MECHTREON 3TA4 Lab 4

PWM Fan Controller

Introduction

You will build a digital thermometer which displays the current temperature and the user selected setpoint temperature on a LCD display. If the temperature exceeds the setpoint you will turn on the fan to cool the sensor.

Prelab

1. This lab assumes you are familiar with the material required for Lab1, Lab2, and Lab3.
2. Review the datasheets for the LM35 temperature sensor, LM358 OpAmp, and FQP7N20 power transistor.
3. Review Zhu, `Embedded Systems with ARM Cortex-M Microcontrollers in Assembly and C` , Chapter 15 General Purpose Timers, Section 15.3 PWM Output.

   Chapter_15_Timer slides give a summary but not all of the details.
4. Review Zhu, `Embedded Systems with ARM Cortex-M Microcontrollers in Assembly and C`, Chapter 20 Analog-to-Digital Converter (ADC)
5. You may find these Lab4 notes helpful.

Procedure

hardware

The LM35 temperature sensor produces a voltage output of 10 mV/degree C. Note that the connections shown in the datasheet are the **bottom view** of the TO92 package (see below).

Figure 1: The **Bottom View** of the LM35 Temperature Sensor (TO-92 Package)

You will need to perform an analog-to-digital conversion using the MCU to read the voltage. For optimum accuracy, you will need to amplify the output of the LM35 by a factor of 2 or 3 by using an LM358 OpAmp before feeding it into the A/D converter. Note that the LM358 is not rail-to-rail. For a 5 volt power supply, the output range is 0 to 3.0 volts. Filtering the OpAmp output may lower spike noise, but the output impedance of the circuit you use to filter should be less than 10k ohms. An example low pass filter and 3x gain signal conditioning circuit is shown below:

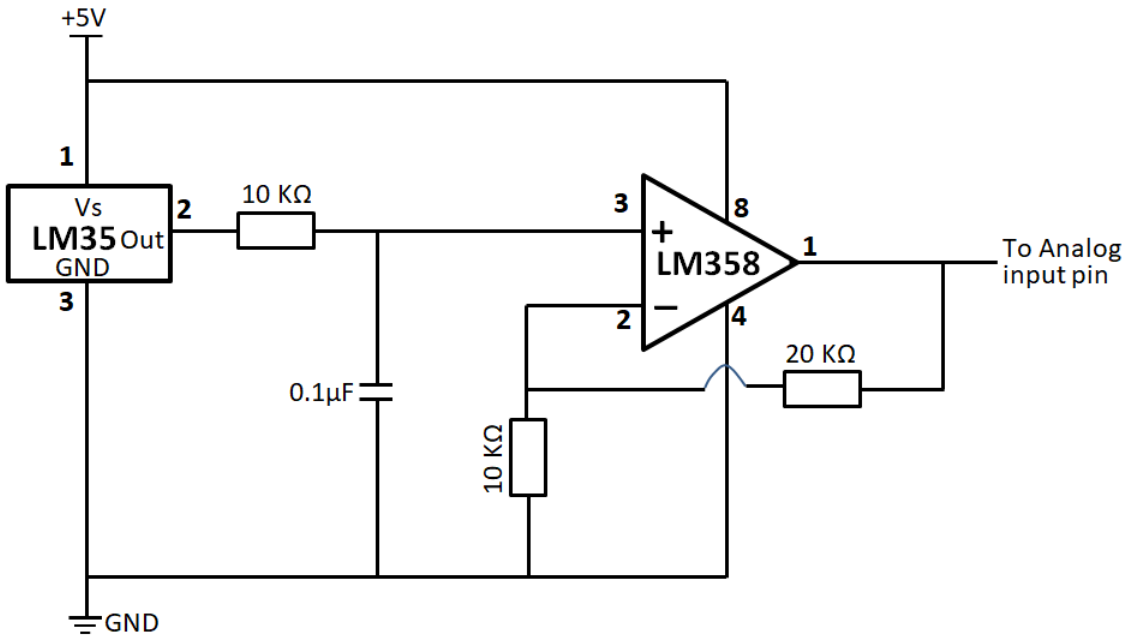


Figure 2: LM35 and Its Signal Conditioning Cirtuit

Use a Oscillopscope to decide if you need a filter.

The most accurate way to use the ADC requires that you use the internal voltage reference. For each different MCU that you use, you will need to measure Aref and use the measured value in your code.

You will need to drive a fan from the MCU. Fans have motors which can cause nasty inductive spikes to wipe out the transistors in the MCU port. The circuit shown below is fairly safe. An optoisolator completely isolates the MCU from the motor. The diode placed across the motor shorts out spikes when the motor is turned off. The resistor grounding the base of the phototransistor should be set for best falltime, probably around 1M ohm. The motor capacitor should start around 0.1uF. Increase it if there is too much spike noise on the analog input, but be sure to use ceramic capacitors, not electrolytic. Electrolytic capacitors are too slow. The pinout of the 4N35 optoisolator and FQP7N20 are also shown.

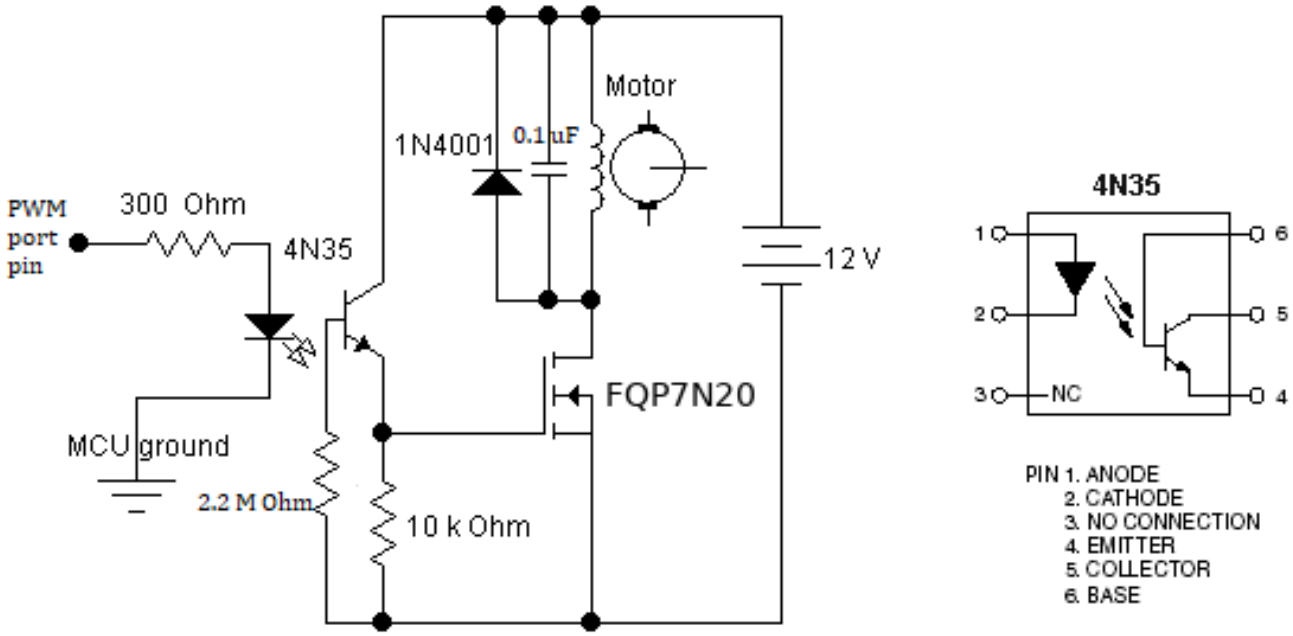
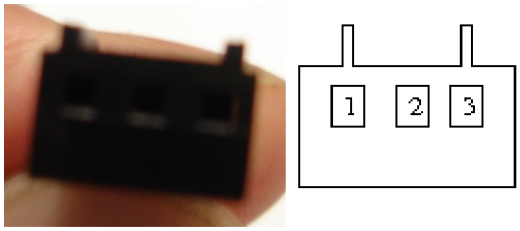


Figure 3: Cirtuit Diagram for Driving a Fan

Figure 4: Pins and Circuit Diagram of FQP7N20



For typical fan:

1	Black	Ground
2	Red	+12V
3	Yellow	Speed sensor wire - usually open collector output (PRM)

Figure 5: Fan Connector Pin Assignment

So for the circuit shown, Fan connector pin 1 gets connected to the FQP7N20 Drain, pin 2 to a 12 Volt source for the benchtop power supply and pin 3 is unused (for now!).

Software

- Download the starter package `lab4starter.zip` and place it in the directory structure described in Lab 1.
- You must configure the ADC channel (optionally in DMA mode).
  - Study the peripheral examples ADC_DMA_Transfer from the firmware package: `STM32Cube_FW_L4_V.1.8.0\Project\STM476G_EVAL\Example\ADC\`
  - The examples use the ADC channel in DMA mode. When you are sampling more than one ADC channel it is best to use DMA. If you are using a single ADC channel it is easier to configure this in independent mode. Read the ADC Manual to gain an understanding of available ADC modes. Read chapters 11 and 18 of the Reference Manual to understand DMA and ADC, respectively, and chapters 6 and 20 of the Programming Manual.
  - You will need to configure the mode of the GPIO pins for ADC signal input as the `GPIO_MODE_ANALOG_ADC_CONTROL` to use the additional functions on the GPIO pins. For the available pins for ADC functions, please consult table 15 of the datasheet for STM32L476.
  - Make sure to choose the appropriate streams and channels when using DMA for ADC. Please see check **Table 45** and **Table 46** For summaries of the DMA requests for each channel.
- You must also configure a PWM output.
  - Study the peripheral example TIM_PWMOutput from the firmware package: `STM32Cube_FW_L4_V.1.8.0\Project\STM476G_EVAL\Example\TIM\`
  - The example uses the four CCRs of Timer 1 to output 4 different PWM signals. For this lab you only need to output one signal.
  - For this lab, you must correctly configure a timer to generate the PWM signal. Bit banging will not receive full marks!
  - Read chapter 58 of the Programming Manual and section 31.3.9 of the Reference Manual to learn more about the PWM capabilities of the STM32L476 MCU.
  - You will need to configure the mode of the GPIO pins for PWM output as `GPIO_MODE_AF_PP` or `GPIO_MODE_AF_OD` to use the alternate functions on the GPIO pins. For the available pins for timers output functions, please consult table 15 of the datasheet for STM32L476.

Requirements

Write a program and construct a circuit that will:

1. **[25 pts.]** Upon RESET, measure room temperature with an LM35 and convert the output voltage form the LM35 to a number using an A/D converter in the MCU. Then display this temperature on the LCD. Make the default fan setpoint temperature (at RESET time) be greater than room temperature.

2. **[5 pts.]** Upon pressing the Selection button of the joystick, the application enters into the state allwing users to change the setpoint temperature. Pressing the Selection button of the joystick again makes the application exit the setpoint setting state.

3. **[5 pts.]** When in the setpoint setting state, each time press the Up button of the joystick, the setpoint will increase 0.5 °C. Each time press the Down button of the joystick, the setpoint will decrease 0.5 °C. Display the new setpoint on the LCD.

4. **[25 pts.]** If the actual sensor temperature is above the setpoint, turn on the fan. The fan should point at the LM35 and turn itself off when the temperature sensor reads a temperature smaller than the fan setpoint.

5. **[10 pts.]** When the fan turns on, it should spin slowly. The fan should increase its speed when the difference between the measured temperature and the setpoint increases.

6. **[5 pts.]** A heavily commented C code. Clear and concise comments are very important for others to understand your code and even for you later on when you need to remember what you did.

7. **[5 pts.]** Upload your project to Avenue. Make sure your porject folder remains in proper order:

   (a) There should only be one folder per lab (ie the only folders after this lab should be: lab1, lab2, lab3, lab4).

   (b) Only source the USER source files should be uploaded into your lab folder. You do not need to upload the library files.

You will demo in the lab all the features above to your TA and also answer the following questions:

1. **[10 pts.]** A calculation of the transfer function for the temperature signal conditioning circuit shown above.

2. **[10 pts.]** A summary of the accuracy of your measurements. You should look at the resolution of the A/D converter and the accuracy of the temperature sensor.

Some notes for this lab:

- To make sure the breadboard circuit is correct:
  - Figure 1 is the **"bottom view"** of the LM35. Connecting the LM35 in a wrong way will easily damage it.
  - *Test the temperature sensor and its conditioning circuit:* Apply 5V and GND signals to circuit shown in Figure 2, messure the voltage at terminal 2 of the LM35 and the voltage at terminal 1 of the LM358. The voltage at terminal 2 of the LM35 should be around 0.2V. The voltage at terminal 1 of the LM358 should be around 0.6V-0.7V.
  - *Test the optoisolator/fan circuit:* Use external power supply to apply 12V (1A) to circuit shown in Figure 3. Connect the 3.3V pin from the STM32L476-Discoverty to the "PWM port pin" of the circuit, and connect the GND pin of the STM board to the circuit GND. The fan should be turned on.

- Programming Notes:
  - For ADC, Vref+ is 3.3v and Vref- is 0v.
  - for STM32L476-Discovery board, ADC has fast channels and slow channels. If a slow channel is used, the sampling time should be longer, otherwiese, the ADC reading may not be correct. To make it safe, you can set the sampling time to 640.5 clock cycles.
  - For converting the ADC readings to temperature in Celsius, DO not let the board do all the calculations. This may give you wrong result since the board is not capable of doing complicated calculations. Instead, do the calculations manually and get a coefficient. Then let the board covert the ADC reading to temperature using this coefficient.
  - For PWM, it is better to use a separate timer. Do not use the same timer for PWM and for interrupt timing control. Using one timer for PWM and for OC interrupt may (most likely will) give you some trouble.
  - It is better to use macro: `__HAL_TIM_SET_COMPARE()` to change the PWM width.
  - For PWM, if duty cycle is less than a certain value (probably 15%), the fan will not run.