This is a PID (Proportional, Integral Derivative) temperature controller. It has been designed around the ATmega328P microcontroller utilizing the Arduino Duemilanove bootloader.

*** NOTE - A separate Arduino board is NOT required, as this kit incorporates a basic Arduino clone.*****

A PID controller works by measuring the error (difference between the Set Point (desired temp) and the Process Value (measured temp). It then performs 3 separate calculations on that error value, the first is PROPORTIONAL to the size of the error, the second is the INTEGRAL, which is the amount of time that the process has been in error, the third is the DERIVATIVE, in which it measures the rate of change of the Process Value. It then calculates the sum of the previous 3 calculations and applies that sum as a percentage of a given window of time in the form of a pulse to a solid state relay (in the case of temperature control loops).

The Arduino code for this project has been adapted with single boiler home espresso machines particularly in mind, such as the Rancilio Silvia and the Le Lit PL041QE and others, however it is easily adapted to any other heating control application, such as Sous Vide Controllers, Kiln Controllers, etc.

The software for the Arduino PID controller itself consists of 3 main parts: the temperature reading code using the MAX31855 chip, the PID control algorithm code, and the menu system code, which allows changing of the Set Point and other parameters, such as temperature offset from boiler to group head.

**Time Required: 2-3 hours depending on experience**

**Experience Level: Advanced**

**Required tools and parts:**

*Soldering iron
*Flux pen
*Fine gauge solder (23 gauge or finer)
*Side cutters
*MAX31855 thermocouple interface chip (Available directly from Maxim Integrated)
*Thermocouple with junction suitable for your application (ie. Probe, Washer, etc)
*Solid State Relay
*USB FTDI 5V Cable (Note:- A single cable can be used for many different microcontroller projects)
*Computer with Arduino IDE v1.0
*Walwart 9V - 12V, 2.1mm Center Positive

**Bill of Materials:**

<table>
<thead>
<tr>
<th>Qty</th>
<th>Jameco SKU</th>
<th>Component Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>31114</td>
<td>47uF Electrolytic Cap Radial</td>
</tr>
<tr>
<td>1</td>
<td>51252</td>
<td>9V Regulator</td>
</tr>
<tr>
<td>2</td>
<td>25523</td>
<td>0.1uF Ceramic Cap</td>
</tr>
<tr>
<td>1</td>
<td>690865</td>
<td>1K resistor</td>
</tr>
<tr>
<td>1</td>
<td>697629</td>
<td>Green T1 LED</td>
</tr>
<tr>
<td>1</td>
<td>36011</td>
<td>1N4007 Diode</td>
</tr>
<tr>
<td>1</td>
<td>101179</td>
<td>2.1mm Power Jack</td>
</tr>
<tr>
<td>1</td>
<td>(Not Included)</td>
<td>Thermocouple</td>
</tr>
</tbody>
</table>
I suggest each kit builder purchase a thermocouple type suitable for their purpose & usage, (eg probe T-type or washer K-type, etc). These can be purchased from companies such as Omega - http://www.omega.com/ and Automation Direct - http://www.automationdirect.com/adc/Home/Home and many others

<table>
<thead>
<tr>
<th>Component</th>
<th>Note</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01uF Ceramic Cap</td>
<td>1</td>
<td>0.01uF Ceramic Cap</td>
</tr>
<tr>
<td>10K resistor</td>
<td>7</td>
<td>10K resistor</td>
</tr>
<tr>
<td>10uF Electrolytic Cap</td>
<td>2</td>
<td>10uF Electrolytic Cap</td>
</tr>
<tr>
<td>6 pin male header</td>
<td>1</td>
<td>6 pin male header</td>
</tr>
<tr>
<td>Tactile button Switch</td>
<td>5</td>
<td>Tactile button Switch</td>
</tr>
<tr>
<td>ATmega328 with Arduino Uno bootloader</td>
<td>1</td>
<td>ATmega328 with Arduino Uno bootloader</td>
</tr>
<tr>
<td>28 pin socket</td>
<td>1</td>
<td>28 pin socket</td>
</tr>
<tr>
<td>16 MHz Ceramic Resonator</td>
<td>1</td>
<td>16 MHz Ceramic Resonator</td>
</tr>
<tr>
<td>NPN General Purpose Transistor (600mA Collector Current)</td>
<td>7</td>
<td>NPN General Purpose Transistor (600mA Collector Current)</td>
</tr>
<tr>
<td>Red LED T1</td>
<td>1</td>
<td>Red LED T1</td>
</tr>
<tr>
<td>2.7K resistor</td>
<td>1</td>
<td>2.7K resistor</td>
</tr>
<tr>
<td>Solid State Relay</td>
<td>1</td>
<td>Solid State Relay</td>
</tr>
</tbody>
</table>

I recommend each kit builder purchase a SSR suitable for their purpose (eg Output DC 10A or Output AC 30A, etc)

<table>
<thead>
<tr>
<th>Component</th>
<th>Note</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HD44780 compatible 16x2 LCD</td>
<td>1</td>
<td>HD44780 compatible 16x2 LCD</td>
</tr>
<tr>
<td>10K potentiometer</td>
<td>4</td>
<td>10K potentiometer</td>
</tr>
<tr>
<td>330 ohm resistor</td>
<td>6</td>
<td>330 ohm resistor</td>
</tr>
<tr>
<td>1N4148 Diode</td>
<td>3</td>
<td>1N4148 Diode</td>
</tr>
<tr>
<td>pin header (16 pins)</td>
<td>1</td>
<td>pin header (16 pins)</td>
</tr>
<tr>
<td>USB FTDI Cable</td>
<td>1</td>
<td>USB FTDI Cable</td>
</tr>
</tbody>
</table>

One cable can be used for many projects. The Jameco FTDI breakout may be suitable, however I have not tested it with this kit.

Step 1 - Check Components. Additional Components Required.

****This kit can control MAINS POWER devices. Mains power electricity is DANGEROUS. Faulty mains wiring due to lack of knowledge can KILL you and/or others! It is YOUR responsibility to correctly wire the controlled devices or find someone who can. ********

Check off all components against the kit list before beginning assembly.

You will need to buy a SSR (Solid State Relay), USB FTDI 5V cable, a thermocouple and MAX31855 IC to suit the device you are controlling.

SSR's are available in both AC and DC output versions, make sure you get the right type for your application. The input voltage must accept 5V. Heating applications require a Zero-Crossing SSR (these make up the majority of available SSR's). The SSR switching current and voltage ratings MUST at least meet, or preferably exceed the ratings of the device you intend to control. Note:- some SSR's need a heatsink, especially if they are operating close to their maximum current. If the SSR is to mounted in a warm/hot area you MUST use a SSR that significantly exceeds the current rating of the device to be controlled.

Thermocouples- If you require a large temperature range (-200 C to +1350 C ), I recommend a K-type thermocouple. If you require a narrower temperature range (-200 to 350 C), I recommend a T-type thermocouple . Note that T-type thermocouples are more accurate than K-type. There are also other types of thermocouples available, make sure you know what type thermocouple you need before proceeding. Thermocouple sensor JUNCTIONS come in many types:- Washer, Probe, Bead, etc. You will need to select a thermocouple junction that best suits your application.

IMPORTANT NOTE:- If you get a MAX31855KASA+ IC, you MUST use a K-type thermocouple. If you get a MAX31855TASA+, you MUST use a T-type thermocouple. If you use the wrong thermocouple you will get incorrect readings. Both of the MAX31855 IC's are available directly from MAXIM Integrated - http://www.maximintegrated.com/datasheet/index.mvp/id/7273/t/al

If you are a student you can get free samples of these IC's from Maxim Integrated.
Step 2 - Solder SMD Chip

Solder the surface mount chip first, as it will be very difficult to solder it when there are other components around it later. The chip is mounted with pin 1 to the lower right hand side of the PCB.

I strongly recommend the use of extra flux, a magnifier and tweezers to carry out this step. Solder paste is NOT required to solder surface mount chips with a soldering iron.

Before placing the chip, solder where pin 1 of the chip will go so there is a little spot of solder on the pin 1 pad. Once the solder cools, add a little flux and then place the chip carefully with tweezers.

When correctly placed (double check the chip orientation is correct!), tin the tip of your soldering iron with a little solder and touch it to pin 1 and the spot of solder you added earlier. The extra flux will help the solder flow without needing to add more solder.

When this is done, check the alignment of the other pins to their respective pads. If correctly aligned, flux each pin & pad, then carefully solder each one (the additional flux will help prevent solder bridges). If the IC is NOT correctly aligned, add some more flux to pin 1, and touch the tip of the soldering iron to pin 1 and align the chip before attempting to solder other pins. Keep the amount of time that the soldering iron is on any of the pins of the MAX31855 chip to a minimum to avoid damaging the chip.

When done clean the area with some isopropyl alcohol and an old toothbrush to remove the flux residue.

Step 3 - 5V Power Supply Construction and Testing

Position the components for the +5V power supply on the PCB. These components will be fitted on the LHS on the PCB. They include the DC jack, 1N4007 diode, 2 x 47uF electrolytic capacitors, the 0.1uF capacitor, the 2.7K resistor and the LM7805 voltage regulator. You will notice they are all grouped together on the PCB.

Make sure the electrolytic capacitors are positioned with the "-" symbol on the capacitor on the opposite side to the "+" marking on the PCB. Make sure the LM7805, and the 1N4007 diode are positioned with the stripe (negative side) as shown on the PCB. Solder the
above parts into position and clip excess lead length.

After soldering and clipping the excess leads, turn the PCB over and fit the green LED to the other side of the board raising it approx 3mm (1/8") above the board. Before soldering, double check the polarity. The flat side of the LED is the "-" lead, the longer lead is the "+" lead. After soldering clip the excess lead length on the other side of the board.

Now connect a wall-wart power supply (2.1mm Center Positive) of a voltage of 9V to 12V DC. The green LED should illuminate. If the green LED does NOT illuminate, something has been positioned incorrectly, or there is a bad solder joint. You will need to rectify any incorrectly positioned or soldered components before continuing the build.

Step 4 - 3V3 Power Supply

The 3V3 power supply components inhabit the lower right hand side of the PCB close to the MAX31855 chip. These components are 3 x 1N4148 signal diodes, 2 x 10uF electrolytic capacitors and 1 x 10K resistor (below the MAX31855 chip). Position and solder the 3 x 1N4148 signal diodes as shown to the lower right side of the PCB. Take care to observe the polarity of the diodes. The stripes on the diodes (negative side) must match the stripes on the PCB.

***Important Note - The diodes are NOT all facing the same way. Double check the polarity before soldering***

Next position and solder the 2 x 10uF Electrolytic Capacitors, taking care to observe polarity as indicated on the PCB. Finally position the 10K resistor & solder into place.

If you have a multimeter, connect your wallart to the PCB and confirm approx 3.3V (3.1V-3.3V) is present on the left hand side of the 10K resistor (with the negative lead of the meter to the center pin of the LM7805 Regulator). If you do not have a multimeter, you can press the leads of the red LED against pins 1 (GND, negative side of LED) and pin 4 (3V3 Vcc, positive side of LED) of the MAX31855 chip.

If no voltage is present, or the LED does not illuminate, you have most likely positioned one or more of the diodes incorrectly. If this is the case, check the other components as well as there may be other polarized components incorrectly positioned.
Step 5 - Solder Remaining Components

As you are soldering components, it is a good practice to clean the PCB periodically with some isopropyl alcohol and a swab or an old toothbrush to remove the flux residue.

Start positioning/soldering the components with the lowest profile (ie, resistors) and work your way up to larger components to make your job easier.

Position all of the resistors, making sure you have placed each resistor in its correct position. Use a resistor color code chart to check against if you don't know the color codes or use a multimeter to check. Solder them into place and clip excess lead lengths on the back of the board.

Fit the red LED on the same side of the PCB as the green LED. Allow 3mm (1/8") of extra lead length so the LED sticks out from the board, just like you did for the green LED.

Now position the socket for the ATmega328PU chip in place taking care to align the notch in the socket with the notch shown on the PCB. Double check the socket is positioned correctly before proceeding. Solder only one pin, check the socket is inserted correctly and fully, then solder the other pins.

Position the ceramic capacitor and the 16MHz resonator and solder into place. Now position the transistor in place making sure the orientation is as shown on the board and solder into place.

Position and solder the 2 position terminal connectors to the PCB, with the terminal holes facing out from from the PCB.

Position the 6-pin right angle male header in place with the short, straight pin side through the PCB and the right angle pin side pointing down, and solder into place.

Position and solder the 16 pin male header in place with the 90 degree pin side through the PCB and the short straight pin side facing AWAY from the PCB.

Alternatively, if you wish to change the form factor of the Arduino PID controller, you can choose to use straight pin header. In this form factor, the mounting holes in the PCB line up with the mounting holes on the LCD.

Position the potentiometer and solder into place.

Now you should have just five button switches and the LCD left.

Turn the PCB over and fit the five switches to the same side of the PCB as the LEDs and solder into place.

With the majority of the components (except the switches and LEDs) facing down, position the LCD. Before soldering make sure that the pin 1 & pin 16 marks on the LCD match the pin 1 and pin 16 marks on the PCB respectively before continuing. After double checking the LCD is positioned correctly, solder it into place.

Step 6 - Fitting ATmega328 chip, Testing and Programming.

Check the notch marked on the ATmega328 chip and align it with the notch on the socket (double check that the socket is facing the correct way!)

You may need to bend the pins of the chip slightly inwards. Carefully push the chip into the socket, taking care not to bend the pins as you insert the chip. Visually inspect the chip/socket before going further.
Download the Arduino IDE V1.0 for your OS from http://arduino.cc/en/main/software
Make sure you get the right version! Install the software.

When installed, connect the USB cable to your computer first, then connect the other end to the PCB making sure the black wire in the cable lines up with the pin marked "GND".

******Note: If you connect it the wrong way you will damage the ATmega328 chip, so make sure you position it correctly!*******

Open the Arduino IDE now installed on your computer. Click on "Tools", "Board" and select "Arduino Uno". Then click on "Tools" and "Serial Port" and select the COM port the PCB is connected to. If you are not sure which COM port is being used, go to your Device Manager, click on the arrow next to "Ports (COM & LPT)". You should see at least 1, possibly more devices listed. Unplug the cable from your computer, and plug it in again. The device that disappears and reappears from the list is the Arduino PID Controller. Take note of the COM port number.

Go back to the Arduino IDE and now select the correct port.

To test the ATmega chip, go to the Arduino IDE and click on "File", "Examples", "Basics", "Blink" This will open a new IDE window with some code in it.

Go to line 13 (line number is at the bottom left of the Arduino IDE window) where it says:

```
  pinMode(13, OUTPUT);
```

change the "13" to "7" (without the quotes)

Now go to lines 15 and 17 and change the "13" to "7"

Now click on the Verify button (looks like a "Tick"), when it compiles, then click on the Upload button (next to the Verify button). If it does not compile, you have made a syntax error.

Immediately press and HOLD the Reset button on the Arduino PID controller, when you see the text "Binary sketch size: xxxx bytes (of a 30720 byte maximum)" appear in the bottom of the Arduino IDE window, RELEASE the Reset button.

If you get an avrdude error, it means you didn't release the button soon enough, or you have the wrong Arduino board selected under Tools. Check the correct board is selected under Tools and try uploading again and release the RESET button at the correct time.

When it uploads correctly, you should see the red LED begin blinking. This indicates that the Arduino is working correctly. You are now ready to upload the PID code.

---

**Step 7 - Program the PID Controller**

Go to https://github.com/CoffeeTronics and locate and download the zip for Arduino PID controller for Espresso that has the highest version number (version 2.0 and above). The "Download Zip" button should be on the right hand side of the webpage, just below the "Clone In Desktop" button. Extract the downloaded zip file to your computer. Go into the extracted folder & copy the CONTENTS of the extracted "Libraries" folder (NOT the Libraries folder itself) to your Arduino "libraries" folder (where you installed the Arduino IDE).

Copy the Arduino_PID_Controller sketch to your Arduino "Sketchbook" folder.

******Note:- the Arduino IDE installation process often places this folder in your "Documents" folder. If not, you will need to search for it.*******

Now close ALL Arduino IDE windows you have open and restart the IDE. Click on "File", "Sketchbook". The file "Arduino_PID_for_Espresso_v2.0" should now be there. Click on it, a new window will open containing the code. (If the "Arduino_PID_Controller" sketch is not there, most likely you did not close all Arduino IDE windows. Close all Arduino IDE windows and restart the IDE and check again)
Once you have opened the code, click on the "Verify" button. After it compiles, upload the code to the Arduino PID Controller (Don’t forget the RESET button!). If it doesn’t compile, you may have downloaded the wrong version of the Arduino IDE, or an obsolete version of the Arduino PID Controller code or you may have placed the contents of the Libraries folder in the wrong folder, or have a Libraries folder inside the Libraries folder.

Once it has uploaded successfully, remove the programming cable and walwart power from the PID Controller, the LCD screen should now be OFF. Now connect the PID Controller OUTPUT pins ("-" is on the lower side, "+" is on the upper side) to the INPUT pins of the SSR you purchased. Make sure the + pin of the output goes to the + pin of the SSR, and the "+" output pin goes to "-" input of the SSR.

*****IMPORTANT:- Make sure the device you intend to control is UNPLUGGED from the power outlet before proceeding with the next step!!*****
Connect the OUTPUT side of the SSR to the device you are controlling (eg. Inline with the heating element).

Connect the Ground ("-" input) of the SSR INPUT side to the Ground of the device being controlled. This helps prevent capacitive ground loops, which can cause erratic temperature readings.

Connect your thermocouple to the thermocouple input terminal connector, taking care to observe polarity. If you have used a K-type, then the Yellow Wire is + and the Red wire is -. If you have used a T-type thermocouple, the Blue wire is + and the Red wire is -. The "+" side of the thermocouple terminal block is on the upper side, while the "-" is the lower side.

Connect your walwart to the DC Jack on the PCB. The Arduino PID Controller will boot up, the LCD will say "Arduino PID for Espresso". After 3 seconds, it will begin displaying temperatures. As the code contains an array for smoothing the temperature readings, the displayed temp will increase rapidly for the first couple of seconds before stabilizing.

Hold the thermocouple junction (the end that senses temperature) in your hand and you should see the temperature reading on the LCD increase from your body heat. If it decreases, you have connected the thermocouple backwards.

Now you can fit the thermocouple to the device being measured. If you are using a washer thermocouple and the place where you are fitting the thermocouple to is Grounded, you may find the temperature readings jump around a bit. If this occurs, you will need to electrically insulate the thermocouple from the device being measured with Mica washers and some teflon tape around the screw/bolt. Don’t forget to use some thermal compound to keep good thermal conductivity.

You are now ready to tune the controller.

---

**Step 8 - Tuning the Controller**

Note: The Autotune function steps the output and observes changes in the Input to calculate parameters for the proportional, integral and derivative bands.

Go to the Google code page for Processing- https://code.google.com/p/processing/downloads/detail?name=processing-1.5.1-windows.zip&can=2&q= and download the zip folder. Extract it to your computer.
Now go here - http://www.sojamo.de/libraries/controlP5/
and download the ControlP5 library. Follow the instructions listed on the webpage under "Installation"

Now go to the where you extracted the Arduino PID ZIP and look for "AutoTune Front and Backend" folder. Copy the folder to your Arduino folder (most likely in your Documents folder).

Open the Arduino IDE, then click File, Open. Navigate to the extracted zip file and open it. Open the folder Autotune Front and Backend, then open the folder AutotuneMAX31855Backend, then open the folder AutotuneMAX31855. Double click on the ino file AutotuneMAX31855. This will open the BackEnd of the Autotune code in the Arduino IDE.
Ctrl F (FIND) "//EDIT" (without quotes) and hit ENTER to find parameters than can be changed by the user (YOU). If you change parameters, it is recommended you save the code with a different name, should you wish to easily change back to the original code. Upload the code to the Arduino PID Controller (make sure programming cable is oriented correctly, and RESET button timing is correct). When finished programming leave the code connected to the Arduino PID controller as it will be used by the processing FrontEnd. Close the Arduino IDE.

Open the Processing IDE, click on File, Open, then find the zip folder you extracted, AutoTune Front and Backend, ProcessingFrontend, PID_FrontEnd_v0_3. Select the PDE file named PID_FrontEnd_v0_3. This will open the code in a new Processing window.
Do a CTRL F (find), type in "//EDIT" (without the quotation marks") and hit ENTER. This will take you parameters that may be adjusted by the end user (YOU) before running the tuning front-end.
If you make any changes, save the updated file under a new name, so it is easy to go back to the original if needed.

Now click "Run" (looks like a Play button). This will open a window where you can manually tune or Autotune the controller. If after about 20 seconds you cannot see the Setpoint, output and Input being graphed (RHS of window), try closing the window and click the Run button again. If you have problems getting Processing to graph values, check you have closed all Arduino IDEs. Occasionally Windows causes problems, and needs to be rebooted.

I will assume you want to use Autotune as it is far easier for beginners. Check the programming cable is still connected.
The values to the right hand side of the buttons in the Processing window are the current values in use. The values to the left hand side or BELOW the buttons can be changed to what you require. These values will be shifted in when the SEND_TO_ARDUINO button is clicked.
Set the TOGGLE_AM button to MANUAL, set the SETPOINT button to your required Setpoint, check TOGGLE_DR is set to DIRECT, check TOGGLE_TUNING is set to OFF. Now click the SEND_TO_ARDUINO button. This puts the controller in MANUAL mode. Start small and slowly increase the OUTPUT button (0-1000) each time you change it, click the SEND_TO_ARDUINO button (Toggle the TOGGLE_AM button to AUTO then back to MANUAL before clicking SEND_TO_ARDUINO). Continue slowly increasing the OUTPUT until your system is in equilibrium with the desired temperature. This can take some time, especially for slow temperature loops, so be patient!
When your system is at equilibrium with the desired SetPoint, change the "Toggle Tuning" button to ON and click "Send to Arduino". The value to the right hand side of the Toggle Tuning button should change to ON. This means it is now AutoTuning the loop. If you watch the OUTPUT graph, you will see the AutoTune function Step the Output several times. When the value beside Toggle Tuning changes from ON to OFF it has finished. The values you now see next to P, I and D are the Autotuned values. Make a note of these values.

Note:- The Autotuned values are a good place to begin manual fine tuning to get better tuning parameters. There are plenty of guides on manual tuning of PID loops. In my personal experience, I have found that running Autotune first, and then tweaking the values gradually gives quite decent results.

Now you can either spend some more time manually fine tuning the loop, or you can upload the Arduino_PID_for_Espresso sketch to the PID controller (inserting the parameters you noted into lines 120-122 of the Arduino PID Controller sketch).

The structure of the menu system is
MENU
| BREW ESPRESSO--This will change setpoint to Brew Setpoint
| STEAM MILK--This will change setpoint to Steam Setpoint
| BREW TEMP--SET BREW POINT--Press L and R keys to change Brew Setpoint
| STEAM TEMP--SET STEAM POINT--Press L and R keys to change Steam Setpoint
| OFFSET--SET OFFSET--Press L and R keys to change Offset (this is used to display "at the group" temps for espresso machines)
| SCALE-- Not currently in use, will be implemented in a software update available through GitHub.