# Lab 3 – Fan Control

## Goals of the Lab

1. Control the speed of a computer fan using PWM.
2. Measure the fan speed using tachometer circuitry.

## Links to Tutorials and Specifications

Intel\_Motherboard\_4\_Wire\_PWM\_Spec.pdf (in Library section of Canvas)

## Parts List for the Lab

[F14 PWM Fan](https://www.arctic.de/en/F14-PWM/ACFAN00078A)

[Optocoupler](https://www.mouser.com/datasheet/2/308/1/HCPL2631M_D-2314497.pdf)

[LM340T5 Voltage Regulator](https://www.ti.com/lit/ds/symlink/lm340.pdf?HQS=dis-mous-null-mousermode-dsf-pf-null-wwe&ts=1627303515034&ref_url=https%253A%252F%252Fwww.mouser.cn%252F)

Resistors: 1k, 10k, 660, 150

Capacitors: 100nF

## Instructions

Some of the grade of this lab will be based on the report you turn in. The report is simply this lab manual (only Lab 3, not the entire manual) with answers and schematics. Please write your answers in green below the questions and paste schematics in from Altium where requested.

## Part 1: Beginning Analysis

### Fan Analysis:

1. What voltage does the F14 PWM fan require to operate?
   1. 12VDC Typical, 6VDC Starting
2. According to the 4-wire PWM Controlled Fans Specification document, what is the maximum steady state current allowed for a 4-wire computer fan?
   1. 1.5A
3. How much current does the F14 PWM fan use?
   1. 0.11A @ 12VDC
4. What voltage is required for the PWM input signal to the fan? Where does this voltage come from?
   1. The voltage is provided by an external power supply, with the signal controlled by the RasPi through an Optoisolator
5. How could you turn the fan off without removing power from the fan? This is not directly stated in the documentation but was discussed in the lecture material.
   1. By Grounding the PWM pin
6. What frequency is required for the PWM input signal to the fan?
   1. 21kHZ -> 28kHz (25kHz target)
7. What is the maximum current for the PWM input signal to the fan?
   1. 5mA
8. Complete the table for the pinout of the fan:

|  |  |  |
| --- | --- | --- |
| **Pin Number** | **Pin Function** | **Voltage Level** |
| **1** | Ground | 0V |
| **2** | Motor Power | 12V |
| **3** | Sense / Tach | 12V |
| **4** | Control / PWM | 0.8V – 5.25V |

### Optocoupler Analysis

1. What is the recommended supply voltage of the optocoupler?
   1. 4.5 – 5.5V
2. What is the typical input forward voltage of the emitter?
   1. 1.45V
3. Design a circuit using the single channel optocoupler described in the datasheet.
   1. The input side of the optocoupler should receive a signal from RPi pin 12.
      1. The current going into the optocoupler should not exceed 5mA
   2. The output side of the optocoupler should mirror the output side of the MOSFET LED circuit used in Pi Lab 2.
   3. When RPi pin 12 is 0V, the LED should be off. When RPi pin 12 is 3.3V, the LED should be on.
   4. The input and output sides of the circuit should NOT be connected in any way, including that they should not share a common ground.
   5. Draw this circuit in Altium. Paste the schematic here.



### Voltage Regulator Analysis

1. What is the maximum output current?
   1. 1.5A
2. What is the maximum input voltage?
   1. 35V
3. Using Table 6.6, what is the output voltage?
   1. 5V
4. In Altium, draw a circuit that produces a 5V output when the input is the fan power supply. Paste the schematic here.



### Combined Circuits

1. Now use the circuits previously designed and combine them into a single circuit where voltage regulator powers the output side of the optocoupler circuit. Paste the schematic here.



## Part 2: Build and Test Power Circuitry

1. On your breadboard, build a power supply circuit that does the following:
   1. Uses 12V power supply connected to DC barrel jack connector
   2. The power rail on the left side of the breadboard is 12V
   3. The power rail on the right side of the breadboard is 5V
   4. The power rails share a common ground
2. Use your multimeter to verify the power rails have correct voltages.

## Part 3: Build and test the Optocoupler

1. Build the ‘Combined Circuit’ designed above. Use 5V from the voltage regulator to power the optcoupler and LED output side of the circuit. Instead of connecting the input side of the circuit to the RPi, temporarily connect the Digilent to supply the input signal .
   1. Please note that connecting a voltage directly to the input pin on the optoisolator will damage the optoisolator. You must use a current-limiting resistor as designed.
   2. Start with just providing 3.3V or 0V
      1. When 3.3V is applied, the LED should turn on. When 0V is applied, the LED should turn off.
      2. When 3.3V is applied, test the current between Digilent and optoisolator
   3. Then use the function generator in Waveforms and make a square wave to emulate PWM
      1. Set the offset to 1.65 and the amplitude to 1.65 to generate a 0-3.3V range
      2. Verify that you can brighten and dim the LED by changing the duty cycle (symmetry).
2. Once it works with the Digilent and you have verified that it does not pull too much current, replace the Digilent with the RPi and run the led\_dimmer code from Lab 2.
   1. RPi Pin 12 should be connected to the resistor on the input side of the optoisolator.
   2. An RPi Ground pin should be connected to the ground pin on the input side of the optoisolator.
      1. This ground should not be connected to the other grounds to maintain electrical isolation.

## START HERE -> Part 4: Test the F14 PWM Fan

1. Using header pins and the breadboard power rails, connect the 12V power rail to the Power and Ground pins on the F14 PWM fan and verify that the fan runs.
2. Why is the fan running with the Control pin disconnected? How can we use the Control pin to run the fan off?
   1. Use your multimeter to measure the voltage of the Control pin on the fan and record the voltage below.
3. What is the value of the pullup resistor in the hub of the fan? You will need to measure this using your multimeter.
   1. Add a 10k resistor to ground, use your multimeter to measure the voltage across the resistor, and then use voltage divider equation to determine the internal resistance. Show your results and work below.
4. How can we create 0V at the Control pin?
   1. Record how you will create 0V at the control pin.
   2. Test it out and make sure that the fan goes off.
      1. The fans with black blades are defective and do not turn completely off. For the rest of the lab, when I say to verify that the fan turns off, we will just verify that it almost turns off.
5. In Altium, draw a circuit diagram of the Control pin (including internal fan components) and a push button connected to the Control pin. The fan should run when the button is not pressed and should turn off when the button is pressed. Insert the schematic below.
6. Based on the circuit diagram, add a push button and verify that you can turn the fan on and off by pressing the push button. Once you have verified that it works, remove the push button as it will no longer be used.

## Part 5: Control Pin Integration

1. Design a circuit to connect the Control pin on the fan to the output side of the optoisolator. The open/closed behavior of the optoisolator should whether the fan is on or off. Note that at this point, you will need to remove the LED circuitry since that was just for testing purposes. Draw the circuit in Altium and insert the circuit schematic below.
2. Build the circuit and test it using the Digilent or RPi on the input side of the optoisolator.
   1. You should be able to control the speed of the fan using the duty cycle.
3. Use the pwm\_fan\_test code provided in Canvas to test the operation of the fan.
   1. The code will ask for a duty cycle value. Provide a value from 0.0 to 100.0
   2. Go through a variety of duty cycle values and verify that the fan speeds up and slows down when the duty cycle is changed.
      1. You will notice something interesting about the behavior at different duty cycle values. Why do we see this behavior?
4. Use the Digilent Oscilloscope functionality to verify that the appropriate 25kHz PWM signal is being provided to the Control pin on the fan.
   1. Capture a screenshot of your PWM signal that shows your measurement of 25kHz.
   2. Insert the screenshot below.

## 

## Part 6: Tachometer (Sense) Signal Design

The tachometer signal is briefly described in Section 2.1.3 in the Intel\_Motherboard\_4\_Wire\_PWM Spec. To explain further, there is a hall effect sensor in the hub of the fan. The fan has 2 magnets that are detected by the hall effect sensor, therefore producing two pulses for every revolution of the fan. The spec tells us that the tachometer signal has an open-collector or open-drain output. It also tells us that the “motherboard” will have a pull up to 12V. In a computer, the motherboard provides the circuitry that is connected to the fan. In our case, we are building the circuitry to connect to the fan. Therefore, we are the “motherboard”. However, there is no need for us to pull the voltage up to 12V, since the rest of our system does not use 12V. In our case, we should pull the output up to 5V.

1. Design a basic Tachometer circuit that only includes the 5V source, 1k pullup resistor, and the Sense pin (and ground of course).
   1. In Altium, draw the schematic for the circuit and insert the schematic at the top of this part.
2. Build and test the basic Tachometer (Sense) circuitry. This circuit should not be connected to the RPi, but you should use the RPi to drive the fan at various speeds using the pwm\_fan\_test program.
   1. Use the Digilent Oscilloscope functionality to analyze the signal produced by the tachometer pin. Try to determine the RPM value based on the output of the Digilent.
      1. Capture a screenshot of the Digilent where you measure the RPM value. Insert the screenshot below, and also show the calculation that you used to determine the RPM value.
   2. Use the laser tachometer to measure the fan speed. Compare the measurement to the RPM value calculated from the Digilent and verify that they are similar. Please note that a fan with white blades will pick up every blade turning, so you will need to divide the laser tachometer reading by the number of blades.
      1. Record your calculated values and your laser tachometer below. Do this for three different duty cycle values.

## Part 7: Integrate Tachometer with the RPi

1. Once the basic tachometer circuit works, use a second optocoupler as an electrical barrier between this tachometer signal and the RPi.
   1. Design a circuit that connects the basic tachometer circuit to the input side of the second optocoupler. The tachometer signal will drive the emitter in the optoisolator.
   2. Build the circuit and test that the tachometer circuit works as expected by using the Digilent to view the voltage of the optoisolator input pin and verifying that it is changing in a similar way that the basic tachometer circuit worked.
2. Design a circuit on the output side of the optocoupler that will read the output of the second optocoupler with RPi Pin 16. Do NOT connect this circuit to the RPi yet. We will test it first.
   1. Draw and insert the schematic for this circuit. This schematic should include the control pin circuitry connected to the input side of the optoisolator as well as the output side of the optoisolator that is connected to the RPi.
      1. This circuit is relatively similar to the push button circuit used in Lab 1. Please make sure to use a 10k resistor, and make sure to check the resistor value when you build it in case the resistors are not sorted well.
      2. All connections on the output side of the optoisolator should be connected to the RPi and should not be connected to the rest of the circuitry on the breadboard. This includes power pins, ground, and the input pin. We should design a circuit that is completely electrically isolated from the 12V source, 5V source, fan, etc.
         1. For Vcc and Ve, you should use the 3.3V power rail from the RPi. Do not use the 5V rail on the RPi.
3. Build the circuit but do not connect it to RPi Pin 16 yet. Test the output of the circuit (where you would connect RPi Pin 16) by checking:
   1. The voltage with the Digilent Oscilloscope
   2. The current to make sure that it is an acceptable current for the RPi.
4. Once tested, remove the Digilent and connect the circuit to pin RPi Pin 16.

## Part 8: Final Schematic

Now that the circuitry is designed and complete, draw a schematic that encompasses all of the circuitry for this lab. Please submit this as a PDF to the Lab 3 assignment in Canvas.

## Part 9: Tachometer Software

1. Now that circuitry is complete, please create a file called fan\_controller\_monitor.py and write software that:
   1. Allows the user to input a desired fan speed between 0.0 and 100.0. If the user inputs 0.0, the fan should be off (or close). If the user inputs 100.0, it should run at full speed.
   2. Reads the RPM value and prints the RPM value to a file called rpm\_value\_file.txt (about once per second). It should only print to the file after getting 10 samples and averaging those samples.
      1. To read the tachometer pin, please use the ‘GPIO.add\_event\_detect’ function and do your calculations and file writing in the callback.
      2. Since the callback is running in a separate thread, you will need to wait for the callback to finish before exiting the program (when you get a keyboard interrupt). To do this, you can just set a flag when you begin the callback and clear it when you end the program. Otherwise, you will get a segmentation fault if the program terminates before the callback finishes.
2. Use the laser tachometer to measure the fan speed and make sure that your RPM values are similar to the laser tachometer’s values.

## Submitting the Lab

1. When complete, demonstrate it for me so that I can verify its operation.
2. Files to submit:
   1. fan\_controller\_monitor.py
   2. Your completed Lab 3 section of the manual.
   3. A PDF of your final schematic.

# Lab 4 - IoT

## Overview

The goal of this lab is to use MQTT to control the fan from Lab 3 from a different RPi. There will be two RPi devices/applications used:

1. mqtt\_fan – the RPi that is running mqtt\_fan will have the fan circuitry connected to it (the other RPi will not have circuitry connected). This RPi will receive a requested speed from the command\_center and adjust the fan speed accordingly. It will also report the RPM value to the command\_center.
2. command\_center – the command\_center will not need circuitry connected to it (although it does not hurt to have it connected). It will simply receive keyboard input of a desired speed (0 to 100 where 0 is off and 100 is full speed) and send it to the mqtt\_fan application running on the other RPi. It will also receive and print the RPM value reported by the mqtt\_fan.

## MQTT Installation

To use MQTT, we need to install Paho MQTT by doing the following:

git clone --depth 1 -b v1.6.1 https://github.com/eclipse/paho.mqtt.python

cd paho.mqtt.python

sudo python3 setup.py install

## Topic Names

With MQTT, all topic names must be unique. Therefore, we need to include our last names in the topic names since we all have different last names. To do this easily, we will pass them in as command-line arguments. You will pass in your name first and then your partner’s name next. For example, if I partnered with someone named Smith and I am the command\_center, I would run the Python script by doing the following:

python command\_center.py potter smith

And since Smith is running the mqtt\_fan, he/she would use the following command:

python mqtt\_fan.py smith potter

The resulting topic names would be:

1. potter/command\_center/desired\_speed
   1. This would be published by command\_center and subscribed to by the mqtt\_fan
2. smith/mqtt\_fan/fan\_rpm\_value
   1. This would be published by the mqtt\_fan and subscribed to by the command\_center

## Software

command\_center.py is provided for you in Canvas. You should not need to modify this code at all. Your job will be to create mqtt\_fan.py. To do this:

1. Start with your code from fan\_controller\_monitor
2. Remove the part that asks the user for input. Instead, set the duty cycle based on the desired\_speed message received.
3. After averaging 10 tachometer readings, you should publish the fan\_rpm\_value rather than writing it to a file.
4. To do this, carefully evaluate the provided command\_center and incorporate some of that code to the extent possible.
5. The final mqtt\_fan.py should include:
   1. A print to the terminal when a new desired\_speed value is received that says something like:
      1. Requested Speed: 75.0
   2. The fan should change speeds according to the desired\_speed.
   3. Publishing the fan\_rpm\_value after 10 samples. You can verify this by viewing the output of the command\_center app.
6. While the fun part is to run with someone else, you can also test this by yourself. Just split the terminal and run the command\_center in one terminal and mqtt\_fan in the other.

## Submitting the Lab

1. Please submit mqtt\_fan.py.