and Excellon files which will allow us to design the layout in the software, if desired, but send the board to get fabricated elsewhere. This would be beneficial to us since the cost of 2 small boards with this manufacturer will cost of $100. The cost and time analysis for this manufacturer is shown in the table below.

PCB Fabrication
PCB Fabrication is an online service that allows you to upload your Gerber files and then customize your PCB's specifications to order for fabrication. This company has allowed for the most flexibility and options when it comes to choosing a board. To request a quote, the form lets you choose: board material, fabrication standard, the routing width, finished copper weight, hole size, silkscreen, solder mask, and board surface finish, just to name a few of the many options PCB Fabrication allows users to choose. I cannot view standard prices without submitting an actual request for a quote.

Manufacturer	Software Price	Board Price	Shipping Cost	Turnaround
ExpressPCB	Free	~$85.40	$9.85	2 days
EAGLE PCB	$69.00	NA	NA	NA
Sunstone Circuits	Free	~121.00	Free	2 weeks
PCB Fabrication	NA	Unknown	Unknown	Unknown

Table 18: PCB Fabricator Candidates

Because of their affordable pricing and quick turnaround, we will go with ExpressPCB. We will use their software to design our PCB layout and also use ExpressPCB to fabricate the board itself.

Display Design

Overview

The display of the temperature controller will be the face of the whole device. Ideally the display should look sleek and high-tech but still be user friendly. The display will need to show the actual temperature of the device at all times. It will also need to provide a series of menus and user settings that are critical to the specific user. Our display will be a touch screen display. In addition, the display must interface with the microcontroller and handle other tasks.

Requirements

The display must have the ability to show the actual temperature of the device. This must be displayed in large, readable numbers that can be viewed from at least ten feet away. The display must show the user the temperature with two digits of precision along with the scale of degrees (.XX °C). The display must also be able to show negative temperatures and have a selectable time constant to smooth the value and reduce flicker. In addition to the actual temperature of the device, it must also show a series of menus and values that are specific to the user. These menus and options will be discussed in more detail in the User Interface section. We are going to use a touch screen for our display. The display we choose needs to be compatible with the microcontroller that we choose. Ideally the display and its associated components must be in production for at least ten years after the completion date of the device. We have size limitations when it comes to our display. Ideally the display we choose should fit in the existing chassis but if modifications are needed then we can work with it. The space we have for the display is 3.9” x 3.9”, the height of the display must not exceed this but the length can be longer if needed. The size will be discussed more in the design section once we know which display we are using.

Display Research

The following subsections will discuss and evaluate various features of the touch screen display such as graphics controllers, interfacing, touch screen options, LED backlight driver, candidates, touch screen controller and size requirements.

Graphics Controllers

When designing a touch screen display we need to decide on which graphics controller we are going to use. There are four main ways to integrate a graphics controller into the design. Out of those four ways only three of them are available to us because of our microcontroller candidates. We are considering the option of external graphics controller not found in glass, external graphics controller and integrated graphics controller.

External Graphics Controller

An external graphics controller is a configuration of two devices. It consists of the microcontroller and the display module that has an onboard graphics controller in itself. This means the frame buffer, display controller and display are all on one device. This option is cost effective and convenient but there are disadvantages to having it all one on device. If the display module is no longer in production then another solution would have to be developed, making this option not a long lasting solution. From our research not many displays come with an onboard graphics controller. This option would be risky for us because we want something that is going to have a long production lifetime and if the display we choose goes out of production we would have to reconfigure our whole design by either getting a new microcontroller or integrating a separate graphics controller. The photo below shows an example of an external graphics controller.

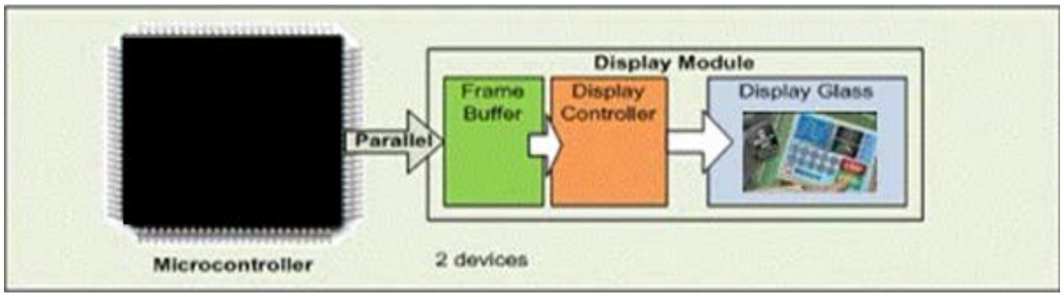


Figure 28 - Example of an External Graphics Controller

External Graphics Controller not Found in Glass

This type of external graphics controller is a configuration of three devices. It consists of the microcontroller, the graphic controller chip and the display module. The display module itself doesn’t contain any onboard graphics controller which means an extra controller needs to be purchased and connected. This option will add an extra cost to our design but the advantage of that is if our display breaks or goes out of production we would only need to purchase a new display. While this option takes up more space and cost more it could be beneficial in the long wrong if we need to make modifications to our design. The photo below shows an example of an external graphics controller not found in glass.

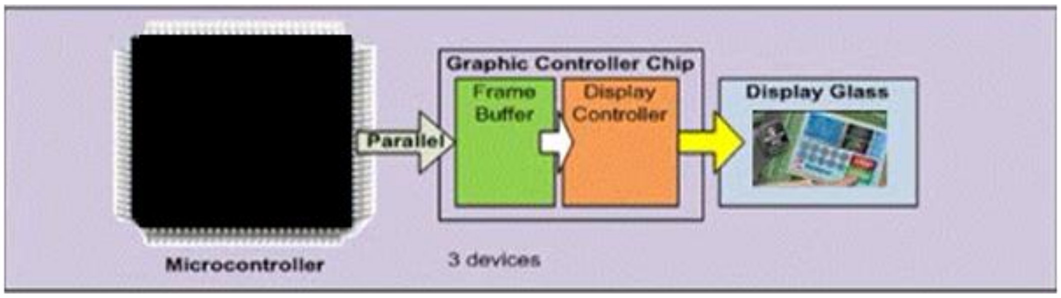


Figure 29 - Example of an External Graphics Controller not Found in Glass

Integrated Graphics Controller

With an integrated graphics controller the microcontroller has a built in graphics module to help drive graphical features. This option is limited to specific microcontrollers but if the microcontroller we pick has this feature then it would be a good choice because it will cut cost and space. The other advantage of this is that a lot of displays don't come with onboard graphics controllers so when choosing a display we have a larger selection to choose from. The photo below shows an example of an integrated graphics controller.

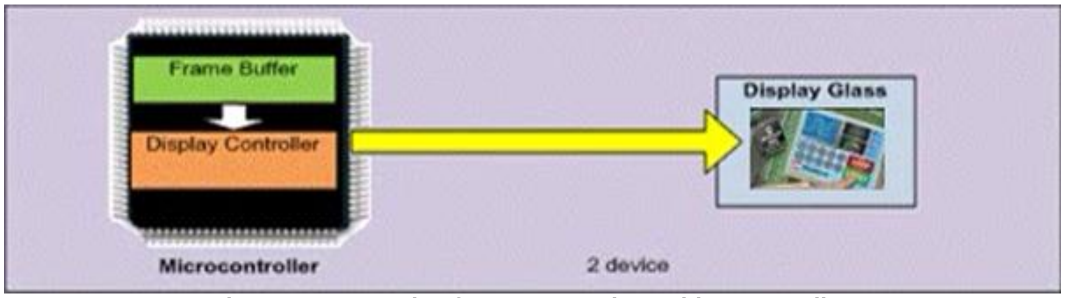


Figure 30: Example of an Integrated Graphics Controller

Interface

Parallel Interface

To interface the display with the microcontroller we have two options. The first option is to use parallel data bus interface. Depending on the display we pick we could use anywhere from 8-bit parallel to 24-bit RGB parallel. If we choose to use the parallel bus it would allow us to quickly refresh the screen, which we don't necessarily need. This option is a more costly and complicated set up because of its complex circuitry and the use of more I/O ports on the microcontroller. The availability of this option with displays is everywhere so depending on what we can find this might be our only option.

Serial Peripheral Interface (SPI)

Our second option is to use SPI to interface the display with the microcontroller. We could use either a three or four wire SPI. This option is slower than the parallel interface but in return it would require only three to four pins, which is beneficial to us. We would want to use a four wire SPI instead of the three wire because the four wire allows you to select between registers and data. Unlike the parallel interface, this option is less available in touch screen displays.

Four Wire Resistive Touch Screen

A four wire resistive touch screen consists of two transparent resistive layers lying parallel to each other separated by a thin space. When the user touches the display the two layers get pressed together and that spot behaves as a pair of voltage dividers. The transparent screen also has four wires coming out of it for its interfacing. These four wires are also used to calculate the coordinates on the touch screen where the user touched. The four wires are labeled X-, Y-, X+ and Y+ and are used to directly connect the touch screen signals. X+ and Y+ are both analog and digital signals while X- and Y- are digital output signals. There is an applied voltage to each side of layer 1 of while layer 2 senses the proportion of voltage at the point the user pushed. This will give the horizontal coordinate. Then there is a applied voltage to the top and bottom of layer 2 and layer 1 will sense the proportion of voltage at that point. This will give the vertical coordinate. The X+ and Y- signals are multiplexed with the touch screen controller. Once we decide on a display we will choose which touch screen controller we will use. This information is then sent to the microcontroller. A four wire resistive touch screen is cost effective and widely available. It can be touched with a finger, stylus or glove and isn't affected by dirt, dust or water.

Capacitive Touch Screen

A capacitive touch screen consists of a layer of glass coated with a transparent conductor. A capacitive touch screen works by sensing a change in the amount of charge on the capacitive layer. When the user touches the display some of the charge is transferred to the user and the charge on the layer decreases. One type of capacitive touch screen detecting is surface capacitance. With surface capacitance only one side of the glass is coated and a small voltage is applied to the layer which results in a uniform electrostatic field. When the user touches the surface it distorts the electrostatic field by drawing current form each side proportionally. The displays controller can determine the location from the change in the capacitance. Capacitive touch screens are more expensive and aren't widely available. The downside of the capacitive touch screens is that it can only be touched with a finger or special type of stylus. Capacitive touch screens support multi-touch but we do not need that feature in our device.

LED Backlight

The LED backlight of the display needs to be driven in some way in order for it to light up. There are three different ways to do this. Since LEDs are current devices a current source could be used to drive the LEDs. This approach is very complex. Another way is to add a current limiting resistor in line from the voltage source. This approach is simple and would work but it takes a lot of voltage to do this. In the displays that we have seen the backlight needs around 20V to power the LEDs. The last option is to use a LED driver. This option allows you to drive the LEDs with a smaller input voltage. Sometimes there is an extra cost to this but it small. This is the best option for our design since our max voltage is around 5V. There are some displays that will come with an integrated driver already on it but most of them do not. Using a LED driver will also allow us to control dimming of the backlight as well. To do this we would use a filtered PWM signal. The PWM signal is filtered by an RC network and fed to the driver. If we choose to do this the microcontroller must support PWM. There are many different LED drivers and each of them have different configurations depending on the number of

LEDs in the backlight, the voltage and current. Most of them use configurations consisting of resistors, inductors and capacitors that get connected to the driver. Once we make the final decision on the display then we will decide on what driver to use depending on if the display has one integrated or not.

Display Candidates

Crystalfontz CFAF320240F-035T-TS

The CFAF320240F-035T-TS is a 320xRGBx240, full-color QVGA graphic TFT display module. It has a 3.5 inch diagonal screen with a white LED backlight. The display is white edge-lit with two parallel rows of LEDs, three LEDs in each row, six LEDs total. The white LED backlight has the anode and cathode pins brought out on the flexible printed circuit (FPC) connector. The CFAF320240F-035T-TS does not come with an integrated LED driver so if we choose this display then we would have to buy a driver that can support all the displays features. The features to consider for the driver would be the input voltage, the number LED strings, the number of LEDs in each string, and the LED voltage. Depending on our microcontroller we could dim the backlight of this display using a filtered PWM signal. The CFAF320240F-035T-TS has 50 pins for interfacing brought out on the FPC and can be connected with standard .5mm ZIF sockets. Crystalfontz recommends the HFJ150CT-ND or the HFK150CT-ND. This display has a 4-wire resistive touch panel with an integrated Solomon SSD2119 display controller. Since this display has the integrated display controller we could use a microcontroller without a graphics controller on it. The issue with this is that if the display module goes out of production then we would have to either get a new microcontroller or get a separate graphics controller which will waste time and money. We would also need to get a 1mm pitch touch panel connector for the 4-wires of the display. Crystalfontz recommends either the 609-1883-1-ND or the 609-1886-1-ND. One advantage of the CFAF320240F-035T-TS is that it has SPI interface as well as 18/16/9/8-bit parallel interface. It also has a wide temperature range of -20°C to +70°C, which is needed since we are dealing with such intense temperatures. It has a 12:00 o’clock viewing angle which is a top viewing angle like what you would see if looking down at the speedometer of a car. The CFAF320240F-035T-TS has logic supply voltage of +2.5V to +4.0V and a supply voltage for I/O signals of +1.4V to +3.6V. This display has a module reliability of 50,000 hours > 50% of initial brightness when operated and stored specification limitations.

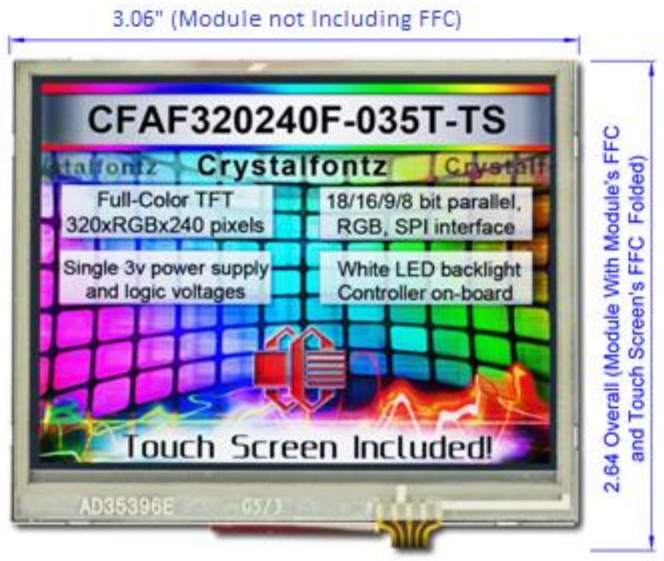

3.06" (Module not Including FFC)
2.64 Overall (Module With Module's FFC and Touch Screen's FFC Folded)
CFAF320240F-035T-TS
Crystalfontz
Full-Color TFT 320xRGBx240 pixels
18/16/9/8 bit parallel, RGB, SPI interface
Single 3v power supply and logic voltages
White LED backlight Controller on-board
Touch Screen Included!
AD35396E

Figure 31: Crystalfontz CFAF320240F-035T-TS - Modified to Appear as the Eye Perceives the Display (showing no FPC connectors)

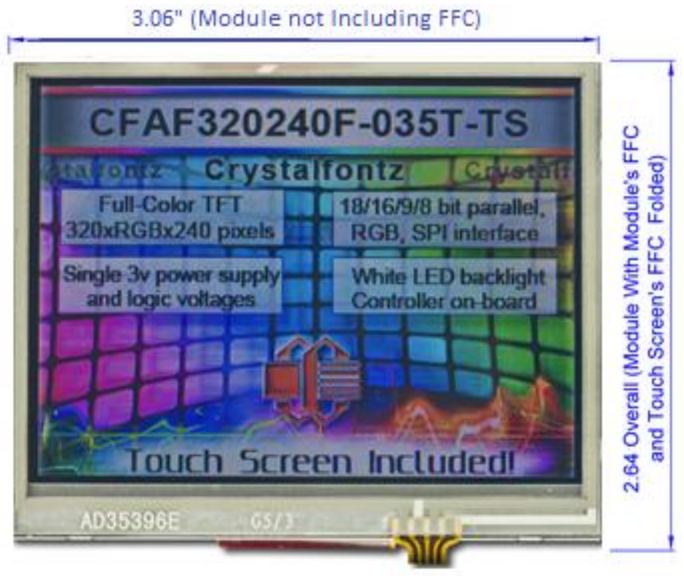

3.06" (Module not Including FFC)
2.64 Overall (Module With Module's FFC and Touch Screen's FFC Folded)
CFAF320240F-035T-TS
Crystalfontz
Full-Color TFT 320xRGBx240 pixels
18/16/9/8 bit parallel, RGB, SPI interface
Single 3v power supply and logic voltages
White LED backlight Controller on-board
Touch Screen Included!
AD35396E

Figure 32: Crystalfontz CFAF320240F-035T-TS - Unmodified, Appears as the Camera Perceives the Display (showing no FPC connectors)

Newhaven NHD-4.3-480272MF-ATXI#-T-1

The NHD-4.3-480272MF-ATXI#-T-1 is a 480xRGBx272, full color TFT display module. It has a 4.3 inch diagonal screen with a white LED backlight. This display has two parallel rows of LEDs, six LEDs in each row, twelve LEDs total. The NHD-4.3-480272MF-ATXI#-T-1 has an Orise OTA5180A LED driver integrated into the display. This can be an advantage from a cost and space standpoint but also a disadvantage if the driver doesn't have all of the features we need. Since it has more LEDs it is going to take more power and the OTA5180A has the feature of using the filtered PWM signal. This display has a 4-

wire resistive touch panel but does not have an integrated graphics controller. This gives us the option to either get a separate graphics controller or get a microcontroller with an integrated graphics controller on it. This is a benefit because we have more options when it comes to designing it the way we want. If we choose to get a separate graphics controller this will be an added cost but it won't be a big one. We would also need to get a 1mm pitch touch panel connector for the 4-wires of the display. The NHD-4.3-480272MF-ATXI#-T-1 only has one option when it comes to interfacing. This display has 24-bit RGB interface. The microcontroller we choose needs to have the capability of driving 24-bit displays. Most of the microcontrollers we are considering only have 16 pins but there is a way to make it drive a 24-bit display. More on this can be found in the microcontroller section under microcontroller design. The NHD-4.3-480272MF-ATXI#-T-1 has 40 pins for interfacing brought out on the flat flexible cable (FFC) and can be connected with standard .5mm sockets. Newhaven Displays recommend the Molex 54132-4097. Since the display has the LED driver on it we will not need to purchase a separate backlight connection. This display has a 6:00 o'clock viewing angle which is a bottom viewing angle like what you would see when looking at a cell phone. It has a power supply voltage and logic supply voltage of 3.0V to 3.6V. The total LED voltage is about 22V and total LED current is about 32mA. The operating temperature range of this display is -20°C to +70°C.

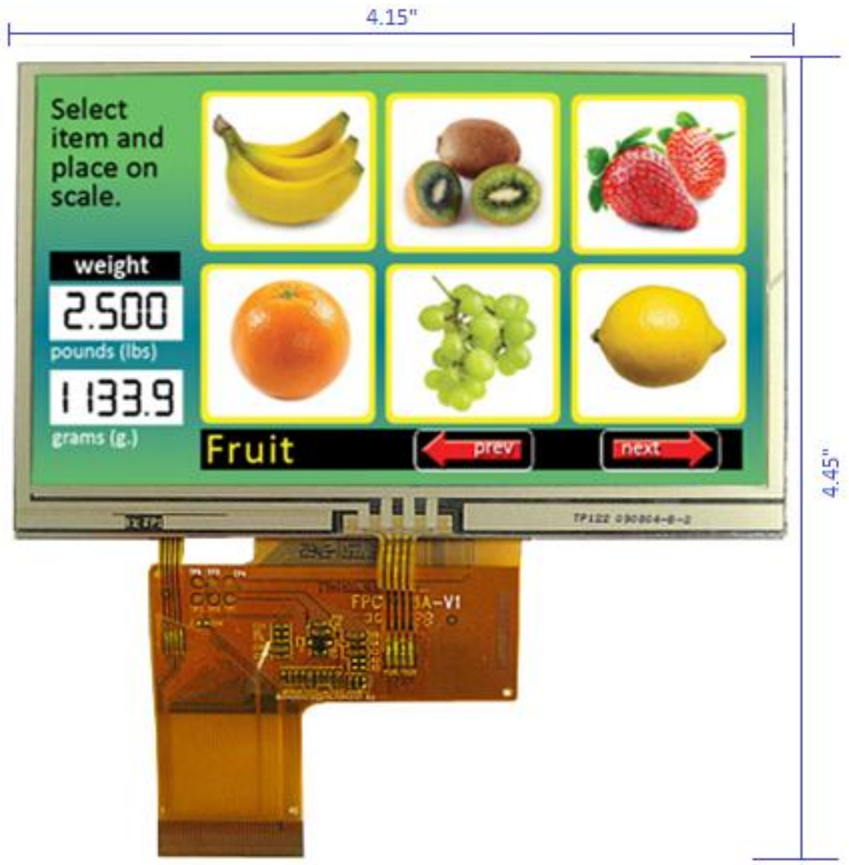


Figure 33: Newhaven NHD-4.3-480272MF-ATXI#-T-1 Display (showing FFC connectors)

Newhaven NHD-3.5-320240MF-ATXL#-T-1

The NHD-3.5-320240MF-ATXL#-T-1 is a 320xRGBx240 pixel, full color TFT display module. It has a 3.5 inch diagonal screen with a white LED backlight. The NHD-3.5-320240MF-ATXL#-T-1 has a Novatek NT39016D LED driver integrated into the display. Again this can be an advantage due to cost and space but also a disadvantage if it doesn't have all of the features we need. The NT39016D will allow us to use a filtered PWM signal when interfacing this with the microcontroller. This display is also a 4-wire resistive touch panel. The NHD-3.5-320240MF-ATXL#-T-1 does not have an integrated graphics controller so we would need to compensate for that. We can either buy a separate graphics controller or use a microcontroller that has an integrated graphics controller. We would also need to get a 1mm pitch touch panel connector for the 4-wires of the display. Newhaven recommends the Molex 52207-0485. This display uses 24-bit RGB interface which can be interfaced to most microcontrollers using 16 pins. The NHD-4.3-480272MF-ATXI#-T-1 has 54 pins for interfacing brought out on the flat flexible cable (FFC) and can be connected with standard .5mm sockets. Newhaven Displays recommend the Molex 51296-5494. Since the display has the LED driver on it we will not need to purchase a separate backlight connection. The NHD-3.5-320240MF-ATXL#-T-1 has a 12:00 o'clock viewing angle, which is a top viewing angle like what you would see if looking down at the speedometer of a car. It has a power supply voltage and a logic supply voltage of 300V to 3.6V. The total LED backlight voltage is about 19.2V and the total LED backlight current is about 20mA. This display has an operating temperature range of -10° to +60°C.

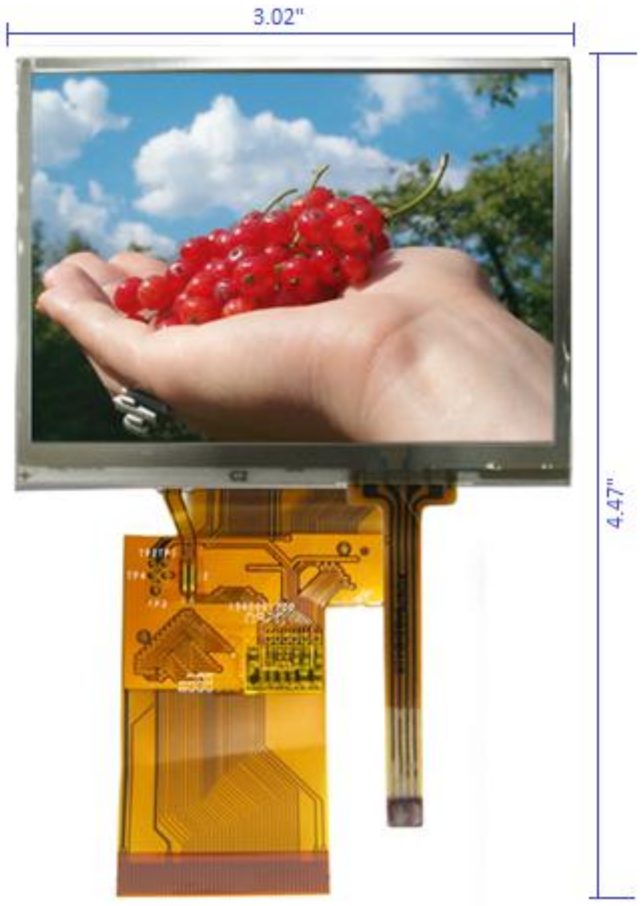


Figure 34: Newhaven NHD-3.5-320240MF-ATXL#-T-1 Display (showing FFC connectors)

Comparison

Candidate	Display Type	Type of Touch Screen	Num. of Pixels (width)	Num. of Pixels (height)	Backlight Type	Num. of LEDs
CFAF320240F-035T-TS	TFT	4-Wire Resistive	320	240	White LED	6 2 rows of 3
NHD-4.3-480272MF-ATXI#-T-1	TFT	4-Wire Resistive	480	272	White LED	12 2 rows of 6
NHD-3.5-320240MF-ATXL#-T-1	TFT	4-Wire Resistive	320	240	White LED	6 2 rows of 3

Table 19: Comparison of Different Display Features

Candidate	Diagonal Dimension	Overall Module Dimension	Graphics Controller	Integrated LED Driver
CFAF320240F-035T-TS	3.5 inches	3.47”x2.64”	Yes Solomon SSD2119	No
NHD-4.3-480272MF-ATXI#-T-1	4.3 inches	4.15”x2.65”	No	Yes Orise OTA5180A
NHD-3.5-320240MF-ATXL#-T-1	3.5 inches	3.02”x2.51”	No	Yes Novatek NT39016D

Table 20: Comparison of Different Display Features

Candidate	Interface	Viewing Angle	Minimum Logic Supply Voltage	Typical Logic Supply Voltage	Maximum Logic Supply Voltage
CFAF320240F-035T-TS	SPI or 18/16/9-bit parallel	12:00 o’clock	+2.5V	+3.0V	+3.6V
NHD-4.3-480272MF-ATXI#-T-1	24-bit RGB	6:00 o’clock	+3.0V	+3.3V	+3.6V
NHD-3.5-320240MF-ATXL#-T-1	24-bit RGB	12:00 o’clock	+3.0V	+3.3V	+3.6V

Table 21: Comparison of Different Display Features

Candidate	Min. LED Current	Typical LED Current	Max. LED Current	Min. LED Voltage	Typ. LED Voltage	Max. LED Voltage
CFAF320240F-035T-TS	10mA/row 20mA total	12.5mA/row 25mA total	20mA/row 40mA total	8.4V/row 16.8V total	9.6V/row 19.2V total	10.2V/row 20.4V total
NHD-4.3-480272MF-ATXI#-T-1	--	32mA total	--	20V total	--	22V total
NHD-3.5-320240MF-ATXL#-T-1	--	20mA total	--	--	19.2V total	--

Table 22: Comparison of Different Display Features

Candidate	Min. Operating Temp.	Max. Operating Temp.	Min. Storage Temp.	Max. Storage Temp.	Cost @ Qty 1
CFAF320240F-035T-TS	-20°C	+70°C	-30°C	+80°C	$53.72
NHD-4.3-480272MF-ATXI#-T-1	-20°C	+70°C	-30°C	+80°C	$38.00
NHD-3.5-320240MF-ATXL#-T-1	-10°C	+60°C	-20°C	+70°C	$31.50

Table 23: Comparison of Different Display Features

The above tables summarize the main features available with our three candidates. The most important features we need to consider is if it has an onboard graphics controller, the size of the overall display, its interfacing options and if it has an integrated LED driver. Since our microcontroller will have an integrated graphics controller the Crystalfontz CFAF320240F-035T-TS display is not a good choice since it has the Solomon already on it. The nice thing about the CFAF320240F-035T-TS is that is has both SPI and parallel interfacing. The Newhaven NHD-4.3-480272MF-ATXI#-T-1 and the Newhaven NHD-3.5-320240MF-ATXL#-T-1 are very similar with the features they come with. The only big difference with these two is the size. Since the box we will be putting our display in is a rectangle the Newhaven NHD-3.5-320240MF-ATXL#-T-1 would probably be too tall. The Newhaven NHD-4.3-480272MF-ATXI#-T-1 is the shape of a rectangle so this should fit nicely into our allowed space. This display also has an Orise OTA5180A integrated LED driver so we would not need to buy one separately. This display has 24-bit RGB interfacing which is a bit more complicated to interface with our microcontroller, but it can still be done. The price is in the middle of our three candidates, but it is way less than we budgeted for. For all those reasons, we will use the Newhaven NHD-4.3-480272MF-ATXI#-T-1 as our display for our temperature controller.

Display Pin-Out Tables

Below are the pins on the Newhaven NHD-4.3-480272MF-ATXI#-T-1. This table shows the pin number, the pin label from Newhaven and the description of the pin. There are 40 pins on this display which control the power of the backlight and the display itself. It also has eight pins each for the red, green and blue data signals. Since this display has 24-bit RGB interface and our microcontroller only has 16 pins we are going to need to make some modifications. In order for our 16-bit RGB 5:6:5 display controller to drive our 24-bit RGB 8:8:8 display we are going to have the red and blue upper three MSBs serve as the upper three MSBs and the lower three MSBs as their 8-bit equivalent. For the green bits we will use the upper two MSBs to serve as the upper two and the lower two MSBs for its 8-bit equivalent. The table for this is found in the microcontroller section. We will be using a Molex 54132-4097 for the LCD connector which has 40-.5mm sockets.

Pin #	Newhaven Label	Description
1	LED-	Ground for Backlight
2	LED+	Backlight Power
3	GND	Ground for Power Supply
4	VCC	Power Supply for LCD and Logic (3.3V)
5-12	[R0-R7]	Red Data Signals
13-20	[G0-G7]	Green Data Signals
21-28	[B0-B7]	Blue Data Signals
29	GND	Ground for Power Supply
30	PLCK	Data Sample Clock Signal
31	DISP	Display ON/OFF Signal

32	HSYNC	Line Synchronization Signal
34	DE	Data Enable Signal
36	GND	Ground Power Supply
37	XR	Touch Panel X-
38	YD	Touch Panel Y-
39	XL	Touch Panel X+
40	YU	Touch Panel Y+

Table 24: Pin Out Table for the NHD-4.3-480272MF-ATXI#-T-1

Touch Screen Controller

If we decide to use a 4-wire resistive touch screen we will need a touch screen controller that will determine the position of where the user touched. This controller will allow us to get the information using serial interface instead of parallel. We have many options when choosing a touch screen controller but we are only going to consider two types based on size, number of pins and the troubleshooting capabilities that each one offers. The two controllers we are comparing are the TI TSC2046 (16TSSOP) and the TI TSC2008 (12DSBGA). The two main differences of these candidates are the pin count and the package type. The TI TSC2008 is a very low-power touch screen controller which can work with a supply voltage as low as 1.2V. This controller is available in different packages but we are looking at the 12DSBGA package. The TSC2008 has 12 pins but since it has the ball grid array (BGA) it can get very complicated when it comes to making repairs and modifications. The good thing about this controller is that it is smaller in size. The TI TSC2046 is a touch controller that supports a low-voltage I/O interface from 1.5V to 5.25V. This controller is available in different packages but we are looking at the 16TSSOP package. This controller has 16 pins and compared to the BGA package it would be a lot easier to make repairs and modifications. The downside to this controller is that it is bigger than the TSC2008. We are going to choose the TI TSC2046 (16TSSOP) for our touch screen controller because of the ability to easily make changes even though it has more pins and is bigger. We will use SPI to interface this to the microcontroller. The pin out table below shows the pins for the TI TSC2046 that we will be using.

Pin #	TI Label	Description
1	+Vcc	Power Supply
2	X+	X+ Position Input
3	Y+	Y+ Position Input
4	X-	X- Position Input
5	Y-	Y- Position Input
6	GND	Ground
9	Vref	Voltage Reference Input/Output
10	IOVDD	Digital I/O Power Supply
12	DOUT	Serial Data Output. Data are shifted on the falling edge of DCLK. This output is high impedance

		when $\overline{CS}$ is high.
14	DIN	Serial Data Input. If $\overline{CS}$ is low, data are latched on the rising edge of DCLK
15	$\overline{CS}$	Chip Select Input. Controls conversation timing and enables the serial input/output register. $\overline{CS}$ is high = power-down mode (ADC only)
16	DCLK	External Clock Input. The clock runs the SAR conversion process and synchronizes serial data I/O

Table 25: Pin Out Table for the TSC2064

Size Requirements

The existing chassis has a 3.9” x 3.9” space for a display. We are able to modify the length of this space but we cannot have our display exceed the allowed 3.9” in height. Since the NHD-4.3-480272MF-ATXI#-T-1 has dimensions of 4.15” x 2.65” we will need to extend the length of the allowed space so that the display can fit. Having a rectangular shaped display will make the device look more sleek and modern. Below is a drawing of the chassis and the existing space for the display.

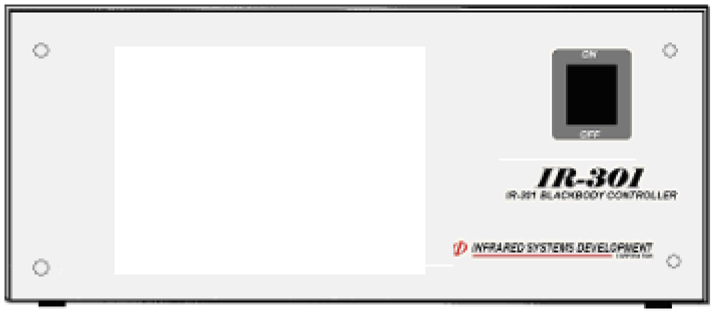


Figure 35: Existing drawing of the chassis that will need to be modified

User Interface

Overview

The user interface is a very important part of this device because many different users are going to be using this so it needs to be intuitive and require little instruction for proper use. The user interface will be designed to look sleek and high-tech but be user friendly. It will be comprised of a series of menus with on screen buttons. There will be a menu for the everyday user and a menu for the administrator.

Requirements

One of the most important requirements of this user interface is that it needs to be use friendly. It needs to be very simple but still look advanced. For the user menu it needs to have an alarm menu, set point menu, backlight dimming menu, the ability to show the actual temperature, the energy radiated by the blackbody when the device is on, and the black body radiance. For the administrator menu it needs to have a menu to set the number of samples, a menu for the offset points, a thermocouple menu, a menu to change the PID coefficients and a menu for the max power to send to the device. These menus will be discussed in more detail in the following sections. The device should also perform power on self testing to be sure all sensors and devices are operating normally. In the event of a failure, the system should report the failure type. The system will also have automatic save and restore to default settings after any value is changed.

User Interface Design

When the user turns on the display the first thing that will appear is the Infrared Systems Development logo. As soon as the display warms up a screen with the device temperature, the alarm output status lights, the power output to heater status lights, the user menu and the administrator menu will come up. The yellow dots are the power output to heater status lights and the red dots are the alarm output status lights. These will be lit when there is power going to the heater or the alarm conditions have been met. More on setting these values will be discussed in the following menu sections.

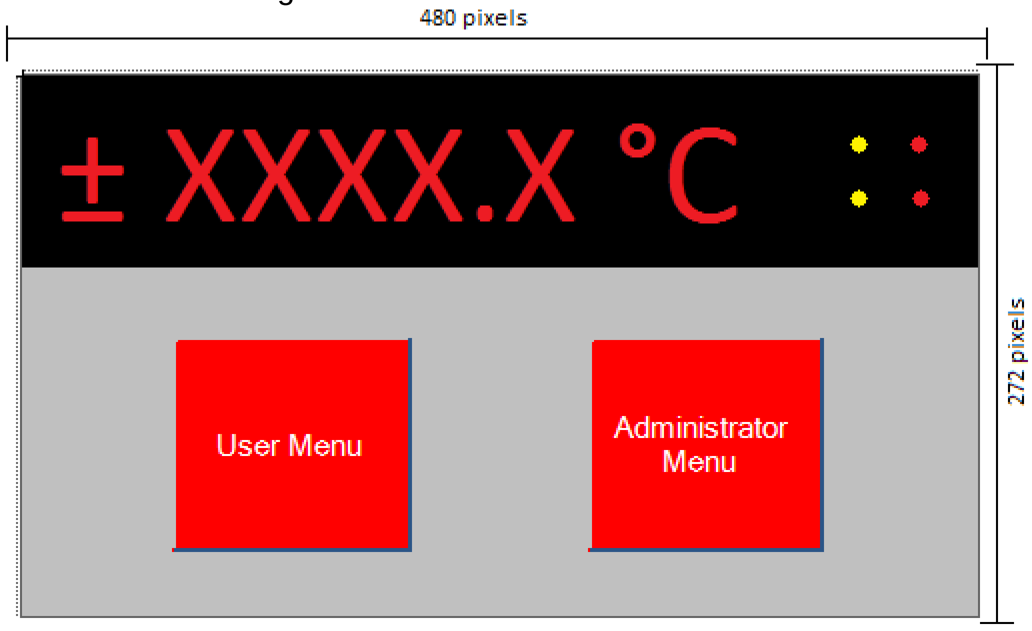
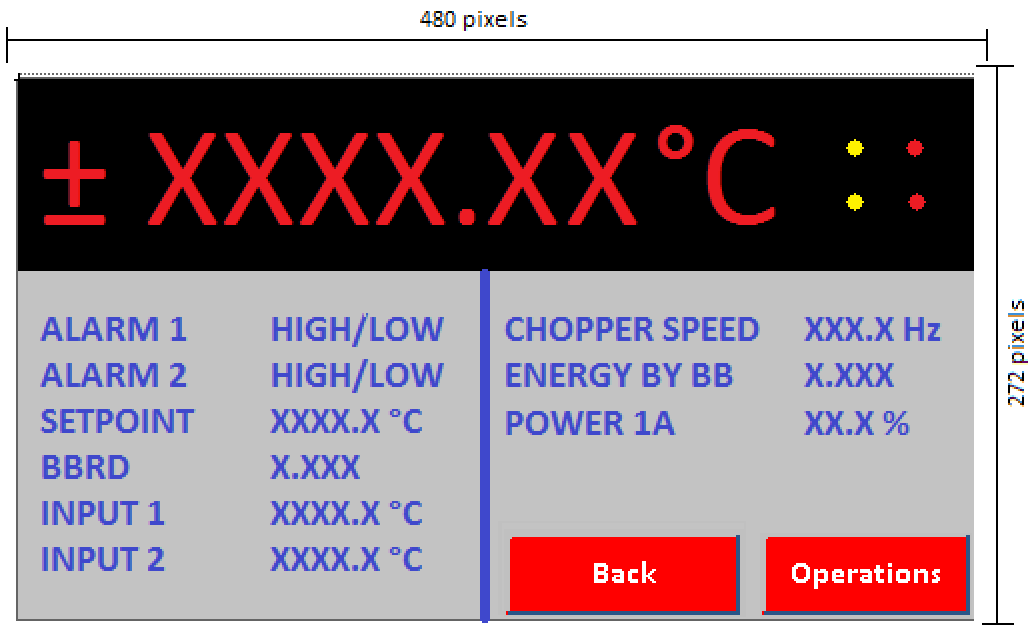


Figure 36: User Menu and Administrator Menu Screen

If the user presses the User Menu button then a new screen will appear with different options. The temperature and status lights will be on the screen at all times. When the User Menu button is pressed it will display a screen showing the settings of the alarm status, the current set point, the BBRD, the input 1 and input 2 and the power 1A. If either ALARM 1 or ALARM 2 says HIGH then that means that the relay is closed allowing the user to add external circuitry that will be triggered once the alarm goes high. If either of them says LOW then that means that the switch is open which will deactivate the circuit. The alarms are typically tripped when the actual temperature is above or below the set point by a certain amount. The SETPOINT is the temperature that the user wants the device to get to. The BBRD stands for black body radiance. This value is calculated by the Stefan-Boltzmann law. This law states that the total energy radiated per unit surface area of a black body per unit time (or the emissive power) is directly proportional to the fourth power of the black body's thermodynamic temperature. The equation will be calculated based on the current temperature of the device. The equation for this is $Q = \sigma T^4$, where σ is the Stefan-Boltzmann constant and has a value of $\sigma = 5.67\,x\,10^{-8}\frac{W}{m^2K^4}$. The INPUT 1 and INPUT 2 show the temperatures that are being read by the thermocouples. The CHOPPER SPEED is the frequency that the blade is spinning at, which is measured in the device. The ENERGY BY BB is the energy radiated by blackbody, which is calculated by Planck's Law. We will be using Planck's Law in terms of wavelength. The equation for this is $B_\lambda(T) = \frac{2hc^2}{\lambda^5}\frac{1}{e^{\frac{hc}{\lambda k_B T}}-1}$ where T is the temperature of the black body, k_B is the Boltzmann constant, h is the Planck constant and c is the speed of light. The user will have to define the upper and lower wavelengths so the energy radiated by the blackbody can be calculated. The POWER 1A is the percent of power to the heater and this is also set by the administrator. The user has the options to change certain values. To do this the user will press the Operations button at the bottom right corner of the screen in Figure 37. If they press the Back button they will be brought back to the menu in Figure 36.



ALARM 1	HIGH/LOW	CHOPPER SPEED	XXX.X Hz
ALARM 2	HIGH/LOW	ENERGY BY BB	X.XXX
SETPOINT	XXXX.X °C	POWER 1A	XX.X %
BBRD	X.XXX		
INPUT 1	XXXX.X °C		
INPUT 2	XXXX.X °C		

Figure 37: Current Values of Device

After the user presses the Operations button it will bring them to a menu where they can change the alarm set points, the ramp to set point value, the current set point value, the wavelength range for the blackbody energy and the backlight brightness. The alarm set points allow you to change the high and low deviations for alarm 1 and alarm 2. The alarm low deviation defines the deviation value on the low side of the set point at which the alarm will be triggered. The alarm high deviation defines the deviation value on the high side of set point at which the alarm will be triggered. The ramp to set point allows you to choose how many °C/min or °C/hour you want the temperature to increase until it reaches its set point. This setting can be turned off, on at start up or you can choose to start up and change. The current set point value is the temperature you want the device to reach and the user can also change the brightness of the backlight. The brightness setting is controlled by the PWM signal through the driver. The user will also have to enter the lower and upper wavelength so that the energy radiated by blackbody can be calculated. If the user presses the Back button they will be brought back to the menu in Figure 37.

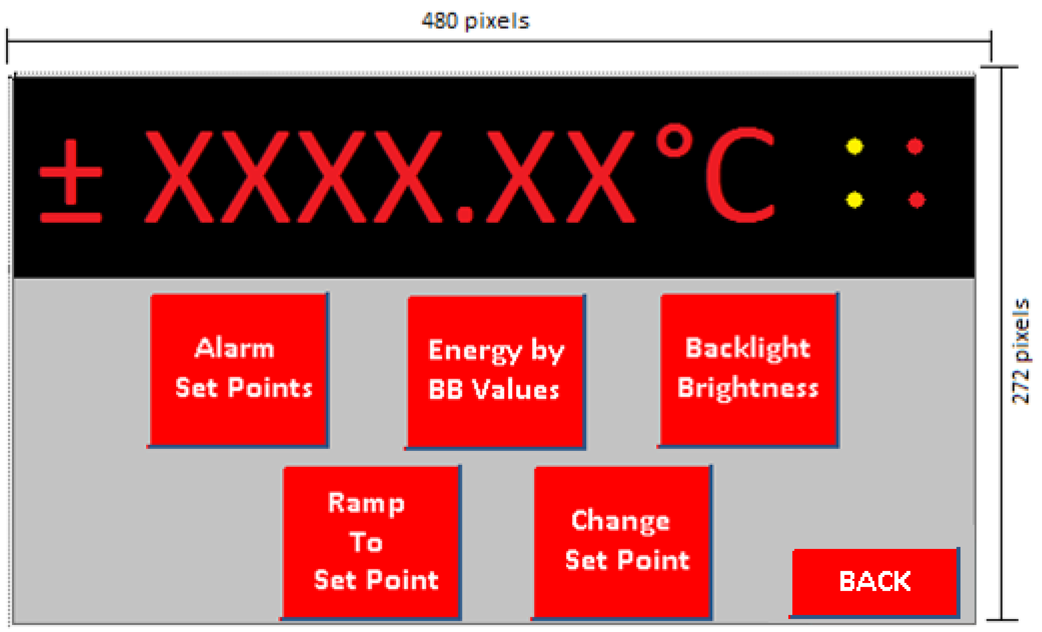


Figure 38: User Operations

If the user presses the Alarm Set Points button then they will be brought to another screen with Alarm 1 and Alarm 2 buttons. If the user presses the Back button they will be brought back to the screen in Figure 38.

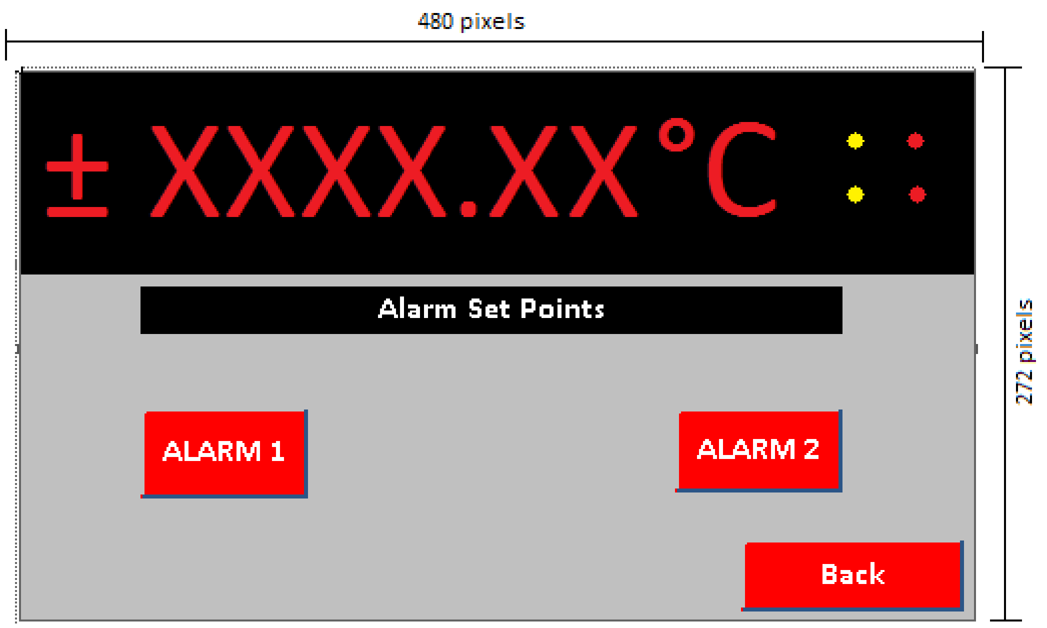


Figure 39: Alarm 1 and Alarm 2

When the user presses either the ALARM 1 or ALARM 2 button they will be prompted to a screen where the user will enter the low and high deviations for each alarm. This will usually be in the form of ± X.X ℃ but the user will have the ability to enter a number as high as ± XXXX.X ℃. The negative (-) is for the low deviation and the positive (+) is for the high deviation. Once the user is done entering this information they will press the DONE button which takes them back to the screen in Figure 39.

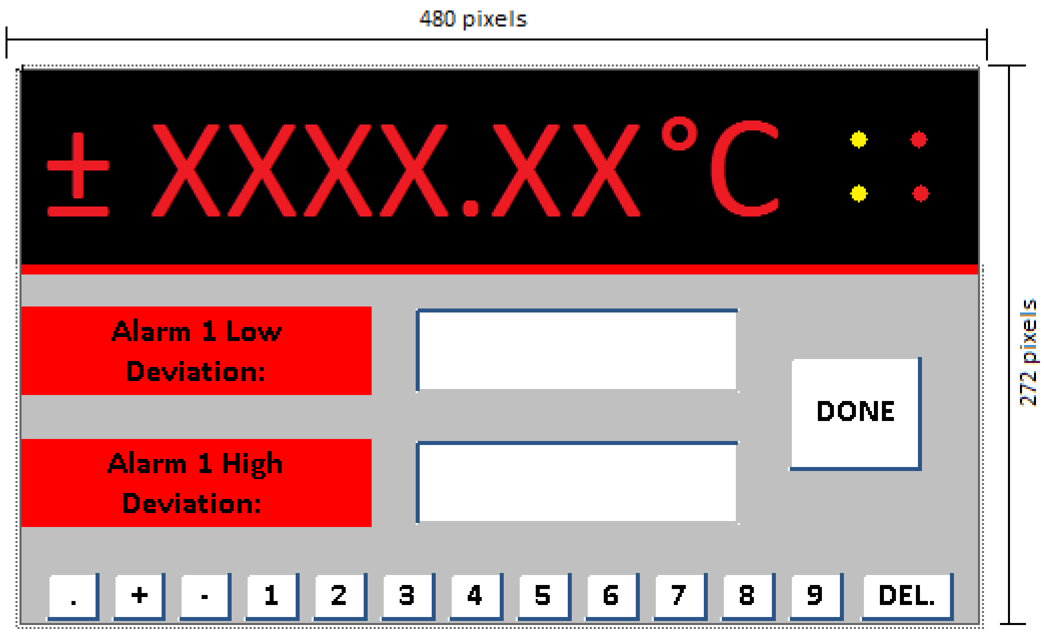


Figure 40: Alarm 1 Deviations

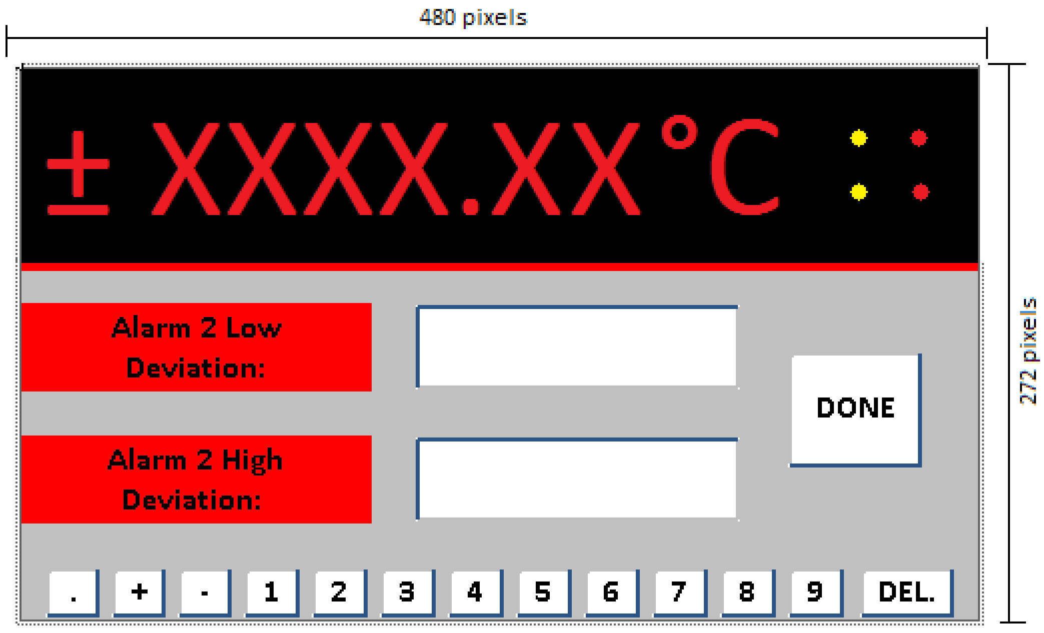


Figure 41: Alarm 2 Deviations

Next the user will need to enter the upper and lower wavelength values so the energy by the black body can be calculated. If the user presses the Energy by BB Values button it will take them to another screen where they will have the ability to enter each of the values. Once the user is done they will press the DONE button and it will take them to the screen in Figure 38.

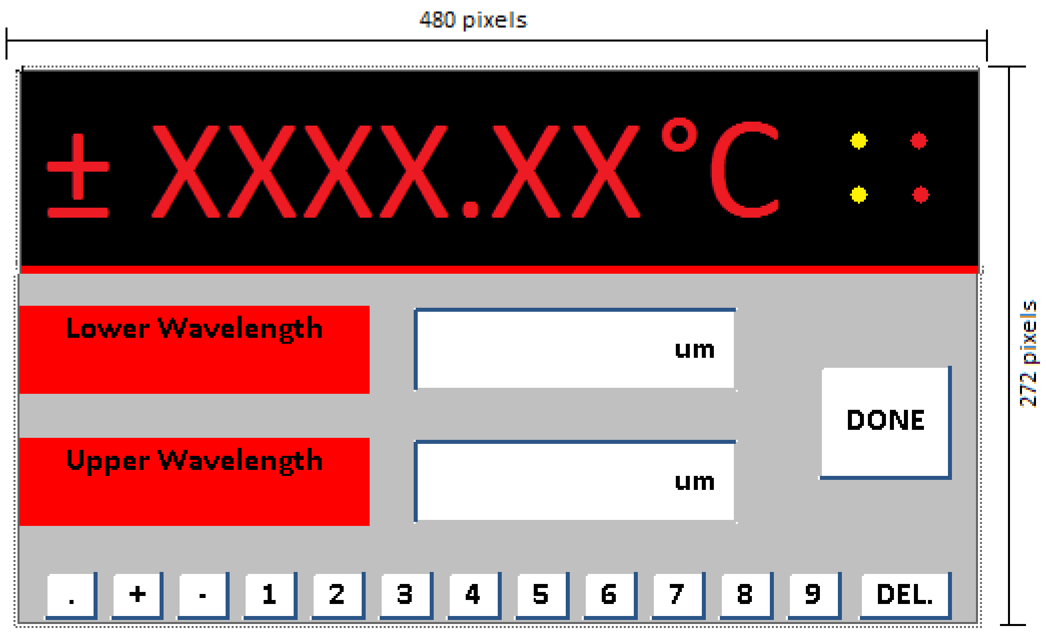


Figure 42: Wavelength Ranges

If the user wants to change the backlight of the screen to make it lighter or darker they will press the Backlight Brightness button. After pressing this button it will take them to a screen that has a scroll bar that ranges from 0 – 100% with an indicator showing what percentage the backlight is at. There will also be arrows on each side of the bar for the user to move the indicator left or right. Once the user chooses their backlight preference they will press the DONE button and return to the screen in Figure 38.

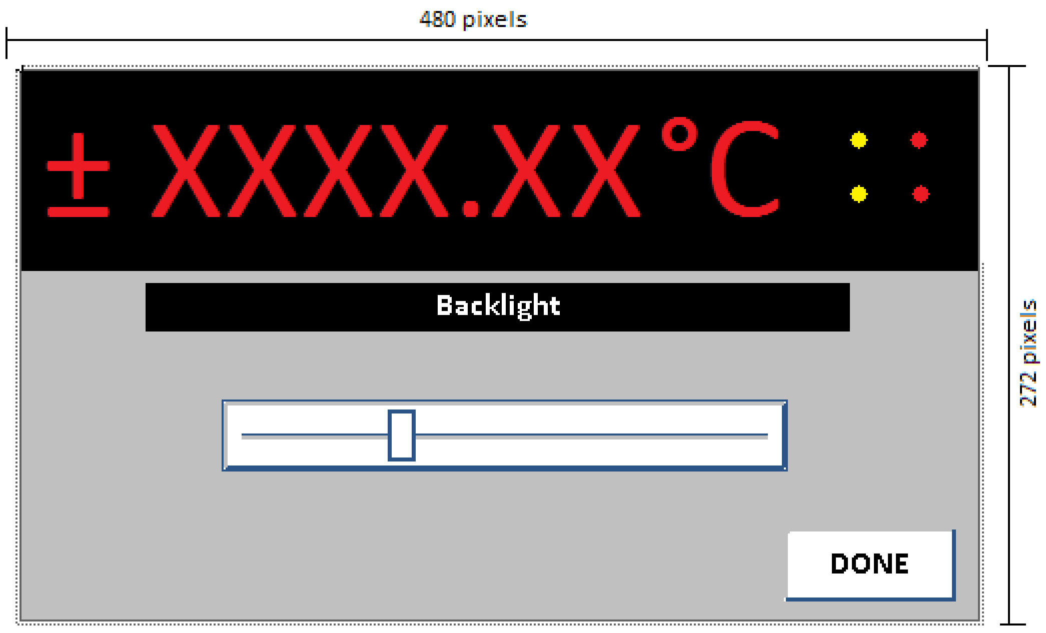


Figure 43: Backlight Setting

The Ramp to Set Point button will allow the user to choose how many °C/min or °C/hour they want the temperature to increase until it reaches its set point. The user has two options for this setting, which are off and start up and change. If the user presses the OFF button then this setting will be turned off and it will automatically shoot up to the desired temperature and return them to the screen in Figure 38. Start up and change means that once you input the value and leave the menu it will start ramping up in the increment that the user chose. Pressing this button will take you to a screen where the user will enter the Ramp to Set Point Rate. The ramp to set point will usually be in the form of X.X °C/min or X.X °C/hour, but the user will have the ability to enter a number as high as XXXX.X °C/min or XXXX.X °C/hour. Once this setting is set the user will press the DONE button to go to the screen in Figure 38.

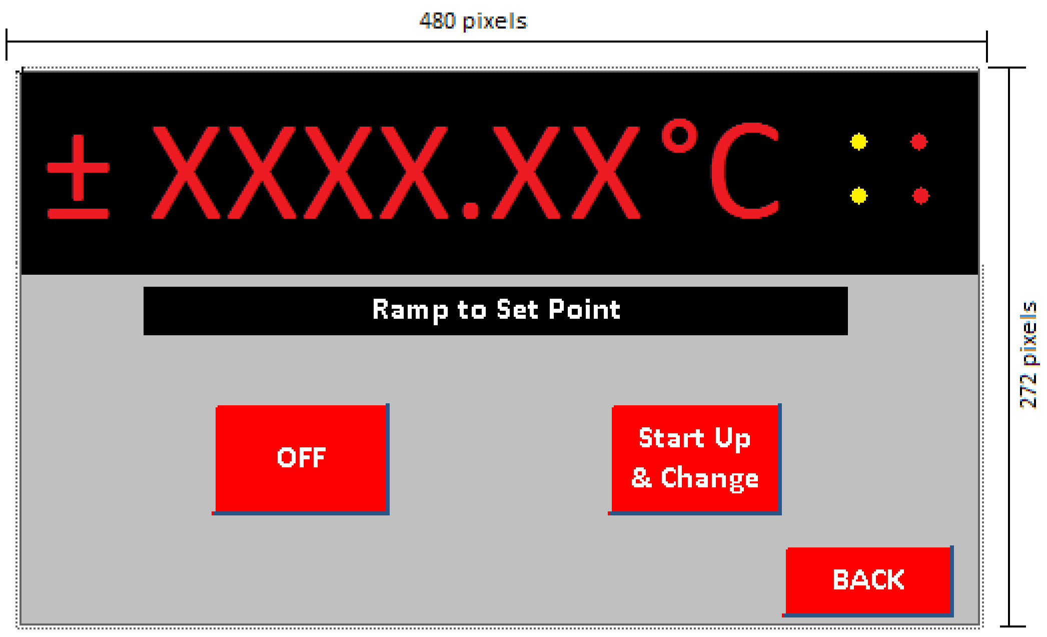


Figure 44: Ramp to Set Point Options

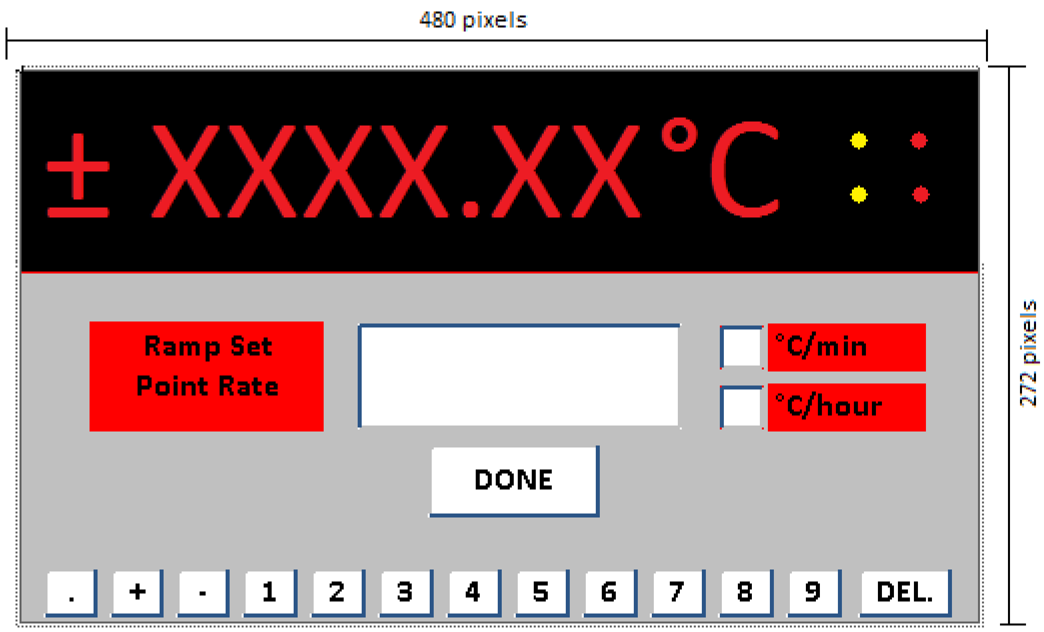


Figure 45: Ramp to Set Point Rate

The last button on this screen is the Change Set Point button. This button allows to user to change the current set point to the desired set point. The value the user inputs will be in the form of XXXX.X °C. Once the user has entered the value they will press the DONE button to return to the screen in Figure 38.

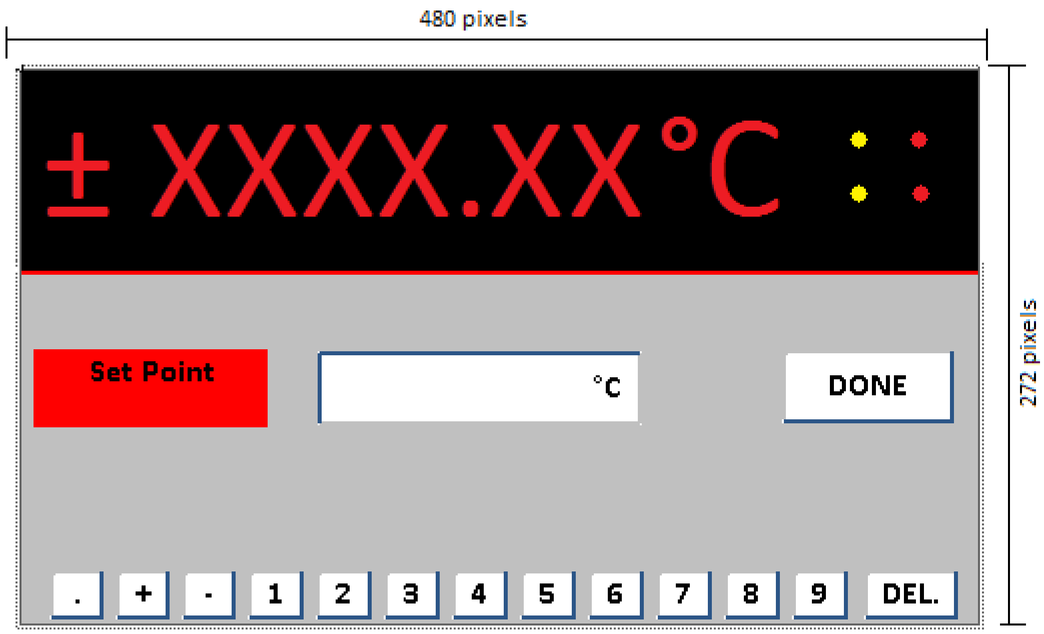


Figure 46: Set Point

After the user enters all of their values then they will press the Back button on Figure 38 to return to the screen in Figure 37. The user will most likely stay on this screen for the remainder of the time that the device is on since this screen shows all the values that they just put in.

This device also has an administrator menu which is protected by a password and will be accessed through Figure 36. The administrator will have access to all the user menus as well as few extra options that should only be accessed by the administrator. Once the Administrator Button is pressed then a new screen comes which will prompt the user to enter a passcode. Once this passcode is entered correctly then the administrator will press DONE and a new screen with all the administrator options will appear. If the user presses the back button it will take them to the menu in Figure 36.

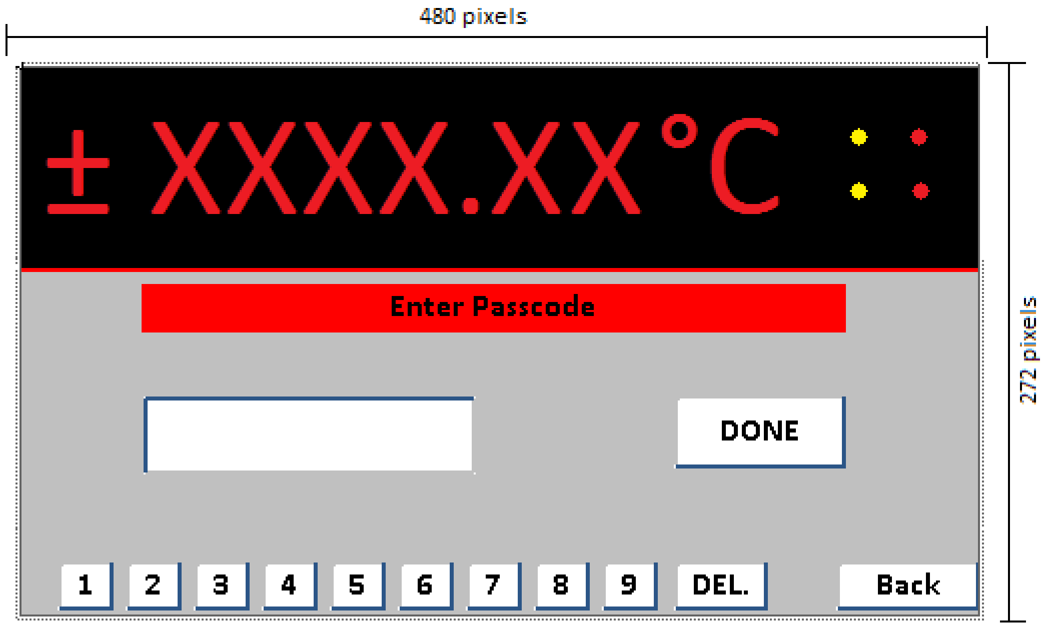


Figure 47: Administrator Passcode

The extra settings on this menu are offset points, thermocouple type, the max power to send to the heater, the number of samples the device takes and the option to change the PID coefficients. The offset points menu will allow the administrator to change any of the offset points. The thermocouple menu and max power menu will allow the administrator to pick the thermocouple type that will be used and choose the amount of power that is sent to the device, respectively. The number of samples menu is the menu where the administrator can choose the amount of samples the device takes in a certain time period before it displays the temperature. The PID menu will allow the administrator to change the crossover points and the coefficients. If the user presses the Back button then they will be brought back to the menu in Figure 36.

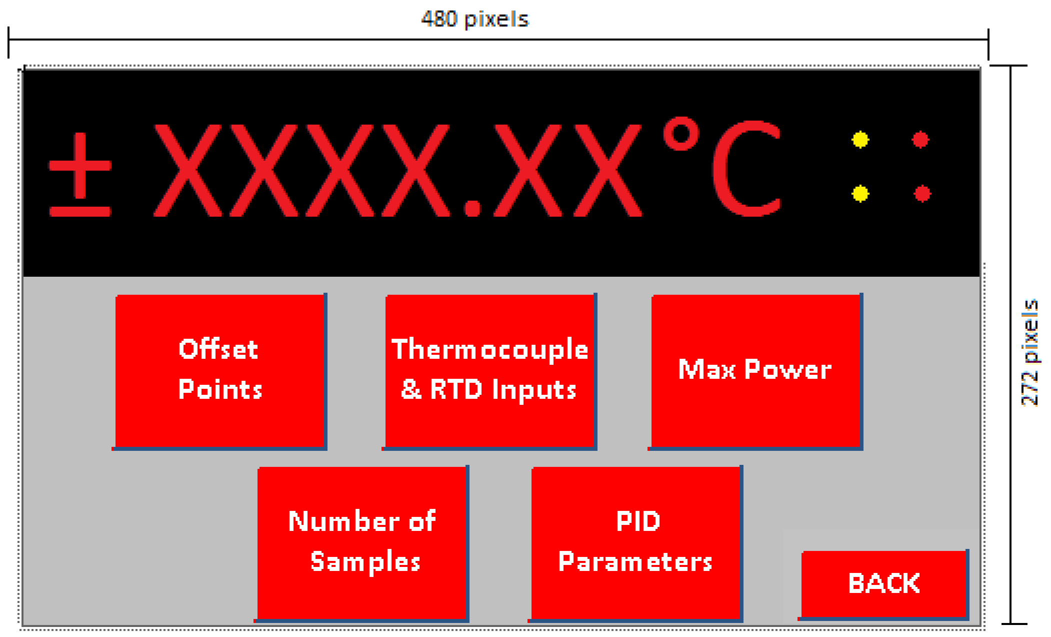


Figure 48: Administrator Settings

If the user presses the Offset Points button then they will be brought to a screen where they can modify the offset point and offset value. A scrolling list menu will come up with Offset 1 through Offset 17 and the user will pick which offset they want to modify. The user can either use the up or down arrows to scroll or they can use their finger to swipe the screen. Once the user picks an Offset then they need to press the continue button to move onto the screen in Figure 50. If the user presses the Back button they will be returned to the screen in Figure 48.

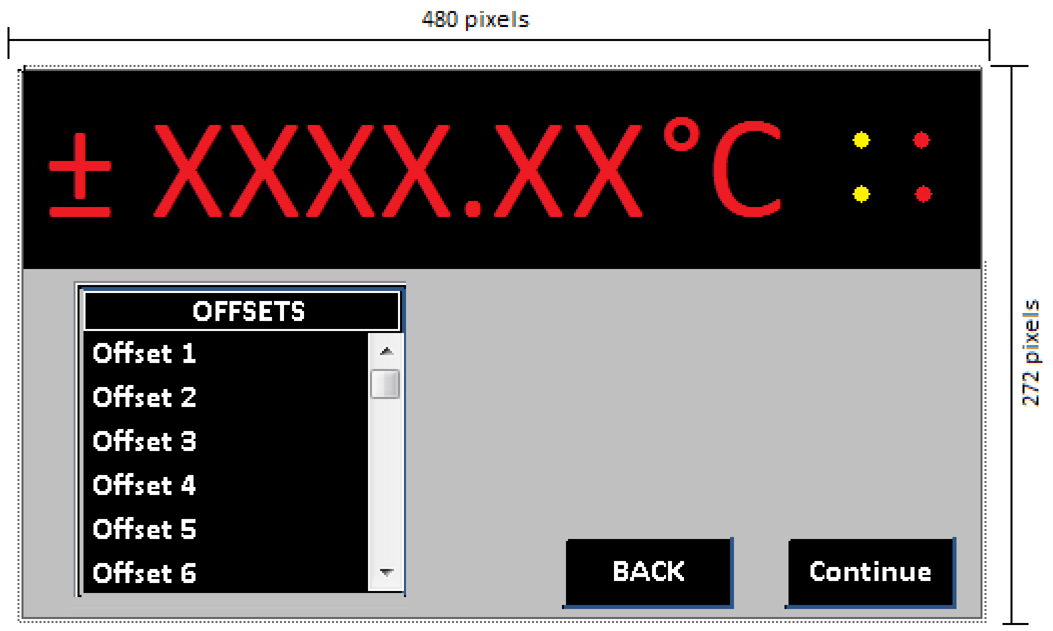


Figure 49: Offset Menu

Once the user picks the offset they want to modify they will get brought to a screen with the Offset Point and Offset Value boxes. The user will input the value in °C for both values. Once the user is done they will press the DONE button which will take them back to the screen in Figure 49.

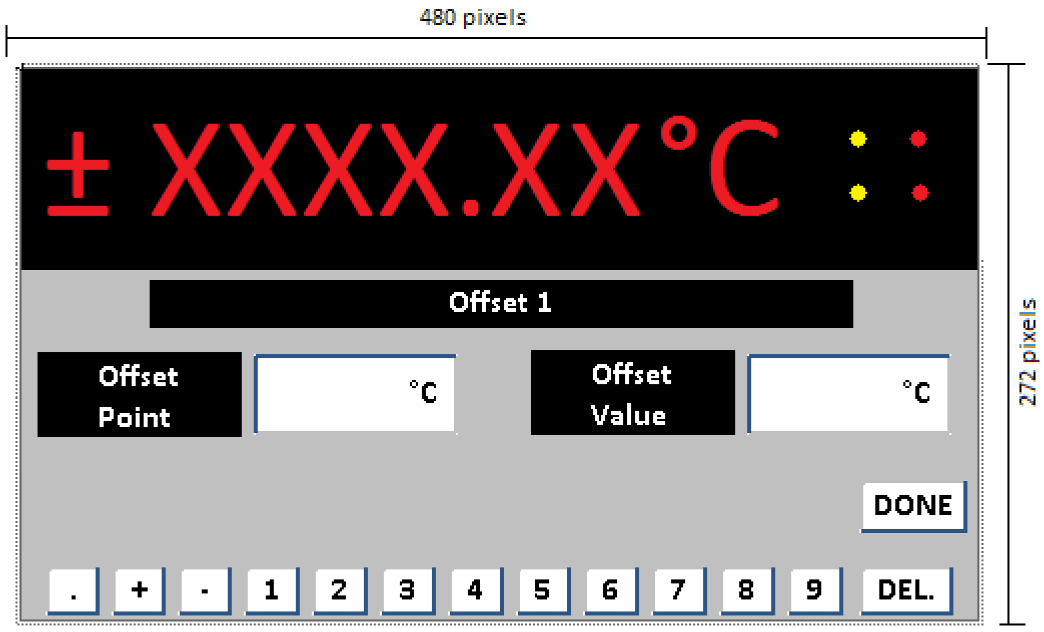


Figure 50: Offset Point and Offset Value

If the user presses the Thermocouple & RTD Inputs button then they will be brought to a screen with Input 1 and Input 2. Both input buttons take you to different screens but they all display the same options. The user will have to set each of these to either a thermocouple, a RTD or turn it off. If the user wants to go back to the screen in Figure 48 then they will press the BACK button.

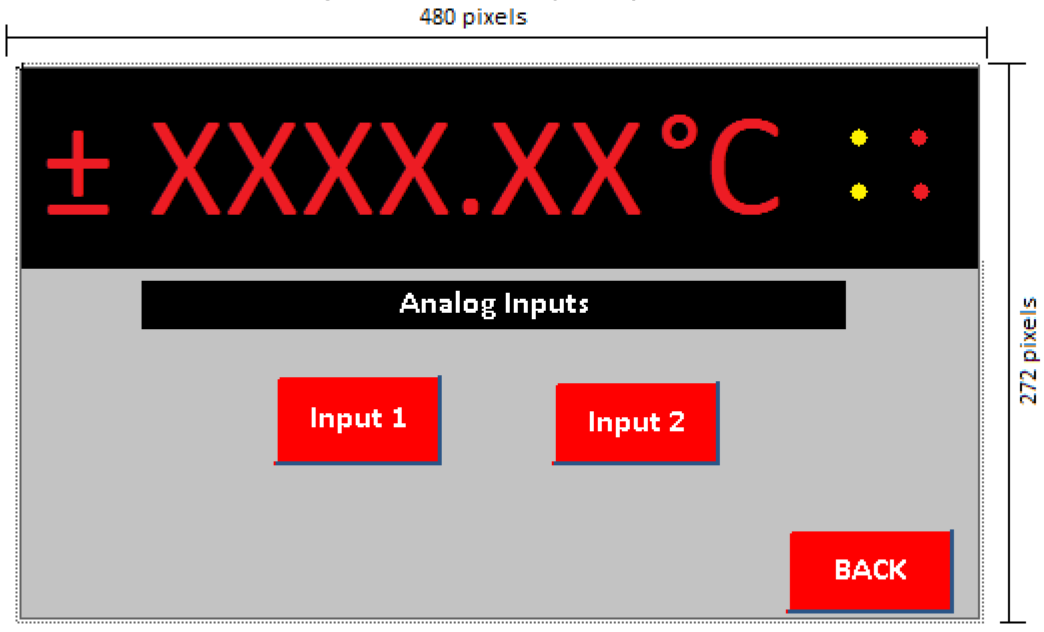


Figure 51: Analog Input 1 and Analog Input 2

Once the user chooses the input they will need to choose the thermocouple type and RTD for inputs. They can choose from type S and T for the thermocouples. If the user chooses RTD then they are finished but if they choose a thermocouple type they will be brought to another screen to choose the Cold Remote Sensing options. The user can press the BACK button to return to the screen in Figure 51.

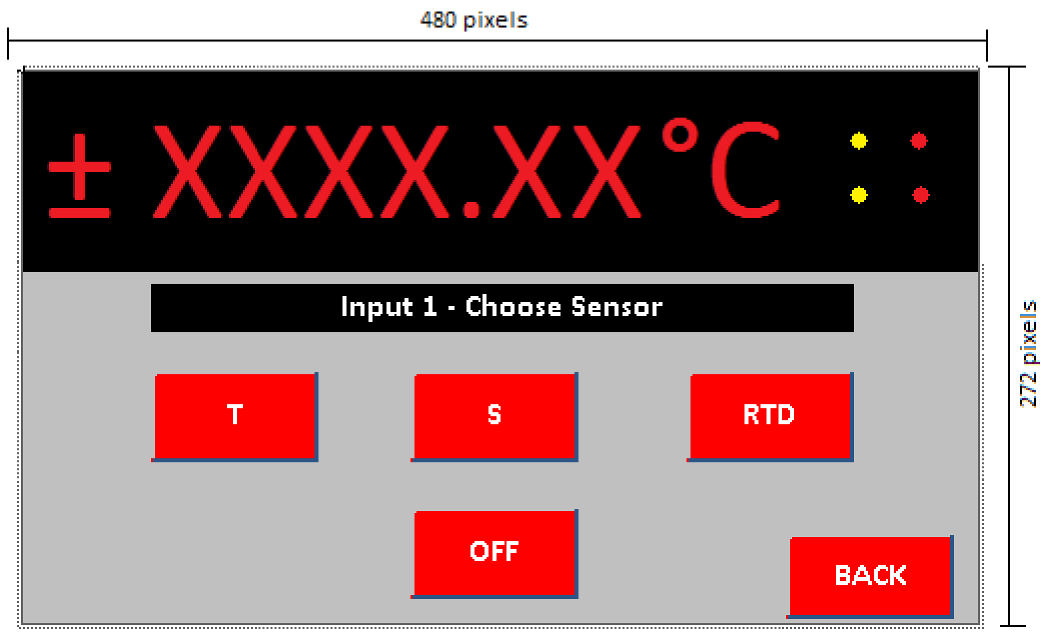


Figure 52: Sensor Type

If the administrator chooses thermocouples they will be brought to a screen to that has Cold Remote Sensing options. The user can choose from Internal, None (Constant) or CJC on Input 2. Once the user makes their selection they will press the DONE button to return to the screen in Figure 48.

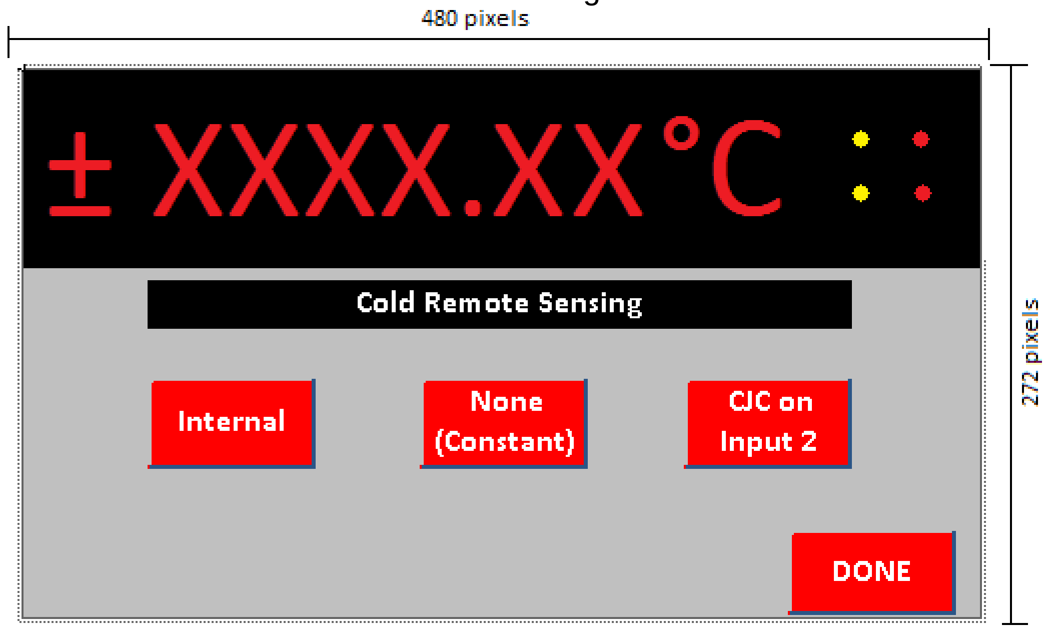


Figure 53: Cold Remote Sensing

The Max Power button allows the administrator to input the maximum percentage of power that is being sent to the device. The user will be brought to a screen where they choose either Heat (Reverse) or Cool (Direct). If the user presses the Back button they will return to the screen in Figure 48.

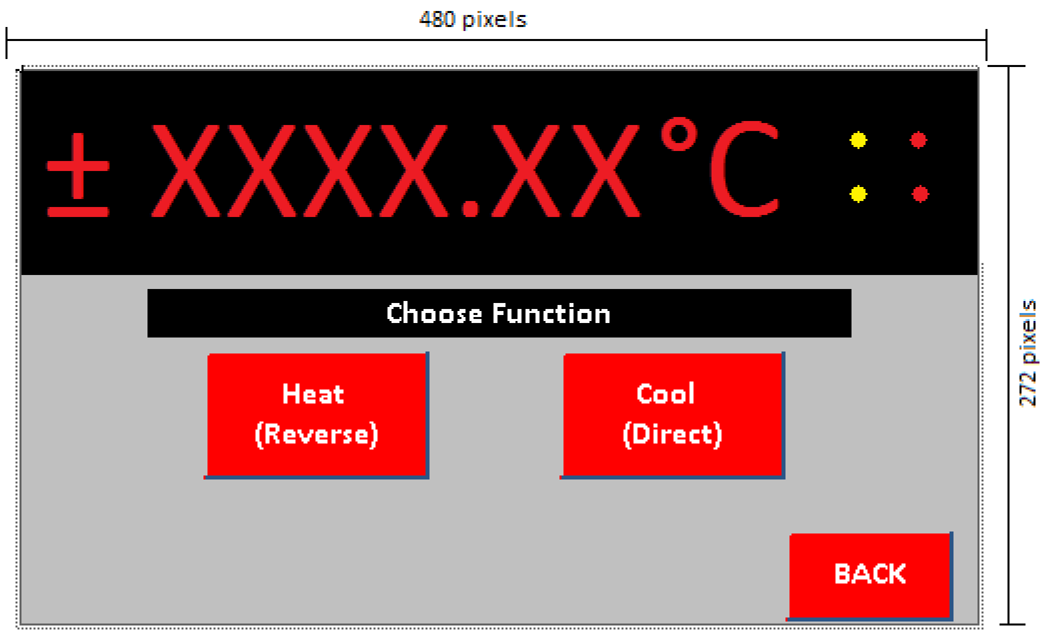


Figure 54: Heating or Cooling Function

Once the user chooses they will be brought to a screen where they enter the percentage of power they want to send to the device. Once this value is entered the user will press the DONE button to get to the screen in Figure 48.

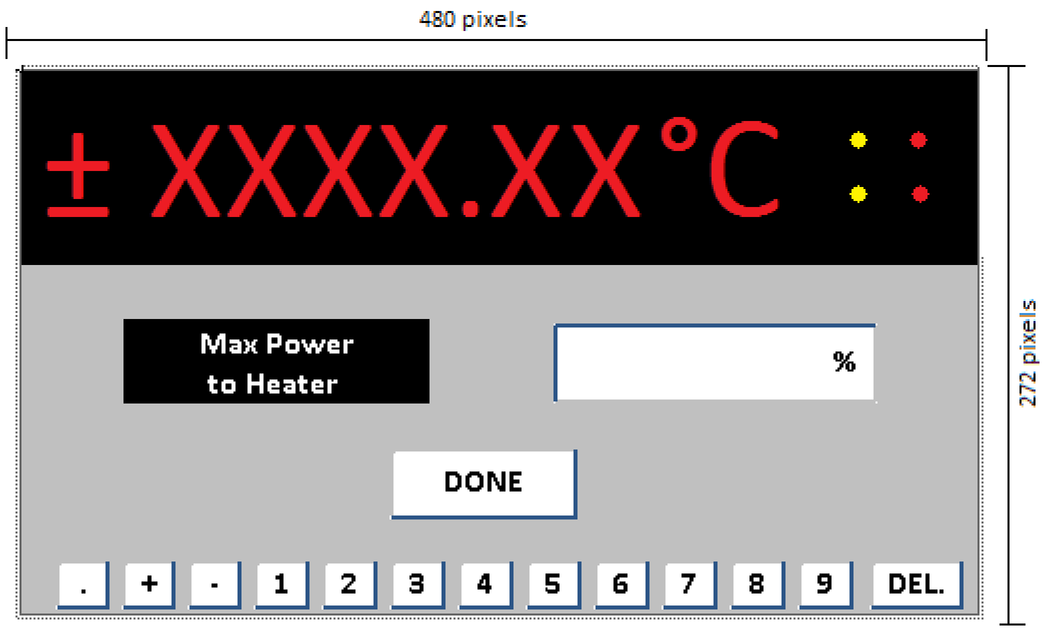


Figure 55: Max Power to Heater

The Number of Samples button will allow the administrator to input the number of samples that the device will take before the actual temperature is displayed. The device will average the temperature for however many samples the administrator enters. Once this is entered the user will press DONE to get to the screen in Figure 48.

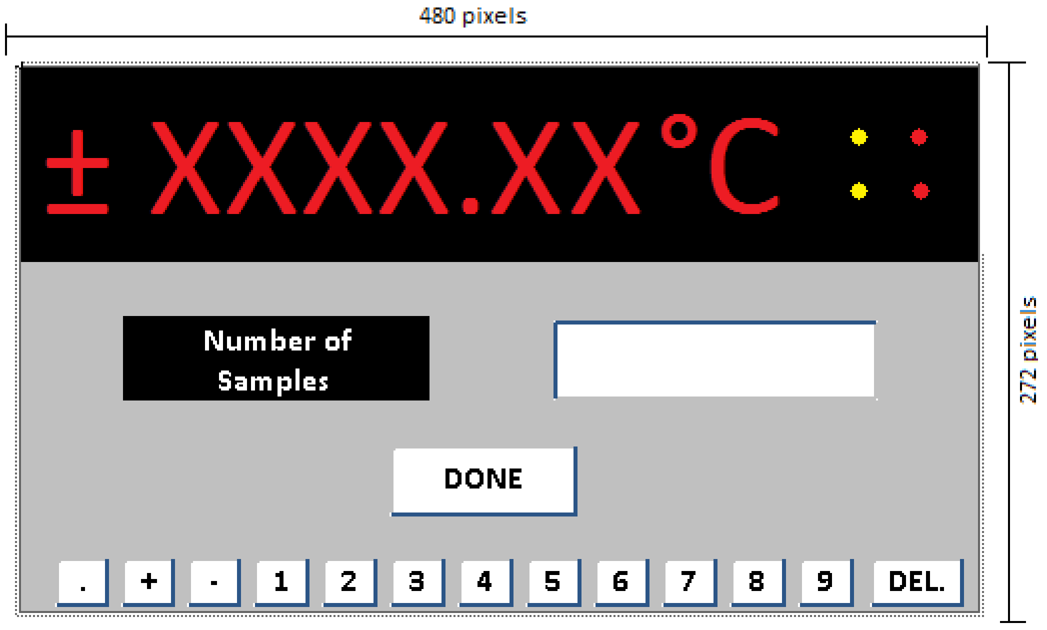


Figure 56: Number of Samples

The last menu that the administrator can control is the PID parameters. The device will use five PID sets for different temperature ranges. Once the user presses the PID Parameter button they will be brought to a screen with five different PID Sets and they will choose which set they want to change. If the user presses the BACK button they will be brought to the screen in Figure 48.

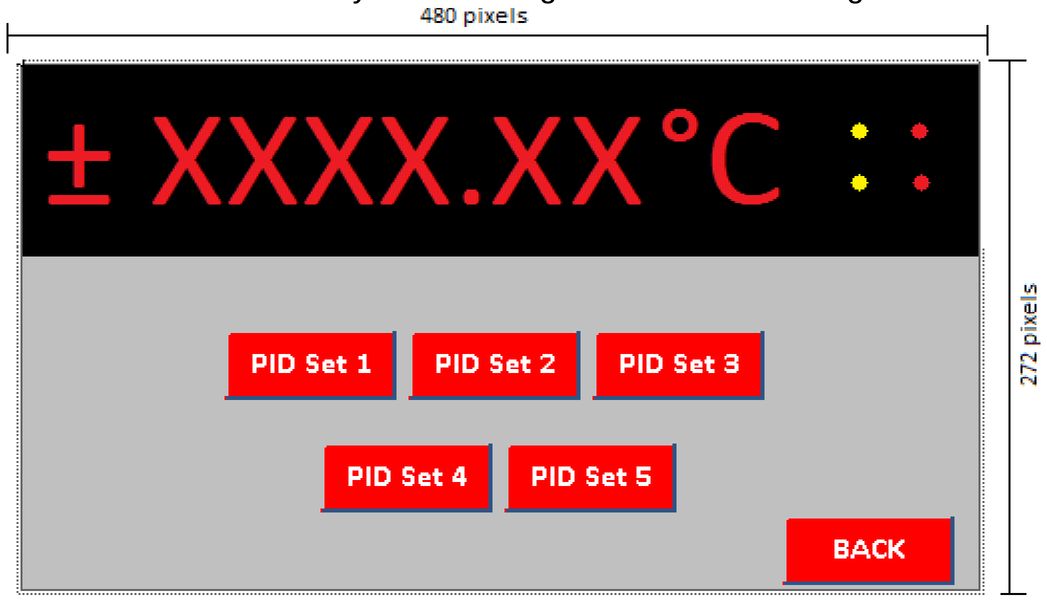


Figure 57: PID Setting Menu

Once they press the PID Set button they will be brought to a screen to enter the parameters for each part of the PID, which are the Proportional part, the Integral (reset) part and the Derivative (rate) part. The proportional parameter will be in units of °C, the integral parameter will be in units of /minute and the derivative parameter will be in units of minutes. Once the administrator enters these values for each parameter and each set they will press the DONE button and return to the menu in Figure 57.

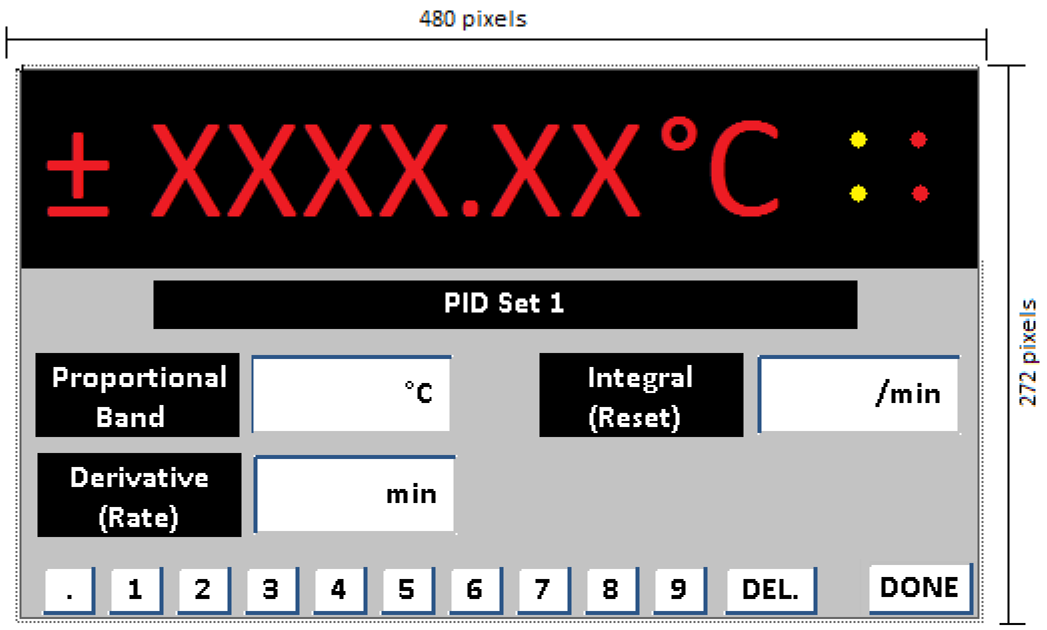


Figure 58: PID Value

The flowcharts below in Figure 59 and Figure 60 summarize the User Menu and the Administrator Menu. The squares are buttons and the hexagons represent a field or static text on the screen. These flowcharts show where each button can lead to you and the options you have when entering information. The dashed boxes represent what is shown on a certain screen. All of this information is found in the legends in Table 26 and Table 27.

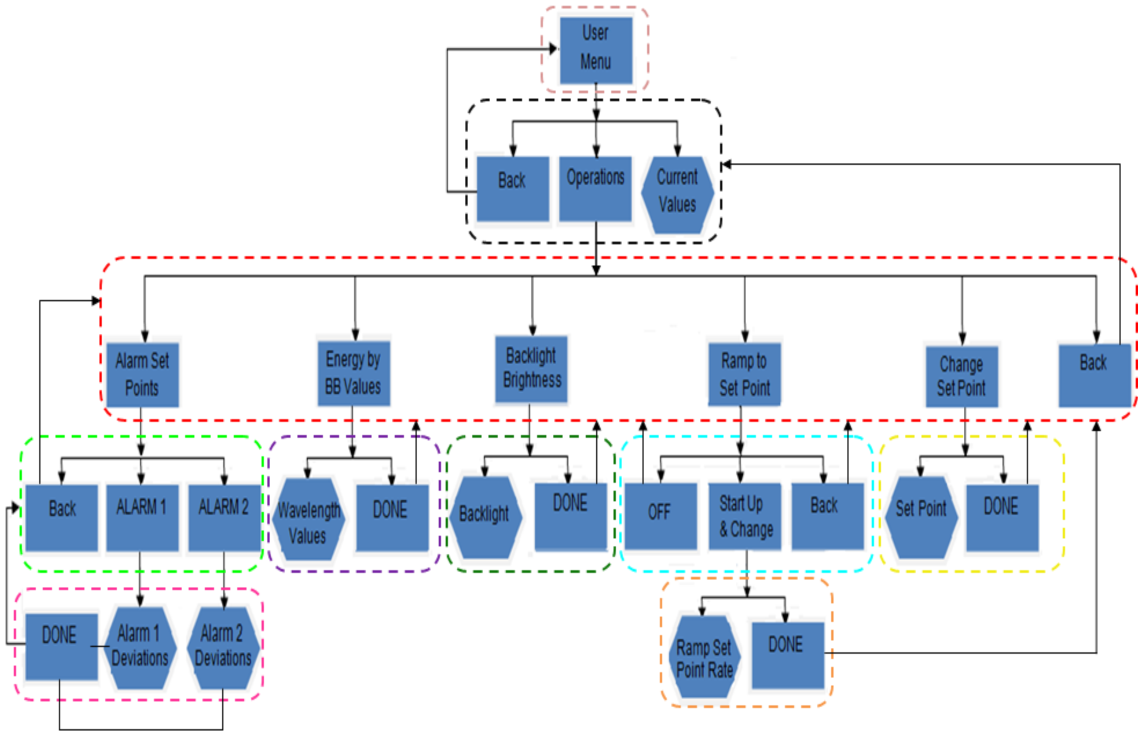


Figure 59: Flow Chart of User Menu

LEGEND FOR USER MENU	
■	A button that the user can press which will direct the user to another screen
⬢	A field where the user has to input a value or it can also represent text
⬚	Represents the User Menu button in Figure 36
⬚	Represents what is in Figure 37
⬚	Represents what is in Figure 38
⬚	Represents what is in Figure 39
⬚	Represents what is in Figure 40 and Figure 41
⬚	Represents what is in Figure 42
⬚	Represents what is in Figure 43
⬚	Represents what is in Figure 44
⬚	Represents what is in Figure 45
⬚	Represents what is in Figure 46

Table 26: Legend for User Menu

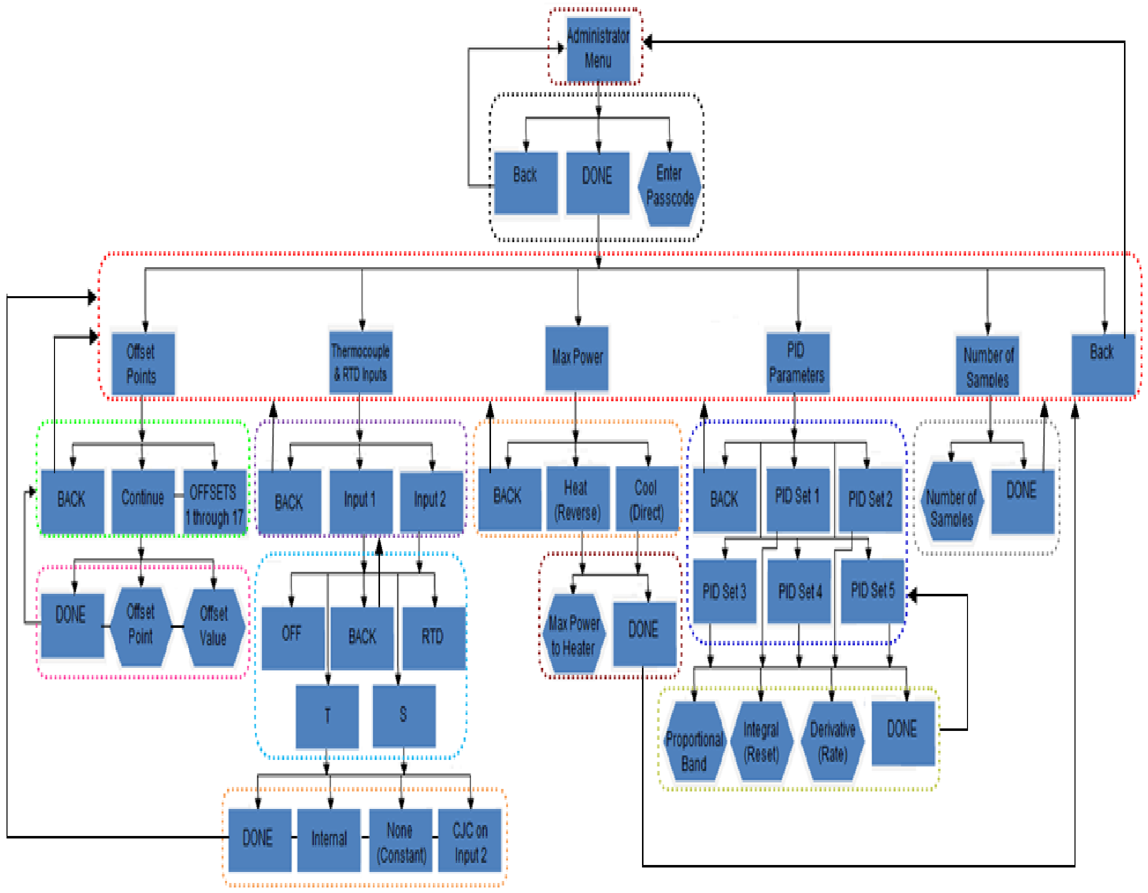


Figure 60: Flow Chart of Administrator Menu

LEGEND FOR ADMINISTRATOR MENU	
■	A button that the user can press which will direct the user to another screen
⬢	A field where the user has to input a value or it can also represent text
⬚	Represents the Administrator Menu button in Figure 36
⬚	Represents what is in Figure 47
⬚	Represents what is in Figure 48
⬚	Represents what is in Figure 49
⬚	Represents what is in Figure 50
⬚	Represents what is in Figure 51
⬚	Represents what is in Figure 52
⬚	Represents what is in Figure 53
⬚	Represents what is in Figure 54
⬚	Represents what is in Figure 55
⬚	Represents what is in Figure 56
⬚	Represents what is in Figure 57
⬚	Represents what is in Figure 58

Table 27: Legend for Administrator Menu

Power

Power Requirements

The input of our device will be 110VAC/220VAC, which will power the heater directly. From the input we will generate the DC voltages we need to power the rest of the components. We also need to have zero cross sensing at the AC relay which will lower the electrical interference generated when switching at a nonzero point. We will use different components to reduce the noise and control the power, which will all be discussed in more detail in the following subsections.

The table below summarizes all of our components that need to be powered by our DC voltage along with the total power in watts. We will have four different rails to power these components, one +5 volt digital rail, one +3.3 volt digital rail, one +5 analog rail and one -5 volt analog rail.

Component	Voltage	Current	Power
Display	3.3V	32 mA	0.1056
Touch Controller	5V	780 uA	0.0039
Backlight LEDs	5V	32 mA	0.16
Microcontroller	5V	3.2 mA	0.016
ENC28J60	3.3V	180 mA	0.594
MAX232	5V	8 mA	0.04
ICS4806	5V	240 mA	1.2
25AA640A	5V	5 mA	0.025
Amplifier (5)	+/-5V	2.4 mA	0.012 * 2 =0 .24
4:1 MUX	5V	1 uA	5.0E -6
2:1 MUX	+/-5V	40 uA	0.0002
1:8 Decoder	5V	3 uA	15.0E -6
ADC (2)	5V	2 mA	0.01 * 2 = 0.02
References (2)	5V	1 mA	0.005 * 2 = 0.01
DC Relay – Alarm (2)	5V	25 mA	0.125 * 2 = 0.25
DC Relay - Heater	5V	25 mA	0.125
AC Relay - Heater	5V	25 mA	0.125
TOTAL POWER			**2.915 Watts**

Table 28: Power for Components

Power Components

The input of our device will be 90VAC/240VAC, which will power the heater directly. Our input will convert the AC input voltage to a DC voltage of 7.5 volts, which we will use to power the rest of the components. TDK-Lambda Americas Inc makes the LS200-7.5 power supply that will do this for us. The LS200-7.5 produces 7.5 volts DC at 26.7 amps. It has an output power of 200W which is more than enough for our power needs. The LS200-7.5 also has an internal fan which is good since the heater has a maximum power of 550W. The power supply itself is a little bigger than what we want but since the power requirements meet our needs we can compromise. We want to have a switch on the device to power the device on and off. We also want to have a fuse for safety purposes. TE Connectivity makes many different kinds of switches with these integrated fuses. Since these switches come in different shapes and sizes, we will wait to decide on the exact one until we start building our device. We can also decide to put an emergency shutdown button on the device, but we will wait for that also until we determine if we definitely need it.

On the digital side of things we will need to power the display and ENCJ286, which run off of 3.3 volts along with the touch screen controller, backlight, microcontroller, MAX232, ICS4806 and 25AA6404A which all run off 5 volts. To do this we will use a universal power supply to take our input power down to 7.5 volts and use voltage regulators to generate our rails. We will have a +5 volt digital rail and a +3.3 volt digital rail to control all of these components. Since we are using an input voltage of 7.5 volts we will use voltage regulators to get our desired +5 volts and +3.3 volts to power our components. There are many different regulators on the market that can produce +5 volts and +3.3 volts. Texas Instruments makes the PTR08060W switching regulator that we will use to produce our +5 volt rail. The PTR08060W has an input voltage range of 4.5 volts to 14 volts, which satisfies our 7.5 volt input. It also has an output range of 0.6 volts to 5.5 volts that is controlled by a single external resistor. This resistor value is determined by the table on the PTR08060W data sheet. The PTR08060W delivers up to 6 A of current which is plenty for our power needs. For our +3.3 volt rail we will use a regulator that produces less current since we only need about 244mA of current for these components, which will reduce cost and noise. Linear Technologies makes the LT1129-3.3 low dropout regulator which has a 3.3 volt fixed output voltage. The LT1129-3.3 has an input voltage range of 4.3 volts to 20 volts and is capable of supplying 700mA of output current which will be sufficient for our components.

On the analog side of things we need to power the five amplifiers, 4:1 MUX, 2:1 MUX, 1:8 decoder, two ADC, two references, and the relays for the alarms and the heater. All of these components have a common voltage of 5 volts. The amplifiers and the 2:1 MUX have dual voltage of +5 volts and -5 volts. We will use the same power supply at 7.5 volts to control these components. We will have one rail for all the +5 volt components and use one voltage regulator to control this. We will also have another rail with -5 volts and use a switching regulator to produce the -5 volts. Using these regulators will also reduce the noise in our device, which is very important. Linear Technologies make the LT1129-5 low dropout regulator which has a 5 volt fixed output voltage. It has an input voltage of 6 volts to 20 volts and is capable of supplying 700mA of output current. Texas Instruments makes the PTN78000A switching regulator that we will use to produce out -5 volt rail. The PTN78000A provides positive-to-negative voltage conversion for loads up to 1.5 A. It has an input voltage of 7 volts to 29 volts, which satisfies our 7.5 volt input. The PTN78000A produces an output range of -3 volt to -15 volts that is controlled by a single external resistor. It also delivers up to 1.5 A of current which satisfies our needs.

The figure below is a power diagram of all the power supplies and regulators we are going to use to power our components. All of our parts are able to run off of 5 volts or 3.3 volts. As you can see we are using four different regulators in total, two that produce +5 volts, one that produces +3.3 volts and one that will produce our -5 volts needs for our dual components.

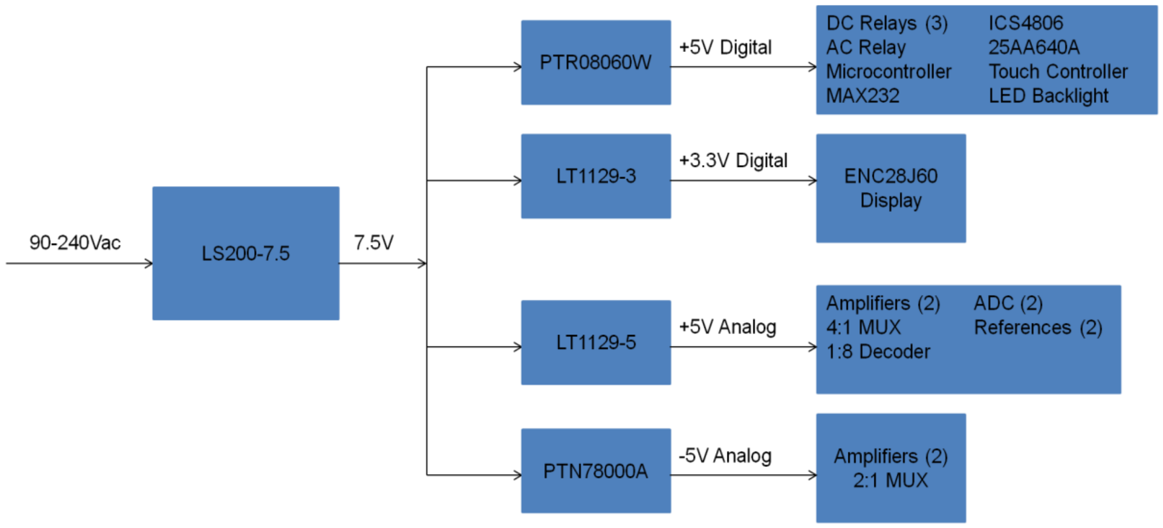


Figure 61: Power Diagram

Network Interfacing

Overview

When looking at the previous product used, it was important to find ways to make our design stand out and more desirable to the customer. One idea was to make it more user-friendly and accessible in the respect of the work place. Previously to see the temperature of the device, or devices in the instance that there is more than one, the user would have to constantly get up and check the readings straight from the device itself. However, these devices are running for extended periods of time, and to continuously stop what they are doing to check the temperature is disruptive and could truly hinder the amount of work a person gets done, especially if the devices are nowhere near their desk. In effort to ameliorate this issue, our device will not only have a user interface through the touch screen, but also through the network.

The initial idea was just to have the device hooked up to the network through an Ethernet connection and have a table available that displayed the current settings of each device. This way the user could check the information they need straight from their desk without having to interrupt what other project they may be working on. This idea was then taken a step further, instead of just receiving data from the device, making it available to send a command to the device while it is running that will change the current set point the device is at. This effectively allows a user to begin a run and monitor its temperature throughout without having to leave whatever else they might be working on, meanwhile should they be unhappy with the process and where the temperature is going they also have the ability to adjust the desired temperature.

Requirements

In order to this we need to have components that are compatible with the rest of the controller, an Ethernet connection to a computer on the network, and a program on the computer that receives data and sends back commands. These components must also need to be able to fit into our chassis in a compact and uncomplicated fashion. Even though our project is for only one controller, there is a possibility that a company may have more than one running at a time. This means that not only does this program need to send and receive data from one controller, but to multiple devices as well across a network. This can be done through either TCP or UDP. The data needs to be displayed in a simple, easy to read, format on a program that can run on various computers. It needs to display data for each individual device and include information such as its current temperature, its set point, and a timer showing how long each respective controller has been running. While the data is available the user needs to use this same program to send direct commands to the device such that the user can change the set point of a device as it is going.

Design

The design process for executing the network design is rather thorough. First we will discuss what chip and boards looked at in trying to find a way to connect to form an Ethernet connection to the device. If it is a board, it has everything on it ready to go, however with a chip there more parts still left to choose in order to complete a connection from one location to another. These parts then have things to consider such as packaging, and mounting. While it is fairly obvious on how a chip would be mounted, the other external components such as the Ethernet port are not as simple. The data needs to be sent across the network in a timely fashion. The Ethernet connection must not only connect to the board, but must also be compatible with the microprocessor. In terms of software, it is essential to know how data is making its way from the microprocessor and how the packets will be sent. Then it is essential to ascertain that the computer can receive said packets. Since we are going both ways in this design, it is also important to know how commands are sent back through the Ethernet connection to the microprocessor to change the set points in the middle of a run.

Once all of the networking capabilities are determined, the next step is to work on the user interface. First there is creating a program that can used on multiple versions of windows, and available on a desktop. Then the program must receive the data from the device and be able to input into a table. This live feed also needs to have a timer going so that the user is not constantly refreshing the program. This is where the user then needs to have a way to send a command to change the set point for any particular device at the users will.

Ethernet Access Candidates

Microchip DM320004

When first looking at Ethernet options it was questioned how difficult we wanted to make our lives. The DM320004 would have been by far the easiest option, mainly because it is a PIC32 Ethernet Starter Kit. As one can assume from its name “starter kit”, it came with everything we could need to start off with and more. The board included components such as: Ethernet, USB host, USB micro-B connectors along with a slew of software assistance such as debuggers and compilers. This product was ruled out because of the price and the amount of unnecessary baggage it came with. For one, it had its own microcontroller which is fairly redundant considering another microcontroller is already being used for the temperature controller itself. Not to mention its two USB compatibilities which are also unnecessary. It was also unsure whether or not the board would fit the space confines that we have with all of the extra components so from here it was deemed necessary to look for individual chips and go from there.

Ethernet Controllers

With the board ruled out an alternative approach was taken where instead of looking for an all in one we only purchase the specific parts we need to execute the desired task. Since we need this chip to interface with our micro controller it seemed like a good place to start looking for Ethernet chips would be the same place we got the microcontroller. When looking at Ethernet chips there are two distinctive categories, Analog and Interface Ethernet chips or Microcontroller Ethernet chips. Once again, since we have already chosen a microcontroller an Ethernet chip with the later would provide unnecessary memory and would be over kill for our design. Thus the focus was then put on the three Analog and Interface options that Microchip offers as compared in Table 29.

Part Number	Interface	Available RAM	Number of Pins	Price
ENC28J60	SPI	8 Kbytes	28	$2.03
ENC424J600	SPI and Parallel	24 Kbytes	44	$2.60
ENC624J600	SPI and Flexible Parallel	24 Kbytes	64	$2.81

Table 29: Microchip’s three Analog & Interface Ethernet options

All three of these stand alone chips are IEEE 802.3™ compliant with an integrated MAC, 10/100 Base- T PHY and have a SPI interface. The first chip is much smaller in comparison to the other two chips which is probably a result of its lack of parallel interface. While the last two have the same amount of available on-chip RAM buffer available, the ENC624J600 has twenty more pins than the ENC424J600 and is thus about twenty cents more expensive. Considering the amount of available RAM on our microcontroller, and the limited amount of space on our printed circuit board both of these options are far more than what we need. The only interface we need is SPI thus it is only rational to choose the ENC28J60 for our design purposes.

Package Types

In terms of packaging, the ENC28J60 Stand-Alone Ethernet Controller provided a few options as seen in Table 30. The considerations when discussing the packaging for this controller are similar to that of the microcontroller; however, for our design purpose, a different packaging type will be used. The main difference between the different types of packaging relies in the overall shape of the chip and the type or orientation of the leads. From the layout of our design, and the basic interface as seen in Figure 6, we can automatically rule out the QFN, Quad Flat No Lead Package. For one it is essential to have leads so that it can be soldered into our board with the rest of the hardware, and as also mention the quad layout would not be ideal and could cause complications in the pin out process due to the pines being on all four sides opposed to only two. Thus in discounting this option our focus will now resort on dual pins opposed to quads.

Package Type	# of Sides with Pins	Leads	Max Length	Max Width	Price
SPDIP	2	Through Hole	35.56 mm	7.49 mm	$3.03
SOIC	2	“J”	17.90 mm	7.50 mm	$2.83
SSOP	2	“J”	10.50 mm	5.60 mm	$2.79
QFN	4	None	6.00 mm	6.00 mm	$2.99

Table 30: Packaging types for the Ethernet chip

The next few packaging types that will be discussed are all the same general size and layout. The next one we will look at is the SPDIP (Skinny Plastic Dual In-Line) package which is the largest, most expensive and uses through hole leads that can be inserted into the board. Due to our desire to conserve space on our printed circuit board, and compared to the dimensions of the remaining options, this can be immediately removed from contention. The SOIC (Plastic Small Outline) is much smaller than the previous option, but when compared to SSOP (Shrink Small Outline) it is still a bit larger and a bit more expensive. The “J” leads of both the SOIC and the SSOP will allow the chip to be soldered directly to the board far more easily than the through hole pins which is another reason these two are more desirable. In the end the ideal package for our design will be the SSOP mainly because it is the smallest, has the ideal leads, and is also the most affordable. As a result the final part number for our chosen Ethernet chip will be Microchip's ENC28J60/SS.

Microchip ENC28J60/SS

Now that the specific part number has been determined, we can further analyze this part in effort to figure out what other components are needed to make the Ethernet connection from the microcontroller to the user's personal computer. We also need to gain a deeper understating of how data will be transferred once this connection is made. The figures used within this section were extracted from the parts' Microchip data sheet. From here we can tell that the next we need to look at is the Ethernet Transformer and from there the RJ45 jack, another alternative is to use a jack that has the magnetic already built in.

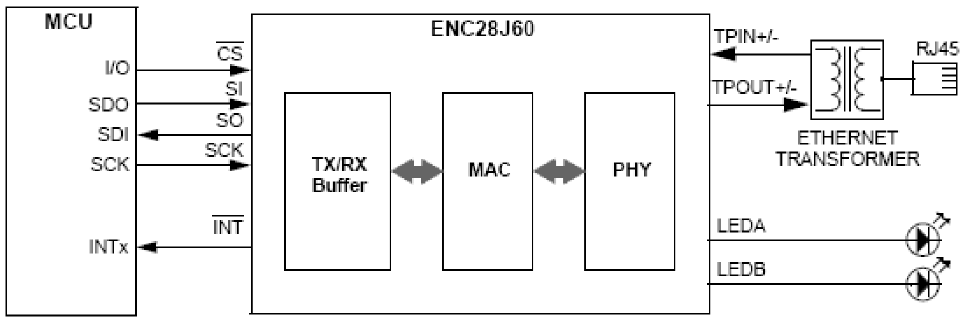


Figure 62: Typical ENC28J60-Based Interface

From Figure 62, we can see the flow of data from the microcontroller to the Ethernet controller and from the Ethernet control out to the RJ455 jack that will be connected with an Ethernet cable to the user's computer. Before we can start understanding how data is sent out of the device, we need to pick the parts from the Ethernet controller to the through the RJ45 Jack.

Ethernet RJ45 with Transformer versus MagJack

As shown in Figure 62, the Ethernet Transformer is connected to a RJ45 jack, from here the hunt for a part can be narrowed down to some extent. Initially it was found that the main choice in trying to pick the exact part was the mounting. Since we already have a chassis that we want to use it is essential to make whatever RJ45 that we use adaptable to this chassis along with easily accessible to the user. Without knowing the difference between the various types of mounting the first guess was to assume that a surface mount would be it and that there would be no further discussion, but upon further examination, it was discovered that a surface mount can be attached through just a simple cut out, which while easy enough would not provide any actual support to holding this component in place. So from there the various types of mounting were then looked into further.

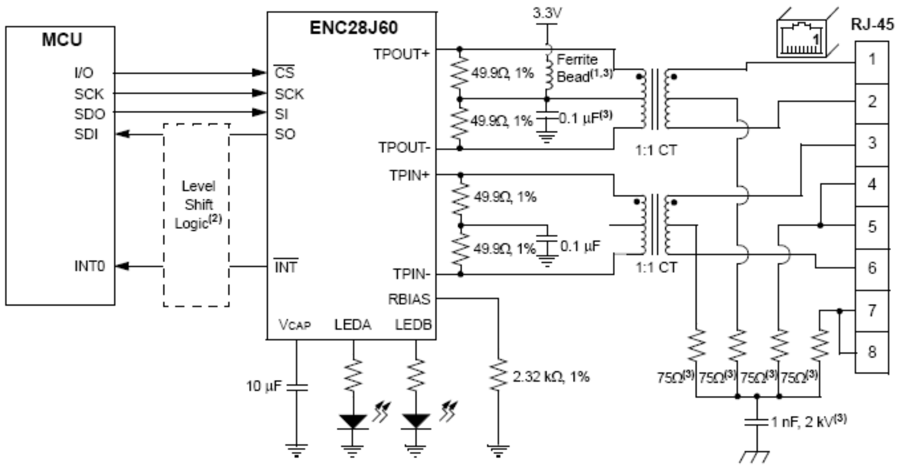
Mounting Type		Company	Part Number	Length	Width	Height	Price
Panel Mount	Bulkhead	TE Connectivity	A-RJ45KU-R	30.5 mm	25 mm	25 mm	$15.90
	Flange	TE Connectivity	1546908-1	44.45 mm	44.45 mm	23.9 mm	$19.33
	Flange, Through Hole	Amphenol Commercial Products	MRJ-5481-01	22 mm	29 mm	14 mm	$13.02
Surface Mount		TE Connectivity	1-338088-3	12.7 mm	14 mm	14 mm	$1.87
Through Hole	Edge Mount	Sullins Connector Solutions	SMJ401-S88W-DS-01YG	16.66 mm	18.40 mm	11.55 mm	$2.45

Table 31: RJ45 Mounting Options

When being the search for the proper part, it was important to find something that can easily be accessed through the chassis but is also small. In Table 31, five different mounting types were compared along with a specific part to represent each of them. Some important factors in picking the best part for our design are cost, size, its ability to adapt to the chassis, and its ease in implementation. While both the Surface Mount and the Through Edge are the smallest and cheapest options, upon further research it was proven that neither part would suit our needs. The surface mount would have to go directly into our printed circuit board, which not have the external access we need and with its connection to the transducers will take up more space than what is available. Meanwhile, the Through Hole Edge Mount proved to have similar if not even more issues. This too would have to be placed onto the printed circuit board itself, but in doing so it would literally have to be mounted through the holes which are not entirely compatible with our current design.

This then leaves us with the three types of Panel Mounts, the bulkhead, flange, and the flange through hole designs. A panel mount would work better than the previous options because they can be mounted on the side of the chassis by cutting a hole to put them through and then screwing them to side panel. This leaves them at a sturdy and easily accessible ninety degree angle so that no stress is put on the Ethernet cord itself nor is it taking up space on the printed circuit board. The Bulkhead Panel Mount was structurally sound but as visible by the dimensions, this connector is a bit more heavy duty than what we needed. This design would be useful in an environment where the receptacle would be in between heavy industrial machinery opposed to being part of a device that is smaller than a shoe box that is just getting plugged into a computer in an office. The flange panel mount is not as industrial, but is still the largest and most expensive of the given options. It consists of the jack being shielded before being placed in the receptacle assembly and finally locked in place with a flange mount. This too, while secure is far more than what we need for our design. Finally this brings us to the flange through hole panel mount, where like the other panel mounts can be attached directly to the chassis however it is much smaller and is the most affordable compared to the other two panel mount options. in order to mount this part to the chassis, a 16 mm by 14 mm rectangular hole would have to be cut along with two 3.45mm in diameter circular screw holes. From there the part can be slipped in and secured by tightly locking it in place with a nut.

Now with this being said if this Ethernet jack is used we then have to find an external transformer and get those put together onto a circuit board which will take up even more space in comparison to the other components that will also be there. A possibility to do this is by using two of the Pulse Electronics Corporation's H1164NL Module Transformer Ethernet LAN and placing them in a circuit layout such as seen in Figure 63. This transformer is used for 10/100 Base-Tx port applications and is compliant with IEEE 802.3. If this route is taken then our printed circuit board will end up being extraordinarily cramped with all of these components.



Note 1: Ferrite Bead should be rated for at least 80 mA.

2: Required only if the microcontroller is operating at 5V. See **Section 2.5 "I/O Levels"** for more information.

3: These components are installed for EMI reduction purposes.

Figure 63: Ethernet Termination and External Connections

There is however an alternative solution that combines both the RJ45 and the Ethernet transformer and it is purely magic. Bel Fuse Inc created a connector series known as "MagJack" the reason for this is that while it is a RJ45 connector with a 10/100 Base-TX speed, it also has the magnetics already included in it. Therefore it is both the jack and the transformers that are needed combined into one package. Its mounting type is a panel mount, through hole, right angle and costs $5.66. While this is not the ideal mounting type, this individual part would make the overall Ethernet design far less expensive and far less complex. The only parts that would be left to purchase are the four 49.9Ω resistors, the two 0.1µF capacitors and the one Ferrite Bead. Typically these components are rather inexpensive making this option far more affordable than buying a RJ45 jack and Ethernet transformers independently. To ensure a strong connection with the Ethernet cable we can make a small hole in the chassis that the cable will be fixed within.

Transmitting and Receiving Packets

As common practice for networks, the internal MAC (Media Access Control) addresses will generate the fields that will be transmitted. The microcontroller will contain all of the data from the PID implementations, along with their current settings. It is this data that will be sent through the buffer memory of the ENC28J60 for transmission to the user’s computer. Each packet will need a control byte to accompany it as it is sent. The layout of transmission is in the form of a stack. First is the control, which is followed by the data, which is then trailed by a status vector that will point to the next designated location. When data is being sent back to the controller, the microcontroller will use the RBM SPI and start reading in the information. In the instance that RAM is needed, the Ethernet controller will manually calculate the proper amount of space needed and buffer itself accordingly. The two LED bulbs shown in Figure 62, LEDA and LEDB, are there for the purpose of informing the user when there is network activity.

TCP/IP versus UDP

Now that the hardware is all figured out, and we have a firm connection from the High Precision Temperature Controller to the computer and we understand how the Ethernet controller is sending and receiving data, we can look further into the network protocol that the Infrared System Development Corporation’s computers will use. At this stage in the network system data is transferred across the Transport Layer. This layer follows, and thus relies on the Internet Protocol from the previous Network Layer. The two main types transport layers we will compare are TCP/IP and UDP, both of which are supported through the Serial Peripheral Interface of the Ethernet controller. Some of the their strong differences are compared in Table 32. Transmission Control Protocol and User Datagram Protocol are used for transmitting data from the High Precision Temperature Controller to the company’s network.

Table 32: Comparison of TCP/IP and UDP

Connection Type	Reliability	Dependency on Connection	Delivery of Packets	Example Uses
TCP/IP	Reliable	Dependent	Ordered	HTTP, FTP
UDP	Unreliable	Independent	Unordered	Streaming, Skype

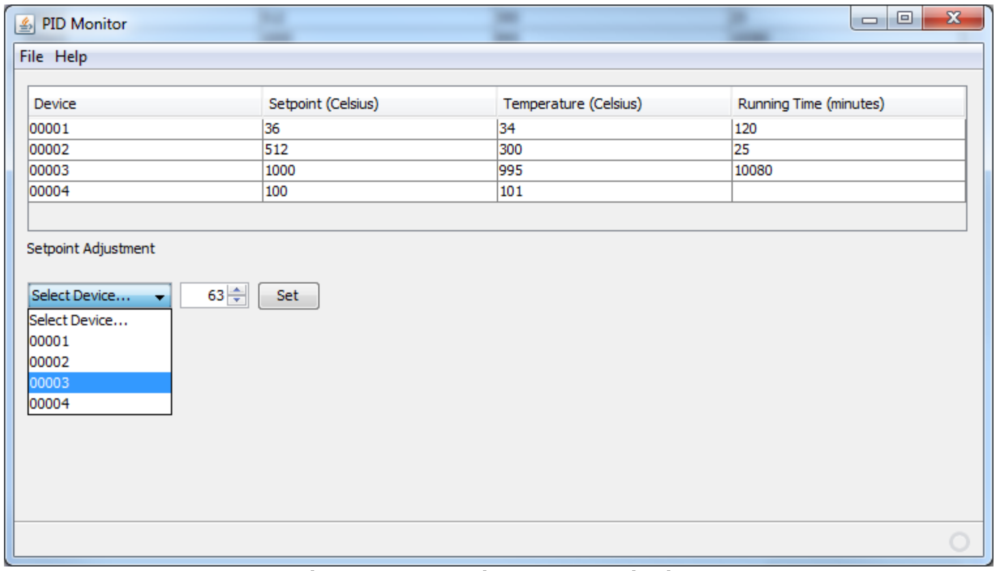
TCP/IP is so named strictly because the TCP breaks the data down into individual Internet Protocol packets before they are sent, and assembles them when they are received. TCP/IP waits for a connection to be made before sending data. Once this connection is made, the data it is sent in an orderly fashion which is done through both congestion control and flow control which is what assists in making this option 100% reliable. If a packet is unable to send, the server will request the lost information to be sent again. Typically data is sent in a specific order, however, if an incidence occurs where the packets are sent in an incorrect order the TCP will buffer the data until they can be arranged in the proper sequence. One of the reasons this protocol is such a strong contender is because of its ability to send data bi directionally. This is important because on top of reading data into a table the program on the user interface will also have the ability to change the set point of a device. In order to do this we need to have the ability to send a command back to the High Precision Temperature Controller to initiate the change.

Meanwhile UDP is named such because the information is sent in the form of a datagram. Unlike the alternative option, UDP does not need a connection to set up and does not care if the network can handle the packets or not. Rather it begins sending data in an unordered process and ignores any segments that may get lost. The reason the packets are not sent in any specific order is because each packet is independent of one another and the data can only be sent in one direction. One of the positives of this process and what makes it a contended despite its unreliability, is that this process is good for servers answering to a multitude of users. It supports packet broadcasting and multicasting where information is sent to all on the network and not just sending information to one host which will then go to another host in a chain of internetworking. Instead it can just to all of the subscribers on the local network. Despite this massive pro, then information we are sending cannot afford to get lost and for the purposes of the user interface on the computer, all of the data needs to appear without needing to wait to buffer. Also, since the data we want to read into the user interface may be coming from several devices the lack of congestion control in the UDP is concerning. Not to mention that part of our design for this network is to be able to send information back through to the devices, since UDP's connection is only one way and attempts and handshaking would be rather complicated. If we were streaming constant live feed or videos, then this approach would be much more desirable, but since this is not the case we will be using the TCP network.

User Interface

This is not the user interface of the device itself, but rather the user interface through the computer. As we have previously mentioned one of main reasons we are connecting our device to the network is so that a user has access to the current devices' settings from the convenience of their desk. Once the information is sent from the Ethernet controller, through the TCP network the data is then accessible from a program on the user's device. We want the program to be easily accessible and thus will have available shortcuts that can be placed on the desktop if desired. Another important factor is the visual appearance and how user friendly the program is. Thus, instead of writing a long code a gooey was and will be used to format the overall layout of the program. For the design see in Figure 64 we used a GUI editor available through Java's NetBeans.

Once this program is opened a display such as that see in Figure 64 will appear. As you can see, the information is visible through a concise and easy to follow table that shows the device, the set point desired, the temperature that it is actually at and the a timer for how long the device has been running. Most, if not all of this information can be found in the PID process within the microcontroller. The reason there is room for multiple devices, is that while right now we are only creating one controller this company usually has more than one Blackbody source being tested at time, which would imply that more than one High Precision Temperature controller will be used. As a result we have added space for multiple devices to accommodate this possibility.



Device	Setpoint (Celsius)	Temperature (Celsius)	Running Time (minutes)
00001	36	34	120
00002	512	300	25
00003	1000	995	10080
00004	100	101	

Figure 64: Sample Program Display

When the idea was proposed to have this information available on the network the idea was discussed and the further built on. Instead of just having a one way connection we could fully utilize the TCP and send data or commands back to High Precision Temperature Controller. This concept is utilized in the small menu beneath the table. From here the user can choose which device they want to change. Once that is decided the user can use the small field to the right and input the set point they want to see the device at. Once this is set the information will go back through the network and the set point will be augmented accordingly. Thus the computer interfacing is complete from the connection of the microcontroller to the ENC28J60 to the Ethernet transforms and the RJ45 Ethernet jack, across to the TCP network to the computer. This process is then repeated in reverse order in sending data back. Overall with the use of the Computer Interface our High Precision Temperature Controller will become even more user friendly.

Final Design

Shown in the block diagram below is our final design. As shown from the diagram, there are 3 main devices we will need to communicate with inside Infrared Systems Development's black body sources: the thermocouples, heater and fan. The fan is communicating directly with the microcontroller with no intermediate devices. Control of the fan will be limited to on and off commands. The heater inside the source is controlled by either an AC or DC relay depending on the heater type, and will turn on and off at different power levels according to the output of our PID algorithm. The thermocouples inside the source are the only devices communicating out of the source. Their end goal is to transmit their readings to the processor; however, this signal is very small and will need to go through an amplification process first. The processor will choose which thermocouple to read at that instant and its input will be passed through by a multiplexer. Once selected, this amplified signal will become digitized through our analog to digital converter and finally end up at the processor. The processor will then communicate with the EEPROM to access data from the thermocouple look up tables and implement the algorithms mentioned in the temperature processing section.

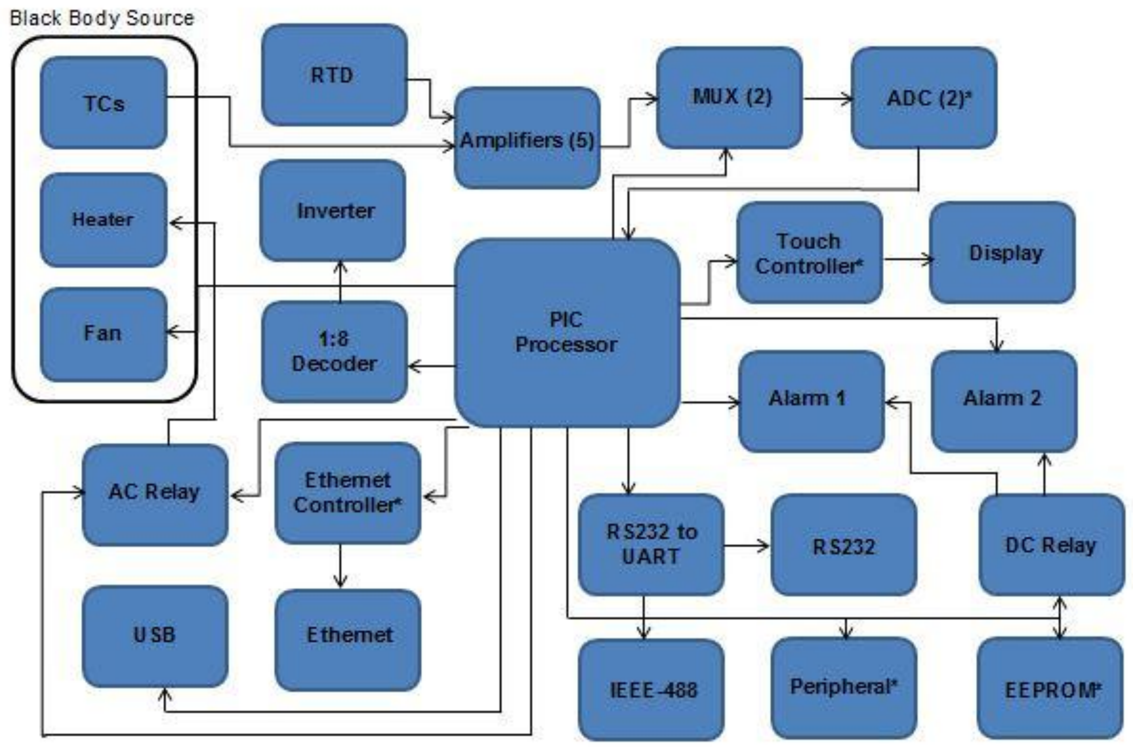


Figure 65: Final Design

Our touch panel display will need to communicate with the touch controller which is getting its instructions from the microcontroller. The display will also need to send information back to the processor depending on certain user inputs. Alarm settings are one of these user inputs, which will be displayed on the touch screen and will go off when the temperature of the source is approaching its set point. These alarms are controlled by a DC relay which will turn on and off appropriately to trigger the alarms.

Data can be taken from the microcontroller or uploaded to via many communication lines included in our design. The user can even monitor the temperature and status of the heating source from an application on a computer desktop via communication with the Ethernet. The Ethernet controller controls this port and allows data to be sent to the user remotely. USB, RS232 and IEEE-488 are also available communication ports included in our design. An example application of data transfer via the USB would be an update to a sensor look up table, but these devices can be utilized in many different ways.

Overall, each of our subsystems collaborates at a central control unit to constantly monitor the temperature of the source and the user inputs. The intuitive user interface also makes it easy for the user to communicate to the controller what kind of data is wanted. Collectively, each device plays its part to ensure maximum accuracy and quick response times as efficiently and economically as possible.

Testing

Microcontroller Testing

To test the Microchip PIC24FJ256DA206, Microchip offers the MPLAB REAL ICE In-Circuit Emulator. This device is capable of debugging through MPLAB's IDE. It offers single-step debugging with variable inspection and modification. In terms of hardware debugging, it features eight logic input/output probes. Also, it has SPI trace capabilities that will allow us to look at the data that is being sent through these lines. This is extremely important because all peripherals on the microcontroller will communicate via SPI. To interface the In-Circuit Emulator with our temperature controller, we will use a CAT-5 connection (Pins 1 – 5) on our temperature controller with connections to V_{DD}, V_{PP}/MCLR, PGC, PGD, and V_{SS} pins on the PIC24FJ256DA206. The testing procedure will ensure that our temperature controller is responding correctly to an appropriate heat stimulus. We will use a mV power supply to serve as our virtual heat source. We will use an SPI trace to read what values are being sent from our ADC to our microcontroller. Based on this value, we will look at the relay control output to detect the percentage of power used to power the heater by looking at the duty cycle of our output. We will compare the response to the response of our simulated PID temperature controller. This will determine if our microcontroller is correctly responding to the temperature stimulus. We will also check the functionality of the two onboard alarms. We will use the I/O probes to determine the predicted alarm response with the actual alarm response. We will have display test to ensure that all portions of the display are addressable and working. We will also use an SPI probe to determine if the correct data is being written to the display controller. We will have a display test to ensure that all pixels on the display are functioning correctly. We will do this by sending out the correct GD0-GD15 values to make all blue, red, and green. We will ensure that the our touch screen is functioning correctly by using the SPI probe to determine what X+, X-, Y+, and Y- values are being returned when the screen is being pressed. Another feature available to us is the JTAG Boundary Scan using the JTAG pins (TMS, TDO, TCK, and TDI). This test will use Shift registers to go through all of the different I/O pins and the external JTAG controller will provide a set of instructions to the pin and read its appropriate response. This will allow us to test the functionality of all 64 pins quickly and easily. The JTAG controller we will be using will be the JTAG USB-1449.1/IE. This has software, ScanExpress Runner that will allow us to quickly and easily write instructions and evaluate the responses. Pictured below is the MPLAB REAL ICE emulator and how we will connect it with our PIC24.

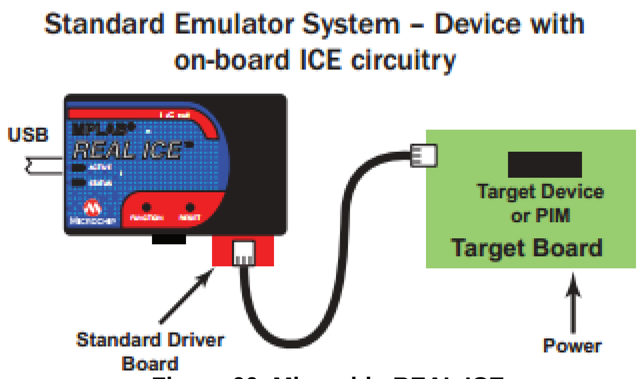


Figure 66: Microchip REAL ICE

Peripheral System Testing

To ensure that we are able to utilize all of the peripherals associated PIC24FJ256DA206, we will need to determine a testing protocol that will be able to quickly and easily confirm that the all devices are working properly. With the computer peripherals, we will use a special command that will instruct the computer to respond with the same message that it just received. If the device is working correctly, the microcontroller will get the same bytes back. We will compare the bytes received with the expected bytes and if they are equal, we know that the particular computer interface is operating correctly. If we get a different response, we know that the peripheral is not working correctly and we will display the error to the user and perform an appropriate action. To test 25AA640A, we will have a byte that will be written to an unoccupied memory address. After the write operation, we will read the value contained at the memory address. If the value we read matches the value we wrote, then we know that we are able to communicate with the external EEPROM. Otherwise, we know that there is an error communicating with this device and we will display this error and take appropriate actions. To test the NHD-4.3-480272MF-ATXI#-T-

1, we will run a color test that will vary the screen from red to green to blue, the tester will confirm if the display performed as expected. If it did not, then we will notify the user and take appropriate actions. To test to see if we can communicate with the TSC2046 touch controller, we will wait for a touch input at the touch screen during a predefined window. If there is not a touch detected, we will timeout and report that there is an error with communicating with the TSC2046.

Display System Testing

To test the touch screen display we will write some test code to have the microcontroller turn the screen different colors. This test will make sure that all of the pixels are working properly and that it can produce the different colors. We will also have it display writing in different areas of the display. Once we know that no pixels are out then we can go on to test the touch sensing. We can write more test code to but buttons on the screen. If we have time it would be a good idea to make a series of different shapes and sizez of the buttons. This will help us detemine how small we can make the buttons while still being easily and accurately touched by the user. One way to test the simplicity of the user interface would be to have someone outside of the group go through the user interface seciton of this paper and try to navigate their way through the menus. This will help us troubleshoot any confusion before we actually write all the code for it. Once we have the code written and everything hooked up we can fix any other bugs that we find when we come to it.

Thermocouple Testing

Infrared Systems Development's black body sources have a thermocouple connector embedded into them. We can use these straight connections to verify the exact voltage output coming from the thermocouples and therefore determine the correct temperature. From this direct reading, we can determine the accuracy of our subsystem.

Multiplexing Testing

To test the multiplexer's reliability to transmit our measurements, we will design a series of test circuits to measure data accuracy, switching time and fluctuations in operating temperature conditions. For testing purposes, we will order some through hole parts as well as the surface mount parts we will need for our printed circuit board. To measure data accuracy, we will use monitor the known input signal and keep a constant watch on the output to check for any offset or noise. We will do this for each input and vary the temperature from 25 to 40 degrees Celsius each time. Here, in this testing environment, we will also observe the effects of charge injection on our data.

Power Testing

Once we get all of our components and our PCB board we can start to test to make sure all of our components work. We can test our power supply easily just by plugging it in and making sure we are getting our desired output. Next we will start by conducting a smoke test by wiring up our board to an input power and to ground. If nothing catches fire, blows up or starts to smoke then we can continue onto testing our other components. Once we conduct this test we will test each component that needs power one by one. We want to make sure that everything is soldered right and that there are no shorts or any other issues with the circuitry. This process can be tedious but it is necessary to do so that we know our components will work properly. Once we test each individual components we can test the system as a whole. Hopefully everything will work but if not then we will have to conduct a series of other tests to find out what is causing the issue. If something with our PCB board is soldered wrong this will be an issue if we have to send it back to get fixed because it will waste time and money. We also need to be very careful when testing our components because we won't have many spare parts lying around. Once we test to make sure that everything is being powered correctly we can go onto other testing and troubleshooting to get our device working the way we want it to.

Complete System Testing

The temperature controller needs to accurately regulate the temperature of a blackbody source with +/- .1°C of error. After the blackbody source and controller undergo calibration, testing will occur to confirm that the controller has been calibrated correctly.

To test the controller and blackbody source, there are two main procedures to follow: Data and Stability. Both processes will check to see if the controller and source were correctly calibrated. No modifications can be made to the offset points or values during testing.

During Data, the controller will bring the blackbody source to the temperature indicated by the offset points minus 1.0°C. For example at an offset of 101.0°C, the temperature controller will bring the heat source to 100.0°C. Once the source reaches stability, a comparison will be made between the display temperature and the external temperature as read from the external thermocouple that is set up in parallel with the internal thermocouple used to calculate the display temperature. If there discrepancy between the two temperatures is greater than .1°C, the modifications can be made to that offset value. Since we are using linear interpolation that only utilizes the two closest points with equal weighting, modifying one of the offset values, should not affect the temperature at any other offset point. However if any modification is made to any offset value during the testing phase, the blackbody source and controller must be retested at all other offsets to ensure that the modifications did not have a negative effect on the accuracy of other points.

During Stability, the blackbody source will be brought to the maximum temperature rating of the source. Once stability has been reached at that temperature, data recording will begin. Once every hour, for five hours, a recording will be made copying the display temperature and the external thermocouple temperature reading. Like Data, these readings should be within +/-.1°C. Unlike Data, changes cannot be made to the offset values during Stability. If any changes are required, the controller and source must be recalibrated.

Administrative Content

Group Responsibilities

In the planning phase of our project, we decided to break down our design into 4 major subsystems: microcontroller and communications, display, temperature sensing and PID optimization. Since there are 4 members in are group, this break down seemed appropriate and distributed the workload fairly. We also assigned miscellaneous research and tasks that did not easily fit into a subsystem to group members according to their workload and interest. The block diagram shown below presents the break down of our design and details the group member responsible.

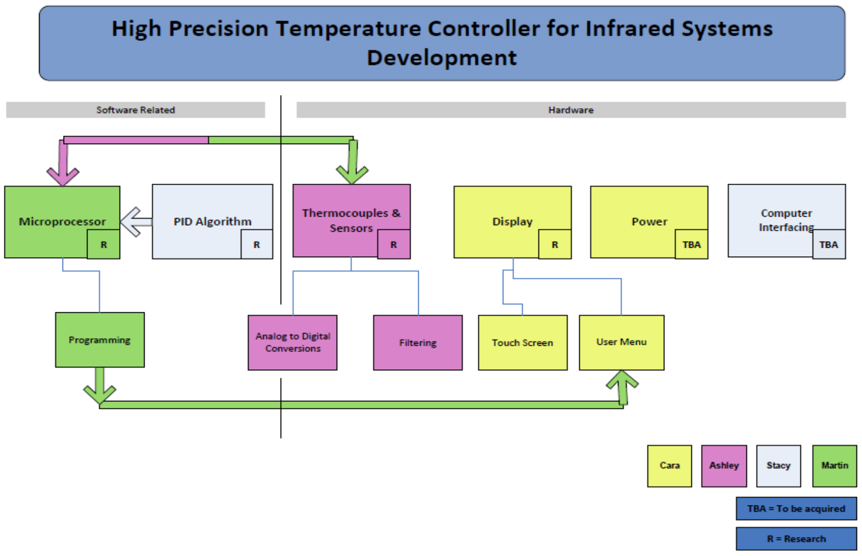


Figure 67: Subsystem Block Diagram and Assignments

As you can see from the diagram, the design research and tasks were distributed fairly. Martin Trang is responsible for the microcontroller, programming and any interfacing devices that communicate with the microcontroller. This is a huge task on its own and therefore, we did not assign Martin any more tasks. He was given this subsystem because of his interest in embedded systems and his vast experience and knowledge in C programming. Ashley Desiongco is responsible for the temperature sensing subsystem which includes research and design of devices embedded in Infrared Systems Development's sources and also many electronic devices needed for data collection. This also is a huge effort by itself and therefore, we did not assign Ashley any more tasks. She was given this task because of her interest in electronics and her experience with circuit design. Cara Waterbury is responsible for the display and power subsystems. She not only had to research and choose an appropriate touch screen and controller, but she also had to design the user interface according to our sponsor's specifications. She was given this subsystem because of her interest in touch panel technologies. The power system research and design was also done by Cara, and she had to work with Martin and Ashley's subsystems in order to find and account for power to all the devices including her own. Stacy Glass is responsible for the PID algorithm and computer interfacing subsystems. Her research on the PID algorithm optimization is critical for the quality of our controller and therefore has been her main focus. She was given the task because of her experience working with genetic algorithms. She was also assigned the computer interfacing because it was her idea to allow users the capability to monitor the temperature remotely via a desktop application.

Budget/Cost Analysis

Before we had started thoroughly researching and designing the devices needed for our project, we drafted a rough estimate budget based upon little initial research done via the internet. We know from our sponsor's requirements that the finished, ready for production controller should cost no more than $500.00 to manufacture. This allows Infrared Systems Development to sell the controller still at a reasonable cost to consumers, but still make a profit. This cost only includes components found within the controller itself, but for our project, we will probably spend more than this manufacturing budget for prototyping, testing and ordering spare parts in case of device failures. In the chart below is a rough budget of what we had previously estimated before we started the project, and the cost of the desired parts we have now discussed.

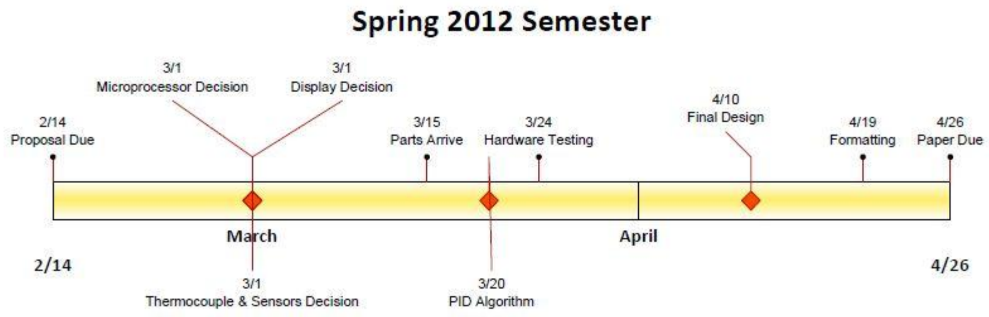
Device	Estimated Budget	Desired Parts Cost
Microcontroller	$10.00	$7.03
Memory	$2.50	$0.90
A/D Converters	$15.00	$8.91
Misc. Peripherals	$15.00	$6.60
Misc. Electronics	$30.00	$14.42
Touch Screen	$150.00	$38.00
TOTAL	$222.50	$75.86

Table 33: Budget

As you can see, we over estimated for many of our components. For example, we estimated $150.00 for the touch screen previously, but since then have found a perfect candidate that meets all of our requirements for a mere $38.00. The miscellaneous electronics category encompasses all the small, but necessary, components in our design such as: amplifiers, multiplexers, decoders, inverters and voltage references. The miscellaneous peripherals category encompasses all the communication devices in our design such as: Ethernet, USB, RS232 and IEEE-488. Certain components such as the thermocouple extension wire and alloy pins and connectors were not included in our cost analysis because Infrared Systems Development manufactures custom thermocouples and extension wire. All final design components will be fully funded by ISDC and any testing equipment costs will be co-funded by all four of our group members. Texas Instruments has also given us an allowance of $200.00 worth of TI parts.

Milestones/Schedule

Early on in the planning stages of the project, we were asked to make a milestone timeline outlining the major events in both the spring and fall semester. We drafted a schedule before we had done major research into our specific subsystems and pre-mature to the full understanding of senior design 1. As you can see by our timeline below, we have a handful of key milestones planned out for both academic semesters. The dates at which the spring semester schedule was set were ideal and ambitious and we slowly realized how unrealistic they were. We initially wanted to order sample parts to see what kind of components we were going to be dealing with, but we decided because of cost and time constraint that we would save this until the fall.



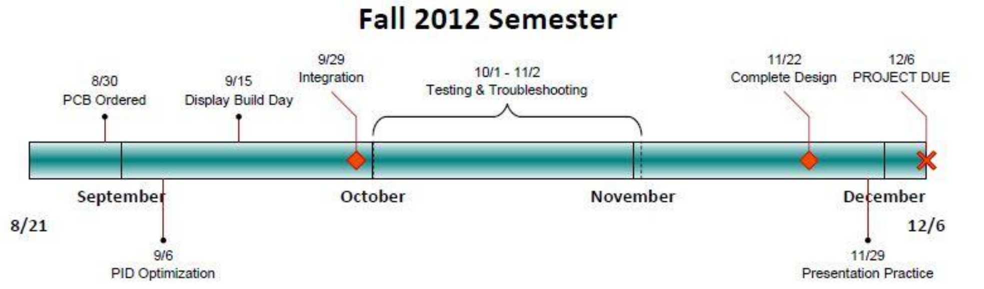


Figure 68: Milestones

The fall semester milestones are still very much realistic. We plan to stick to this schedule in order to maintain a steady development and testing plan for our design. We would like to include our sponsor as much as possible in this development and will therefore need to consider their schedule with ours. These dates may be changed closer to the fall semester, due to unforeseen events but we will still complete the milestones in a reasonable proximity to the scheduled dates. Each group member will be enrolled in at least 3 classes and be working 20 hours a week minimum during the semester; therefore, time management will be critical in order to complete our project in a timely manner.

Appendices

Table of Tables

Table of Figures

Works Cited

LS200 – Power Supply Datasheet
http://us.tdk-lambda.com/lp/ftp/Specs/ls200.pdf

PTN78000A – Switching Regulator Datasheet
http://www.ti.com/lit/ds/slts246b/slts246b.pdf

PTR08060W – Switching Regulator Datasheet
http://www.ti.com/lit/ds/slts289f/slts289f.pdf

LT1129-3/LT1129-5 – Switching Regulator Datasheet
http://cds.linear.com/docs/Datasheet/112935ff.pdf

Chassis, Heater
http://www.infraredsystems.com/index.html

Display Research
http://www.newhavendisplay.com/

Newhaven 4.3" Display Data Sheet
http://www.newhavendisplay.com/specs/NHD-4.3-480272MF-ATXI-T-1.pdf

Orise OTA5180A Data Sheet
http://www.newhavendisplay.com/app_notes/OTA5180A.pdf

Graphics Controllers
http://www.microchip.com/

Texas Instruments Parts
http://www.ti.com/

Crystalfontz 3.5" Display
http://www.crystalfontz.com/product/CFAF320240F-035T-TS

Thermocouple Introduction and Theory - Omega
http://www.omega.com/temperature/z/pdf/z021-032.pdf

Lookup table for RTD
http://www.omega.com/temperature/Z/pdf/z255.pdf

Calculating Temperature Coefficient (Tempco) and Initial Accuracy for Voltage References
http://www.maxim-ic.com/app-notes/index.mvp/id/3998

64K EEPROM

http://ww1.microchip.com/downloads/en/DeviceDoc/21830E.pdf

PIC24FJ256DA206
http://ww1.microchip.com/downloads/en/DeviceDoc/39969b.pdf

ENC28J60
http://ww1.microchip.com/downloads/en/DeviceDoc/39662c.pdf

IEEE-488
http://www.icselect.com/pdfs/4816_ds.pdf

TSC2046
http://www.ti.com/general/docs/lit/getliterature.tsp?genericPartNumber=tsc2046&fileType=pdf

Looking up Ethernet Parts
http://www.digikey.com/

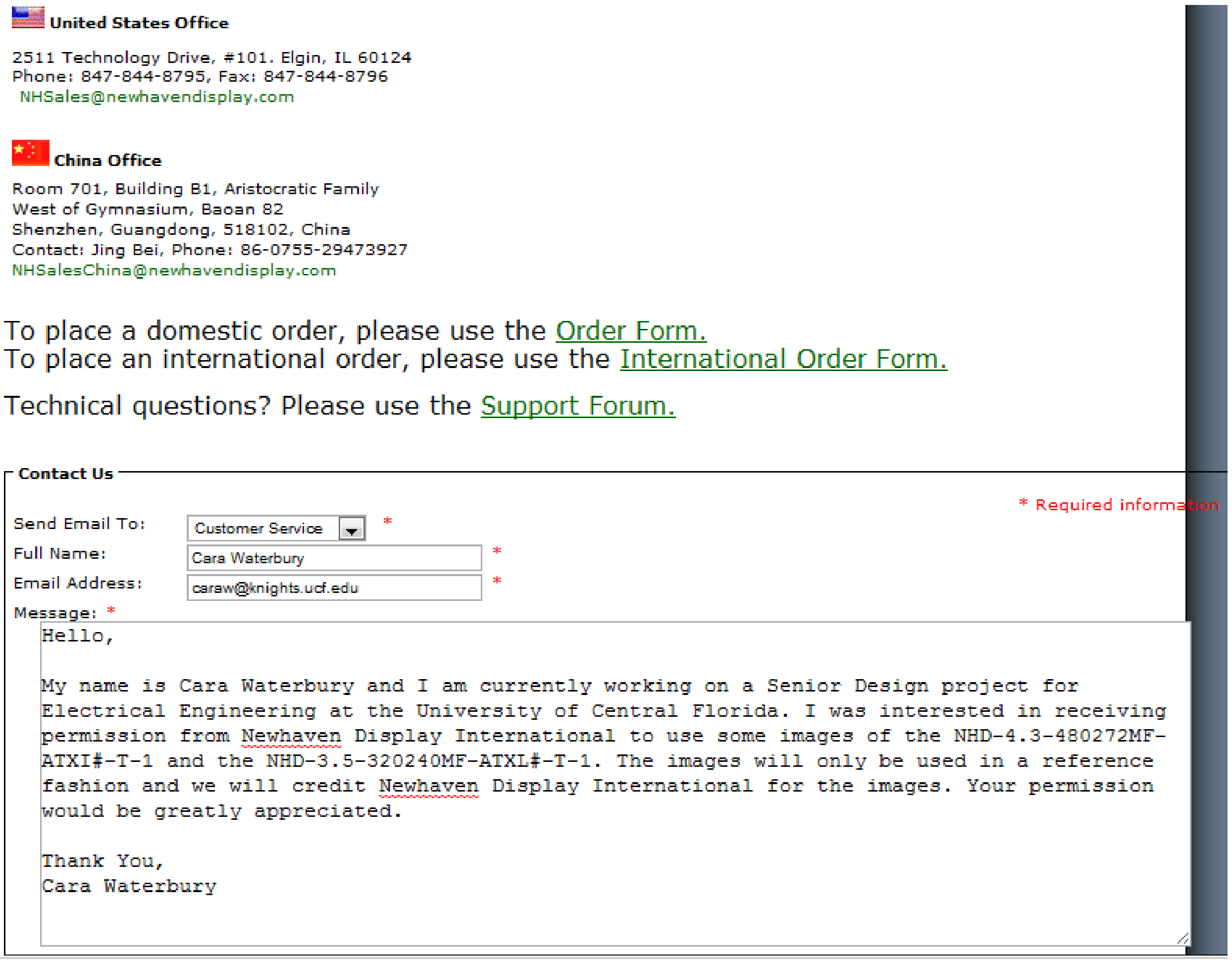
Networks Research
Notes from Computer Communication Networks EEL 4781

PID Research
http://arri.uta.edu/acs/jyotirmay/EE4343/Labs_Projects/pidcontrollers.pdf

PID Research
http://www.vandelogt.nl/datasheets/pid_controller_calculus_v320.pdf

Cascade and Feed Forward PID Research
http://www.engr.uconn.edu/control/pdf/isa04-1.pdf

Permission Emails



United States Office

2511 Technology Drive, #101. Elgin, IL 60124
Phone: 847-844-8795, Fax: 847-844-8796
NHSales@newhavendisplay.com

China Office

Room 701, Building B1, Aristocratic Family
West of Gymnasium, Baoan 82
Shenzhen, Guangdong, 518102, China
Contact: Jing Bei, Phone: 86-0755-29473927
NHSalesChina@newhavendisplay.com

To place a domestic order, please use the Order Form.
To place an international order, please use the International Order Form.

Technical questions? Please use the Support Forum.

Contact Us

* Required information

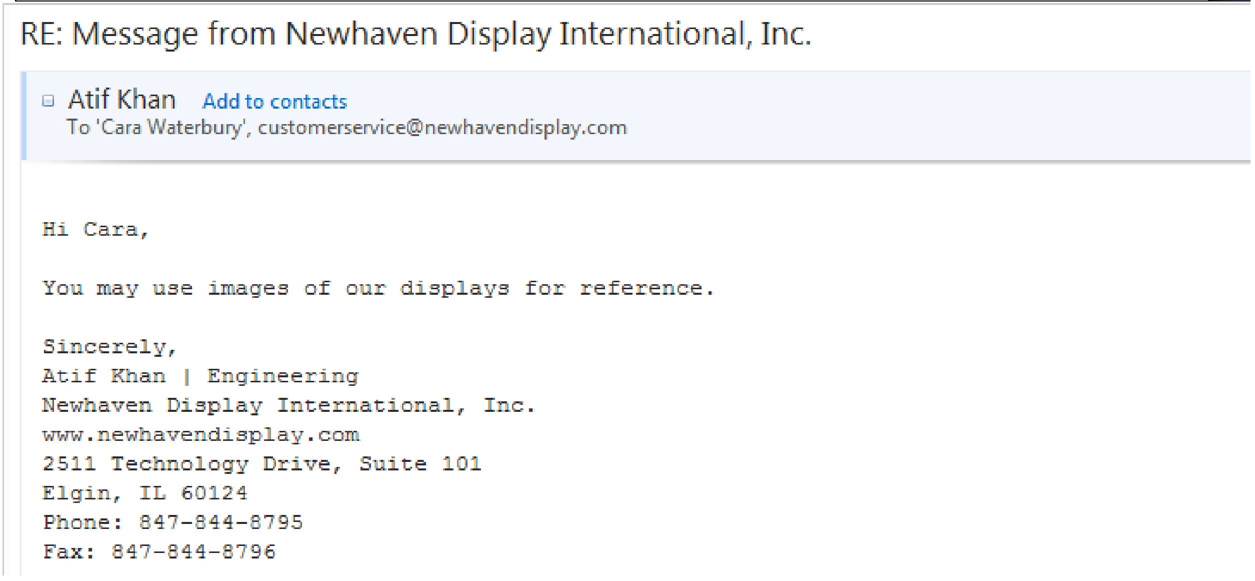
Send Email To:	Customer Service *
Full Name:	Cara Waterbury *
Email Address:	caraw@knights.ucf.edu *

Message: *

Hello,

My name is Cara Waterbury and I am currently working on a Senior Design project for Electrical Engineering at the University of Central Florida. I was interested in receiving permission from Newhaven Display International to use some images of the NHD-4.3-480272MF-ATXI#-T-1 and the NHD-3.5-320240MF-ATXL#-T-1. The images will only be used in a reference fashion and we will credit Newhaven Display International for the images. Your permission would be greatly appreciated.

Thank You,
Cara Waterbury



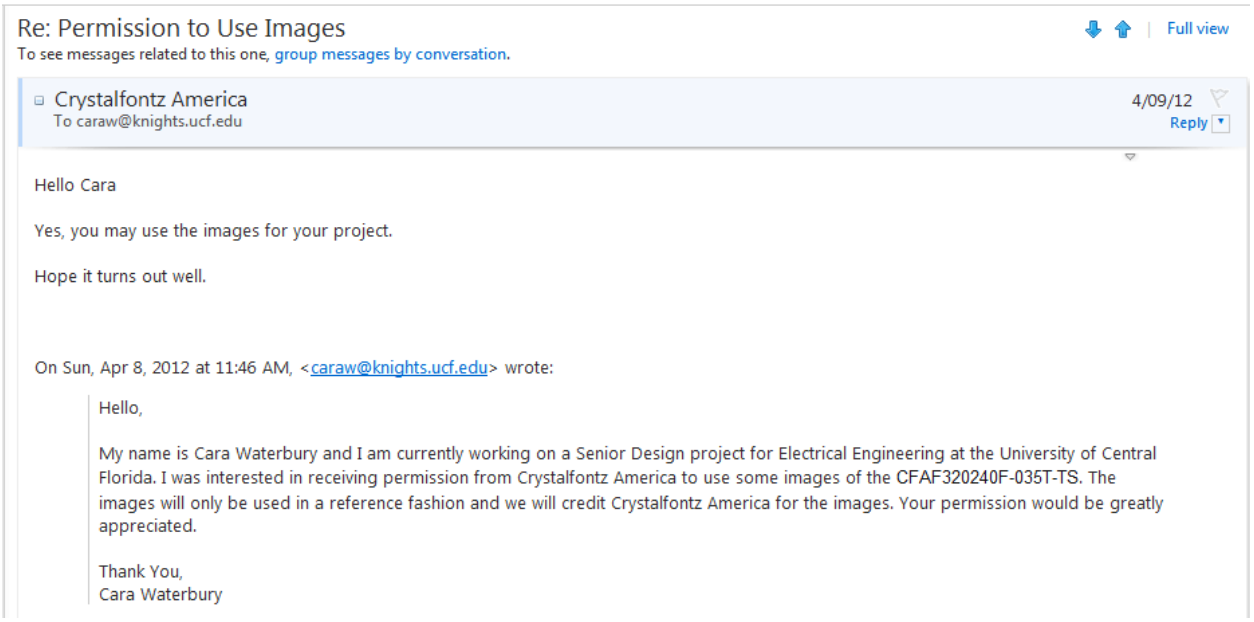
RE: Message from Newhaven Display International, Inc.

Atif Khan   Add to contacts
To 'Cara Waterbury', customerservice@newhavendisplay.com

Hi Cara,

You may use images of our displays for reference.

Sincerely,
Atif Khan | Engineering
Newhaven Display International, Inc.
www.newhavendisplay.com
2511 Technology Drive, Suite 101
Elgin, IL 60124
Phone: 847-844-8795
Fax: 847-844-8796


Re: Permission to Use Images

To see messages related to this one, group messages by conversation.

Full view

Crystalfontz America
To caraw@knights.ucf.edu

4/09/12
Reply

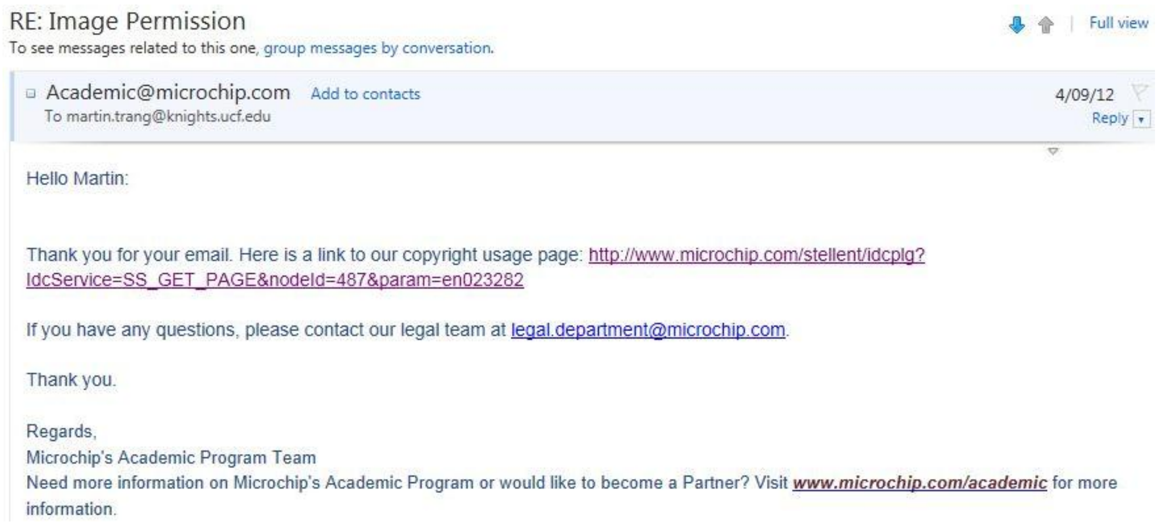
Hello Cara

Yes, you may use the images for your project.

Hope it turns out well.

On Sun, Apr 8, 2012 at 11:46 AM, <caraw@knights.ucf.edu> wrote:

> Hello,
>
> My name is Cara Waterbury and I am currently working on a Senior Design project for Electrical Engineering at the University of Central Florida. I was interested in receiving permission from Crystalfontz America to use some images of the CFAF320240F-035T-TS. The images will only be used in a reference fashion and we will credit Crystalfontz America for the images. Your permission would be greatly appreciated.
>
> Thank You,
> Cara Waterbury


RE: Image Permission

To see messages related to this one, group messages by conversation.

Full view

Academic@microchip.com Add to contacts
To martin.trang@knights.ucf.edu

4/09/12
Reply

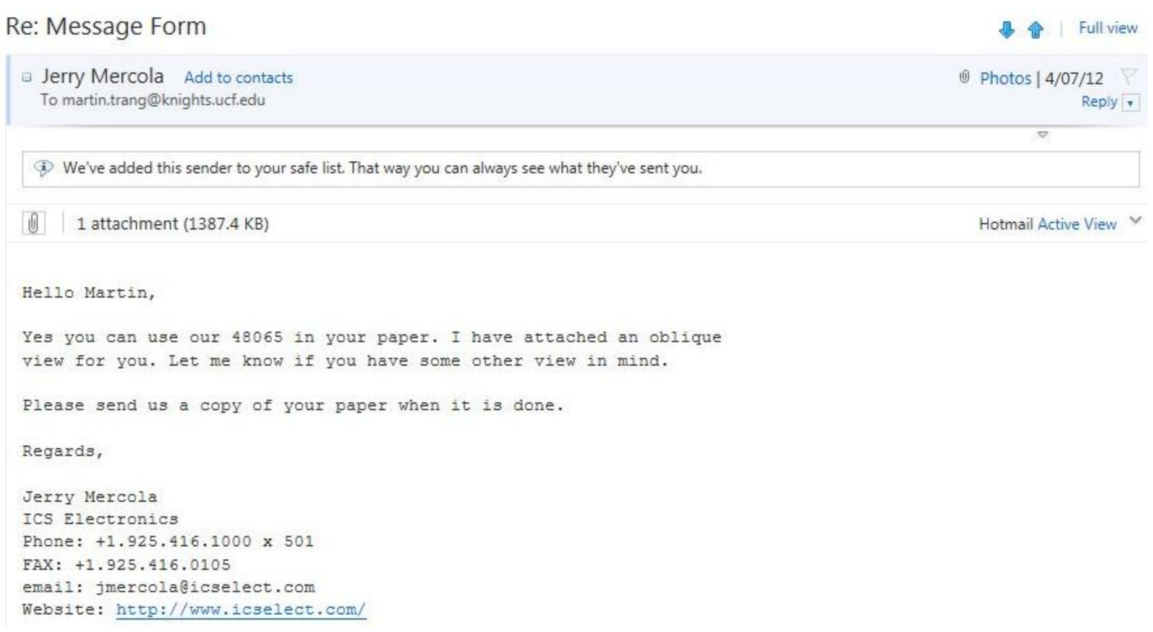
Hello Martin:

Thank you for your email. Here is a link to our copyright usage page: http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=487&param=en023282

If you have any questions, please contact our legal team at legal.department@microchip.com.

Thank you.

Regards,
Microchip's Academic Program Team
Need more information on Microchip's Academic Program or would like to become a Partner? Visit ***www.microchip.com/academic*** for more information.


Re: Message Form

Full view

Jerry Mercola Add to contacts

To martin.trang@knights.ucf.edu

Photos | 4/07/12

Reply

We've added this sender to your safe list. That way you can always see what they've sent you.

1 attachment (1387.4 KB)

Hotmail Active View

Hello Martin,

Yes you can use our 48065 in your paper. I have attached an oblique view for you. Let me know if you have some other view in mind.

Please send us a copy of your paper when it is done.

Regards,

Jerry Mercola
ICS Electronics
Phone: +1.925.416.1000 x 501
FAX: +1.925.416.0105
email: jmercola@icselect.com
Website: http://www.icselect.com/