Milestone 2: Temperature Control Loop

R. Boylan, A. Steel, R. Michaelchuck
Rowan University

November 27, 2019

1 Design Overview

The objective for this Milestone is to be able to control a fan in order to keep a voltage regulator at a specified temperature. The specified temperature is supplied over UART, and the chosen microprocessor will speed up or slow down a fan in order to reach that temperature. The microprocessor for this task is the TI MSP430G2553. The system is evaluated on how well it can maintain the specified temperature. This system portrays the basic principles of how temperature control for a computer processor works, or how a HVAC system can maintain a desired temperature in a room.

1.1 Design Features

The following design features represent the top level functions and capabilities of this project.

Design Features:

- Variable PWM: used to control fan speed
- Analog to Digital Conversion: used to measure temperature
- Communication via UART: used to get a desired temperature value

1.2 Featured Applications

Listed below are the featured applications of this project.

- Temperature control of heat dissipating circuitry
- Controlling Fan Speed

1.3 Design Resources

The following links are to the Github repository where this lab is stored as well as the documentation for the microprossor, fan, voltage regulator, and the low voltage temperature sensor used in this lab.

- Github Repository
  https://github.com/Intro-To-Embedded-Systems-RU09342/milestone-2-milestone2-teamwsnh.
- MSP430 Datasheet
  http://www.ti.com/lit/ds/symlink/msp430g2553.pdf.
- MSP430G2553 User's Guide
  http://www.ti.com/lit/ug/slau144j/slau144j.pdf.
- AUB0912VH Fan Datasheet
  https://www.delta-fan.com/Download/Spec/AUB0912VH-CX09.pdf.
- TMP36 Temperature Sensor
  https://www.analog.com/media/en/technical-documentation/data-sheets/TMP3-53637.pdf.
- L7805CV 5V Voltage Regulator
  https://cdn-shop.adafruit.com/product-files/2164/L7805CV.pdf.

1.4 Block Diagram

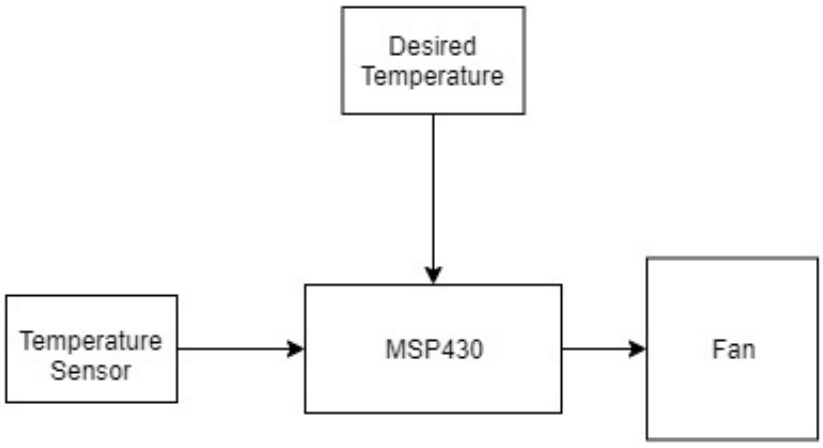

Figure 1: System Block Diagram

1.5 Board Image

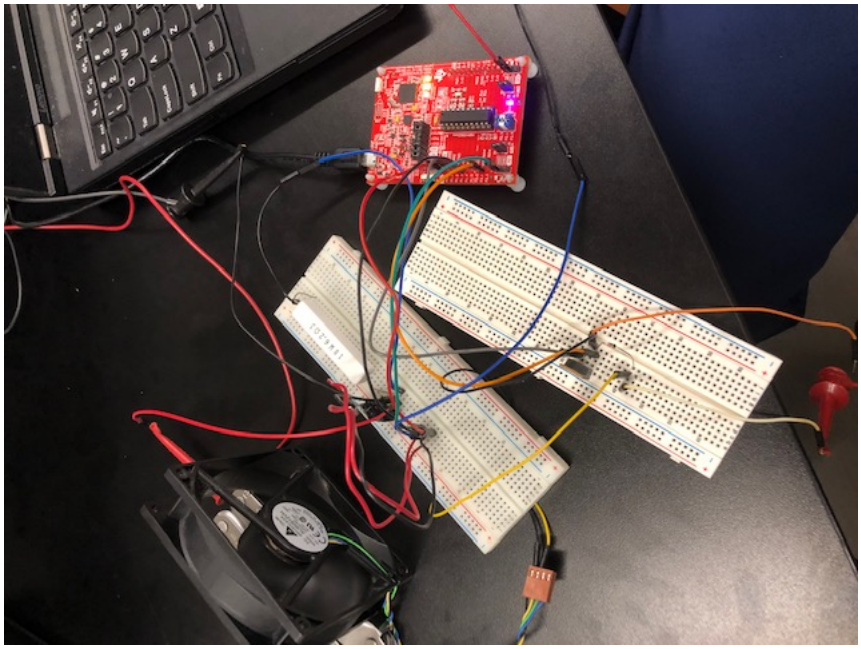

Figure 2: Breadboard Image

2 Key System Specifications

Table 1: Table of System Specifications

Parameter	Specification	Details
SMCLK	1 MHz	PWM Timer Speed
Baud Rate	9600	UART Communication Speed
TA1CCR0	1025	PWM Period
TA1CCR1	P2.1	Fan and Red Status LED PWM
TA1CCR2	P2.5	Blue Status LED PWM
ADC10MEM	P1.3	ADC Input pin

3 System Description

The objective is to use a microprocessor to control the temperature of a heat source. The temperature of the source is read using a temperature sensor and the desired temperature is provided over UART. To control the temperature, a fan is PWM'd to

speed up or slow down, which will increase or decrease air flow across the heat source. At some fan speed, the heat being produced by the source will equal the cooling potential of the air flow across it, and will remain at a steady state, ideally at the desired temperature that was provided over UART. The way the microprocessor changes the fan speed is by comparing the desired temperature with the current temperature that is read by the sensor. If the desired temperature is higher than the current temperature, then the fan should slow down to allow the source to heat up. Otherwise, if the desired temperature is lower than the currently sensed temperature, then the fan should spin faster in order to force more air across the source and cool the source down.

3.1 Detailed Block Diagram

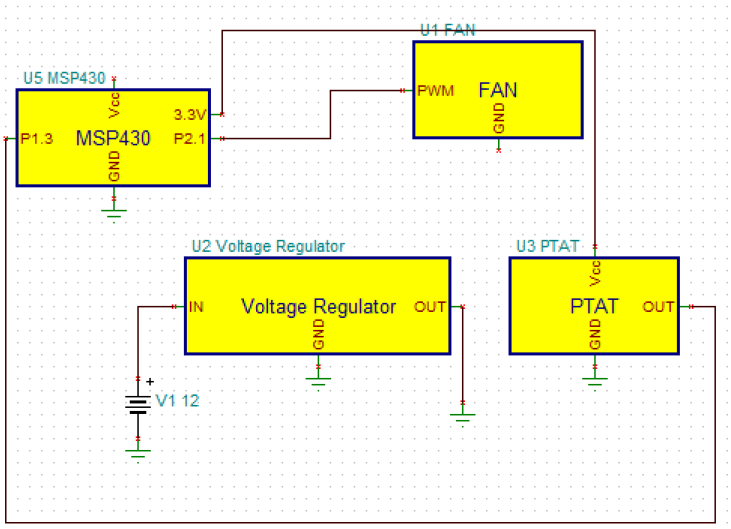

Figure 3: Block Diagram showing parts of system interacting

3.2 Highlighted Devices

- MSP430G2553: Microprocessor to read temperature and control fan speed
- Fan Control Circuitry: Read the temperature of the heat source and control the fan speed to keep maintain the desired temperature
- Heat Source: Voltage regulator outputting current to dissipate power as heat

3.3 MSP430

The MSP430 is the microprocessor used to control this system. It has two inputs, which are the desired temperature and the current temperature of the temperature

sensor. The desired temperature is set via UART, and is an 8-bit hexadecimal number that corresponds to the desired temperature in degrees Celsius. The current temperature is gotten from the temperature sensor, which gives a voltage that is directly proportional to the temperature. This relationship is shown in Equation 1 below, where V_r is the voltage reading.

$$T = \frac{V_r - 0.750}{0.01} + 25 \tag{1}$$

This equation is formed by the characteristics of the temperature sensor, as it has a 0.750V offset at 25 degrees C, and then is 10 mV/°C after that. The read temperature is then compared to the desired temperature, and ran through a simple PID control loop to determine how fast the fan should spin.

3.4 Fan Control Circuitry

The fan is is the output of this system, and creates a loop as it affects the input from the temperature sensor. The MSP430 takes a temperature reading from the temperature sensor, and controls the fan speed based off of the difference of the current temperature and the desired temperature. Since the fan operates off of 12V at up to 600 mA, an IRL520 Silicon Carbide MOSFET is used as a low side switch. Since the fan acts as an inductor, a catch diode is utilized to prevent an induced emf when the fan shuts off from damaging the MOSFET or even the MSP430. The setup for controlling the fan speed is shown below in Figure 4.

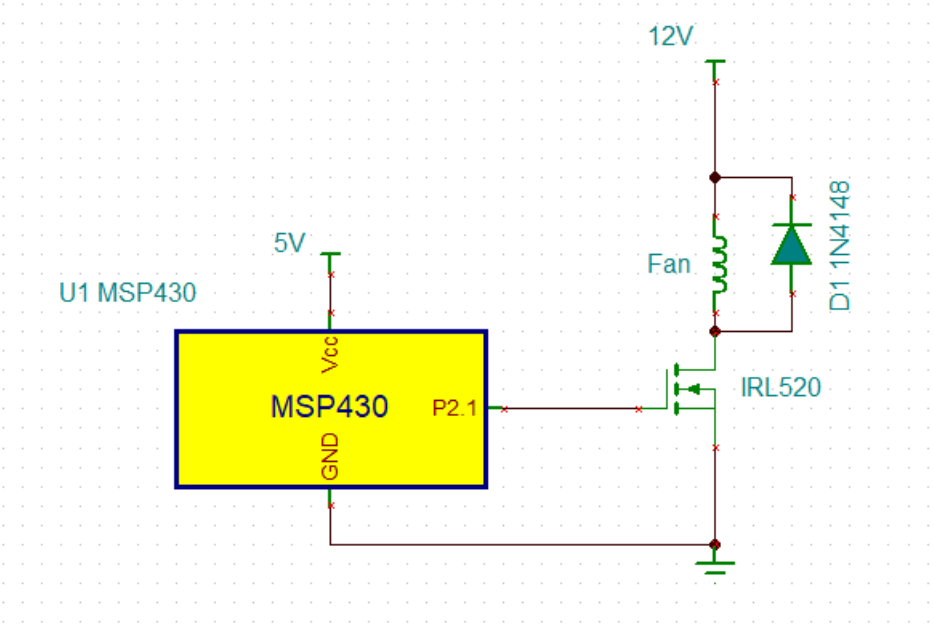

Figure 4: Circuit Diagram to Control Fan Speed

The IRL520 Silicon Carbide MOSFET should be used due to its low $R_{ds(on)}$ of only 0.27 Ω, which means that it will dissipate less power than a different MOSFET such as a 2N7000 which has an $R_{ds(on)}$ of 5 Ω. The equation for power is shown below in Equation 2, and shows that power dissipated is directly proportional to resistance.

$$P = I^2 R \tag{2}$$

Since the fan can draw up to 600 mA, this means that the SiC MOSFET could dissipate up to 97.2 mW, while the 2N7000 could dissipate up to 1.8 W. The IRL520 is rated for a peak power dissipation of 60W, while the 2N7000 is only rated for 350 mW, which means that it would likely not last very long, and justifies the use of a more expensive MOSFET.

3.5 Heat Source

The heat source in this setup is a L7805CV 5V Regulator that has an input of 12V and outputs 5V. By connecting a low resistance power resistor in series to ground as shown in Figure 5, the voltage regulator will output current and dissipate the power as heat.

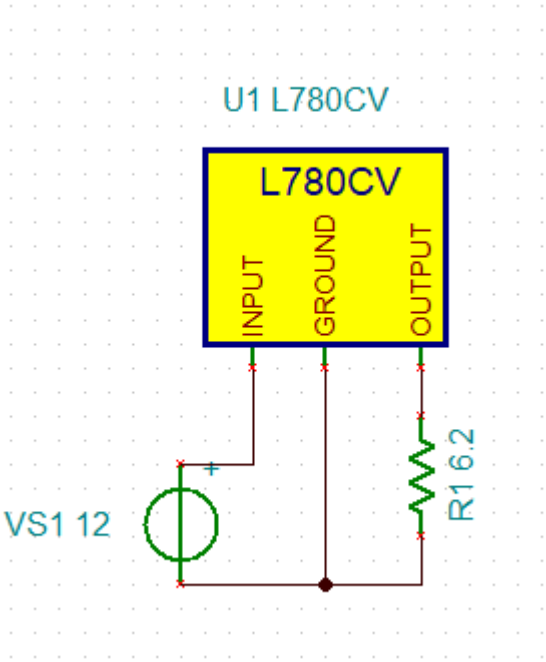

Figure 5: Circuit used to dissipate heat across the 6.2 Ω resistor

The current can be calculated by using Ohm's Law and solving for current as

shown in Equation 3 below.

$$I = \frac{V}{R} \tag{3}$$

The voltage is 5V, and the resistance is 6.2 Ω, which means that the current being outputted by the voltage regulator is 0.806A. The power dissipated by the voltage regulator can be found using Equation 2 above, with the voltage being the difference of the input voltage and the output voltage. This yields a power dissipation of 0.806A * (12V-5V) which is 5.6W. By using a lower input voltage, the power dissipation is lower which is more manageable for the fan.

4 SYSTEM DESIGN THEORY

This system is made up of only a few parts. There is a temperature sensor that provides a temperature reading input, UART communication is used to provide a desired temperature, a voltage regulator is used as a heat source, and a fan is the output of the system, used to control the temperature of the voltage regulator. The major part of the system is the microprocessor, which in this case, is the MSP430G2553. It is responsible for taking in the inputs and controlling the output based off of the inputs.

4.1 Design Requirement 1 - Communication

The first design requirement that had to be taken care of was the transmission of the desired temperature to the microprocessor. The communication is done via UART which stands for Universal Asynchronous Receiver/Transmitter. The input signal is very simple in this case, as it is only two bytes long and shows the desired temperature in °C. The returned signal is the temperature that is sensed at that time.

4.2 Design Requirement 2 - Pulse Width Modulation

The second design requirement was outputting a Pulse Width Modulated (PWM) signal used to control the speed of the fan. With a higher PWM duty cycle, the output is set high for more time, which would translate to a faster fan speed. An example of different duty cycles is shown below in Figure 6.

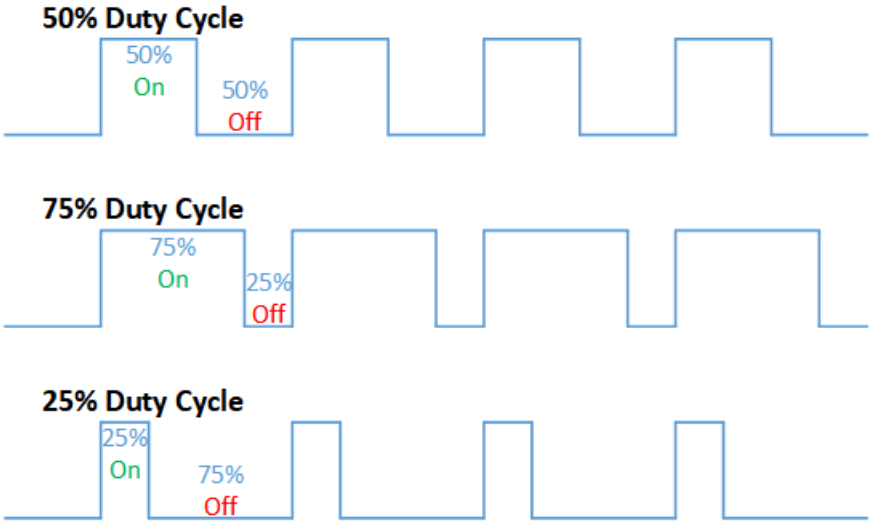

Figure 6: Different PWM duty cycles

Timer A1 is used in the up mode to achieve this. The reset value is set to the max PWM period of 1025 or 0x401 in hex. TA1CCR0(Timer A1 Capture Compare Register Zero) is used to store that reset value. TA1CCR1 is used to control the PWM of the fan, and TA1CCR2 is used as a blue status LED to visually show how hot the desired temperature should be. The reset frequency is 1025 or 0x401. Up mode in the timer works by counting from 0 to the period of 0x401, and then resetting back to 0. When the value of TA1CCR1 is equal to the value of the timer, it will cause an interrupt and flip the state of the pin attached to TA1CCR1, which is P2.1 as shown in Table 1. This pin is also attached to a red status LED, which visually shows how fast the fan is spinning. This processes is the same for TA1CCR2, which is connected to P2.5 as again shown in Table 1, and is a blue status LED which shows how hot the desired temperature is. When the value of the timer reaches the period of 0x401, all Capture Compare Registers will flip states again, which create the PWM. Figure 7 below is a section from the MSP430G2553's user guide that explains how the up mode feature works in Timer A1.

12.2.3.1 Up Mode

The up mode is used if the timer period must be different from 0FFFFh counts. The timer repeatedly counts up to the value of compare register TACCR0, which defines the period, as shown in Figure 12-2. The number of timer counts in the period is TACCR0+1. When the timer value equals TACCR0 the timer restarts counting from zero. If up mode is selected when the timer value is greater than TACCR0, the timer immediately restarts counting from zero.

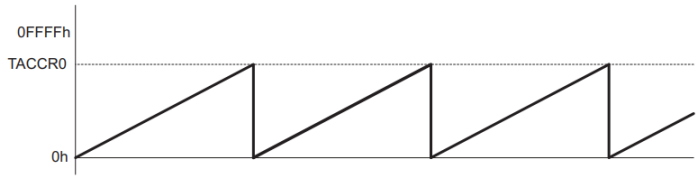

Figure 12-2. Up Mode

Figure 7: User Guide Info on Up Mode

4.3 Design Requirement 3 - Closed Control Loop

The final design requirement was utilizing a closed control loop in order to maintain the temperature of the voltage regulator. Due to time constraints, a full Proportional, Integral, Differential (PID) control loop was not utilized. Instead, a crude control loop that looks at the difference between the measured temperature and the desired temperature is used. If the difference is negative, then the desired temperature is higher than the measured, and therefore the fan should spin slower in order to allow the voltage regulator to heat up. As the temperature approaches the desired temperature, the fan should spin faster to slow the momentum and keep the temperature at the desired temperature. The algorithm used to implement the desired control of the system is shown below.

$if(difference > 3)TA1CCR1 = 1025;$
$else\ if(difference > 1)TA1CCR1 = 800;$
$else\ if(difference == 0)TA1CCR1 = 600;$
$else\ if(difference < 0)TA1CCR1 = 200;$

5 Getting Started/How to use the device

To set up the device for testing and operation, begin by wiring the circuit as displayed in the schematic shown in Figure 9. After initial setup, using the device is easy and user friendly. First, the code must be uploaded to the board from Code Composer and then the code needs to be ran. Next, using RealTerm, the user sends the processor a value. This value is the max temperature, in degrees Celsius, that is desired for the voltage regulator. This is all that is required from the user to use the system. Simply set up the circuit and enter the voltage regulators desired max temperature.

6 Getting Started Software/Firmware

There are two pieces of software that are necessary for operation of this system. These would be Code Composer, a compiler used for coding the MSP430, and RealTerm, a serial terminal used to send a data package to the MSP430. Code Composer is necessary to upload the projects code onto the MSP430. RealTerm is necessary in order to communicate via UART to the device and interact with the system.

6.1 Communicating with the Device

Once the code found in the main.c file in the Github Repository has been uploaded to the MSP430, communicating with it involves connecting it to an open serial port at the correct baud rate.

6.2 Device Specific Information

On the MSP430G2553, there is a header with several jumpers on it that change the configuration of the board. In order to switch between hardware and software UART, the jumpers labeled RX and TX must be in the orientation that is printed onto the silkscreen. In this case, hardware UART is used, so the two jumpers must be turned so that they are pointing perpendicular to their original positions, which is shown in Figure 8 below.

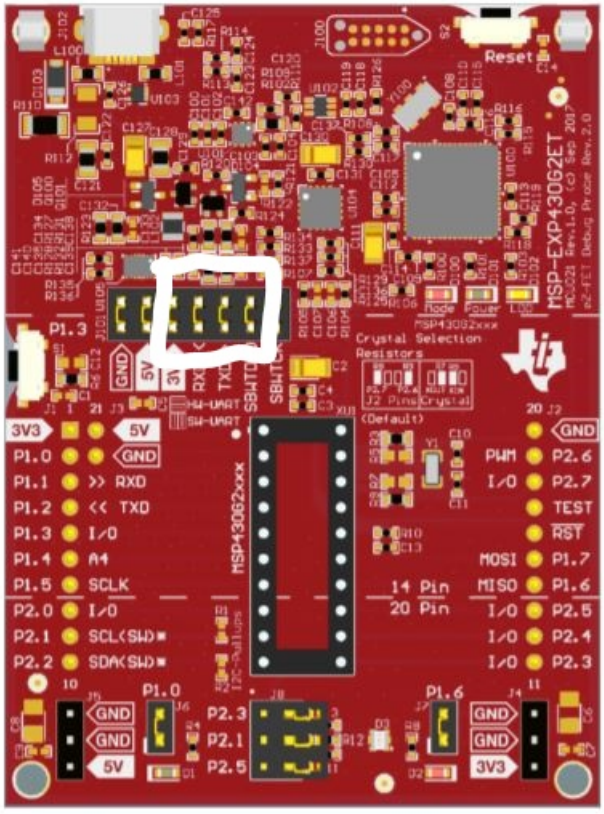

Figure 8: Image showing the original configuration of the MSP430G2553 and showing the specified jumpers in highlighted white box

7 Test Setup

To setup for test, circuit must be wired as shown in Figure 9.

- Supply Power to board from computer.
- Create a 12V Power rail
- Connect fan to 12V and Drain terminal of MOSFET
- Insert Diode across fan pins
- Voltage regulator input to 12V
- Voltage regulator output to power resistor
- Supply 3.3V from MSP430 to PTAT

- Connect PTAT output to P1.3 on MSP430
- Attach PTAT insulated top to thermal heat sink of voltage regulator
- Ground respective terminals of PTAT and voltage regulator to MSP430.

8 Design Files

8.1 Schematics

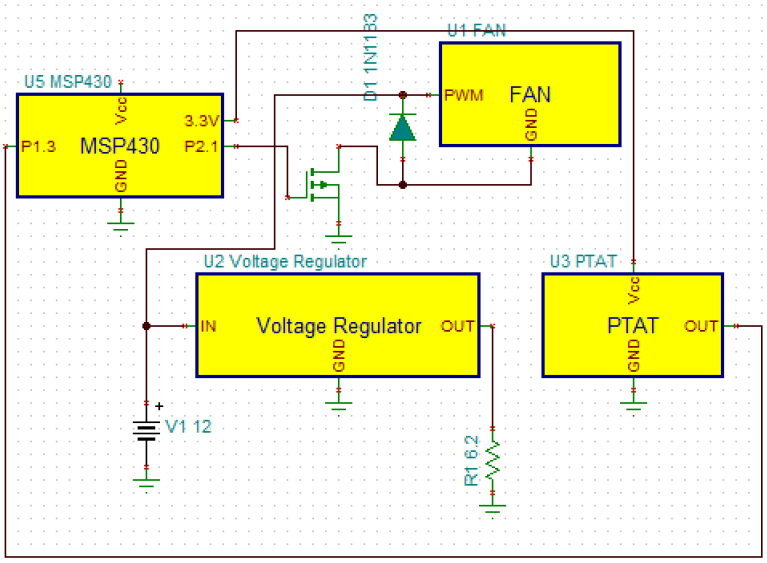

Figure 9: Schematic showing how the MSP430 connects to the LED and how it would connect to other nodes

8.2 Bill of Materials

- MSP430G2553 x1
- AUB0912VH 12V Fan x1
- TMP36 Low Voltage Temperature Sensor 1x
- L7805CV 5V Regulator 1x
- 1N4148 Signal Diode 1x
- IRL520 SiC Mosfet 1x
- 6.2 Ω 10W Power Resistor

- Various Jumper Cables (M-M) (F-M) (F-F) for various connections
- Arctic MX-4 Thermal Paste to create a thermal connection between the TMP36 and the L7805CV