Milestone 2: Closed Loop Control System

Skylar Adams
Rowan University

December 3, 2018

1 Design Overview

For this milestone, students were required to design code for the MSP430F5529 to create a closed loop control system, to keep a voltage regulator at a set temperature. The code must be able to convert an analog voltage to a temperature, receive a temperature value to hold the regulator at, and be able to hold the regulator at the set temperature within plus or minus 3 degrees Celsius. A thermistor was used as a temperature sensor and UART was used for communication to and from the board. A low side switch using an N-channel MOSFET was used so a PWM signal could control the speed of the fan which cooled the voltage regulator.

1.1 Design Features

The key design features include:

- MSP430F5529 used for PWM, UART, and as an ADC
- IRLB8743 power N-channel MOSFET to control fan speed via PWM
- NTCLE100E3 NTC thermistor for cheap and accurate temperature sensing

1.2 Featured Applications

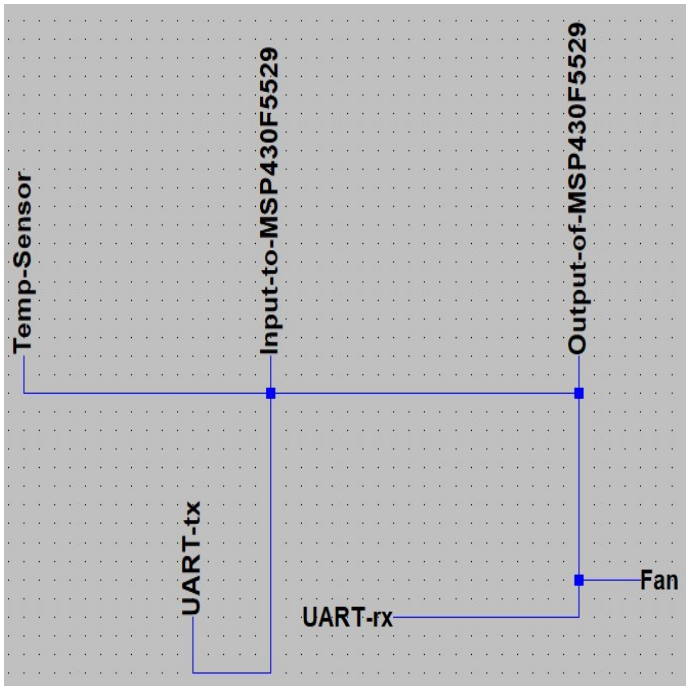
- Temperature control of a system
- Temperature monitoring of a system
- Conversion of analog signals to digital signals

1.3 Design Resources

The github link for the design folders and code can be found here:
https://github.com/RU09342-F18/introtoembedded-f18-milestone2-team-2

1.4 Simple Block Diagram

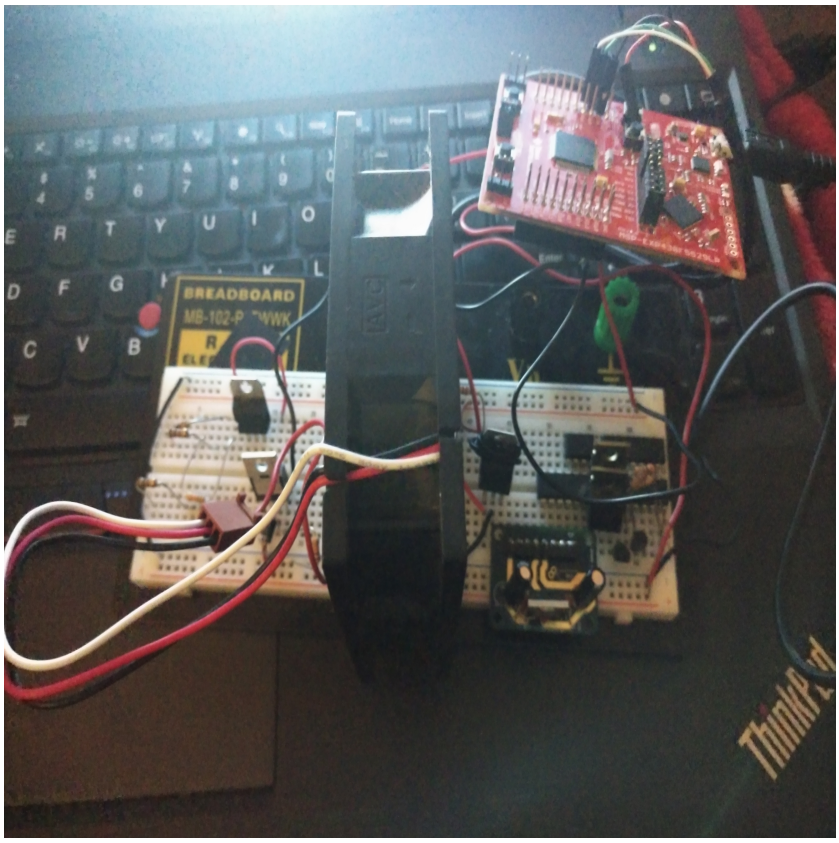
Figure 1: Simple Block Diagram



The input and output of the MSP430 being connected is due to there being a feedback loop written into the code to utilize proportional control of the fan speed.

1.5 Board Image

Figure 2: Image of Board



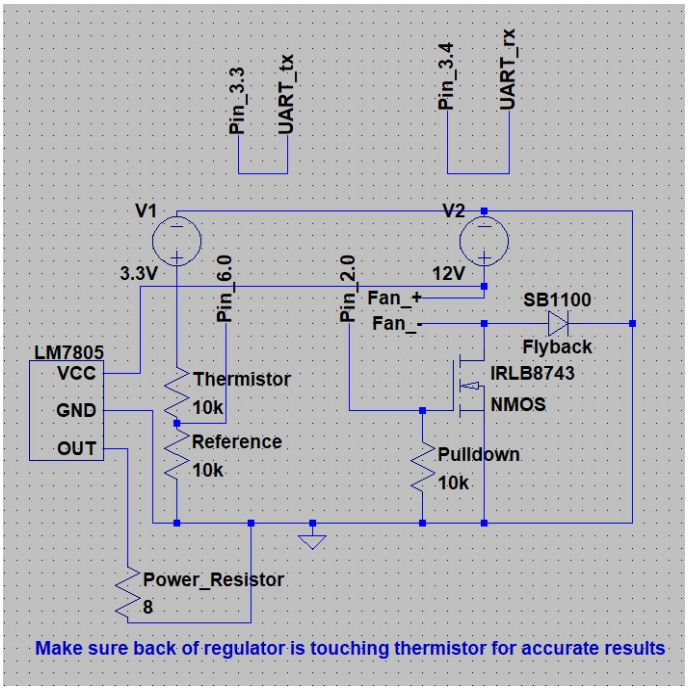
2 System Description

The problem to be solved is to create a closed loop control system that can steadily hold the temperature of a 5 volt voltage regulator under load. The temperature to hold the regulator at is sent to the board via UART, and depending on the difference between the set temperature and the current temperature of the regulator, the PWM of the fan is increased or decreased accordingly. A thermistor was used as the top resistor of a voltage divider to measure the temperature of the regulator. A thermistor was used instead of a PTAT due to a thermistor being able to perform the same job, and be just as accurate, but at a much lower hardware price. In the code, due to the thermistor being a non-linear device, its behavior had to be linearized to ensure the code ran at a proper speed for accurate measurements, so the code was written

to bound the temperature measurements between 20 and 100 degrees Celsius. The output of the voltage divider is analog, and for it to be used in the control system, it had to be converted to digital, which was accomplished with the onboard 12 bit ADC on the MSP430F5529. The MSP430F5529 also handled UART communication, and provided a PWM signal to the MOSFET which drove the fan. To keep the oscillations of temperature within the accepted plus or minus three degrees, a proportional control algorithm was implemented. In the code. The algorithm found the error between the set temperature, and the current temperature of the regulator by subtracting the voltage of the regulator from the set temperature. This error is multiplied by the proportional gain, in this case, 60, and that output is used to set the duty cycle of the PWM signal, which held the temperature within the allowed range. The MOSFET was used as a switch to a PWM signal could be used to control the speed of the fan. A power MOSFET was required due to the current drawn by the fan being too high for a small signal MOSFET to handle.

2.1 Detailed Block Diagram

Figure 3: Detailed Block Diagram



2.2 Highlighted Devices

The Devices used for this project were:

- MSP430F5529 for PWM, UART, and ADC
- Fan switching Circuit
- Temperature Sensing Circuit

2.3 MSP430F5529

The MSP430F5529 was used for generating a PWM signal for the fan, converting the analog output of the temperature sensing circuit to a digital signal, usable by the MSP430F5529, and for UART communication. The Timer A peripheral was used to create a hardware based PWM signal which controlled the fan, and it was used to continuously allow for ADC conversions. the ADC12 peripheral was used to convert the analog output of the temperature circuit into usable data for the MSP430. Timer A was initialized using SMCLK for both generating a PWM signal and for the continual conversion in ADC12. The ADC input pin was pin 6.0 of the board, and the PWM output was set to pin 2.0. UART was used to send a temperature to hold, and to display the current temperature of the voltage regulator. For communication, pints 3.3 and 3.4 were set to Tx and Rx respectively.

2.4 Fan Switching Circuit

The fan switching circuit was used to allow speed control of the fan with a PWM signal and was created with one N-channel power MOSFET in a low side switch configuration. A flyback diode was connected in parallel with the fan to prevent high voltage spikes from slowly damaging the MOSFET when the fan switched off suddenly. A pulldown resistor of 10k ohms was connected to ground on the gate of the MOSFET to ensure that when no signal is sent to the MOSFET, it will remain off. The fan was connected to a separate 12V source to ensure it would run at full speed, and not draw too much current from the board.

2.5 Temperature Sensing Circuit

The temperature sensing circuit was used to sense the temperature of the voltage regulator. The circuit was a simple voltage divider, using the NTCLE100E3 10k ohm thermistor as the top resistor, and a 10k ohm resistor as the bottom resistor. A 10k ohm resistor was used as the bottom resistor to make the voltage to temperature calculation easier to implement in code, and because the datasheet recommended using a reference resistor of the same value of the thermistor. The output of the voltage regulator was connected to pin 6.0 of the MSP430F5529, which was the input of the ADC. The top leg of the voltage divider was connected to the 3.3V from the board, and the bottom leg was connected to ground.

3 SYSTEM DESIGN THEORY

Within this project, the micro-controller was used to create a closed loop control system, more specifically a black box closed loop control system. A black box closed loop system means that the internals of the system are unable to be controlled by the user, and the only controllable aspect is the input to the system. Closed loop means the output of the system loops back to the input of the system in what's called a feedback loop, which is essential for the proportional control algorithm of the system. Essentially, the system works on a temperature input from the user, and outputs a PWM which holds the temperature within the required bounds. The system was also designed to use full PID (proportional integral differential) control, which would allow for even tighter temperature oscillations, but it was found through testing that only proportional control was needed to keep the system within the plus or minus 3 degrees Celsius required, and in a real life situation, tuning of the system would result in the system taking more time to complete. In this system, a MOSFET was required to control the speed of the fan, due to it not having the onboard circuitry for PWM control. Without the MOSFET, the fan would only be able to be either on or off, which will also work at keeping the regulator cool, but with much larger oscillations than required.

3.1 Design Requirements

The requirements were to maintain the temperature of a voltage regulator within plus or minus 3 degrees Celsius, with the range of temperature being the range of acceptable oscillation. First to actually measure the temperature of the voltage regulator, the output of a voltage divider with a thermistor was input into the onboard 12 bit ADC of the MSP430F5529, ADC12. The ADC converted the analog voltage into a digital signal, usable by the MSP430F5529, and that voltage was converted to temperature through arithmetic on the board itself. The ADC was set to convert continuously by enabling and disabling sampling and conversion using timer interrupts via the Timer A peripheral. Timer A was set to up mode, and an interrupt was generated whenever the clock counted to 12000 by setting the CCR0 register to 12000. To provide a temperature hold, and to monitor the current temperature, UART was used for communication between the board and a laptop. To generate a PWM signal to control the fan, the Timer A peripheral was set up to create a hardware PWM signal, using OUTMOD, in this case, OUTMOD 7, or reset/set mode. Reset/set will toggle the output register on the first interrupt, CCR0, and set the value on the second, CCR1, which creates the PWM signal. In this case, CCR1 was created using a simple proportional control algorithm. The algorithm calculated the error between the set signal and current temperature, and increased or decreased the duty cycle of the PWM waveform, which directly controlled the speed of the fan. The lower the set temperature, the higher the PWM and vice versa. This is due to the fact that if one wants to cool down the regulator, the PWM must be high to have the fan run at full power, and to heat up the regulator, the fan must either be off or running at low power to allow for the regulator to heat up. For the fan to even run off of a PWM signal, the power MOSFET was required to be a switch, due to the fan only being a three pin fan, which is power, ground, and

a tachometer. A four pin fan would have an input for PWM and the circuitry for proper PWM control. The MOSFET itself needed a flyback diode to prevent short, sudden, high voltage spikes from slowly damaging the component, and a pull-down resistor to ensure that when no signal is sent, the fan will remain off.

4 Getting Started/How to use the device

Subsection 4.1 will go over how to connect hardware components, and subsection 4.2 will go over how to set up the required software.

4.1 Hardware Setup

First, view the detailed block diagram in order to see how to properly set up the circuitry on a breadboard. A power supply that can provide at least 12V at 2A is required for proper fan and voltage regulator operation. The resistor values, component models, and any extra voltages are all listed in the detailed block diagram. Be sure when setting up to avoid accidentally plugging any pins into the 12V power rail, at risk of one damaging or destroying the board. Plug the output of the voltage divider into pin 6.0 of the MSP430, and connect pin 2.0 to the gate of the MOSFET. Plug both the MSP430 power cable and the UART cable into USB ports, connect the white wire of the UART cable to pin 3.3, connect the green cable to pin 3.4, and connect the black cable to any ground pin on the board. DO NOT connect the red cable, for it is a 5V cable and can damage or destroy the board.

4.2 Software Setup

First, be sure both code composer studio and Realterm are installed on the computer. In code composer studio, once the board is connected via the USB, debug the project, proceed on the low power mode message, and once complete, click the resume button on the top bar to start debugging. In Realterm, under the ports tab, be sure the baud rate is set to 9600, and make sure the COM port of the UART cable is selected. To find the COM port, open up device manager (if on windows) and expand the ports tab. The COM port of the UART cable should be labelled. Once set up, navigate to the display tab and check off ASCII to see the current temperature of the voltage regulator. Finally, navigate to the send tab, and prepare a temperature in the provided text box.

5 Getting Started Software/Firmware

This section will go into more detail on how to setup the software, mainly Realterm setup.

5.1 Communicating with the Device

To communicate with the device, the program Realterm, and a UART cable are required. To start, start up Realterm as shown in the Software Set up subsection. Once navigated to the text box, one can start sending temperatures. Send all temperatures using either hexadecimal or binary values, with a minimum of 20, and a maximum of 100. Plese only send one temperature value at a time, as that is all the code can handle. The hexadecimal input in Realterm is not case sensitive. Once typed in, click the send button, which will send the code to the board, and set the referance temperature to keep the regulator at. Once inputted, watch the current temperature displayed over uart change to the set value.

6 Test Setup

To setup the system for test, follow all guidelines in the hardware and software getting started sections. Once Realterm is open and ready to send data, and the voltage regulator is at room temperature, turn on the 12V power supply and send the following temperatures:

- Room temperature to 60 and hold until steady oscillations
- 60 to 40 and hold until steady oscillations
- 40 to 50 and hold until steady oscillations
- 50 and reduce the voltage of the power supply to see how the system reacts

6.1 Test Data

The following subsection includes pictures of the oscilloscope connected to the output of the temperature circuit, which shows the current temperature of the voltage regulator.

Figure 4: Oscillation at 60 Degrees

Figure 5: Oscillation at 40 Degrees

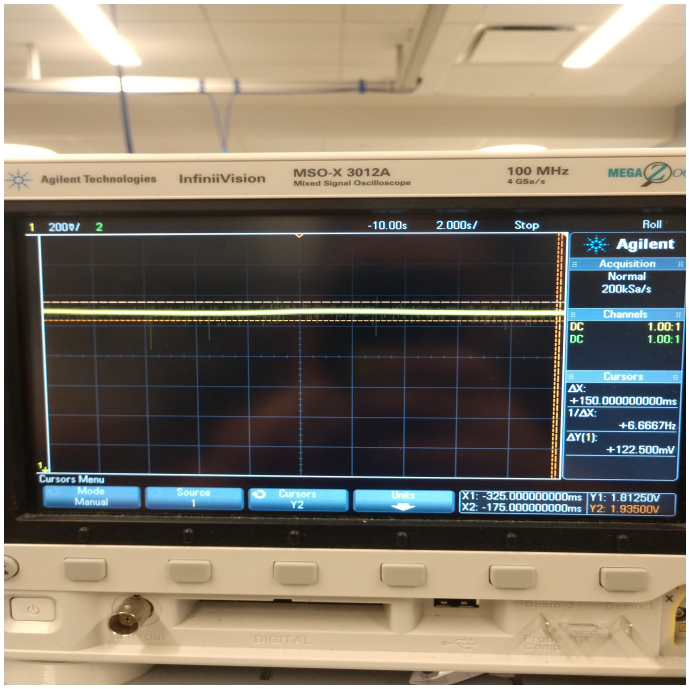
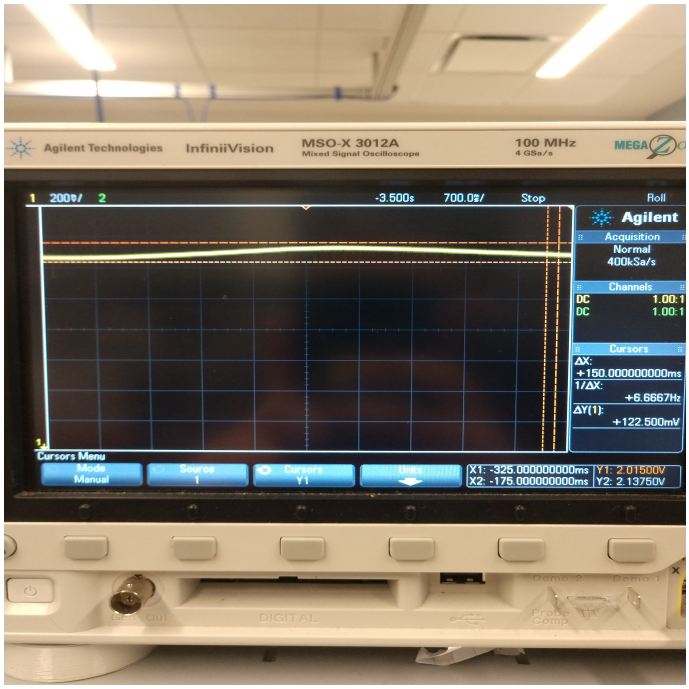


Figure 6: Oscillation at 50 Degrees



As shown, all of the temperatures remained in the range of plus or minus 3 degrees Celsius from the set temperature.

7 Design Files

Design Files can be found in the github repository located at:
https://github.com/RU09342-F18/introtoembedded-f18-milestone2-team-2

7.1 Schematics

The Schematic of the temperature sensing circuit, fan driver circuit, and voltage regulator circuit can be found in the detailed block diagram section of this Application Note.

7.2 Bill of Materials

Used materials include 1 salvaged CPU heatsink fan, the MSP430F5529, 1 IRLB8743 N-channel power MOSFET, 2 10k ohm 5 percent tolerance resistors, an 8 ohm power resistor, 1 NTCLE100E3 10k ohm thermistor, 1 LM7805 5 volt 1 amp voltage regulator, 1 SB1100 power diode, a UART cable, and a breadboard.