Milestone 2 Report - 3000 Wolves

N Foy, M Wolf, J DiBenedetto
Rowan University

December 3, 2018

1 Abstract

During the completion of milestone 2, our main goal was to understand all of the internal attributes of the MSP430 itself. Manipulation of the MSP430 to control basic circuit components was also a desired outcome; as well as how UART plays a factor into their behavior. Milestone 2 would introduce us to how our microprocessor uses Analog to Digital (ADC) conversion to interact with real world signals. In basic terminology, milestone 2 dealt with converting a physical, real world signal to a digital code based signal that our microprocessor could understand and process. In order to do so, a temperature control system was built using a fan, a temperature sensor (thermistor), and a heat source (voltage regulator). By manipulating these three tools on a breadboard, along with using the micro controller's built in ADC port, UART was used successfully to control our closed-loop "temperature control" system.

1.1 Design Features

Milestone 2 had the following design features:

- Utilizes the MSP430F5529 microprocessor's ability to receive messages over UART
- Uses analog to digital (ADC) conversion to convert temperature readings from the thermistor to the proper format for UART to communicate with
- Uses Pulse-Width Modulation (PWM) to control the speed of the fan
- Heats and cools the heat source (voltage regulator) as directed by UART software.
- Connections from the F5529 to the fan, voltage regulator, and thermistor are made via breadboard circuitry wiring

1.2 Featured Applications

The functions of Milestone 2 are capable of being used in the following applications:

- Room temperature reading/sensing
- Temperature control
- ADC manipulation (temperature to UART)

1.3 Design Resources

This device functions via the main.c code on the team's Github. This code can be seen in the link below.

Closed Loop System.

1.4 Block Diagram

The following block diagram shows a general overview of the system and it's devices.

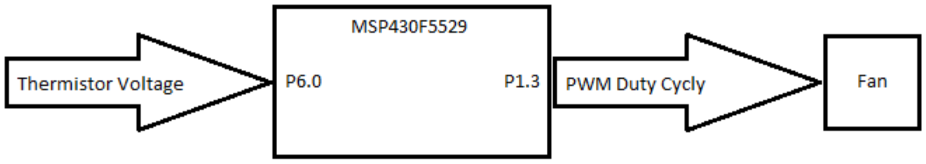


Figure 1: Closed System Block Diagram

1.5 Board Image

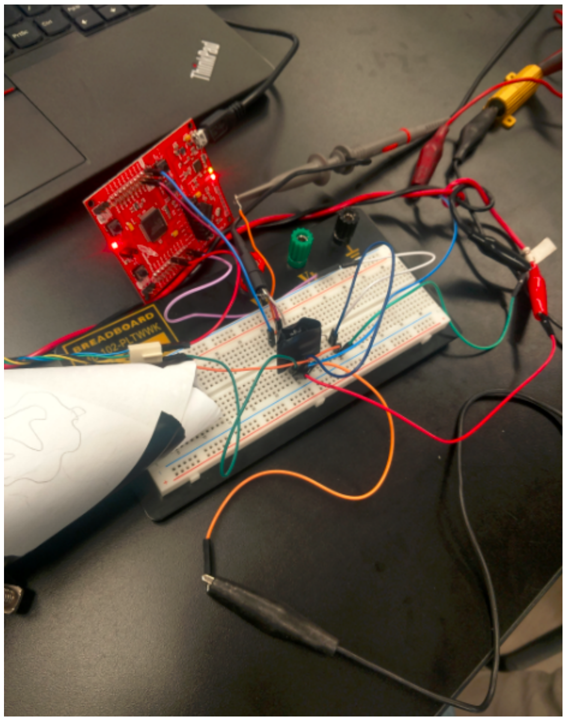


Figure 2: Full Closed-Loop Temperature Control System

Pictured above is the full system construction, with proper voltage sources hooked up as necessary. A funnel was used on the fan to provide more control of air flow. A 2 watt, 8 ohm power resistor was used to handle the current that would be drawn from our system. This allows the voltage regulator to heat up quickly without burning the entire circuit.

2 Key System Specifications

PARAMETER	SPECIFICATIONS	DETAILS
Microprocessor	MSP430F5529	Microprocessor needed for corresponding code parameter
Thermistor	NTCLE100E3	The given temperature sensor Specification
RealTerm	UART	Program for communication with microprocessor
L7809	Voltage Regulator	Heat Source used in circuit
2 watt 8 ohm, 1 watt 10k	Resistors	Resistors used in circuit

3 System Description

The manufactured closed-loop system that we built works as a temperature controller. We see many of these in the real world, such as heat/AC in your house or vehicle. The problem in this case can be seen as more of a ”request” by someone. Figuratively, they may ask you to design a heating/cooling system for them to utilize, and our response would be to build this closed-loop system as a prototype; assuming that the customer would want a much larger-scale creation.

To complete this task, we had to design a closed-loop system that is able to detect the temperature from a source of heat; and respond to that detection based on the instructions that are given to the system. A small fan, a thermistor sensor, and a voltage regulator were the three main components of the system. The voltage regulator acted as the heat source, and the thermistor acted as the temperature sensor; which hooked back up to the micro controller. The fan was used control the temperature of the voltage regulator based on the thermistor reading being received by the micro controller. UART is used to input a temperature value that the system should maintain. If the current temperature value being read by the thermistor exceeds the temperature input through UART, the fan will activate to bring the temperature back down to the input value. The fan then ”maintains” that temperature value input into UART by turning on and off periodically (or by varying its speed). ADC and PWM techniques are used to control the fan speed and to receive and convert temperature readings to a configuration that UART is able to interface with.

More information about the separate components of this system can be found in the upcoming sections.

3.1 Detailed Block Diagram

Elaborating from the block diagram earlier in the app note, the following figure contains a more detailed block diagram of the system’s various components. The multi-nodal blocks consist of the microprocessor, fan, and voltage regulator. A 12 volt power source supplies voltage to both the regulator and fan. A 3.3 V pin off the microprocessor powered the thermistor which in the figure looks similar to an additional resistor. The thermistor is in series with a 10k resistor and in between the junction of the two is the input of the microprocessor. This set-up creates a voltage divider and this voltage level is the input to the microprocessor’s ADC. This input and the program running on the processor will determine whether or not to turn the fan on via PWM.

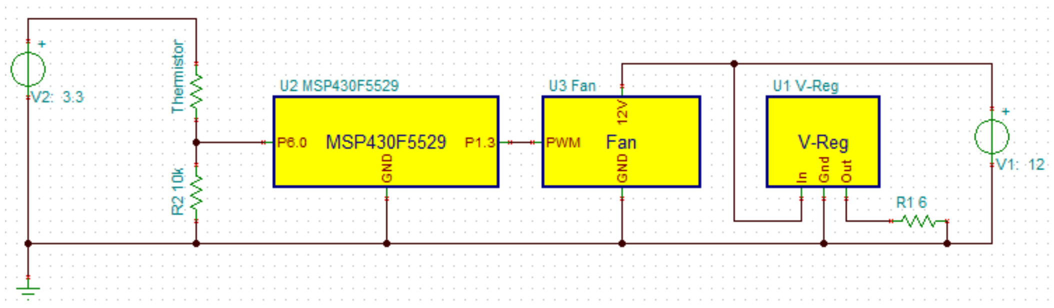


Figure 3: Detailed Closed System Block Diagram

3.2 Highlighted Devices

The following list encompasses all of the major devices used within this system.

- MSP430F5529
- NTC Thermistor
- 2 Watt power resistor (8 ohms)
- Voltage Regulator
- Cooling Fan
- Breadboard along with jumper cables/wires

3.3 Device 1: MSP430F5529

As was the case in Milestone 1, the microprocessor used in this project is the brain of our system. The features on the F5529 used are the built in ADC channel and UART peripheral, timers, interrupts, and input/output ports. The interrupts are enabled so the coded program can perform an action only when a timer reaches a set value. This interrupt routine would act as the Pulse-Width Modulation (PWM) of the fan used. Thanks to previous work with PWM functionality in Milestone 1, implementing the PWM for the fan was straightforward. The ADC on-board channel on the F5529 allows for conversion of the analog/physical signal (temperature) to a digital signal (voltage) so that UART could properly understand and interface with the temperature being sensed.

3.4 Device 2: Thermistor and Voltage Regulator

The second major device enabling this closed loop to function is the combination of the thermistor and voltage regulator. A voltage regulator is a device which maintains a constant voltage output regardless of the input. In this system, the input was 12 V and the output was 5 V. This drop in voltage created a large source of heat from the regulator. Assembling the circuit on the breadboard so that the thermistor makes direct contact with the regulator allows the system to read the temperature. The thermistor is assembled in a voltage divider setup so that a voltage fed back to the microcontroller can be interpreted as an accurate temperature.

3.5 Device 3: PWM Fan

The fan that was used was a small, basic fan with 4 input connection wires. These wires were black, yellow, green, and blue. As mentioned earlier, the speed of the fan needed to be controlled for the system to be able to operate correctly. The blue wire would be connected to the GPIO port (P1.3), where a PWM is programmed to control the speed of the fan. The fan also needed to be powered in some way. The yellow wire would act as this input, which was hooked up to a 12V power supply. The black wire would be connected to the ground terminal in our circuit. Through making all of these connections, the fan was capable of cooling the heat source down in our system, as well as keeping it at a steady temperature if commanded to do so (Through UART).

4 System Design Theory

In order for the system to properly work, ADC conversion has to occur. Thanks to the 12-bit ADC converter built into the F5529 board, ADC conversion was made possible. The multiple interrupts in the ADC converter can be manipulated to desired functionality. The analog signal that we would be dealing with came from a thermistor, an analog device that reads temperature. The thermistor reads the temperature as a voltage, which is then sent to the built in ADC on the F5529. This could have been done with a PTAT as well. However, by the time that the PTATs were handed out, we had already calculated the conversion equations needed for our code and UART. These 5 equations are listed below, with each equation varying based on the temperature range being read by the thermistor.

- 15℃-30℃
  $Temp = \frac{-(165000-(171225*ADCVoltage))}{2729*ADCVoltage}$
- 30℃-45℃
  $Temp = \frac{-(330000-(262460*ADCVoltage))}{2729*ADCVoltage}$
- 45℃-60℃
  $Temp = \frac{-(330000-(169120*ADCVoltage))}{1383*ADCVoltage}$
- 60℃-75℃
  $Temp = \frac{-(330000-(169120*ADCVoltage))}{737*ADCVoltage}$

- 75℃-90℃
  $Temp = \frac{-(330000-(145640*ADCVoltage))}{411*ADCVoltage}$

All of these equations are needed because of the non-linear relationship between temperature and voltage from the thermistor. These equations proved to be very useful when testing out our closed loop system. On the oscilloscope, we were able to determine where our system should stabilize at (The equations provided us with the voltage values that our signal should be in between.

5 Getting Started/How to use the device

In order to set up the system, you need to make the correct connections to the fan. 4 wires came out from the fan: black, yellow, green, and blue. The blue wire was the PWM which controlled the rate at which the fan would blow out air to cool-down the regulator. The yellow wire was the power supply; we ran 12V to it, the required amount to power the fan. The black wire connects to the common ground. Lastly, the green wire was left open and not connected to the circuit. The 12V power supply is ran to the voltage regulator as well, which connects to a 8Ω power resistor rated for 2W in order for it to conduct enough to heat to reach approximately 65C. Lastly, the MSP430F5529 was connected to the breadboard using the designated ports for us to communicate with the thermistor through UART. Using UART, we were able to reach any desired target temperature by typing a number into RealTerm to send to the board. Once received, the regulator then heat up to the desire temperature within about a minute. After the temperature was reached the fan would then constantly switch on and off/vary its speed to maintain temperature.

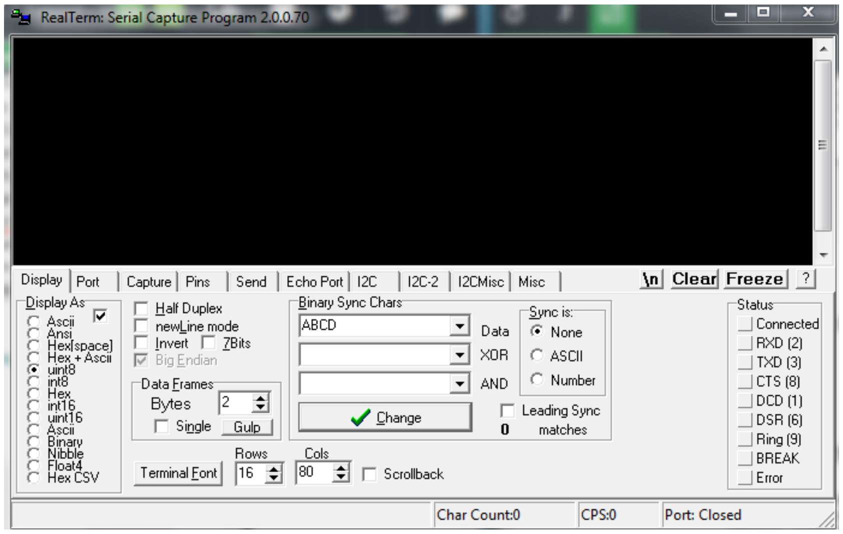
6 Getting Started Software/Firmware

The software needed to use this milestone device is Code Composer Studio (CCS) and the program RealTerm. CCS is used to generate the code/create the functions that are necessary for the system to operate. Once the code is debugged without errors, it can be successfully programmed into the board.

6.1 Communicating with the Device

Serial communication is the main communication between the computer and the F5529 micro controller. Realterm was the program used here, which utilizes the UART peripheral built into the F5529 board to properly communicate. However, in order for Realterm and UART to be properly used, the settings must be completely accurate and must reflect the specific settings of the processor that is being used. The baud rate and COM port are two important settings to mention. The baud rate for this device should be set to 9600, and the COM port must match up with the port of the processor. By going to the ”Device Manager” on your computer, you will be able to find the COM port by looking under ”Ports” of the USB hooked up to your computer.

Listed next to the processor USB port will be the COM port number. To change this in RealTerm, go to the Port tab, change the baud rate to 9600 and the COM port to the number found in the device manager, and hit ”Change”.



Display As: Ascii, Ansi, Hex[space], Hex + Ascii, [✓ uint8], int8, Hex, int16, uint16, Ascii, Binary, Nibble, Float4, Hex CSV

Half Duplex ☐
newLine mode ☐
Invert ☐ 7Bits ☐
Big Endian ✓
Data Frames: Bytes 2, Single ☐
Binary Sync Chars: Data ABCD, XOR, AND
Sync is: [✓ None] ASCII Number
Leading Sync ☐ 0 matches
Rows 16 Cols 80 Scrollback ☐
Status: Connected, RXD (2), TXD (3), CTS (8), DCD (1), DSR (6), Ring (9), BREAK, Error

Char Count:0 CPS:0 Port: Closed

Figure 4: RealTerm program used for UART communication

Once the COM port is properly configured, the communication settings must be changed. For starters, we want to be reading the temperature from the thermistor as two-digit numbers. In order to do this, the messages must be set to int8. Under the Display tab, in the section ”Display as”, select int8. Go over to the port tab and hit ”Change” to complete the configuration of RealTerm.

7 Test Setup

First, the correct RealTerm settings are configured (as mentioned above) in order to communicate with the F5529. Once done, go to the send tab, where you can type in a desired temperature for the regulator to heat up to. Make sure the proper connections are made between the regulator, the fan, and the F5529 board before sending the numbers. Once you send the numbers, a signal should be generated. To view the signal generated from sending the temperature number, hook up the system to an oscilloscope.

8 Test Data/Conclusion

During our test, our system showed the capability of being able to heat up to 65C, and cool to around 30 C. These processes took about a minute and a half to fully complete. Our system was also successful in holding a temperature/keeping the temperature at a steady value. When viewing our signal on the oscilloscope, we saw that it was right in the middle of the constraints determined by the equations used. There was little to no noise observed on the oscilloscope.