Plastic Machine Embedded IOT Controller

SDDEC22-01

Website sddec22-01.sd.ece.iastate.edu

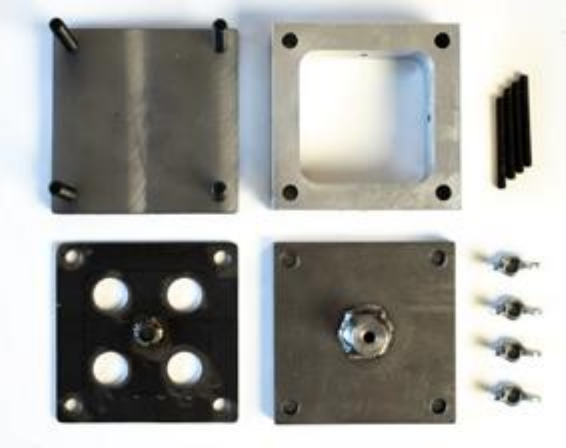
Team - Joshua Baringer, Evan Pasero, Charles Sang, Rachel Teaberg, Stone Widder

Client - Mark Hansen

Faculty Advisor - Dr. Philip Jones

Our Project

- Injection Molding
- Improve on Existing Technology
- Long Term Mission



Requirements

Hardware

- Thermocouple data input
- Relay Controlled Heating
- Microcontroller from market

Software

- Display Pertinent Data
- Dynamic User Profiles
- AWS- Not implemented

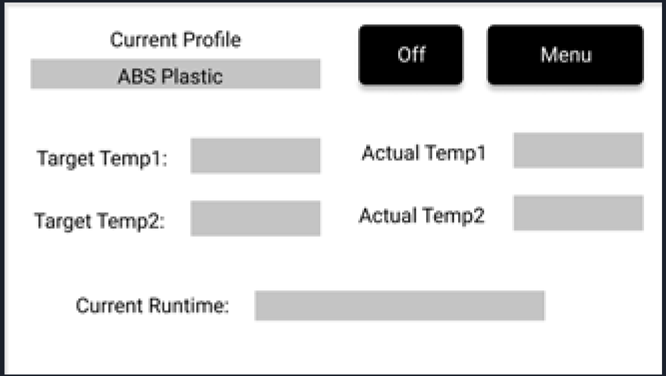
Non-Functional

- Budget of $250
- User-Friendly
- Minimal Upkeep

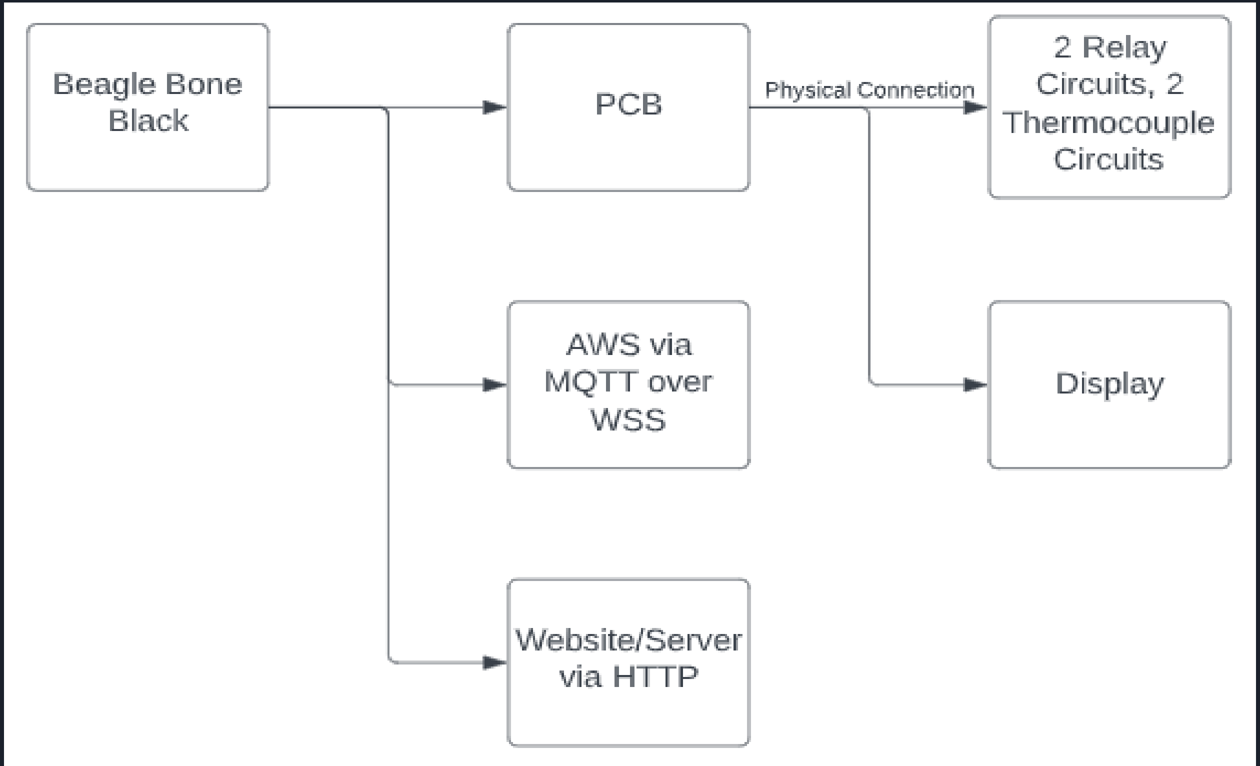
Technical Ideation

Options	Mikroe-55	Mikroe-495	E35KA-FW1000-N	PH480272T005-IHC03	104990243	ATM0430D19A	AFY480272B0-4.3N12NTM-C
Price	5	5	3	3	2	5	4
Size	3	4	5	4	2	4	4
OBC	3	5	3	3	5	3	5
Connectivity	3	1	3	5	5	1	1
Res	1	3	3	4	5	4	4
Amount Availabl	5	1	2	1	1	5	5
	32	**28**	**32**	**34**	**33**	**36**	**37**

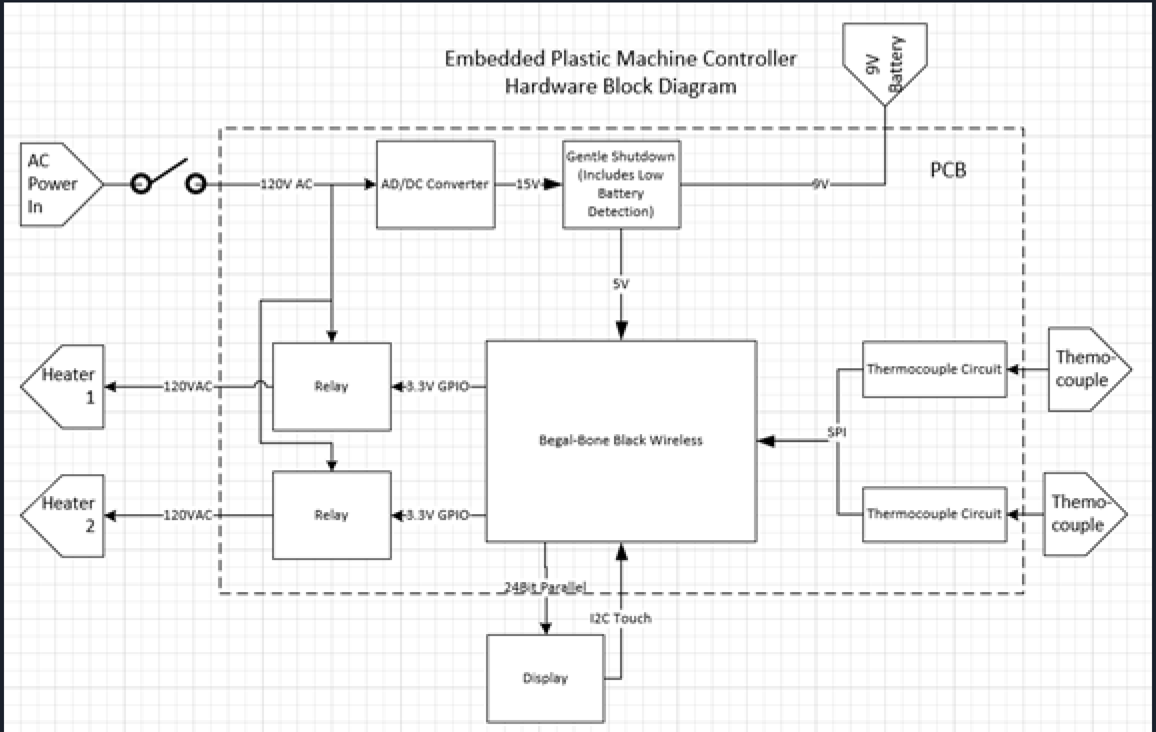
Decisions and Research

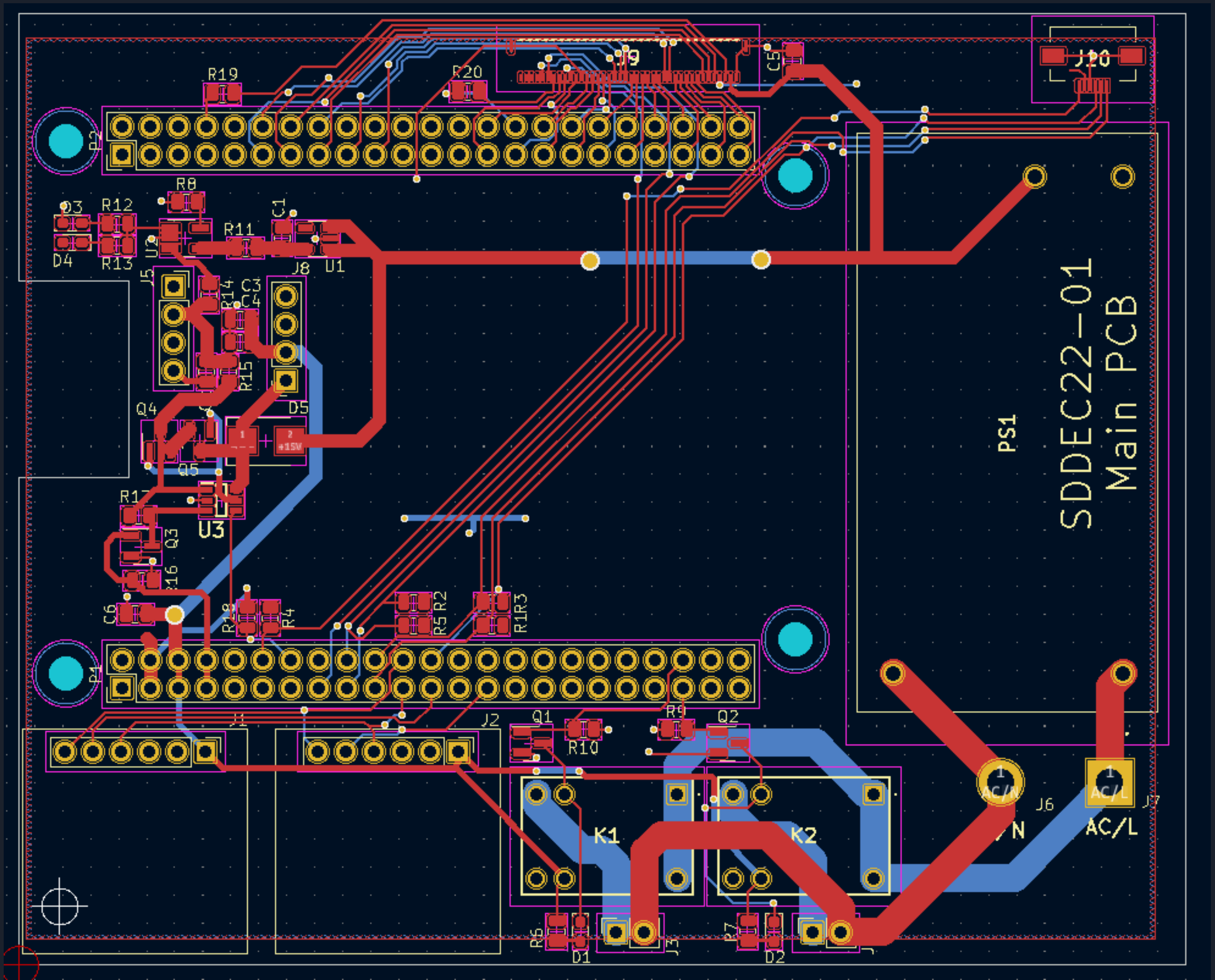


Software Block Diagram



Hardware Block Diagram

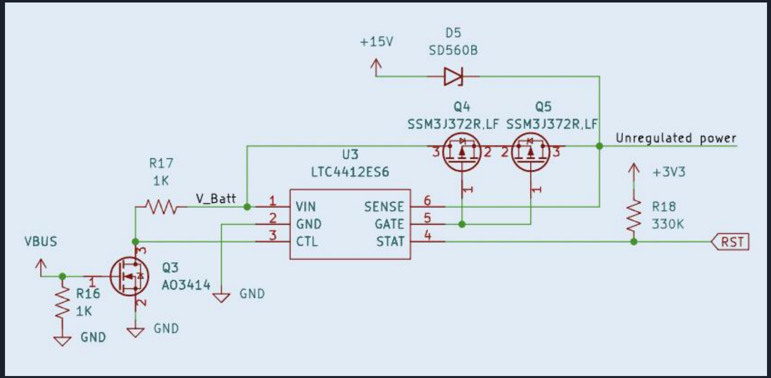



SDDEC22-01
Main PCB
PS1
J9
J10
R19
R20
P2
P1
R8
R12
R13
D3
D4
R11
C1
U1
U2
J8
J5
R14
C3
C4
R15
D5
Q4
Q5
R17
U3
Q3
R16
C6
R18
R4
R2
R5
R1
R3
C5
J1
J2
Q1
Q2
R9
R10
K1
K2
J3
D1
D2
R6
R7
J6
J7
AC/N
AC/L

Circuits of Interest

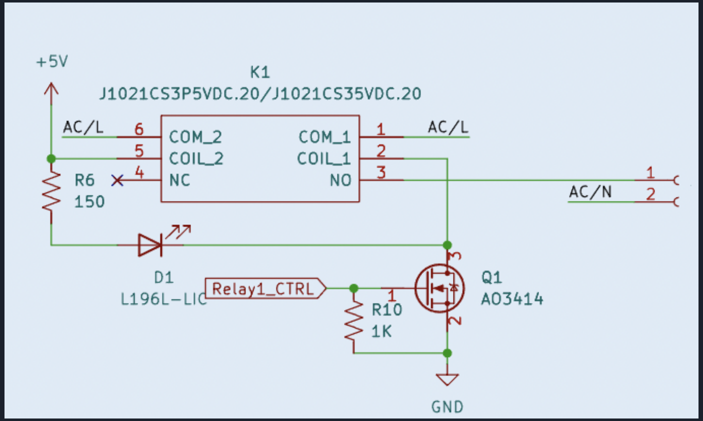
Gentle Shutdown Circuit

- Purpose : Prevent OS corruption when BBB shuts down
  - When 15V line shuts down, MOSFETS and a load sharing IC will connect 5V regulator to LIPO battery
- First Iteration
  - Used a regular 9V battery
  - Tried to use only passive components
- Second Iteration
  - LIPO battery with charging circuit
  - Load sharing ic to handle switching rails
- Third Iteration
  - Added MOSFET Q3 to prevent battery rail leakage



Relay Control Circuit

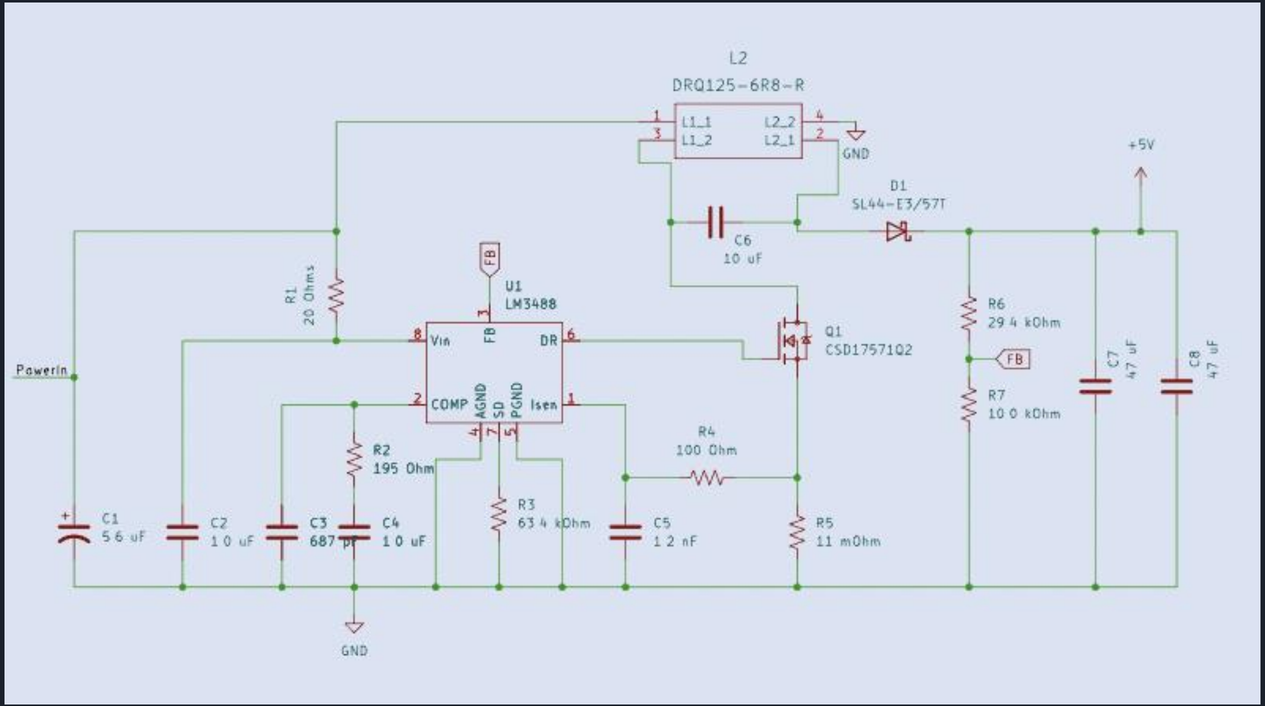
- Purpose: quickly turn on and off power to heating element to regulate temperature
  - Interfaces with AC
- First iteration:
  - Relays would switch but not maintain the switch
  - Caused by insufficient power and current
- Second/Final Iteration
  - Changed relays used
  - Added Mosfet to control switching
  - Powered via 5V instead of 3.3V
- Future iteration
  - Use solid state relay instead of mechanical



5V Regulator Circuit

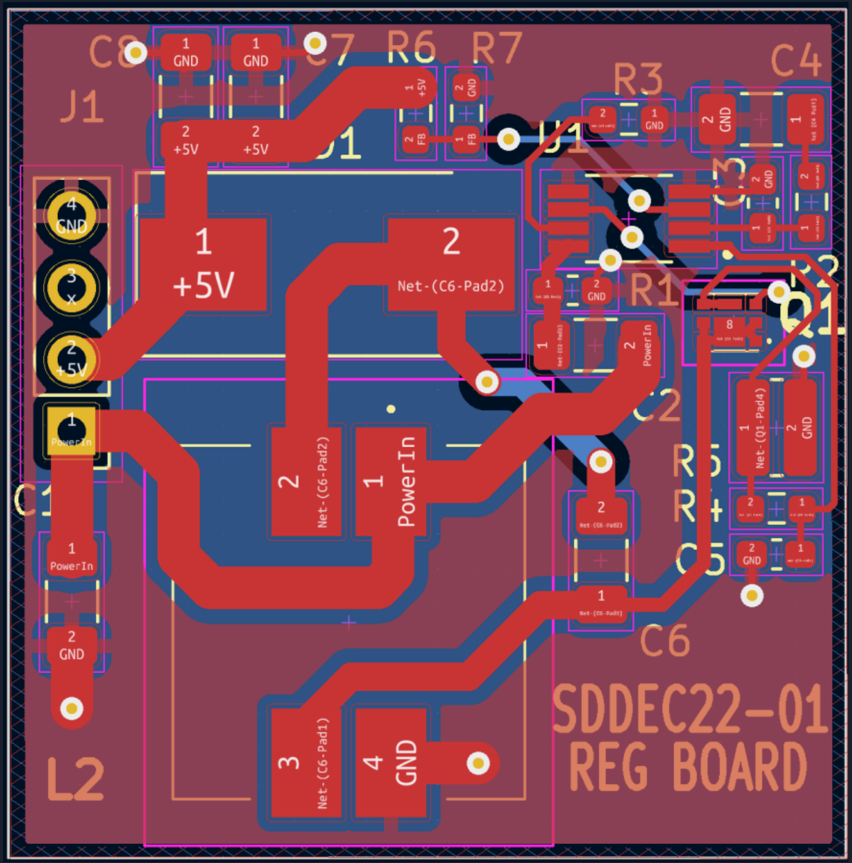
- Purpose: Provide 5V to Beagle bone from the 15V power rail
- First Iteration:
  - We used a Linear 5V regulator
  - Didn't work because
- Second Iteration:
  - Switched to a 5V switching regulator with a buck boost convertor due to switch in gentle shutdown circuit
  - Pulls 3.7V from LIPO battery up to 5V and 15V from power rail down to 5V
  - While testing discovered it had a ripple of 1V
- Final Iteration:
  - Fixed ripple issue by changing layout of feedback traces
  - Shuts off with load higher then .05Amp
  - Tested for shorts, voltage of sd/freq/sink pin, isense pin resistor was lower then short circuit detection level
- Future Iteration: More extensive testing and help from external sources

5V Regulator Schematic



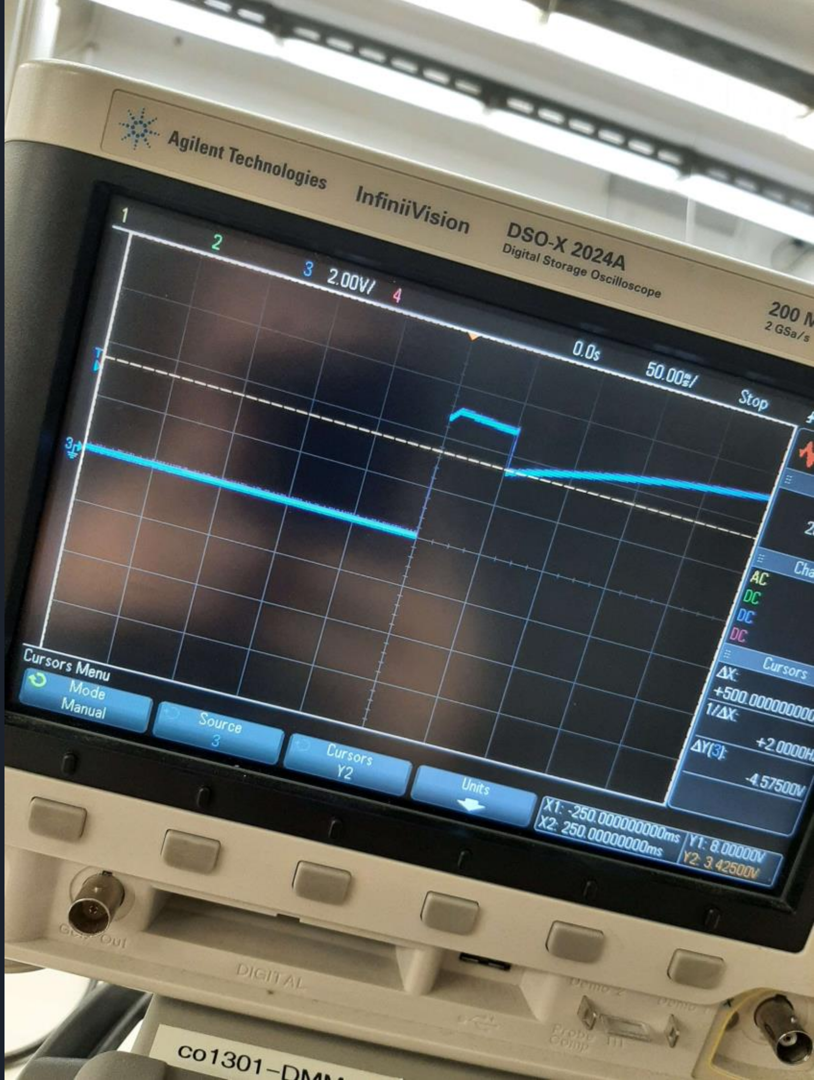
5V Regulator Layout

- First iteration
  - Voltage swings between 5.4 and 6.6V
- Second iteration
  - Steady voltage output
  - Fails to maintain voltage with load
  - Switching loop as small as possible
  - Fixed footprint mistakes
  - More extensive ground plane



Hardware Testing

- For main board circuits, bread board first
- Switching reg must be on board to test.
- Using Oscilloscope to watch individual signals
    - Pictured: Trying to reduce 5V Reg Startup Time
- Following high heat signatures
- Thermocouples temperature are tracked to within 10^0C of set point.
- Verify relays heating up correctly as per thermocouples.



Software

Graphic User Interface

- The display should be interactive with the ability to scroll up and down the profile menus and, should have a touchscreen functionality.
- We used a python library called Tkinter since it is lightweight, easy to program.
- GUI allows the user to create, edit and save different profiles needed to melt a variety of plastics.
- Communicates with an underlying command line interface to do things such as
  - Accessing the databases to work with profiles.
  - Loading and tracking PID values.
  - Running the main program.
- It should open on screen when the system has been powered up and the setup script ran.

Boot up Script

- Single bash script
  - Updates the image
  - Sets up the databases used to store the profiles,
  - Sets up history and PID values,
  - Configures the script needed for the SPI bus to be run at boot..
- Operation manual explains how to run script
- After running and rebooting, the system is ready to use

Command Line and Display Overlay Interface

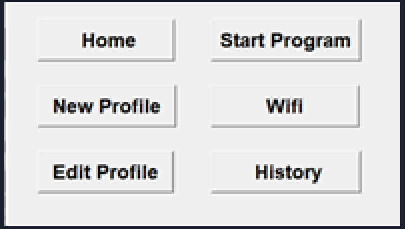
- Command Line Interface
  - Using a command line interface allow for easy future use and GUI changes
  - Uses
    - Handles SQL query's,
    - Reads the thermocouples
    - Turns on the relays

- Device Tree Overlay
  - Had to create our own device tree overlay
  - Must use with specific image of Debian OS
  - Uses
    - Configures the pins needed to control the display
    - Sets resolution,
    - Sets timings of frames
    - Sets clock frequencies.

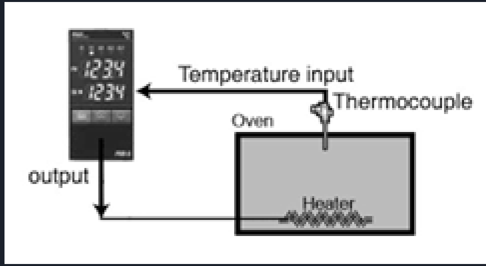
PID

- Manual
  - The PID controller can be manually tuned by entering new values in the GUI and observing the results by running the system.
- Autotune
  - Written in python but based on Arduino autotune library
    - Implements Ziegler-Nichols(ZN) method
  - Basic process
    - Heats to max setpoint
    - Cools to min setpoint
    - Tracks time it takes to do a cycle
    - Using that time and ZN constants calculates Kp, Kd, and Ki
    - Repeats for set number of cycles averaging gains together
    - Returns PID gains
  - Was not able to fully implement due to time constraints

Testing



- Unit Testing
  - GUI functionality
  - Command line
  - Thermocouple circuit and PCB
  - Safe Power down
- Integration Testing
  - Verify that reading from thermocouple controls the relay
  - PID controller is able to hold the temperature within 10 degrees of setpoint
- System Testing
  - Everything happens behind the scenes and is set via the GUI



Questions?