USERS MANUAL

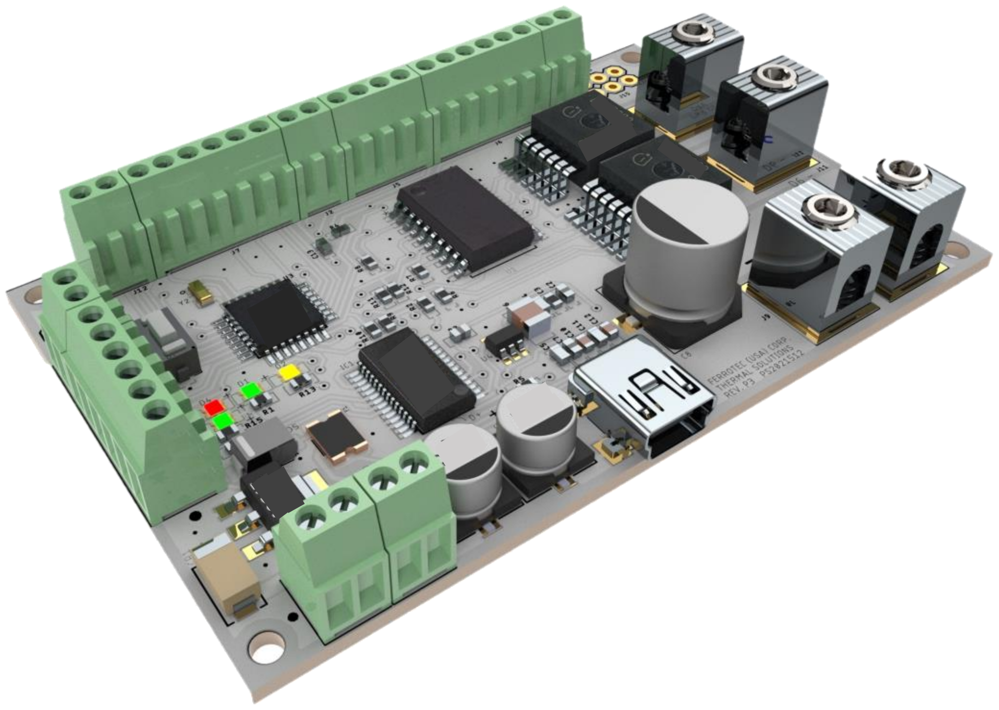
FTCP-1200

Programmable Multi-Purpose Controller

Thermoelectric Controller (TEC)

FerroTec

Thermal Solutions Group

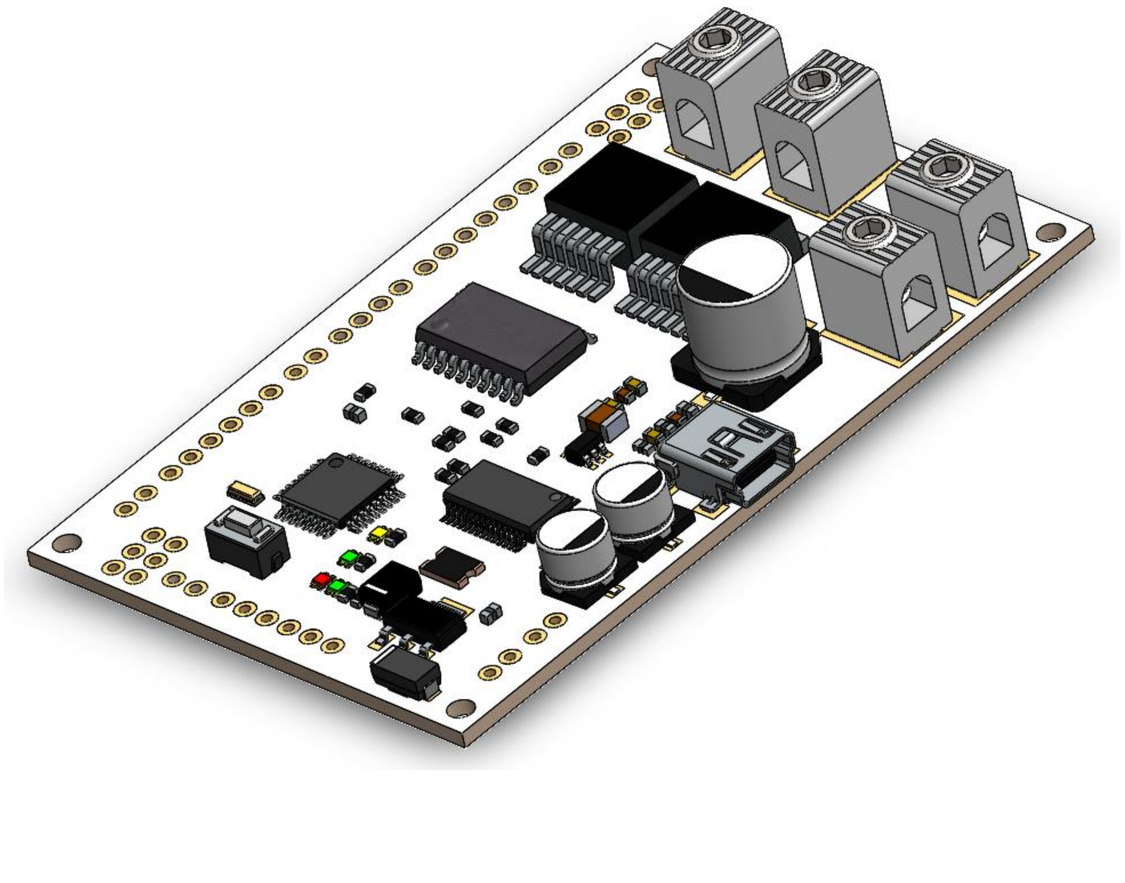


FTCP-1200MAN REV-D, Apr. 2024

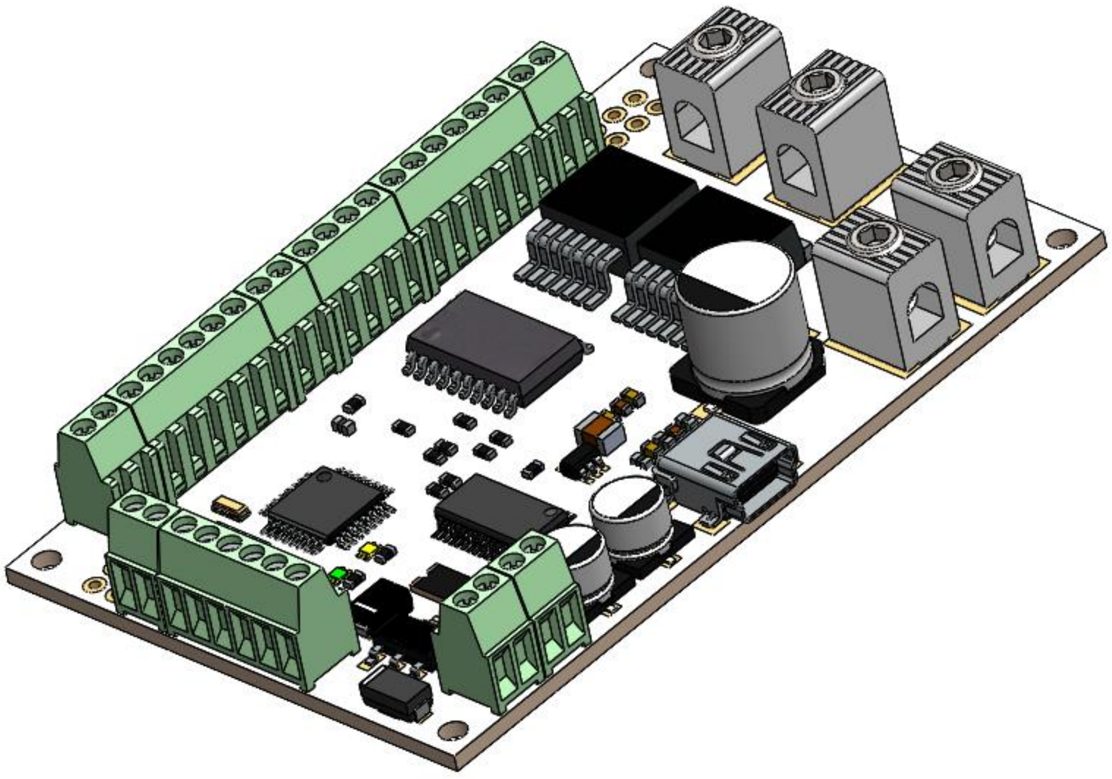
FORWARD

The Ferrotec FTCP-1200 thermoelectric temperature controller is a programmable microcontroller-based device that can be incorporated into a thermoelectric assembly or other DC motor and control applications. This manual currently covers the 202116 and the 202117 models. The main difference between the 202116 and 202117 models is that the 202117 model has screw terminals for the ease of attaching wires whereas the 202116 model has thru hole solder points in the same positions.

202116: With solder connection thru holes



202117: With screw terminal connection points



April 4, 2024

566 Exchange Ct

Livermore, CA 94550

Phone 1-800-522-1215 • Fax 925-449-4096

CONTENTS

1. **Description**

The Ferrotec FTCP-1200 thermoelectric temperature controller is a programmable microcontroller-based device that can be incorporated into a thermoelectric assembly or other suitable application. It has an on-board bi-polar driver; thus, can be used to drive thermoelectric modules in both cooling and heating modes (e.g., via polarity switching). Temperature monitoring/feedback can be used to control temperature to a high degree of accuracy in an application. The controller can be used in high power applications (up to 40V @ 30A). It is highly flexible, with a variety of inputs and outputs (digital/analog), allowing it to be used to power fans, read temperatures, control relays, set alarms, etc.

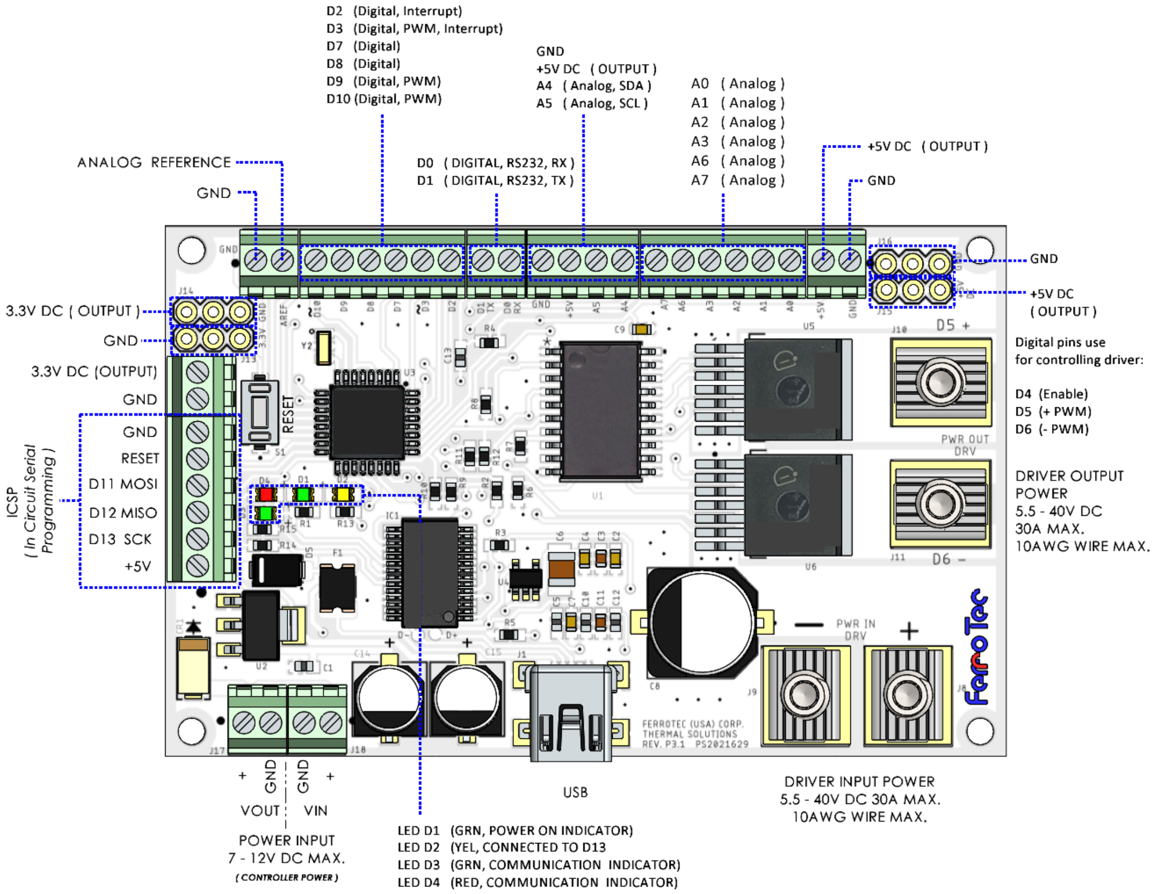
Microprocessor details:
- Type: ATmega328P
- Architecture: AVR
- Operating voltage: 5V
- Flash memory: 32 kB (2 kB used for bootloader)
- SRAM: 2 kB
- EEPROM: 1 kB
- Clock speed: 16 MHz
- Analog in pins: 8
- A/D resolution: 10-bit (1024 counts)

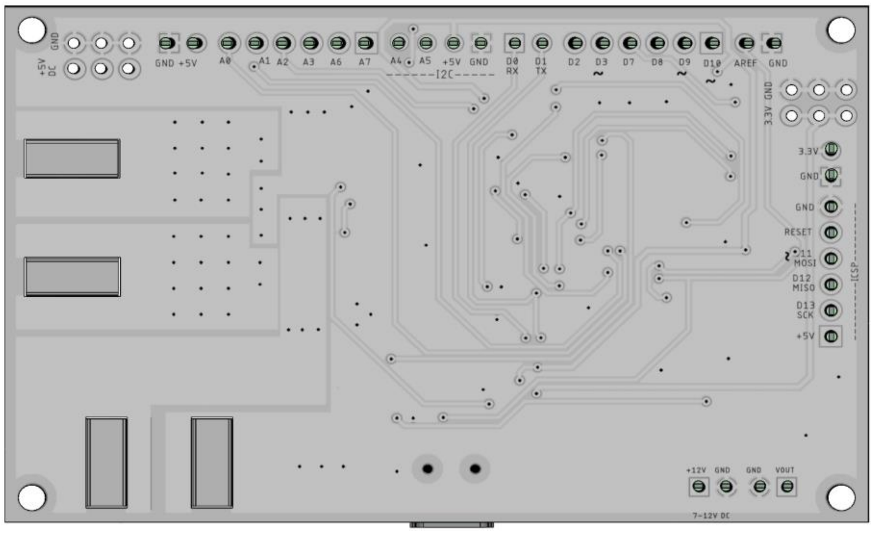
Controller-specific details:
- USB Connector Input 5 VDC
- Input voltage: 6-12 VDC to V in pins
- Digital I/O (pins): 14 total, 11 available (D4, 5, and 6 used for H-Bridge)
- 40 ma / pin max current
- 200 ma max current total
- Available voltage outputs:
  - 5 VDC: 6 (max total draw 1.5A, requires V in supply above 500 ma)
  - 3.3VDC: 4
  - 6 - 12 VDC: 1 (V out pins)
- Communications: UART, I2C, and ICSP
- LEDs: 4 each
- Temperature sensors: Any analog signal <= 5 VDC. (Thermocouples require the use of an amplifier such as MAX31856 or MCP9600)

Driver details:
- Type: Dual H-bridge Driver
- Voltage: 5.5 – 40V (Must be 0.5 V > VCC)
- Current: 30A max

* Driver input power is supplied from a separate DC power supply

2. <u>202117 Controller with Screw Terminals Pin Layout and Function. 202116 controller uses same positions but with no screw terminals</u>


D2 (Digital, Interrupt)
D3 (Digital, PWM, Interrupt)
D7 (Digital)
D8 (Digital)
D9 (Digital, PWM)
D10 (Digital, PWM)
GND
+5V DC (OUTPUT)
A4 (Analog, SDA)
A5 (Analog, SCL)
A0 (Analog)
A1 (Analog)
A2 (Analog)
A3 (Analog)
A6 (Analog)
A7 (Analog)
+5V DC (OUTPUT)
GND
ANALOG REFERENCE
GND
D0 (DIGITAL, RS232, RX)
D1 (DIGITAL, RS232, TX)
GND
+5V DC (OUTPUT)
3.3V DC (OUTPUT)
GND
3.3V DC (OUTPUT)
GND
GND
RESET
D11 MOSI
D12 MISO
D13 SCK
+5V
ICSP (In Circuit Serial Programming)
Digital pins use for controlling driver:
D4 (Enable)
D5 (+ PWM)
D6 (- PWM)
DRIVER OUTPUT POWER 5.5 - 40V DC 30A MAX. 10AWG WIRE MAX.
RESET
PWR OUT DRV
PWR IN DRV
D5 +
D6 −
FerroTec
FERROTEC (USA) CORP. THERMAL SOLUTIONS REV. P3.1 PS2021629
+ GND GND +
VOUT VIN
POWER INPUT 7 - 12V DC MAX. (CONTROLLER POWER)
USB
DRIVER INPUT POWER 5.5 - 40V DC 30A MAX. 10AWG WIRE MAX.
LED D1 (GRN, POWER ON INDICATOR)
LED D2 (YEL, CONNECTED TO D13
LED D3 (GRN, COMMUNICATION INDICATOR)
LED D4 (RED, COMMUNICATION INDICATOR)



Bottom Side

2.1. **Input Power**

There are 2 methods for powering the controller:

2.1.1. USB port

2.1.2. Screw terminals for Vin and GND, 6-12 VDC

Current consumption: 19 mA (board electronics, quiescent state)

2.2. **Output Power**

There are a variety of output power options:

5 VDC: Screw terminals [202117] and 3 solder through holes, 1.5 A max
3.3 VDC: Screw terminals [202117] and 3 solder through holes, 150 mA max
5 VDC and 3.3 VDC: 500 mA max total with USB power input
VOUT screw terminals 6 - 12 VDC (same as board input voltage): 1 pin, 3A max

2.3. **Digital Input and Output**

Each of the 14 digital pins (some pins have dedicated functions) can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. Each operates at 5 volt logic level.

Digital I/O (pins):	11 useable pins, D0 – D3, D7 – D13; D3, D9, and D10 are PWM-type
Digital I/O (LED):	D13, shared with LED D2, see below
Digital I/O (hardwired):	D4 (H-Bridge enable), D5 & D6 (H-Bridge polarity control)
DC current per I/O pin:	40 mA

2.4. **Serial**

D0 (RX) and D1 (TX): Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the FTDI USB-to-TTL Serial chip.

2.5. **External Interrupts**

D2 and D3 can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.

2.6. **PWM**

Some digital I/O have PWM function capability. These are D3, D5**, D6**, D9, D10, and D11, and are controlled using the analogWrite() function.

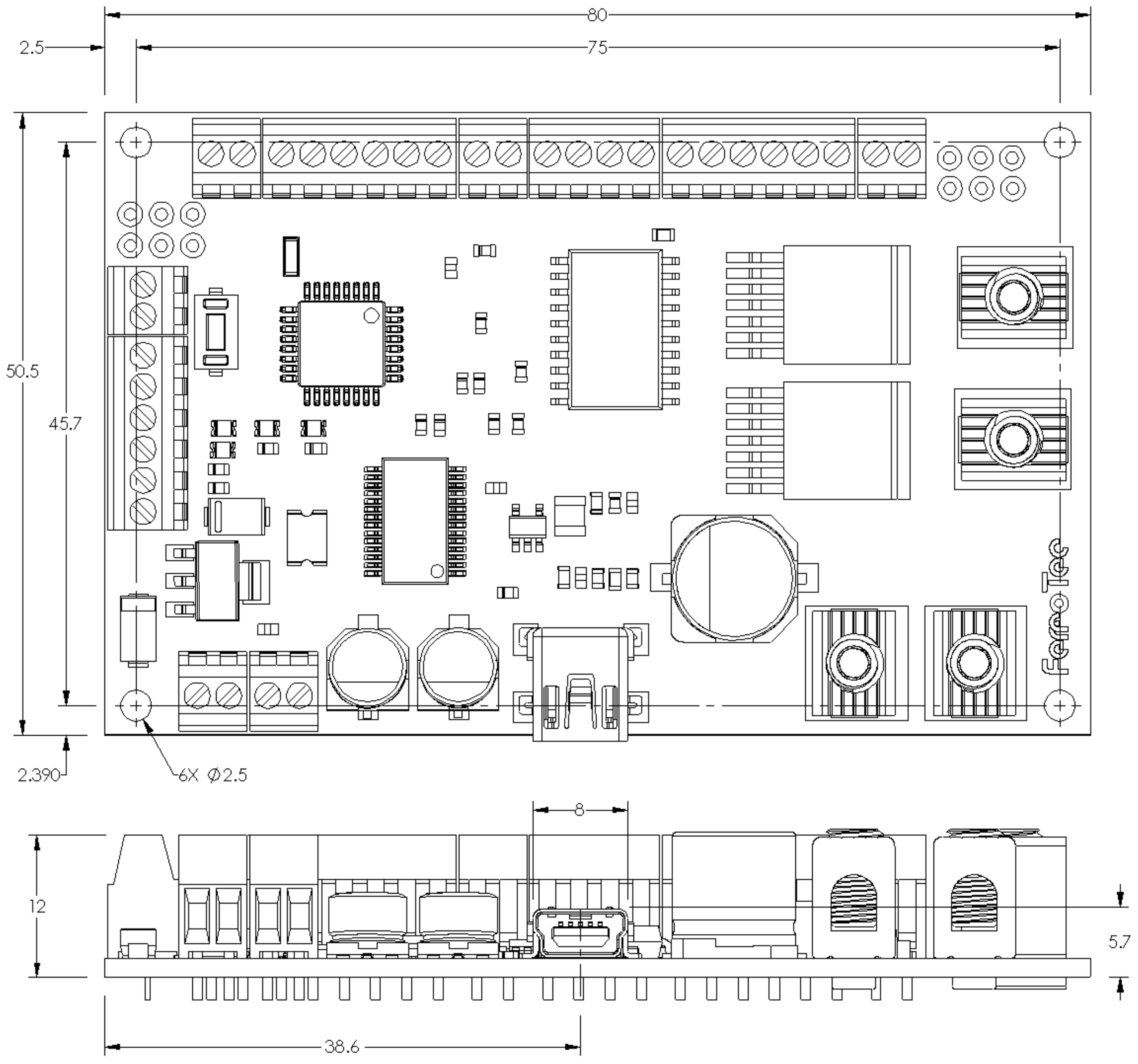
** Dedicated to H-Bridge

2.7. **SPI (Serial Peripheral Interface), ICSP (In-circuit Serial Programming)**

6 pins → GND, Reset, MOSI (D11), MISO (D12), SCK (D13), +5V

2.8. **LEDs**

D1 (green):	power on indicator; if not lit, when power is applied, there is an error
D2 (yellow):	driven by digital pin 13; if pin is high, light is on, if pin is low, light is off
D3 (green):	RX communication
D4 (red):	TX communication



202117 Model Depicted Above

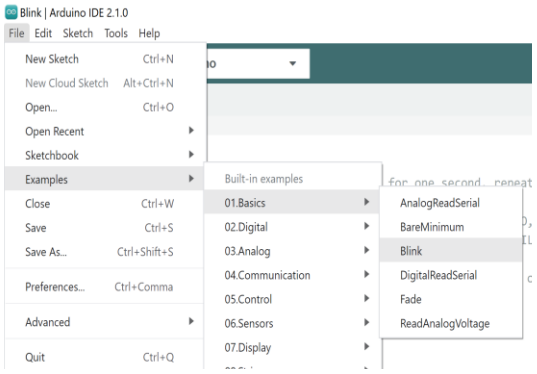
3. SOFTWARE SETUP

The Ferrotec FTCP-1200 thermoelectric temperature controller is compatible with the most popular integrated development environment (IDE) software for programming the on-board ATmega328P microcontroller (example: Arduino IDE). Arduino IDE is an open-source platform with many different outlets and guides for help with using the program. There is a very extensive set of libraries available to make programming easier.

Other methods include MicroPython, assembly language, and C#. There are alternative languages such as XOD and Snap as well.

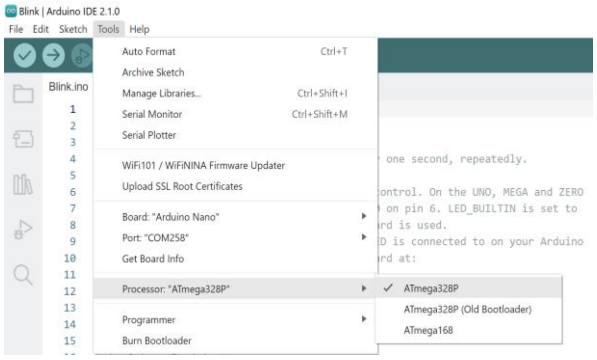
In this example, to program the FTCP-1200, we are going to use the compatible IDE from Arduino. The IDE is free to download from https://www.arduino.cc/en/Main/Software. Get the latest version of the Arduino Desktop IDE from the download page for your operating systems. When the download finishes, proceed with the installation. Allow the driver installation process when you get a warning from the operating system. Once the installation has finished, we are going to program the FTCP-1200 with a sample code provided by Arduino.

To set up the Ferrotec FTCP-1200 thermoelectric temperature controller to receive the program, you will need a mini-USB cable (see left). To begin, connect the board to your computer using the USB cable. A green power LED on the controller board should turn on. Double-click the icon to start the IDE.

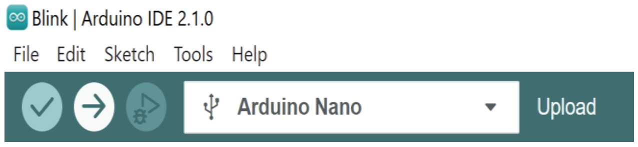


Once the software starts, you can either create a new project or open an existing project example. We will be using an existing project to demonstrate how to load code onto the FTCP-1200. The example code has the name Blink (see right). This code turns an LED on and off with time delay.

To avoid any error uploading a code to the FTCP-1200 thermoelectric temperature controller, you must select the correct com port, board name, and microcontroller type which matches with the board connected to your computer. Go to Tools, Board, Arduino Nano (see right)


Blink | Arduino IDE 2.1.0
File Edit Sketch Tools Help
Blink.ino
Auto Format Ctrl+T
Archive Sketch
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter
WiFi101 / WiFiNINA Firmware Updater
Upload SSL Root Certificates
Board: "Arduino Nano"
Port: "COM258"
Get Board Info
Processor: "ATmega328P"
Programmer
Burn Bootloader
✓ ATmega328P
ATmega328P (Old Bootloader)
ATmega168

Once you've selected correct com port and the Arduino Nano board, verify that the correct microcontroller chip is selected. (Right) Now you can upload the code by pressing the upload arrow (Below). After the board has been programmed, you should see the yellow LED light blinking.



4. HARDWARE SETUP

4.1. Temperature Sensor

4.1.1. Thermistor

A thermistor is a thermal resistor, a resistor whose resistance changes with temperature. Technically, all resistors are thermistors, their resistance changes slightly with temperature. However, the change is usually very, very small and difficult to measure. Thermistors are made so that the resistance changes drastically with temperature. It can be 100 ohms or more of change per degree.

There are two kinds of thermistors, NTC (negative temperature coefficient) and PTC (positive temperature coefficient). In general, NTC sensors are used for temperature measurement. PTCs are often used as a resettable fuse; an increase in temperature increases the resistance, which means that as more current passes through them, they heat up, and 'choke back' the current and are quite effective for protecting circuits.

Thermistors have some benefits over other kinds of temperature sensors, such as analog output chips or digital temperature sensor chips or thermocouples.

First, they are much cheaper than all the above! A 5% accuracy thermistor can cost as little as 10 cents each in bulk.

They are also much easier to waterproof since they are just resistors.

They work over a wide range of voltages (e.g., digital sensors typically require 3 or 5V logic).

Compared to a thermocouple, they do not require an amplifier to read the minute voltages. Any microcontroller can read a thermistor.

They are difficult to break or damage and are much simpler and more reliable.

On the other hand, they require a little more work to interpret readings, and they do not work at very high temperatures as thermocouples do.

Their simplicity makes them very popular for basic temperature feedback control. For example, let's say you wanted to have a fan that turns on when the temperature gets high. You could use a microcontroller, and a digital sensor, and have that control a relay to power the fan. Alternately, you could use the thermistor to feed the base of a transistor; as the temperature rises, the resistance goes down, feeding more current into the transistor until it turns on. (This is a rough idea. You would need a few more components to make it work).

The FTCP-1200 thermoelectric temperature controller can be used with thermistors, analog output temperature chips, and digital temperature sensor chips. It cannot be directly connected to thermocouples.

4.1.2. Analog output temperature sensors

There are a variety of temperature sensors that output an analog voltage proportional to temperature. This voltage can be input into any of the A/D pins (A0 – A3, A6, and A7) on the controller and the temperature converted according to the conversion factor.

4.2. Thermoelectric Module

4.2.1. Techniques used to install thermoelectric modules in a cooling system are extremely important. Failure to observe certain basic principles may result in unsatisfactory performance or reliability. Some of the factors to be considered in system design and module installation include the following:

4.2.1.1. Thermoelectric modules have high mechanical strength in the compression mode, but shear strength is relatively low. As a result, a TE cooler should not be designed into a system where it serves as a significant supporting member of the mechanical structure.

4.2.1.2. All interfaces between system components must be flat, parallel, and clean to minimize thermal resistance. High conductivity thermal interface material (TIM) is often used to ensure good contact between surfaces.

4.2.1.3. The “hot” and “cold” sides of standard thermoelectric modules may be identified by the position of the wire leads. Wires are typically attached to the hot side of the module, which is the module face that is in contact with the heat sink. For modules having insulated wire leads, when the red and black leads are connected to the respective positive and negative terminals of a DC power supply, heat will be pumped from the module’s cold side, through the module, and into the heat sink. Note that for TE modules having bare wire leads, the positive connection is on the right side and the negative connection is on the left when the leads are facing toward the viewer and the substrate with the leads attached presented on the bottom.

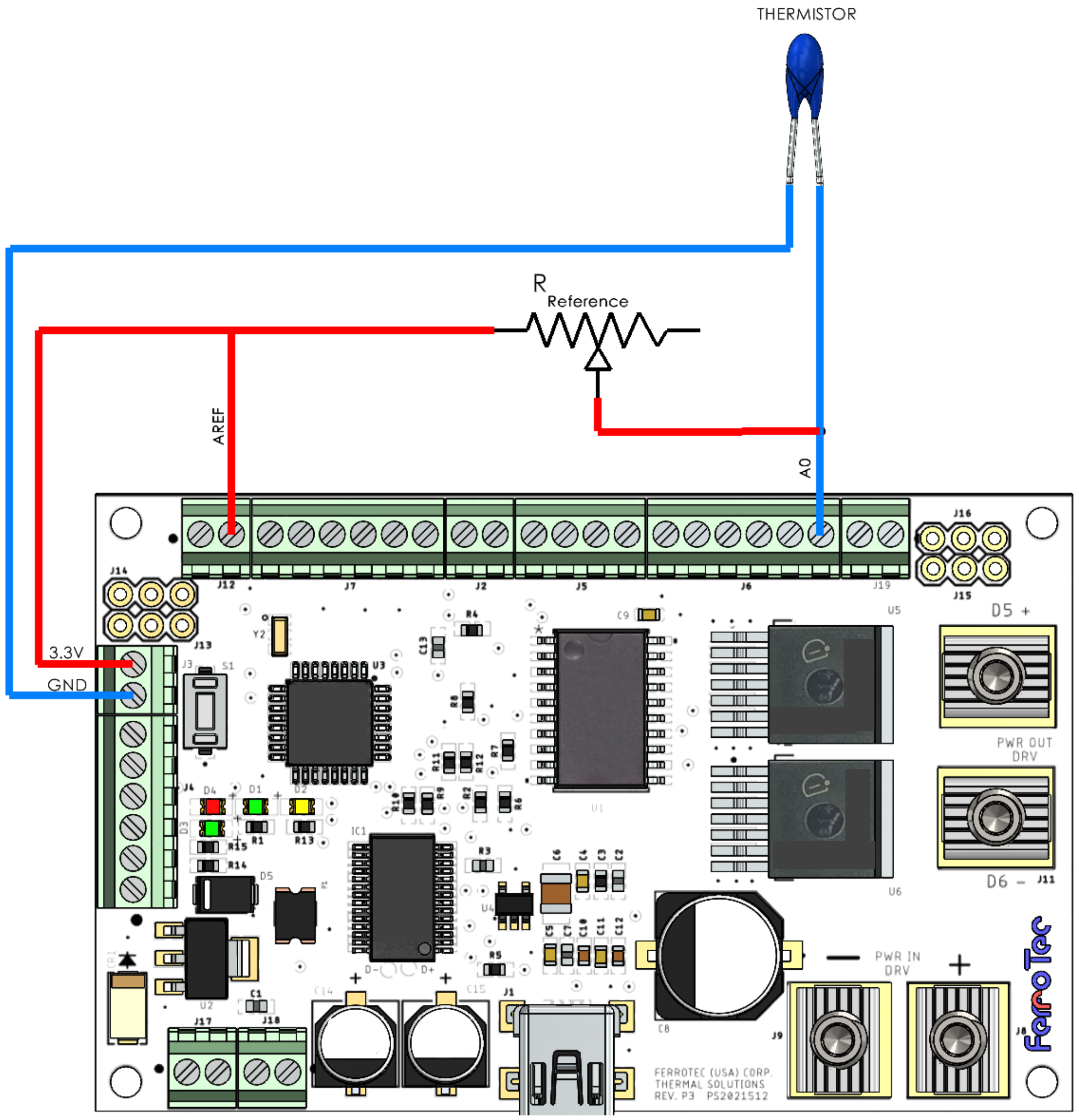
4.2.1.4. When cooling below ambient, the object being cooled should be insulated as much as possible to minimize heat loss to the ambient air. To reduce convective losses, fans should not be positioned so that air is blowing directly at the cooled object. Conductive losses also may be minimized by limiting direct contact between the cooled object and external structural members.

4.2.1.5. When cooling below the dew point, moisture or frost will tend to form on exposed cooled surfaces. To prevent moisture from entering a TE module and severely reducing its thermal performance, an effective moisture seal should be installed. This seal should be formed between the heat sink and cooled object in the area surrounding the TE module(s). Flexible foam insulating tape or sheet material and/or silicone rubber RTV are relatively easy to install and make an effective moisture seal. Several methods for mounting thermoelectric modules are available and the specific product application often dictates the method to be used.

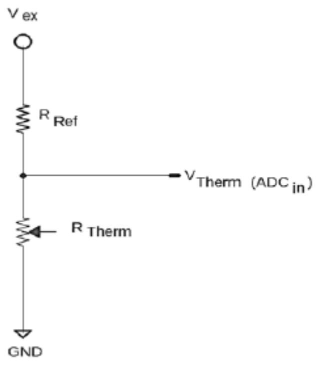
4.3. Power Supply

4.3.1. Thermoelectric coolers operate directly from DC power. Suitable power sources can range from batteries to simple unregulated “brute force” DC power supplies to extremely sophisticated closed-loop temperature control systems. A thermoelectric cooling module is a low-impedance semiconductor device that presents a resistive load to its power source. Due to the nature of the Bismuth Telluride material used in thermoelectric modules, modules exhibit a positive resistance temperature coefficient of approximately 0.5 percent per degree C based on average module temperature. For many noncritical applications, a lightly filtered conventional battery charger may provide adequate power for a TE cooler provided that the AC ripple is not excessive. Simple temperature control may be obtained through the use of a standard thermostat or by means of a variable-output DC power supply used to adjust the input power level to the TE device. In applications where the thermal load is reasonably constant, a manually adjustable DC power supply often will provide temperature control on the order of +/- 1°C over a period of several hours or more. Where precise temperature control is required, a closed-loop (feedback) system generally is used whereby the input current level or duty cycle of the thermoelectric device is automatically controlled. With such a system, temperature control to +/- 0.1°C may be readily achieved and much tighter control is also possible.

4.4. **Thermistor connection using a voltage divider circuit [202117 depicted].**



NTC (negative temperature coefficient) thermistors are a common choice for temperature measurement. They are basically a variable resistor that decreases in resistance value as temperature increases and vice versa. They are relatively low cost and can be very accurate (</= 0.1°C) when properly selected and properly read. They cover a relatively large temperature range (e.g., typically -25°C to 125°C or even wider for quality thermistors). Lastly, when using one of the many readily available smaller components, they have a relatively small thermal response time (e.g., low thermal mass). An easily implemented method of reading them with a microcontroller is through use of a voltage divider circuit, where an excitation voltage is applied to one side of a reference resistor with the thermistor in series and the opposite side of the thermistor connected to ground. The voltage drop is read across the thermistor. This voltage drop can be used to obtain the resistance of the thermistor.



$V_{thermistor} = R_{thermistor} / (R_{thermistor} + R_{reference}) * V_{ex}$

Where V_{ex} is the power supply voltage (3.3 or 5 VDC).

Reading this into a microcontroller with a 10-bit ADC (analog to digital converter) results in:

ADC value = $R_{thermistor} / (R_{thermistor} + R_{reference}) * V_{ex} * 1024 / V_{aref}$

If V_{ex} and V_{aref} are the same value, they cancel out.

Simplifying the reduced equation results in the following.

$R_{thermistor} = R_{reference} / (1024 / ADC - 1)$

$R_{reference}$ is selected based on the temperature range you intend to operate within using the following equation. Note that lower spans result in lower error.

$$R_{ref} = \sqrt{R_{therm@lowerlimit} \times R_{therm@upperlimit}}$$

The thermistor resistance value is then used to obtain the temperature by one of three means. One means is to use a look-up table, with the values in the table taken from the thermistor manufacturer's chart of resistance vs. temperature. The second is the use of manufacturer supplied temperature vs. Resistance for the thermistor in the form of coefficients to use with the Steinhart-Hart equation of temperature vs resistance. (The Steinhart-Hart equation is a third order polynomial fit to the temperature vs. resistance characteristic of the thermistor.) A third method is to create a curve using readily available but abbreviated temperature resistance table data and deriving a fifth or sixth order polynomial equation using the manufacturers resistance vs. temperature data. The terms in the polynomial equation can be generated using the Excel "polynomial curve fit" function on a "log resistance vs. temperature" chart. An example of temperature measurement and output using this third method is shown in the sample code below.

```
// The following code is an example on how to obtain accurate temperature measurements
// between two temperatures

//How to display the temperature on a serial moitor when using a 10K ohm thermistor with
//the 202117 programmable controller. The values in this code are for illustration only
//and must be substituted with actaul values from user's setup.

void thermTE () {

  // Multiple number of samples give you more accurate readings by limiting the noise

  average = 0;
  for (n = 1; n < NUMSAMPLES; n++) {
    samples[n] = analogRead(tempPin);     // Reads ADC output on tempPin (A0)
    average += samples[n];
  }
  average /= NUMSAMPLES;
  rTherm = rRef / (1024 / average - 1);  // Calculates resistance for thermistor leg of half
  logrTherm = log10(rTherm);             // Obtain log of thermistor resistance for temperature

  // Use a higher order polynomial for accurate conversion from resistance to temperature

  // Amphenol Thermistor
  tempTherm = -.0416421 * pow(logrTherm , 5) + 1.08312 * pow(logrTherm , 4) - 12.0168 * pow(logrTherm , 3)
            + 75.0873 * pow(logrTherm , 2) - 300.331 * logrTherm + 559.369;

  // Sentech Thermistor
  tempTherm = .07191332 * pow(logrTherm , 6) - 1.791422 * pow(logrTherm , 5) + 18.67046 * pow(logrTherm , 4)
            - 105.4027 * pow(logrTherm , 3) + 350.9841 * pow(logrTherm , 2) - 729.505 * logrTherm + 833.2693;

  Serial.println(tempTherm, 1);         // Displays temperature to 1 decimal point on serial monitor
  delay(150);                           // Delay to obtain readings in .15 second intervals
}
```

The 10-bit A/D converter on the controller provides a fairly accurate temperature measurement. The actual temperature resolution and accuracy depends on the accuracy of the thermistor selected and the temperature range of interest used in the design of the voltage divider network.

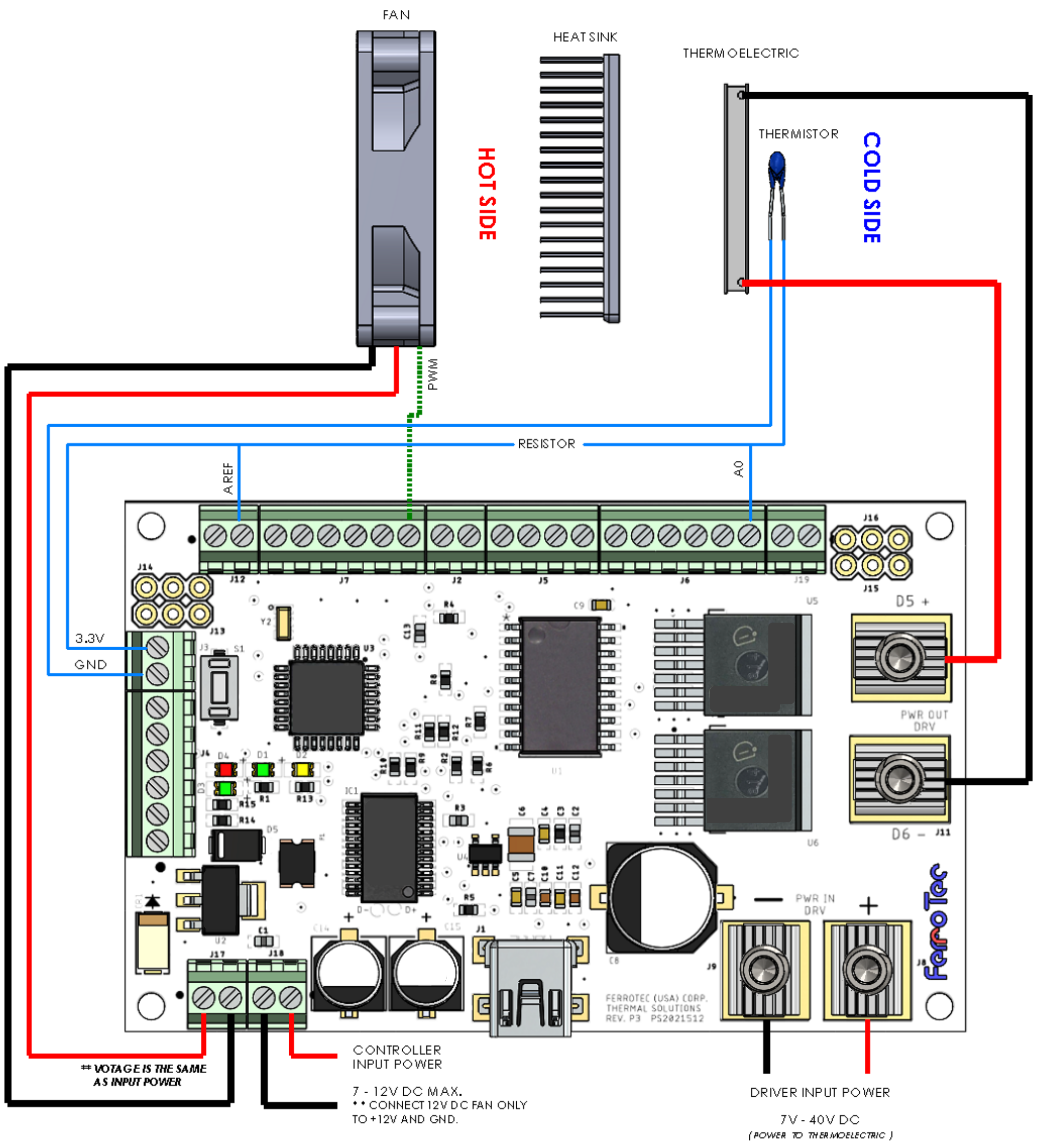
Another significant factor when using thermistors is to be aware of power dissipation constants for the selected thermistor and ensuring that the measurement is not affected by thermistor self-heating. If you are operating in air or other poor heat dissipation medium it is best to turn monitoring on and off as readings are required instead of constantly powering the thermistor.

When using thermistors for temperature measurement with this control board, it is recommended that the 3.3 VDC bus be connected to analog Reference (aRef). Doing so, isolates the thermistor from noise that is present on other 5 VDC connections within the control board and results in more stable readings. Using this method does require a line of code specifying “analogReference (EXTERNAL)”.

More accurate temperature measurements can be obtained by collecting 6 to 10 samples and averaging them.

4.5 TEC-fan-thermistor example [202117 model depicted]

Below illustrates an example of a typical thermoelectric assembly, using a thermistor for measuring the temperature of the main (cold) side of the TEC. It incorporates the voltage divider network shown in the section above.



```
// The following code is an example on how to obtain accurate temperature measurements
// between two temperatures

//How to display the temperature on a serial moitor when using a 10K ohm thermistor with
//the 202117 programmable controller. The values in this code are for illustration only
//and must be substituted with actaul values from user's setup.


#include <Math.h>                   // Enables extended math in polynominal solving
#define aref_voltage 3.349          // Setting analog reference voltage
#define NUMSAMPLES 20               // Setting desired sampling rate
int samples[NUMSAMPLES], n;

const int tempPin = A0;              // Voltage divider ADC input pin
const int rRef = 5610;               // Setting the refernce resistor value

int maxTemp = 50;                    // The maximum temperature before cooling
int minTemp = 10;                    // The minimum temperature before heating
float setPoint = 10;
unsigned long holdTime = 120000;     // The value is the amount of time to hold in miliseconds

// The coefficients below are for sample only.
// Each assemble must be tuned for it's characteristics.

float kp = -150; float ki = -0.46; float kd = -280.0;
float ITerm = -105;

float average;
float rTherm, logrTherm, tempTherm;
unsigned long currentTime, previousTime;
float elapsedTime;
float input, output;
float error, lastError;
float dInput, lastInput;

const int enablePin = 4;
const int coolPin = 5;
const int heatPin = 6;


void setup() {

  analogReference(EXTERNAL);            // Isolates noise on 5V bus from othr IOs that may be used by connecting 3.3V to aRef
  pinMode(enablePin, OUTPUT);
  pinMode(coolPin, OUTPUT);
  pinMode(heatPin, OUTPUT);
  pinMode(tempPin, INPUT);
  digitalWrite(enablePin, HIGH);        // Turns enablePin on

  Serial.begin(115200);                 // Select correct braud rate for the serial monitor
}

void loop() {

  Serial.println("1 = Temp Cycle");
  Serial.println("2 = Steady State");
  Serial.println ();
  Serial.println("Input number for function to run. ");      // Waits for user to select which program to run in serial monitor
  Serial.println ();

  while (Serial.available() == 0) {
  }

 int menuChoice = Serial.parseInt();

 switch (menuChoice) {

   case 1:

     thermTE ();

     for (int x = 0; x < 2; x = x + 1) {  // Use for loop to set the intial value of the variable,
                                          // The condition to exit the loop, and the action of the variable each time around the loop
       thermTE ();
       analogWrite(heatPin, 255);         // Turns heatPin on

       for (float c = (tempTherm); c < maxTemp; c = (tempTherm)) {
         thermTE ();
       }

       analogWrite(heatPin, 0);          // Turns heatPin off
       delay (50);                       // Delay to obtain readings in 0.05 second intervals

       thermTE ();

       analogWrite(coolPin, 255);        // Turns coolPin on

       for (float c = (tempTherm); c > minTemp; c = (tempTherm)) {    // Use for loop to set the intial value of the variable,
                                                                      // The condition to exit the loop, and the action of the variable each time around the loop
         thermTE ();
       }

       analogWrite(coolPin, 0);         // Turns coolPin off
       delay(50);                       // Delay to obtain readings in 0.05 second intervals
     }

     break;                             // Used to exit the for loop
// The following code is an example on how to obtain accurate temperature measurements
// between two temperatures

//How to display the temperature on a serial moitor when using a 10K ohm thermistor with
//the 202117 programmable controller. The values in this code are for illustration only
//and must be substituted with actaul values from user's setup.


#include <Math.h>                    // Enables extended math in polynominal solving
#define aref_voltage 3.349           // Setting analog reference voltage
#define NUMSAMPLES 20                // Setting desired sampling rate
int samples[NUMSAMPLES], n;

const int tempPin = A0;               // Voltage divider ADC input pin
const int rRef = 5610;                // Setting the refernce resistor value

int maxTemp = 50;                     // The maximum temperature before cooling
int minTemp = 10;                     // The minimum temperature before heating
float setPoint = 10;
unsigned long holdTime = 120000;      // The value is the amount of time to hold in miliseconds

// The coefficients below are for sample only.
// Each assemble must be tuned for it's characteristics.

float kp = -150; float ki = -0.46; float kd = -280.0;
float ITerm = -105;

float average;
float rTherm, logrTherm, tempTherm;
unsigned long currentTime, previousTime;
float elapsedTime;
float input, output;
float error, lastError;
float dInput, lastInput;

const int enablePin = 4;
const int coolPin = 5;
const int heatPin = 6;


void setup() {

  analogReference(EXTERNAL);           // Isolates noise on 5V bus from othr IOs that may be used by connecting 3.3V to aRef
  pinMode(enablePin, OUTPUT);
  pinMode(coolPin, OUTPUT);
  pinMode(heatPin, OUTPUT);
  pinMode(tempPin, INPUT);
  digitalWrite(enablePin, HIGH);       // Turns enablePin on

  Serial.begin(115200);                // Select correct braud rate for the serial monitor
}

void loop() {

  Serial.println("1 = Temp Cycle");
  Serial.println("2 = Steady State");
  Serial.println ();
  Serial.println("Input number for function to run. ");      // Waits for user to select which program to run in serial monitor
  Serial.println ();

  while (Serial.available() == 0) {
  }

  case 2:

    thermTE ();

    unsigned long startStage = millis();           // Initial start time
    unsigned long stageTime = startStage;

    while (stageTime - startStage <= holdTime) {   // Will loop continuously, and infinitely, until the expression becomes false
      thermTE ();
      input = tempTherm;
      currentTime = millis();                      // Get the current time
      elapsedTime = (currentTime - previousTime);  // Formula to obtain how much time has passed
      error = (setPoint - input);                  // Formula to obtain error value
      dInput = (input - lastInput);                // Formaula to obtain dInput value

      ITerm += ki * error;
      if (ITerm < 0)  {
        ITerm = 0;
      }
      if (ITerm > 255)  {
        ITerm = 255;
      }

      float output = kp * error + ITerm - kd * dInput;
      if (output < 0)  {
        output = 0;
      }
      if (output > 255)  {
        output = 255;
      }
      analogWrite(coolPin, output);         // Setting coolPin to be an output
      lastInput = input;
      previousTime = currentTime;
      stageTime = millis();
      delay(100);                           // Delay to obtain readings in 0.1 second intervals
    }

    break;                  // Used to exit the while loop
  }
}
```

5.0 FURTHER INFORMATION

Additional information regarding TEDs and their usage may be obtained on the Ferrotec website at:

https://thermal.ferrotec.com/technology/thermoelectric-reference-guide/

HISTORY

Revision:	Date:	Description of Changes:
B	8/10/23	Format and text revisions
C	12/18/23	Was 2021xx is 2021XXMAN , Removed footer “Initial Release”
D	4/4/23	Updated part number from 2021xx to FTCP-1200