PID Control for Espresso Brewing Temperature

Nicholas Natsoulas * and Yanran Kuang[†]
Cornell University Mechanical and Aerospace Engineering, Ithaca, New York, 14850

Dmitry Savransky[‡]
Cornell University Mechanical and Aerospace Engineering, Ithaca, New York, 14850

This study introduces a Proportional-Integral-Derivative (PID) control system to enhance the active brew temperature control of a typical home espresso machine, surpassing the performance of its default thermostat control. Utilizing thermocouples for real-time temperature sensing, a microcontroller for reading sensors and commanding control output, and a Solid-State Relay (SSR) for modulating the heating element, the PID control system ensures precise and responsive temperature regulation during espresso brewing. Experimental results demonstrate superior temperature stability, minimizing fluctuations and enabling quick responses to varying conditions. The simplicity of the implemented system makes it an accessible, low-cost upgrade for coffee enthusiasts seeking refined control without extensive modifications to their Rancilio Silvia Espresso machine. This research contributes to improving espresso brewing optimization and highlights the potential of PID control in enhancing the performance of consumer-grade coffee equipment.

Nomenclature

(unless specified, all entries are either unitless or use base S.I. units)

e	=	error
e_d	=	derivative error
e_i	=	integral error
K_d	=	derivative gain
K_i	=	integral gain
K_p	=	proportional gain
μ	=	arithmetic mean
$\bar{\mu}$	=	arithmetic mean of arithmetic means
σ^2	=	variance

*Undergraduate Researcher, Cornell Mechanical and Aerospace Engineering
[†]Graduate Researcher, Cornell Mechanical and Aerospace Engineering.
[‡]DGS for Theoretical and Applied Mechanics, Associate Professor, Cornell Mechanical and Aerospace Engineering.

σ_μ^2 = variance of arithmetic means

T = temperature in degrees Celsius

u = control signal

Contents

I. Introduction

Coffee is the most consumed beverage globally besides water. High-quality espresso coffee is in high demand since it is the base of popular coffee drinks such as espresso, cappuccino, latte, macchiato, mocha, and many iced coffee beverages. One major problem facing the average coffee consumer is accessibility to high-quality espresso coffee at home. In addition to coffee bean and ground quality, the key determinant of an espresso shot's quality is the brew water's temperature. Typical espresso machines utilize the thermostat, the simplest form of active control for brewing water temperature. Just as a home thermostat works for heating during the winter, there is a temperature threshold for when the heating should be on or off. This threshold is relative to the desired temperature, the setpoint. This thermostat method is an example of bang-bang control, often resulting in a large steady-state error. Large steady-state errors mean that the setpoint is often over or under-shot when brewing, resulting in low-quality espresso that lacks a robust flavor profile. PID control offers an intuitive, high-precision alternative to thermostat control. An increased precision translates to consistently maintaining a temperature within the range required for the best extraction and flavor profile [1]. Considering this range is around 92-96°C, there is insufficient margin for large steady-state brew temperature error. The following sections present the design of a low-cost PID control system for temperature control on a Rancilio Silvia home espresso machine and showcase statistical analysis that compares it against the thermostat temperature control system.

II. Design

A. Parts selection

The main objective of this project is to improve the brewing function of a Rancilio Silvia Espresso machine (the Miss Silvia Model) by achieving precise temperature control within a 1-degree Celsius margin. To this end, the brew thermostat on the espresso machine is replaced with the PID control system components. Each component is explained below.

Type K thermocouples are the most widely used due to their broad temperature range, high sensitivity, and low cost. Type K thermocouples have an operational temperature range from -210°C to 1090°C, which makes them suitable for recording temperature at the boiling element of the espresso machine and directly within the water stream of the machine.

The Arduino Uno is a microcontroller board that provides a platform for programming and executing the PID control algorithm. The open-source Arduino integrated development environment(IDE) makes the development and upload of software to any Arduino board quick. The Arduino Uno has General Purpose Input/Output (GPIO) and analog pins that allow it to interface seamlessly with the control system's actuator and sensors. The control algorithm stabilizes the water's brew temperature. The thermocouple amplifier interfaces the voltage signals the thermocouples send to the microcontroller. The AD8495 is designed explicitly for interfacing with thermocouples and converting their output to a

voltage signal that the Arduino can process. Compared to other Adafruit models, such as the MAX3185, the initial model used, the AD8495, produces less noise and can sample signals at a higher frequency, leading to more accurate results.

The Solid State Relay (SSR) controls the power supplied to the heating element of the espresso machine. SSRs are preferred for PID controllers due to their fast switching capability and reliability. The SSR receives scaled control output from the Arduino Uno and outputs load current to the espresso machine, thus effectively activating the heating element. The SSR acts as a switch controlled by the DC input signal. The SSR replicates the typical behavior of the heating process in espresso machines: the heating element is turned on and off to maintain a stable temperature. The thermostat monitors the water temperature and activates the heating element as needed. Similarly, the SSR turns the heating element on and off as needed.

Lastly, miscellaneous components are used to build and assemble the controller circuit. These components include a breadboard, spade connectors, ring connectors, jumper wires, 14-gauge wires, and a crimping kit.

The list of parts used in the build is shown in Table 1 below:

Table 1 Parts List

Parts Name	**Quantity**
Arduino Uno Rev3	1
Adafruit Analog Output K-Type Thermocouple Amplifier - AD8495	2
2M K-Type Temperature Sensor - Thermocouple	2
SSR-40DA 40A Solid State Relay	1
Breadboard Jumper Wires - Assorted ; 10cm	10+
Piggyback Spade Connectors 1/4" 22-10 AWG	2
Crimp Connectors - Assorted	10+
14 Gauge Wire	10 feet
Ring Connectors 1/4"	2

B. Build

The build of the brew temperature control system for the Rancilio Silvia Espresso Machine consists of the assembly of the control system hardware and the system's integration with the machine.

The parts in Table 1 are configured to wire the thermocouples' breakout boards and the SSR to the microcontroller. This allows the embedded software to actuate the SSR by providing voltage to an output pin and sense temperature by reading the voltage of input pins. The SSR is wired to the heating element so that upon the microcontroller's command, it sends a current to the heating element, thus increasing the system's temperature. Since the control algorithm discussed in the next section regulates the temperature of two distinct features of the system —the boiler's external surface temperature and the brewing water temperature— the two thermocouples are fixed to their respective locations on the machine.

The visual walk-through (Figures 1, 2, 3, 4, 5, 6, and 7) below provides a clear sense of the final assembly, which includes the build's wiring and thermocouple placement.

1. Visual Walk-through

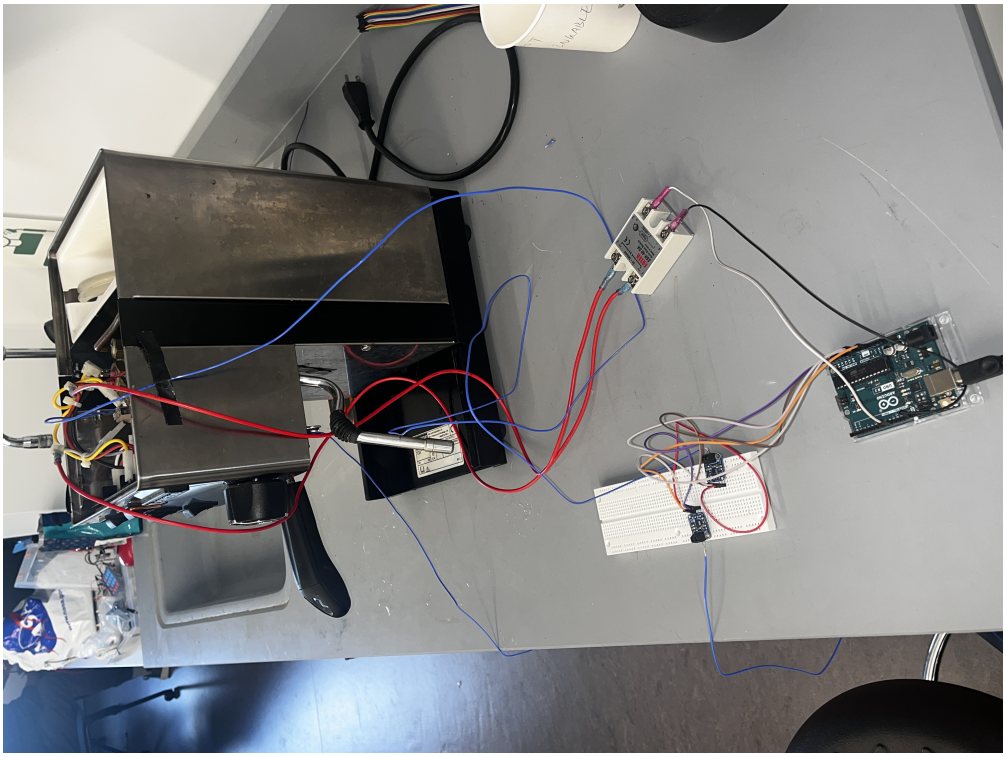

Fig. 1 The Full Assembly for Brew Tests

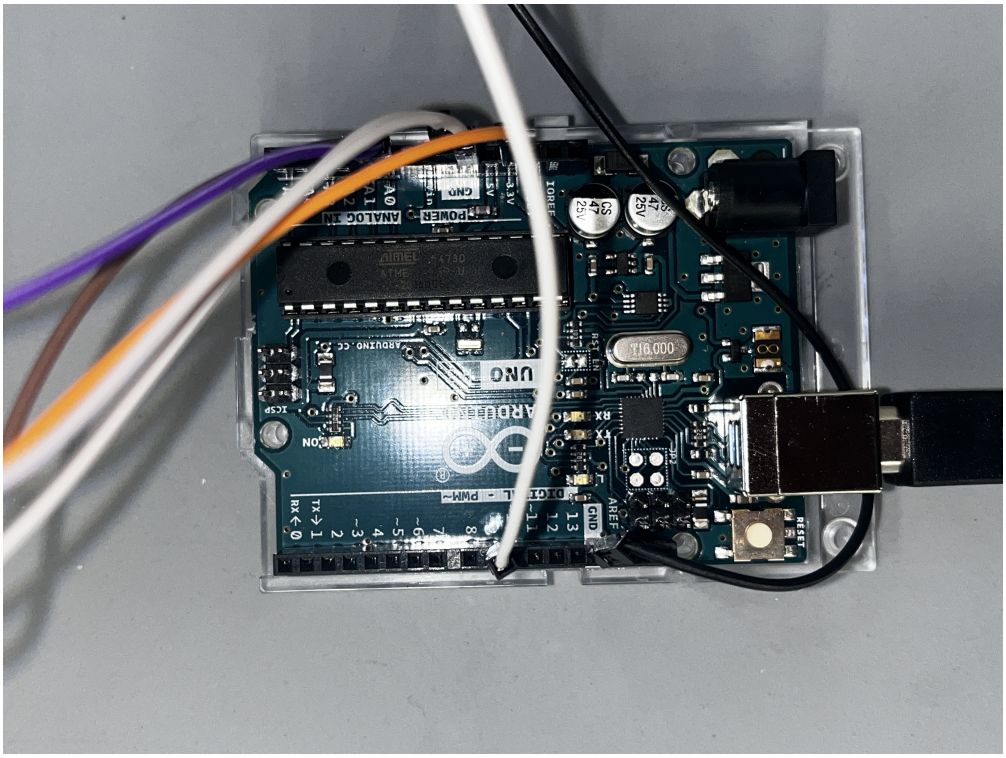

Fig. 2 The Arduino Uno Microcontroller

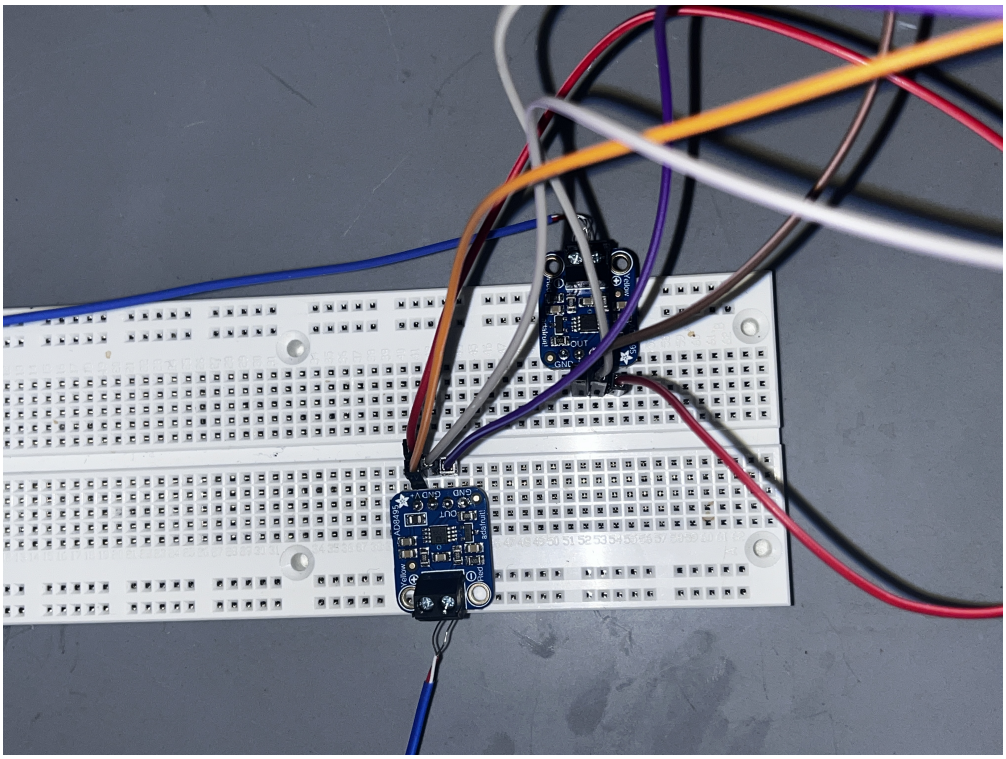

Fig. 3 AD8495 Type-K Thermocouple Amplifiers (Breakout Boards)

In Figure 3, the Type-K thermocouples are attached to their respective ADA8495 breakout boards. These breakout boards are on a breadboard to simplify design iteration and debugging.

Fig. 4 Solid State Relay

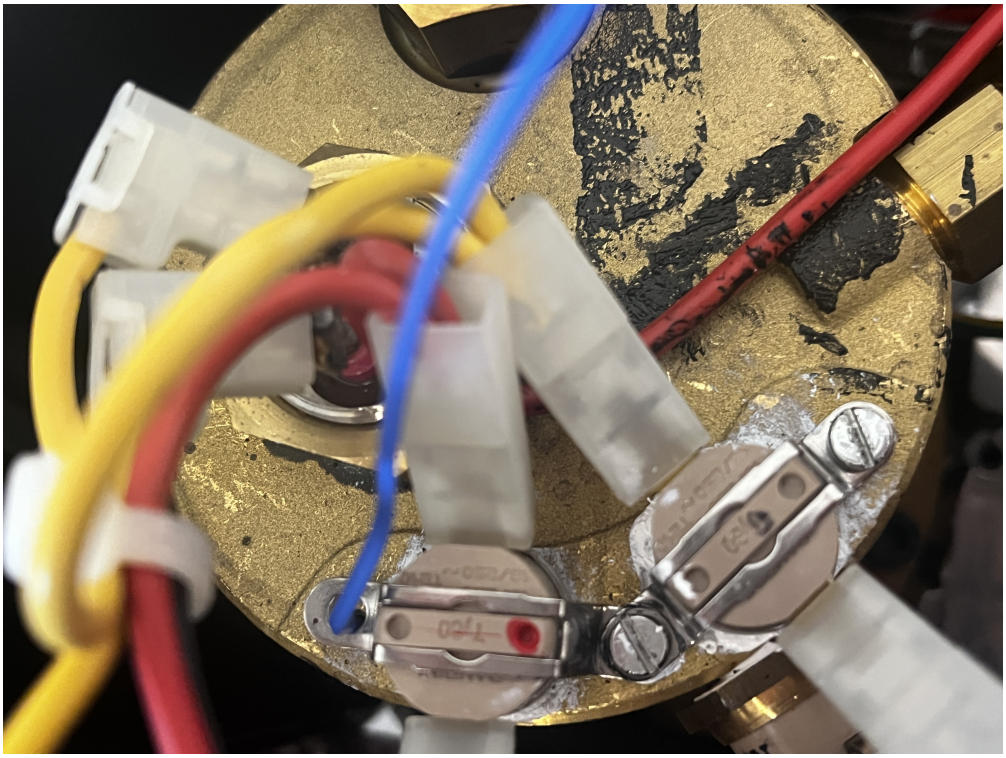

Fig. 5 Water Boiler with Thermostat Installed

In Figure 5, the boiler beneath the machine's top casing is in its default configuration for thermostat temperature control. The brew thermostat is the tan cylindrical component with a red dot on the right side of its tab. The blue wire, albeit not an original part of the system, is the boiler thermocouple and is necessary for data acquisition during testing of the default temperature control.

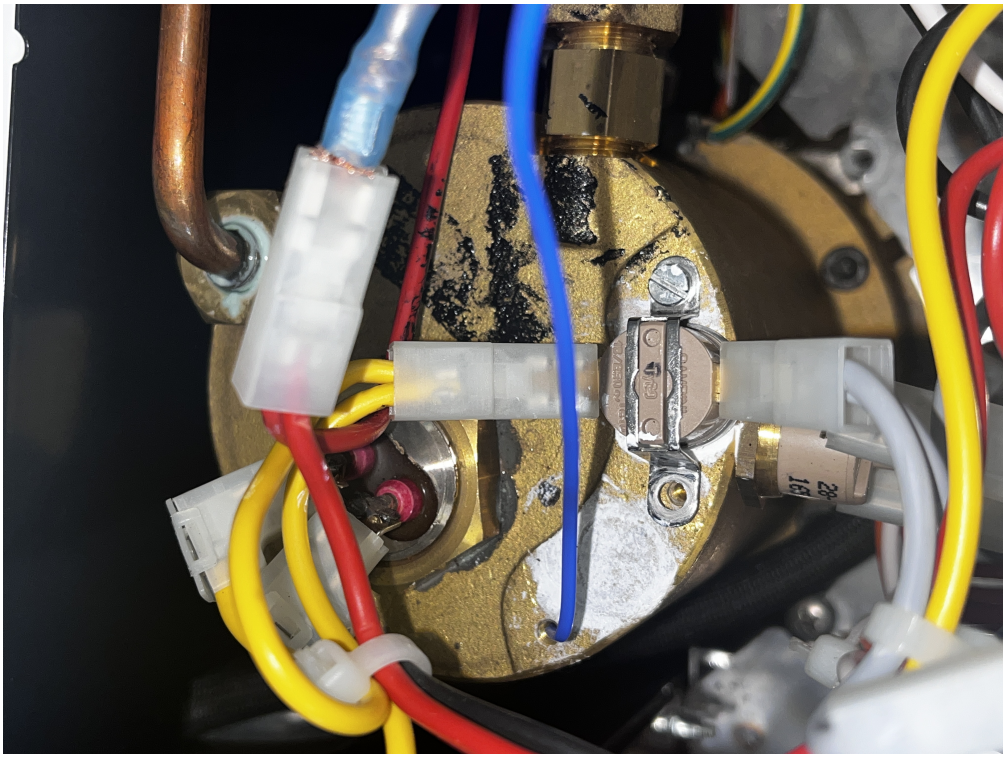

Fig. 6 Water Boiler with PID Control System Installed

In Figure 6, the boiler is in its modified configuration for PID temperature control. Notice that the brew thermostat is no longer included and that the thermocouple remains fixed in the same threaded hole. The spade connector with a white plastic protector and a translucent blue crimp connects the SSR's output wire (14-gauge) and the heating element.

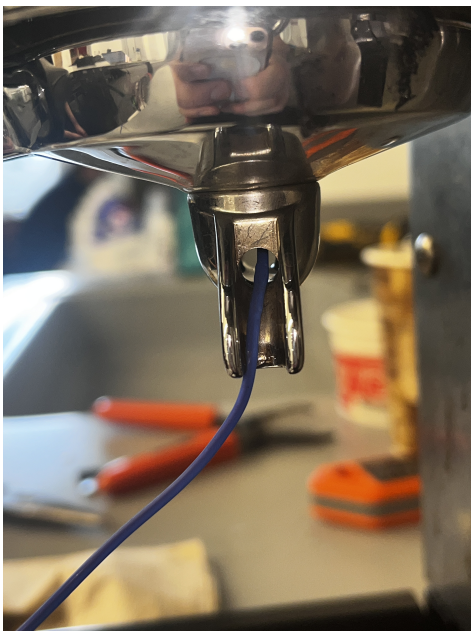

Fig. 7 Brewing Water Thermocouple Placement

This placement for the brewing thermocouple, as shown in Figure 7, permits the best brewing water temperature measurement without modifying the machine's hardware or structure. Testing without a puck of ground coffee in the portafilter ensures that the water flows directly over the thermocouple and is closest to the temperature of the water if it were brewing coffee in the portafilter. Note that the placement of this thermocouple inside the portafilter and above the coffee grounds yields inaccurate and noisy brew temperature readings compared to the chosen placement.

2. Build Iteration Process

Several locations to attach the thermocouple were considered.

The boiler thermocouple was initially placed directly beneath the brew thermostat. However, a layer of insulating paste between the thermostat and the heating element prevented accurate measurements due to the thermocouple's sensitivity. The boiler thermocouple was eventually placed inside the threaded hole on top of the heating element, where it once secured one of two screws for the brew thermostat. This positions it closer to the highly electrically resistant coils that generate heat.

Determining the placement of the brew thermocouple involved trials and testing. Initially, the consideration was to place the thermocouple sensor downstream of the water stream after the water passed through the heating element and before making contact with the puck between the eco space group nickel and gasket undertorque miss (to understand these part names, refer to Figure 14 in the appendix). However, experimentation revealed that this placement interfered with the seal of the space group nickel, leading to leakage. Placement within the filter holder, or metal puck, was also considered. However, testing showed that the temperature profile was extremely noisy and inconsistent. The temperature distribution within the metal puck, especially with the coffee ground, is not uniform, likely resulting in unreliable measurements by the thermocouple. Thus, the optimal location chosen was in the whole 2-dose beak of the filter holder, shown in Figure 7.

The initial model utilized the Adafruit MAX31865 breakout boards/amplifiers. Testing revealed that this model was not ideal for this project. The control algorithm samples and sends out control outputs a thousand times per second. However, the MAX3185 breakout board had a lower sampling frequency. This was evident from the time delay between the control output and SSR activation. During testing, it was observed that the boiler and brewing temperatures consistently overshoot because the temperature measurements received by the controller are several time steps behind. As a result, the Adafruit AD8495 model was selected to reduce time delay.

C. PID Control Algorithm

The control algorithm employs Proportional-Integral-Derivative (PID) control to regulate the temperatures of the brewing water and the boiler. The subsections 1-5 below provide a high-level summary of PID control.

1. Proportional Gain (K_p)

The proportional term is based on the current error, which is the difference between the desired setpoint (SP) and the actual process variable (PV). The proportional gain is denoted by K_p. The controller output is proportional to the current error.

$$P = K_p \cdot \mathrm{e} \tag{1}$$

2. Integral Gain (K_i)

The integral term is based on the accumulation of past errors over time. It helps to eliminate the steady-state error and reduce the impact of accumulated error. The integral gain is denoted by K_i.

$$I = K_i \cdot \int_0^t \mathrm{e}\, dt = K_i e_i \tag{2}$$

3. Derivative Gain (K_d)

The derivative term is based on the rate of change of the error. It anticipates future behavior and helps to dampen the system's response to prevent overshooting. The derivative gain is denoted by K_d.

$$D = K_d \cdot \frac{d}{dt}\mathrm{e} = K_d \cdot \frac{d}{dt} e_d \tag{3}$$

4. Overall PID Controller Output

The overall controller output (u) is the sum of the proportional, integral, and derivative terms:

$$u = P + I + D \tag{4}$$

5. PID Controller Tuning

- Increasing K_p amplifies the response to the current error.
- Increasing K_i reduces steady-state error and helps eliminate accumulated error over time.
- Increasing K_d dampens the system's response, reducing overshooting.

Balancing these gains is crucial for achieving a stable and responsive control system. Adjusting these gains to achieve the desired system performance is called tuning.

The PID gains for the brew water temperature and the boiler temperature are specified by the constants:

$$K_{p_{\text{brew}}}, K_{i_{\text{brew}}}, K_{d_{\text{brew}}}$$

$$K_{p_{\text{boiler}}}, K_{i_{\text{boiler}}}, K_{d_{\text{boiler}}}$$

which provides one set of gains to the brew temperature control and another to the boiler temperature control.

The system measures temperatures using thermocouples connected to analog pins of the microcontroller for the brewing water and the boiler. The raw analog readings are converted into temperature values using predefined calibration constants and a linear conversion formula.

The PID control loop calculates the error between the desired temperature setpoints T_{brew} and T_{boiler} and the current temperatures. The integral and derivative terms are accumulated and computed to adjust the control output, which is then scaled to fit within the microcontroller's voltage range of zero to five volts. A heuristic is applied to this voltage output to create a switch functionality. Any voltage above a threshold is mapped to the full five-volt signal, whereas anything below is mapped to zero volts. If the voltage input is 3 volts or above, the SSR supplies current to the heating element and otherwise provides no current. The switch for the control output is helpful since it is a fundamental implementation of a pulse modulator for the control output signal.

The control output is applied to the system via the microcontroller's assigned output pin, which sends voltage to the SSR. The brew water control output is used when the brew process is active; otherwise, the boiler control output is used.

The algorithm continuously monitors the system, updating control outputs based on temperature readings and maintaining PID calculations. It also tracks the progress of the brew process and constantly checks if the boiler's setpoint is reached, which permits the brew functionality. Brew must occur after the boiler's setpoint has been reached, as the brew temperature depends on the boiler's temperature before brewing.

This control algorithm design ensures precise regulation of both brew water and boiler temperatures during operation.

There are two distinct PID controllers utilized for temperature control. One for each operating mode: heating and brewing. The boiler temperature is the control variable during the heating mode, whereas the brew water temperature is the control variable during the brewing mode. The diagram below in Figure 8 is a high-level overview of this control scheme.

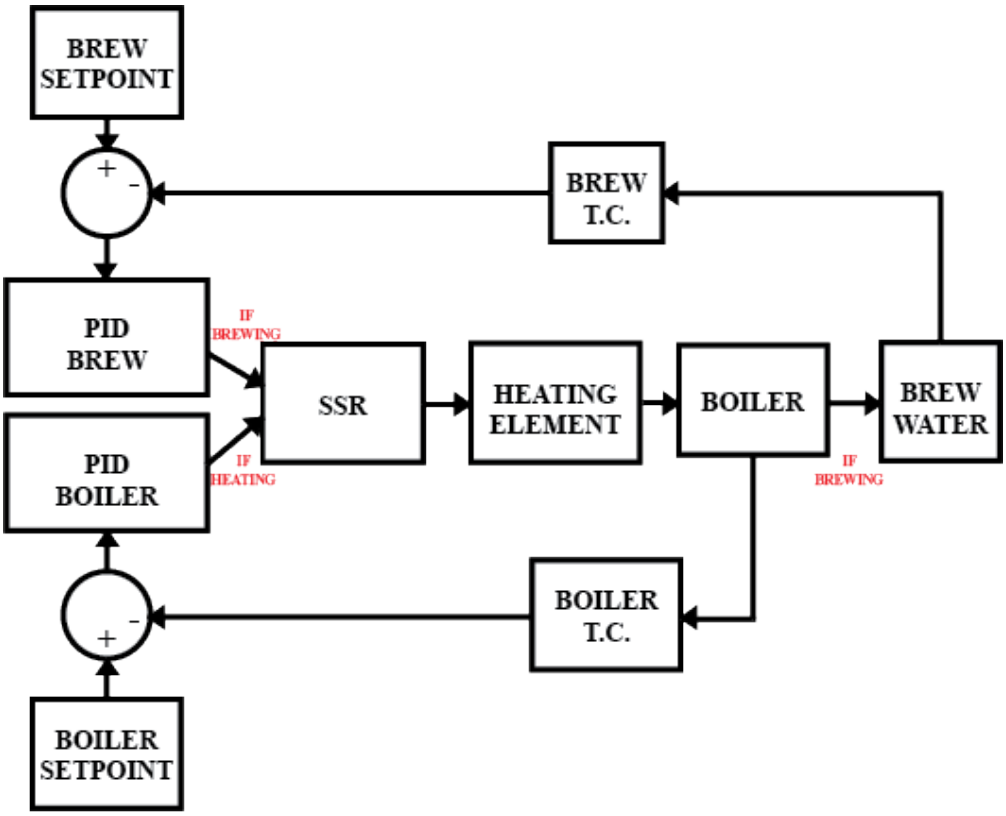

Fig. 8 Control Algorithm Block Diagram

D. Control Algorithm Software

The control algorithm software is in C++ and is compatible with the Arduino Uno microcontroller. This software is uploaded to the microcontroller via the Arduino IDE and runs continuously as long as the controller is powered. The software runs on the microcontroller with a sampling time of one millisecond. This high-frequency sampling time is a safe design choice since the brewing process is on the order of seconds, and the controller requires hundreds of steps to converge. The software utilizes many function definitions for modularity and readability of the main control loop. For more detail, consult the software in Listing 1 in the appendix.

E. Controller Tuning

Performance metrics helped to determine the gain values for the PID controllers. The controller should stabilize the overall system, have a rise time <= 1.00 seconds, minimize overshoot, and avoid integral windup. The gain values found to meet these goals are summarized in table 2

Table 2 Gain Values for Boiler and Brew PID Controllers

Controller	Gain Values		
	K_p	K_i	K_d
Boiler	1	0.001	10
Brew	10	0.00001	15

1. Gain Tuning Iteration Process

A simplified PID model that estimated the dynamic of the heating element warming up was used in Simulink to experiment with different gain values. The preliminary gain values selected were K_p = 0.431, K_i = 0.0205, K_d = 0.04. These gain values were initially implemented on the brew and boiler controllers and tested in real time. Testing revealed that integral error accumulates rapidly in both systems, especially for the brew system because the espresso machine was heating up from room temperature. As a result, in the first two to three minutes of the trial, the brew thermocouple measures at or close to room temperature, making it prone to integral windup. Thus, the integral gain for both systems was adjusted such that the integral term is orders of magnitude smaller than the proportional and derivative gains. Furthermore, the brew and boiler temperatures were overshooting past their set points. To minimize overshoot, the proportional and derivative gains were increased. Multiple trials and testing were conducted to adjust the three gain values.

III. Testing

Once the build and the tuning are configured, as explained in a prior section, the general testing procedure is outlined in the following subsections:

A. Embedded Software Integration

Upload the embedded software to the Arduino Uno microcontroller via the Arduino IDE. Connect the Arduino Uno to a personal computer throughout the test for ease of design iteration and data acquisition.

B. Start Data Acquisition

Start the data acquisition immediately. The data printed per iteration, as mentioned in Table 3 in the Analysis section, is collected from the IDE's serial monitor into a CSV file via a plug-in called Arduspreadsheet [2]. Arduspreadsheet is an open-source tool that can be installed onto legacy versions of the Arduino IDE.

C. Heating

Assuming the machine starts from room temperature, the PID controller heats the boiler until the setpoint is reached. This typically takes around two minutes. The heating is finished once the boiler temperature settles within one °Celsius

of its setpoint, which in this case is 106 °C. A flag is printed to the IDE serial monitor when the thermocouple reads within one °C of the setpoint.

D. Brewing

Initiate brewing at any point following the heating process. The brewing is started by manually flipping the brew switch on the espresso machine, just as one does for operating the default configuration. Brewing should last between three and ten seconds to acquire sufficient data for post-processing.

E. Save-off Data

Manually name and save the data onto a personal computer for further analysis.

IV. Analysis

A. Data Collection

Data was collected from the start of each trial while the boiler was at room temperature to the end of one shot worth of coffee, which is approximately three seconds of the dispensing from the espresso machine. For each timestep, an array of measurements were recorded. The array consists of the following variables.

Table 3 Variables in Data Array

Variable Name
Time (μs)
Brewing Temperature (°C)
Scaled Control Output for Brewing
Brew Start Flag
Brew Counter
Proportional Error for Brewing
Integral Error for Brewing
Derivative Error for Brewing
Boiler Temperature (°C)
Scaled Control Output for Boiler
Boiler Error percentage
Proportional Error for Boiler
Integral Error for Boiler
Derivative Error for Boiler
Boiler Setpoint Reached Flag

Measurements recorded at each time step are stored in a CSV file. For data analysis, the data set was parsed, such that the elapsed time starts at time of brew (when the brew button was hit), and 4.5 seconds after. This period accounts for the time for the espresso machine to dispense one shot worth of coffee and for the brew temperature to stabilize into a steady state.

V. Results

Five trials each were conducted for the thermostat-controlled machine and the PID-controlled machine. Temperature results across all five trials were collected in an array. The mean was calculated and stored in another data array for each trial. Both arrays were utilized for statistical analysis. The Table 4 below summarizes the findings.

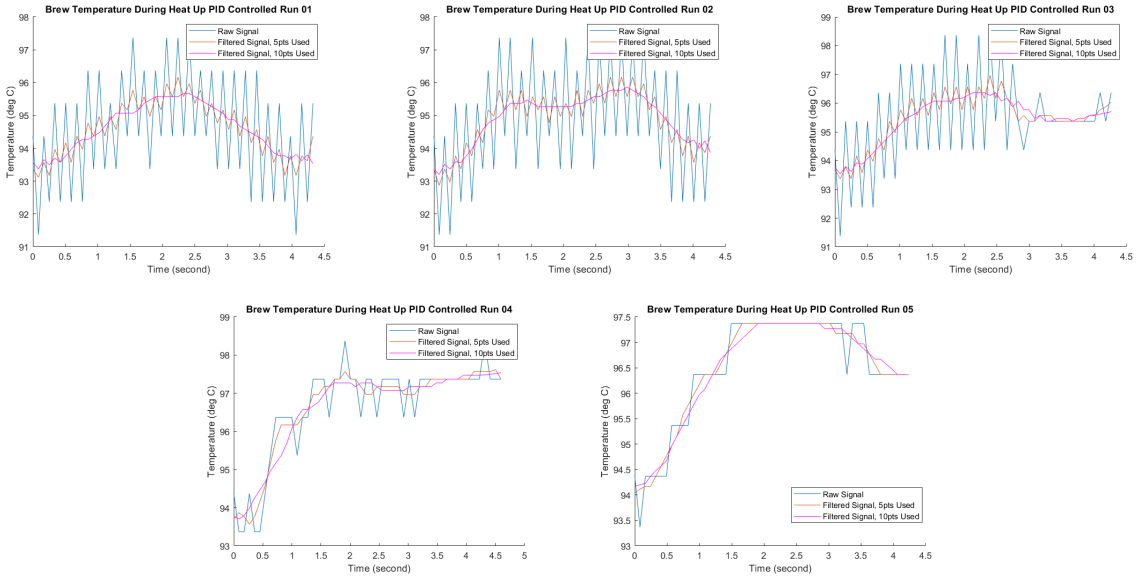

Fig. 9 Temperature of Brew vs Time Elapsed. Set of Five Trials in PID Control Testing

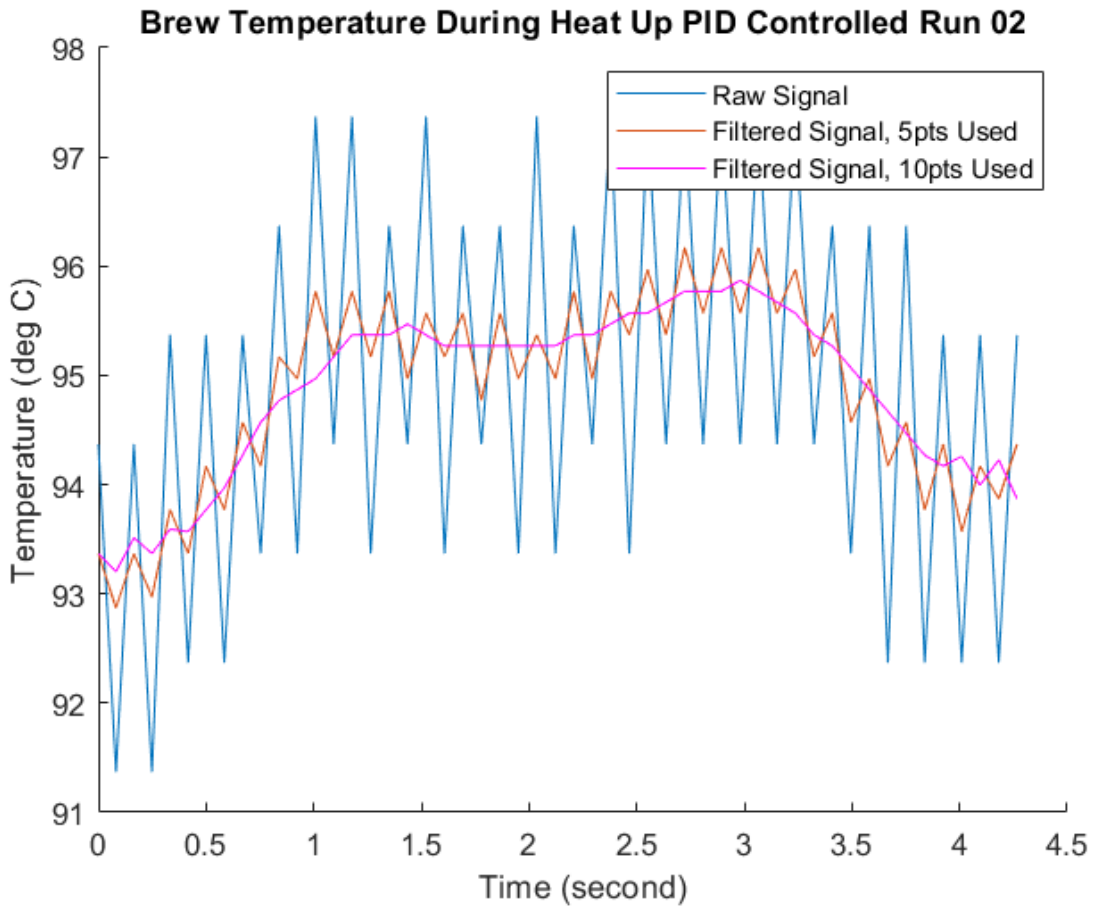

Fig. 10 Exploded View of a Brewing Trial with PID Control

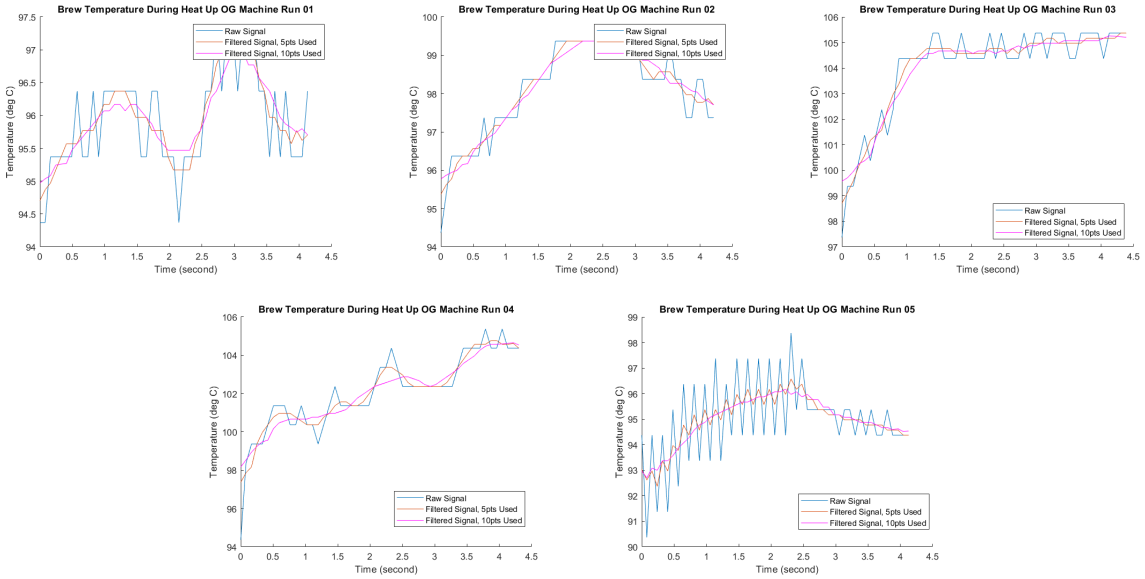

Fig. 11 Temperature of Brew vs Time Elapsed. Set of Five Trials of the Thermostat-Controlled Machine

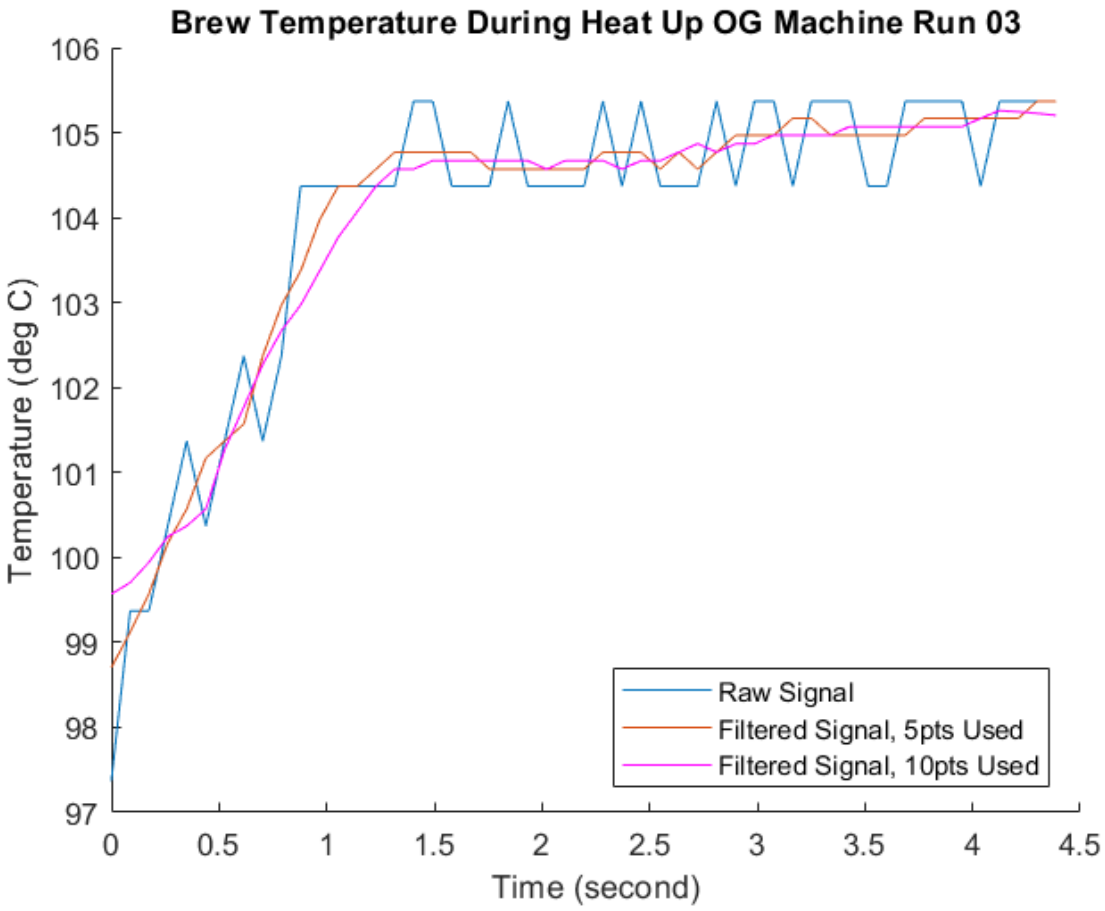

Fig. 12 Exploded View of a Brewing Trial with Thermostat Control

Table 4 Summary of Trials

Control Scheme	$\mu_{T_{brew}}$ (°C)	$\sigma^2_{T_{brew}}$	$\bar{\mu}_{T_{brew}}$ (°C)	σ^2_{μ}
Thermostat	98.991	13.7181	98.991	15.0845
PID	95.5838	1.4929	95.5838	0.824

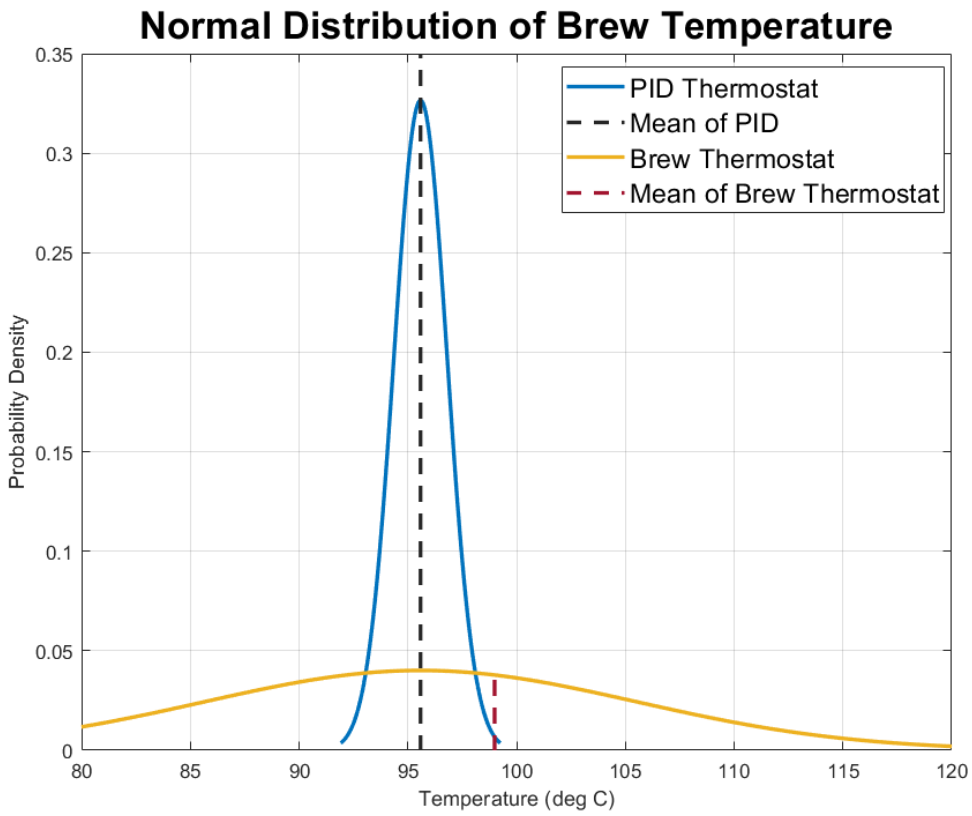

Fig. 13 Distributions of Brew Temperature

The temperature of the brew thermostat, in direct contact with the water stream, was recorded over time. The raw signals often demonstrate a noisy pattern due to the thermocouple's measurement error limitation. The raw signals were filtered by calculating the moving average. For Figures 9 to 12, the raw signal, the moving average with a five datapoint window, and the moving average with a ten datapoint window are plotted.

A. PID Performance

For the PID controlled machine, the desired temperature for brewing is 95 °C. Figures 9 and 10 show the temperature in the hot water stream over the 4.5 second period. The average brew temperature in five trials was 95.5838 °Celsius, same as the average of the five means. The variance in temperature across five trials was 1.4929, and the variance of the means was 0.824. The average brew temperature is within the optimal range of 92-96°C [1]. The percent difference between the average temperature and the set point of 95 °Celsius is 1.4929 %. The percent difference between the average of the mean temperatures and the set point of 95 °Celsius is 0.8240 %.

B. Comparison to Default Thermostat

The thermostat control's performance was compared in Figures 11 and 12.

The average temperature maintained by the thermostat control is not within the proposed optimal brew range. Furthermore, the variance of the average temperature for the thermostat control is greater than that for the PID control. Figure 13 shows the average temperature distributions. This implies that the PID-controlled machine can maintain a more consistent and stable temperature profile during brewing. The lower variance in temperature indicates that the PID control system effectively regulates the heating element, minimizing temperature fluctuations and ensuring that the water temperature remains closer to the desired set point.

C. Next Steps

Note that the steps are ordered by priority.

1. Hardware Solution for Thermocouple Placement

The thermocouple placements of this build are intended to measure the temperature change near the heating element and the temperature of the brewing water. Hardware modification is necessary to place the thermocouples to measure these temperatures more directly and reliably with less delay and noise. Disassembly of the espresso machine may be required, especially for boiler and filter assembly modifications.

2. Controller Tuning

The gain values for the brewing and boiler controllers are sufficient to drive the brew temperature to the setpoint of 95 °C. However, the boiler system slightly overshoots over the set point of 106 °C when the boiler first starts at room temperature. Further tuning of PID gains for the boiler system is necessary to minimize overshoot. Otherwise, when the boiler overheats, more time is allocated to converging to the setpoint, and espresso enjoyment incurs significant delay.

3. Filtering

Live data processing using filtering helps mitigate the noisiness of the temperature sensors. The moving average strategy can be applied in live data collection. As new data points are collected, the moving average can be continuously calculated and replaced with the raw data point.

VI. Conclusion

This study successfully introduces and demonstrates the effectiveness of a PID control system for enhancing the active brew temperature control of a home espresso machine. Through the integration of thermocouples, a microcontroller, and an SSR, the PID system provides precise and responsive temperature regulation during the espresso brewing process, surpassing the performance of the default thermostat control with a temperature variance that is an order of magnitude smaller. The mean, variance, and other experimental results affirm the superior temperature stability achieved by the PID control system, effectively minimizing fluctuations and enabling rapid responses to changing conditions, making

for better consistency and accuracy in the espresso brewing quality.

This research contributes to espresso brewing optimization and underscores the potential of PID control in elevating the performance of low-cost, consumer-grade coffee equipment. Overall, the findings presented in this study open avenues for further exploration and application of PID control systems.

VII. Appendix

The embedded software utilized for all the testing and results discussed in the report is shown below in Listing 1.

Listing 1 PID Control Algorithm Embedded Software

```
#include <Arduino.h>
#include <SPI.h>

// set sampling time
const int Ts = 1; // milliseconds
int iteration = 1;

// Pin assignments
const int controlPin = 9;

// PID gains
const double Kp_brew = 1, Ki_brew = 0.001, Kd_brew = 10;
const double Kp_boiler = 10, Ki_boiler = 0.00001, Kd_boiler = 15;

// Voltage range
const double min_voltage = 1, max_voltage = 255;
Setpointsnts and thresholds
const double setpoint_brew = 95.0, setpoint_boiler = 106.0;
const double brew_threshold = 90.0, boiler_error_margin = 1.0, boiler_threshold =
    20.0;


// Adafruit AD8495 instances (Thermocouple breakout board)
int     analogPin_brew = A0;     // Pin for brew thermocouple
int     val_brew = 0;      // variable to store the ADC value from A0
float   temperature_brew;  // Temperature value in celsius degree
```

```
26 float   setup_gain_brew = 0.005;
27 float   setup_ref_brew  = 1.26313;
28
29 int     analogPin_boiler = A1;     // Pin for brew thermocouple
30 int     val_boiler = 0;      // variable to store the ADC value from A1
31 float   temperature_boiler;  // Temperature value in celsius degree
32 float   setup_gain_boiler = 0.005;
33 float   setup_ref_boiler  = 1.26313;
34
35 // Variables for PID control
36 unsigned long startTime = 0;
37 double integral_brew = 0.0, integral_boiler = 0.0;
38 double previous_error_brew = 0.0, previous_error_boiler = 0.0;
39
40 // Variables for brew
41 int brew_counter = 0;
42 double temperature_brew_old = 22.0;
43
44 // Variables for boiler
45 bool boilerSetpointReached = false;
46 double temperature_boiler_old = 22.0;
47 int percentage_boiler;
48
49
50 void setup() {
51   Serial.begin(9600);
52   Serial.println("DONE.");
53   pinMode(controlPin, OUTPUT);
54 }
55
56 void loop() {
57   unsigned long currentTime = millis();
58   unsigned long elapsedTime = currentTime - startTime;
59
60   // double temperature_brew = readTemperature(thermocouple_brew, temperature_brew_old
```

```
    );
// double temperature_boiler = readTemperature(thermocouple_boiler,
    temperature_boiler_old);
val_brew = analogRead(analogPin_brew);                    // read the input pin
double temperature_brew = (float(val_brew) * setup_gain_brew - setup_ref_brew)/0.005
     ;

val_boiler = analogRead(analogPin_boiler);                    // read the input pin
double temperature_boiler = (float(val_boiler) * setup_gain_boiler -
    setup_ref_boiler)/0.005 ;

double error_brew = setpoint_brew - temperature_brew;
double error_boiler = setpoint_boiler - temperature_boiler;

integral_brew += error_brew;
integral_boiler += error_boiler;

double derivative_brew = error_brew - previous_error_brew;
double derivative_boiler = error_boiler - previous_error_boiler;

double output_brew = calculateOutput(Kp_brew, Ki_brew, Kd_brew, error_brew,
    integral_brew, derivative_brew);
double output_boiler = calculateOutput(Kp_boiler, Ki_boiler, Kd_boiler, error_boiler
    , integral_boiler, derivative_boiler);

double scaled_output_brew = scaleOutput(output_brew, boiler_threshold);
double scaled_output_boiler = scaleOutput(output_boiler, boiler_threshold);

updateSetpointReached(error_boiler);

bool brewstarted = checkBrewStatus(temperature_brew);

updateControlOutput(brewstarted, scaled_output_brew, scaled_output_boiler);

updatePreviousErrors(error_brew, error_boiler);
```

```
90
91    iteration++;
92    delay(Ts);
93  }
94
95
96  double calculateOutput(double Kp, double Ki, double Kd, double error, double &integral
        , double derivative) {
97    return Kp * error + Ki * integral + Kd * derivative;
98  }
99
100 double scaleOutput(double output, double threshold) {
101   if (output > threshold) {
102     return max_voltage;
103   } else {
104     return min_voltage;
105   }
106 }
107
108 void updateSetpointReached(double &error_boiler) {
109   percentage_boiler = abs(error_boiler) / setpoint_boiler * 100;
110   boilerSetpointReached = (percentage_boiler < boiler_error_margin);
111 }
112
113 bool checkBrewStatus(double temperature_brew) {
114   bool brewstarted = (temperature_brew > brew_threshold);
115   if (brewstarted) {
116     brew_counter++;
117     if (brew_counter == 1) {
118       integral_brew = 0;
119     }
120   }
121   return brewstarted;
122 }
123
```

```
124 void updateControlOutput(bool brewstarted, double scaled_output_brew, double
        scaled_output_boiler) {
125   if (brewstarted) {
126     analogWrite(controlPin, scaled_output_brew);
127   } else {
128     analogWrite(controlPin, scaled_output_boiler);
129   }
130 }
131
132 void updatePreviousErrors(double error_brew, double error_boiler) {
133   previous_error_brew = error_brew;
134   previous_error_boiler = error_boiler;
135 }
```

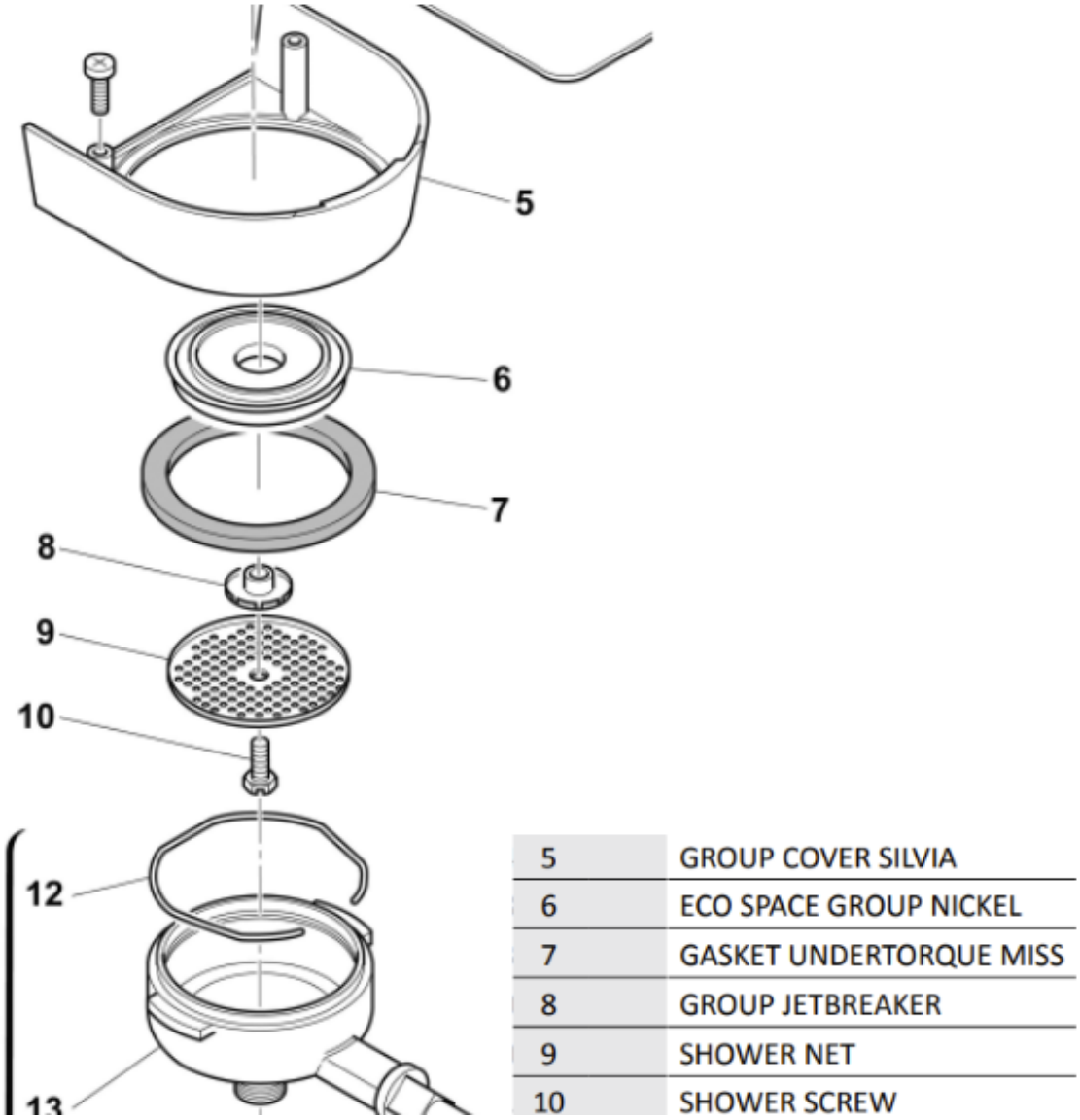

Fig. 14 Portion of Official Rancilio Silvia Parts Diagram

The diagram in Figure 14 originates from the official Rancilio Silvia parts manual sourced from a well-known espresso coffee enthusiast website [3].

VIII. Funding Source

Cornell's Sibley School of Mechanical and Aerospace Engineering funded this project.

IX. Acknowledgments

Along with his role as supervisor, Professor Dmitry Savransky generously donated his Rancilio Silvia espresso machine to this design project.

Additionally, espresso coffee has played a major role in the feasibility of finishing Cornell's MAE degree program.

References

[1] Batali, M. E., Ristenpart, W. D., and Guinard, J. X., "Brew temperature, at fixed brew strength and extraction, has little impact on the sensory profile of drip brew coffee," *Sci Rep*, Vol. 10, No. 1, 2020, p. 16450. https://doi.org/10.1038/s41598-020-73341-4.

[2] Luuk, I., "Arduino Serial to Spreadsheet," , 2021. URL https://circuitjournal.com/arduino-serial-to-spreadsheet.

[3] Love, W. L., "Rancilio Silvia M V1, V2, and V3 Parts Diagram," , ???? URL https://support.wholelattelove.com/hc/en-us/articles/4403765194131-Rancilio-Silvia-M-V1-V2-and-V3-Parts-Diagram.