Application Note: Milestone 2

Alex Marino & Cameron Bendzynski
Rowan University

December 2, 2018

1 Design Overview

The Temperature Sensing and Regulation program is designed to measure and adjust temperatures. It does so by first measuring the current temperature using a thermistor over analog-to-digital conversion. This can either be an ambient temperature, or the temperature of a specific device. In initial testing, room temperature was measured, but in most of the testing performed, a 5V regulator was measured. The second function of the system is to modify temperatures using a CPU fan. First, a user inputs a desired temperature over UART. Then, using pulse width modulation, the CPU fan is powered in order to cool the 5V regulator. Once the system detects that the regulator is at the desired temperature, it switches to maintaining it by lowering the speed of the fan.

1.1 Design Features

- Receive UART Rx input (9600 baud rate) of desired temperature
- Measure current voltage of thermistor using ADC
- Convert voltage reading to temperature value using thermistor equations
- Send current temperature over UART Tx each time target temperature is set
- Calculate distance from current temperature to target temperature
- Calculate distance from current temperature to previous temperature
- Modify PWM output to fan based on temperature differences
- Maintain desired temperature by maintaining proper PWM value

1.2 Featured Applications

- Temperature Measurement
- Ambient Temperature Control
- CPU Temperature Modulation

1.3 Design Resources

The entire project, including the initial assignment, are stored on the team GitHub repository for ease of access. All files can be found here.

1.4 Block Diagram

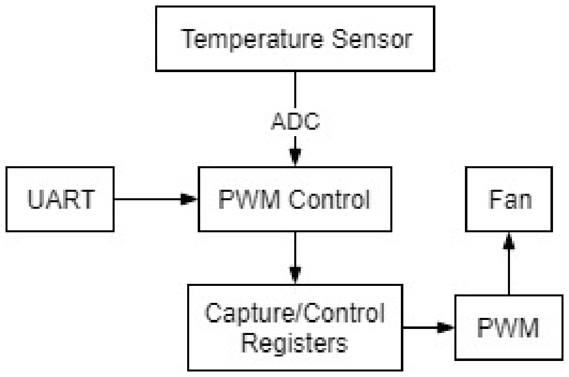

Figure 1: System Block Diagram

1.5 Breadboard Image

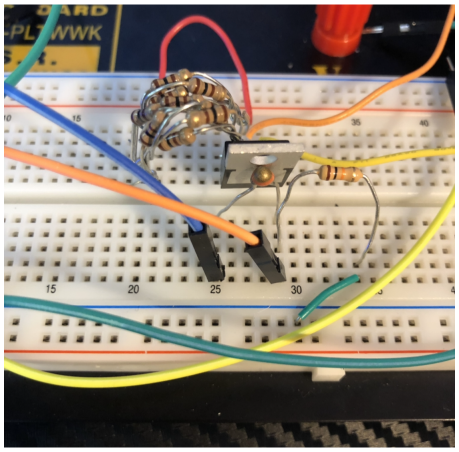

Figure 2: Thermistor Circuitry (wires left to right: 3.3V, ADC input, ground)

2 Key System Specifications

PARAMETER	SPECIFICATIONS	DETAILS
CCR Limit	0 - 65,535	Limited to 16 bits
Board Input Voltage	5V	
Thermistor Circuit Voltage	3.3V	
Fan and Regulator Voltage	12V	
UART Rx/Tx Buffer Size	0 - 255	Limited to a byte

Table 1: Key System Specifications

3 System Description

This system has four main components: measuring the voltage of the thermistor, converting this value to a temperature, receiving a desired temperature over UART, and controlling the fan. The thermistor is placed in a voltage divider circuit with a 10,000 Ohm resistor, the output of which is connected to the ADC pin of the development board. The ADC samples this value, and it gets converted to a resistance value based on the voltage divider circuit equation. Once a resistance value is found, one of three linear equations is used to convert this value to temperature. Which equation is used depends on the current temperature, and the process for choosing the equations will be explained in a later section. Once the resistance is converted into a temperature, the system compares the current temperature to the desired temperature which was received over UART, and to the previous temperature reading. Using these two values, the system modifies the PWM output such that the fan speed increases when the regulator is too hot, and decreases when it is cool enough.

3.1 Detailed Block Diagram

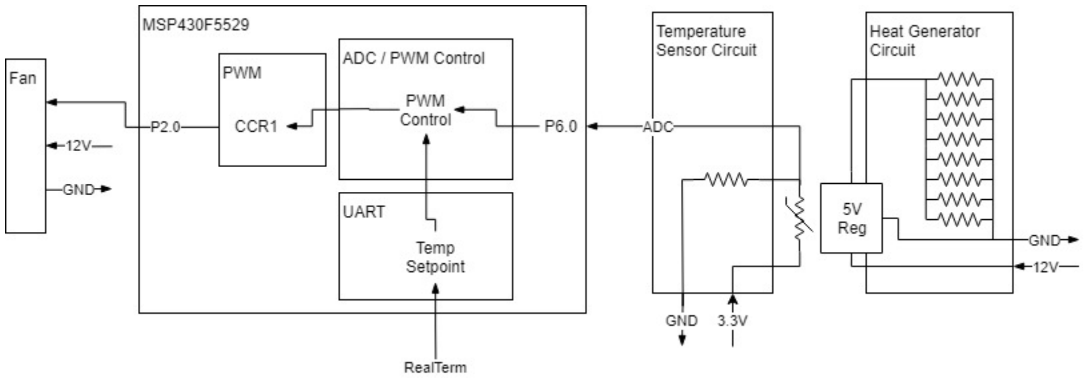

Figure 3: Detailed System Block Diagram

3.2 Highlighted Devices

The following devices were required to implement this system:

- MSP430F5529: Receives and sends UART transmissions, recieves ADC voltage reading from thermistor, controls fan speed with PWM output
- Breadboard with circuitry: houses thermistor voltage divider circuitry, 5V regulator circuitry, fan circuitry, and connections to MSP430F5529, including:
  - NTCLE100E3 thermistor with R25 value of 10 kOhm
  - CPU fan
  - UA78M33C 5V Regulator

3.3 Device 1: MSP430F5529

The board chosen for this project was the MSP430F5529LP. This board was responsible for carrying the software in order to measure the temperature and control the fan speed. This board was chosen for its ease of use and familiarity. The system required an ADC, hardware PWM, and multiple timers, all of which are easily accessible on the F5529. The other development boards were considered, but ultimately the F5529 was chosen for its wide range of available peripherals. Every time the system received a new voltage reading from the ADC it was converted to a resistance value, which was then converted to a voltage value. The sampling of the ADC was sourced from Timer 0, which used SMCLK divided by 4. The F5529 also received UART communication over USB so that a target temperature could be set. Pins P3.3 and P3.4 were used as UART Tx and Rx, but they were connected with a jumper so as to be accessible via USB. Finally, the F5529 was responsible for driving the PWM to regulate the fan speed based on the current, previous, and desired temperatures. Pin P2.0 was used as the PWM output. The PWM used reset/set mode on Timer 1, which also had SMCLK divided by 4 as its source.

3.4 Device 2: Thermistor and Regulator Circuitry

In addition to the F5529, the system also required two external circuits: a 3.3V and a 12V. Both circuits shared a common ground with the F5529. The 3.3V was powered by the 3.3V output of the MSP430F5529, but could also be powered externally. This circuit was used to power the voltage divider circuit. This circuit, as shown in Figure 4 in the bottom half, was used as a method to find the resistance of of the thermistor. If the thermistor was connected by itself, the voltage measured would always read 3.3V, because this would always be the potential difference to ground. However, by placing the thermistor in a voltage divider circuit with a known resistance of 10 kOhm, the output voltage (between the thermistor and the 10k resistor) of the divider could be measured by the ADC. This way, the input, output, and 10k resistance values are all known, and the unknown value of the thermistor's resistance can be solved for algebraically.The 12V circuit was used to power the CPU fan and the 5V regulator. The fan was connected directly to the power supply, with the ground connected to the power supply ground and common ground. The PWM pin of the F5529 was also connected to the control pin of the fan as shown by the CPU fan pin-out in Figure 4 in the top half. The 5V regulator was placed in parallel with ten 100 Ohm resistors, which made it heat up faster. The exact configuration of the regulator was not important for the implementation of the system, as it was simply used as a component whose heat could be increased by voltage and decreased by controlling the fan. In a more realistic setup, the regulator could be replaced with a CPU or the ambient temperature in a room.

4 System Design Theory

4.1 UART Communication

The UART communication routine in the software is what allows the device to receive a target temperature from the user. To enable UART over USB, UCA1 was used. To set the baud rate at 9600 with the 1 MHz clock, the UCA1 baud rate register 0 was set to a value of 6, and the baud rate register 1 was set to a value of 0. Rx interrupt also needed to be enabled such that, whenever a new target temperature was received, the system would include it in the new calculations, as well as return the current temperature to the user.

When an interrupt was detected on the UART Rx, the system would enter the UART interrupt service routine. Here, the UART interrupt vector was checked using a case statement. For the purposes of receiving the target temperature, the only interrupt the system was concerned with was a receive interrupt flag (RXIFG), which would set the UART interrupt vector UCA0IV to 2. Within case two, a variable for current temperature stored the received data. The incoming data was only considered if the Tx buffer was ready, meaning there was no pending data still on the pin. The Tx buffer was also loaded with an integer of the most recent temperature reading, so that the user knows what temperature the system is currently at each time they send a new target temperature.

4.2 Analog To Digital Conversion (ADC)

The MSP430F5529's Analog-to-Digital Converter (ADC) allows for an input of a voltage in reference to a source voltage to be converted to a digital signal. This is accomplished by turning the ratio of the input voltage to the source voltage into the same ratio in reference to the highest resolution of the ADC. For example, the MSP430F5529 has an ADC with a resolution of 12 bits, or 4095. The reference voltage is 3.3V and the input voltage is the output of a voltage divider, which output a voltage between 0V and 3.3V based on the resistance of the thermistor. The digital value calculated by the ADC is determined by the following equation:

$$\frac{V_{IN}}{3.3V} = \frac{ADC}{4095}$$

The value of V_{IN} is determined by the following voltage divider equation:

$$V_{IN} = 3.3V(\frac{10000}{Res + 10000})$$

The value of Res is determined by a series of linear equations produced from the exponential characteristic graph that the thermistor produces. That linearization is covered in the next section.

The ADC was configured to have 16 ADC12CLK cycles in the sampling period for ADC12MEM0, and was sourced from the sampling timer. ADC12MEM0 was the primary register of interest as this is where the voltage value from the thermistor circuit is stored. Pin P6.0 was used as the ADC input, with interrupt enabled so that it could take a new reading every time the timer generated an interrupt. The sampling timer used was SMCLK in UP mode, divided by 4, with a CCR0 value of 10,000.

4.3 Equation Linearization

Even though the chosen thermistor changes its resistance value as its temperature changes, it does not do so linearly. While logarithmic and other such operations are supported in C, they put unnecessary load on the microprocessor. To alleviate this, the temperature equation was divided into three temperature ranges, each with its own linear equation. The system then checks which range the current resistance value is in, and then processes it using the corresponding equation to find the resulting temperature. This provides reasonable enough accuracy without needing to implement the full, non-linear equation.

4.4 Pulse Width Modulation

The Pulse Width Modulation (PWM) system provides the microcontroller with the ability to operate the CPU fan at different speeds. The PWM is achieved through the use of a single timer (Timer A1) operated at 1MHz through the use of SMCLK. Only two Capture/Control Registers were necessary for this project, CCR0 and CCR1. CCR0 is used to turn off the fan after 1000 clock cycles, while CCR1 was able to be set to any number between 50 and 950 (for error-avoiding purposes).
Rather than polling, the PWM utilizes interrupts and a timer to control how long each LED is on for. The timer is set in Up Mode just for ease of use, especially with the built-in hardware PWM. The value of CCR1 decides how long the fan will be on for, shutting the fan off once the timer reaches the value set in the register. The fan will remain off until the timer reaches the value of CCR0 at 1000, where the fan turns on and the timer resets back to zero. The fan modulates so rapidly that the fan appears to be increasing and decreasing in speed rather than turning on and off intermittently. Additional logic is used in the system to ensure the CCR1 never reaches a value less than zero (which would result in the controller getting stuck) or greater than CCR0. The value of the CCR0 is never altered, but the other Capture/Control Register is manipulated through the temperature value and the UART-received setpoint temperature.
In order to reach a fan speed that allowed the thermistor to sit steadily at a certain temperature, a PWM control system had to be implemented. All this consisted of was a measurement of the difference of the temperature from the setpoint, as well as a difference in temperature since the last time the temperature was recorded. These measurements were used to increment or decrement the CCR1 value depending on how far off the temperature was from its goal, as well as how rapidly it was changing. The final equation was as follows:

$$TA1CCR1 = TA1CCR1 + 0.5 * (deltaTemp + timeTemp)$$

In the equation, $deltaTemp$ represents the difference between the current temperature and the selected temperature ($deltaTemp = temp - targetTemp$), while $timeTemp$ is the change in temperature over time ($timeTemp = 9 * (temp - lastTemp)$). Due to the fact the temperature never quite changed too rapidly, the value of $timeTemp$ was amplified by a factor of 9 to give it a more noticeable effect. The equation allows the fan speed to increase when the temperature is hotter than what is selected, due to the large value of $deltaTemp$. When current temperature begins to approach the setpoint at a rapid pace, the value of $timeTemp$ then starts to cancel out $deltaTemp$, slowing down the fan until the ideal fan speed is reached.
The control for the fan is done on Pin 2.0. The fan itself is power by 12V of electricity while accepting 3.3V for the PWM.

4.5 Design Requirement 1: UART Communication

To properly enable UART communication, several considerations must be met.The first is which Universal Serial Communication Interface (USCI) to use. USCI A1 allows for UART over USB, and so was used for this implementation. The next consideration is the baud rate. On the MSP430F5529, the baud rate is controlled by the baud rate registers. The values of these registers should be referenced from the device data sheet. The USCI also needs to be reset, and then enabled, before it can begin sending and receiving. This is controlled by modifying the UCSWRST bit of the USCI control register. Finally, a consideration for how UART communication is going to be used is recommended. For this design, only a UART receive would generate an interrupt, and therefore interrupt only needed to be enabled for receive.

4.6 Design Requirement 2: PWM/CCRs

The value of CCR1 was originally determined by an equation that did not factor in limits, which created many problems. The incrementation equation would bring the value of CCR1 higher than that of CCR0, never turning off the fan. While this is not innately a problem, it kept the PWM from changing since the value of CCR1 would continue to increase to ever-higher values while not having any effect. This was solved by adding in a buffer system, preventing the value of CCR1 from being decreased while below 50 and increased while above 950. The system would keep the register at the value as long as it wanted to move further into the buffer. However, once the value was to moved out of either buffer, it would be allowed to do so. This prevented many issues that previously occurred.

4.7 Design Requirement 3: Thermistor, Fan, and Regulator Circuit

A resistance of 10 kOhm was chosen for the thermistor voltage divider circuit because the thermistor used is 10 kOhm at room temperature. The parallel resistors for the 5V regulator were chosen as 100 Ohm so that they would be fairly low, and 8 were placed in parallel because adding more resistors made the regulator heat up more quickly. This circuit was ran on the 12V line because the CPU fan requires 12V.

5 Getting Started / How To Use The Device

This software does not interface with other devices, but can easily use any replacement fan, as they all practically work the same way. Most of the work is done in the code itself, but for testing purposes, a few things are important to take note of. When using RealTerm to send setpoint temperatures to the board, a number will be returned. This number is the current temperature of the thermistor. The current temperature will only be returned upon sending the board a setpoint. There is no consequence to sending the same setpoint multiple times to receive the temperature value again. In fact, this was done many times during testing.

The board should be configured according to Figure 3. Only two pins off the board are utilized (P2.0 and P6.0), but it should be noted that the ground of all the systems should be connected together. RealTerm is used through the use of the USB on the board, so no pin setup is necessary.
To use the system, simply use RealTerm to send the board a setpoint temperature (in decimal), and the system will automatically change its fan speed to reach the set temperature. The only user input necessary is through RealTerm, as long as the circuit is correctly assembled. This can be done easily on a breadboard.

5.1 Communicating with the Device

Communication with the device is performed via UART over USB. For UART over USB, USCI A1 is utilized in conjunction with a serial communication software such as RealTerm. RealTerm should be configured to display ”int8” values to properly view current temperature. To set the desired temperature, simply send an integer value using the ”send” function. The returned value will be what the system measures as the current temperature, and then the system will begin to modify the fan speed.

6 Test Setup

In order to verify functionality of the system, additional software needed to be utilized. First, the milestone project was flashed to the MSP430F5529. Then, the software RealTerm was installed on the test computer. This program allows communication via USB with the MSP430, and allows test packets to be sent to the board. This was only used to send setpoint temperatures to the board, and receive the current temperature. The thermistor / voltage divider and 5V regulator circuits were assembled, according to the circuit diagram in Figure 4, and the fan was connected to 12V and the PWM from the MSP430F5529. An oscilloscope was also used to measure the output voltage from the voltage divider, primarily to check the accuracy of our circuit.
Once power was connected, the 5V regulator began to heat up, and the fan was configured to keep the thermistor at a temperature of 30°C. Once a steady fan speed was reached, the setpoint was changed to 65°C, and allowed to heat up while the fan sat idle. At such high temperatures, even the fan spinning at a low speed had a major effect on the temperature.
Further testing was done at other temperatures, and the system was also tested for functionality at lower voltages than 12V simply for error-checking purposes.

7 Design Files

7.1 Schematics

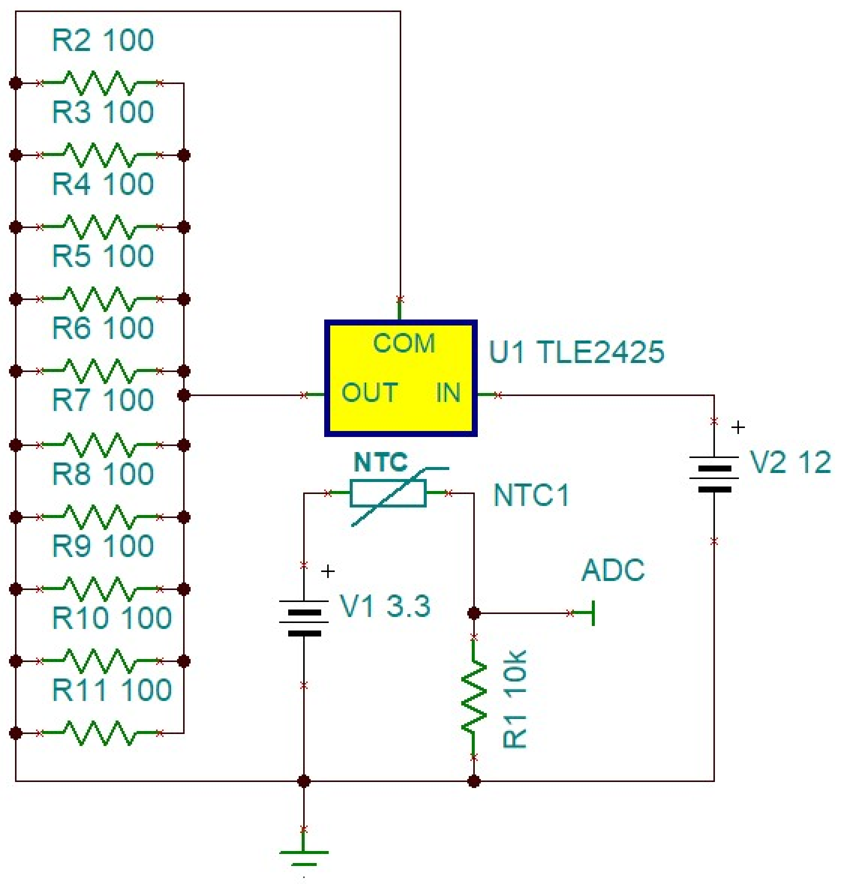

Figure 4: Thermistor & Heat Generator Setup

7.2 Bill of Materials

- MSP430F5529
- Micro USB Cable

- Breadboard
- NTCLE100E3 thermistor with R25 value of 10 kOhm
- Resistors
  - 1x 10 kOhm for voltage divider
  - 10x 100 Ohm in parallel with 5V regulator
- Jumpers and other miscellaneous wires
- CPU fan (4-wire)
- Laptop with RealTerm Software