

Experiment M6 Robot Cheetah Leg I: DC Motor Control Procedure

Deliverables: Checked lab notebook, In-lab demonstrations

Overview

In this lab, we will implement a sophisticated feedback controller to turn a regular DC motor into a servo motor, where we can precisely control the shaft position and dynamic response. A simple circuit model of a DC motor is shown in Fig. 1. The motor is given a voltage input V . This voltage produces current I in the motor, which places torque τ_m on the motor shaft, and then causes a change in the output angle θ of the motor.

The feedback controller you will develop will use real-time data for the measured motor shaft angle, angular speed, and current to adjust the motor voltage. The output torque τ_m is proportional to the motor current $\tau_m = k_T I$, where k_T is the motor torque constant. Thus, using a fast feedback loop to directly control current will allow us to directly control the motor torque.

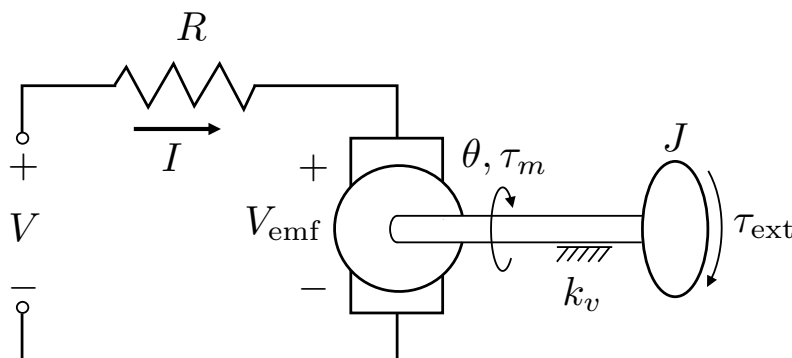


Figure 1 – Model of DC motor with applied voltage V , winding resistance R , current I , back-EMF voltage V_{emf} , output angle θ , motor torque τ_m , viscous friction coefficient k_v , rotational inertia J , and external torque τ_{ext} .

Importantly, a motor acts as a generator that produces a voltage or “back EMF” proportional to the motor speed, $V_{emf} = k_B \dot{\theta}$. Using Kirchhoff’s loop rule, we obtain the equation

$$V - IR - k_B \dot{\theta} = 0,$$

where V is the applied voltage, I is the current, R is the motor winding resistance, k_B is the back EMF constant, and $\dot{\theta}$ is the motor speed in rad/s. It is important to note that the back EMF constant is equal to the motor torque constant, $k_T = k_B$.

Part 0: Teensy 4.1 Microcontroller and Coding Environment

In this experiment, we will use a much faster microcontroller: the Teensy 4.1, which is equipped with a 600MHz ARM CPU (same architecture as your phone), compared to the Arduino UNO's ATmega328P with clock speed of only 16MHz. A pinout for the Teensy is shown in Fig. 2. Note that the Teensy uses 3.3V instead of 5V for its digital HIGH signal, which allows it to run at a faster clock speed.

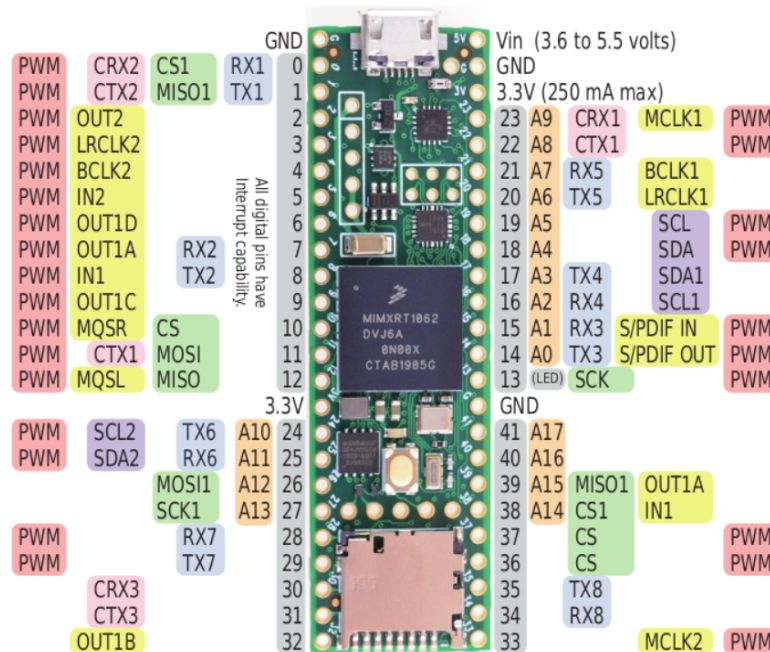


Figure 2 – Pinout for the Teensy 4.1 Microcontroller.

Procedure

1. Align the Teensy pin with the breadboard, such that it is straddling the trench across the middle. Firmly press down on the teensy with both thumbs to connect it to the breadboard. (This step is not necessary if it is already in the breadboard.)
2. Open the **Arduino IDE 2.3** software on the lab computer (NOT the 1.8 version). Go to Tools > Board and select Teensy 4.1. If you do not see Teensy, you will need to install it. Go to https://www.pjrc.com/teensy/td_download.html and follow the instruction at the top of the page. Ask the lab instructor for the admin password.
3. Connect the Teensy to the lab computer using the USB Cable. Got to Tools > Port to make sure the port with the Teensy is selected.
4. To test the Teensy, we will run the example Blink code. Go to File > Examples > 01.Basics > Blink. Upload the code as you normally would.
5. Adjust the code so the LED is on for 2 seconds and off for 2 seconds. Upload the code to the Teensy, and **demonstrate it to the lab instructor or TA to receive credit on your score sheet.**

Part 1: Motor Encoder

You will now connect use the Teensy to measure the motor shaft angle and angular speed. We will be using the single motor (not the ones connected to the cheetah leg).

Procedure

1. Make sure the Teensy is disconnected from any power source including the USB cable. (The Teensy is a bit more fragile than the Arduino UNO, and short circuiting any of the pins will likely fry the entire board.)
2. Use the 3V and G pins on the Teensy to create 3.3V and Ground bus lines on the breadboard. (Important: It is a 3.3V bus line—NOT a 5V bus line.)
3. Make the following connections with the Teensy and motor encoder wires. Use jumper wires that match the color of the motor wires.
 - a. Encoder White → Teensy Pin 6
 - b. Encoder Yellow → Teensy Pin 5
 - c. Encoder Blue → Teensy 5V
 - d. Encoder Green → GND bus line on breadboard
4. Download the Encoder Example code from the lab webpage. Upload it to the Teensy.
5. Open the Serial Monitor, then press the reset button on the Teensy. Rotate the shaft, and you should see the time (microseconds), shaft angle (degrees), and shaft speed (rad/s) printed in the serial monitor. Note that the program will re-zero the shaft angle whenever you press the reset button on the Teensy.
6. Test the sub-system to make sure it works. **Demonstrate it to the lab instructor or TA to receive credit on your score sheet.**

Part 2: Motor Driver and Analog Current Sensor

You will now connect the motor to the motor driver board and current sensor pictured in Fig. 3.

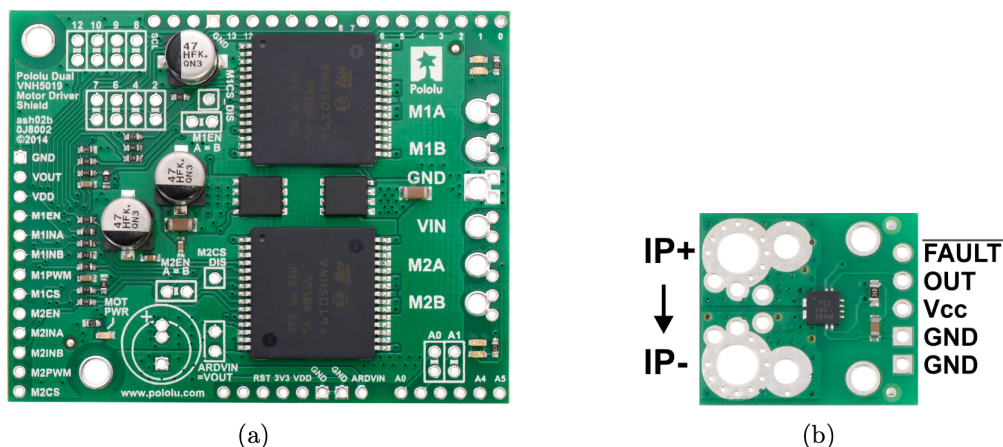


Figure 2 – (a) Pololu dual VN5019 motor driver shield for Arduino, (b) ACS711EX Current Sensor.

Procedure

1. Make sure the Teensy is disconnected from any power source including the USB cable. Make sure the DC power supply is off, as well.
2. Use thick 18 or 20 gauge wire to connect the green “POWER” screw terminals on the Cytron board to the 12V DC power supply. Make sure you use the kill switch.
3. Make the following connections between the Teensy and motor driver board:
 - a. Motor driver GND → GND bus on breadboard
 - b. Motor driver M1INA → Teensy Pin 4
 - c. Motor driver M1INB → Teensy Pin 3
 - d. Motor driver M1PWM → Teensy Pin 33
4. Use a black jumper wire to connect M1A to the black motor wire.
5. Connect the yellow wire (IP+) on the current sensor to M1B.
6. Configure the orange Extech DMM to measure current in the 10A range. Make sure the red grabber cable is plugged into the 10A port.
7. Connect a red jumper wire to the red motor wire. Complete the circuit by connecting the free red wire on the current sensor to the red grabber on the DMM and the other DMM cable to the red motor wire. This should create a path for current to flow out from M1B, through the current sensor, then through the DMM, then the motor, then back into M1A.
8. Download the Motor Driver code from the lab webpage. Note that it defines a function `setVoltage()` at the bottom of the code. Replace the `***` at the beginning of the code with the correct pin numbers.
9. Make sure the 12V DC power supply is switched off. Upload the to the Teensy.
10. Press the reset button on the Teensy, and turn on the 12V DC power supply. The motor should begin spinning. Try changing the voltage in the `setVoltage()` function in the main loop to a negative value. It should spin in the opposite direction when you upload the code. You should also see a change in the current on the DMM.
11. **Demonstrate it to the lab instructor or TA to receive credit on your score sheet.**

Part 3: Current Sensor Calibration

The analog current sensor outputs a voltage that is linearly related to the current flowing through it. You will now use the DMM to calibrate the analog current sensor.

Procedure

1. Similar to the Arduino UNO, the Teensy has analog-to-digital convertors that map a 0 – 3.3V input to a 10 bit integer from 0 -1023. The Motor Driver code from the previous part prints this integer value.
2. Change the voltage in the `setVoltage()` function in the main loop to be 10V.
3. Hold the blue servo horn attached to the motor shaft to stall the motor. Open the serial monitor. **Do NOT try to stop it while it is spinning—it will wreck your knuckles!** Press the reset button on the Teensy, and turn on the 12V DC power supply. Record the current measured on the DMM and the integer value printed in the serial monitor.
4. Change the voltage in the `setVoltage()` function in the main loop to be -10V. Repeat the previous steps. As you hold the servo horn, you should feel it pushing in the opposite direction, and the current on the DMM should change signs.
5. The motor current I is linearly related to the integer value VAL from the sensor via the calibration formula $I = a \cdot VAL + b$. Use the two data points you just measured to determine the calibration constants a and b for the analog current sensor. Record them in your lab notebook.
6. Implement the calibration formula in your code to convert the 10 bit integer value for current into the value in Amps.
7. **Demonstrate the working system to the lab instructor or TA. The current printed in the serial monitor should be within 10% of the current measured on the DMM.**

Part 4: Measuring the Motor Parameters

You will now perform an experiment to determine the motor winding resistance and back EMF constant k_b . These parameters will be important for controlling the motor torque and the motion of the Cheetah leg next week.

Procedure

1. Sketch Fig. 1 and copy down Eq. (1) into your lab notebook.
2. Make sure the 12V DC power supply is switched OFF.
3. Download the Motor Parameters code from the course website. Fill in the ***s in the variable declarations with the correct pin numbers and the current sensor calibration constants you just measured.
4. Change the voltage in the `setVoltage()` function in the main loop to be $V = 10V$.
5. Hold the blue servo horn attached to the motor shaft to stall the motor. Open the serial monitor. Press the reset button on the Teensy, and turn on the 12V DC power supply. Record the current printed in the serial monitor. This is the motor stall current at 10V.

6. Use the stall current you just measured and the 10V applied voltage to calculate the winding resistance R . Record the value in your lab notebook.
7. Release the servo horn and allow the motor shaft to spin freely at the no-load speed. Record the motor speed (rad/s) and current (Amps) printed in the serial monitor.
8. Use Eq. (1) to determine the back EMF constant k_B . Record the value in your lab notebook.
9. **Show your value of the winding resistance R and back EMF constant k_B to the lab instructor or TA to receive credit on your score sheet.**

Design Challenge 1 – Simple Position Control

You will now use the measured angle θ and angular speed $\dot{\theta}$ to compute the applied voltage via proportional-derivative (PD) feedback $V = k_p\theta + k_d\dot{\theta}$.

- Combine your codes from the previous sub-systems into a code that will measure the angle θ , then use the measured value to compute the motor voltage value using the proportional-derivative feedback formula.
- Be sure to keep the 20ms delay in the code, as this is necessary to eliminate noise from the measured angular speed.
- Include print statements that display the angular displacement, angular speed, and applied voltage.
- Estimate the proportional gain k_p to be the maximum voltage of 12V divided by a typical angular displacement $\Delta\theta$. A similar approach can be used for the derivative gain.
- Test the system with only proportional feedback, where $k_d = 0$. The motor shaft should behave as a torsion spring and exhibit damped oscillation.
- Test the system with only derivative feedback, where $k_p = 0$. You should feel viscous damping when you try to turn the shaft: it should be more difficult to turn the faster you turn.
- The final feedback controller should return the shaft to within 5 degrees of its home position.
- **Demonstrate the working system to the TA to receive credit on the score sheet.**

Design Challenge 2 – Current Control

You will now design a feedback controller to control the motor, and thus the torque, using the feedback formula $V = RI_S + k_B \dot{\theta} + K_p(I_S - I)$, where I_S is the desired current. This will be incorporated as a “fast loop” running within the main loop of your code.

- Do NOT try to add this to the code from the previous design challenge. Do this as a new code.
- Create a for-loop within the main loop to measure the current and adjust the feedback. It should run for 100 iterations at a rate of ~5kHz. (Use `delayMicroseconds()` within the loop to obtain the ~5kHz rate.)
- Delete any delays you have outside of the fast loop.
- Include print statements that display the angular displacement, angular speed, applied voltage, and measured current.
- Starting with a set-point of $I_S = 2$ Amps, hold the blue servo horn to stall the motor when you test it.
- Try it first with only proportional feedback. It should get close to the set-point, but never quite reach it. Then, add in the other terms of the formula.
- The final feedback controller should reach a steady state current within 0.1 Amps of the set-point for values between -4 and 4 Amps.
- **Demonstrate the system to the TA to receive credit on the score sheet.**

Design Challenge 3 – Servo with Current Control

Now that you can control the current, you can also control the torque. The desired torque can be reached by having a current set-point $I_S = \tau_S/k_B$ (recall that $k_T = k_B$). Modify the code from the previous design challenge to use PD feedback $\tau_S = -K\theta - D\dot{\theta} + \hat{k}_v\dot{\theta}$ to compute motor torque just before the fast loop. Then the fast loop will adjust the current to the corresponding set-point of $I_S = \tau_S/k_B$.

- Do NOT try to add this to the code from the previous design challenge. Do this as a new code.
- Test it with only proportional feedback first, where $\hat{k}_v = D = 0$. It should behave as a damped harmonic oscillator due to the natural viscous damping of the grease in the gear box.
- Adjust the value of $\hat{k}_v$ to make the damping disappear, so it oscillates indefinitely. This is called “friction compensation”. **Record the value of $\hat{k}_v$ in your lab notebook and demonstrate it to the lab instructor or TA.**
- With the friction compensation in place, adjust the derivative gain D to make the response have a damping ratio close to $\zeta = 1$. That is, it should get to the set-point angle as fast as possible without overshoot and oscillation. **Record the value of D in your lab notebook and demonstrate it to the lab instructor or TA.**

Clean-up

To receive full credit, you must return the lab bench to its initial state:

- On the Extech DMM, move the red grabber cable back to the voltage port. Failure to do this will cause students in AME Lab 2 to fry their circuits!
- Unplug the DC power supply. Remove the kill switch and screw terminal adapter.
- Leave the Teensy in the breadboard.
- Disassemble the circuit. Disconnect the wires from the Cytron board, Encoder shield, and Arduino.
- Put anything that belongs in your tool kit back into the tote bin.

Appendix A

Equipment

- Standard Mechatronics lab kit in small tote (above lab bench)
- C-clamp
- Teensy 4.1
- Medium size breadboard
- USB C to Micro USB cable, 6 ft length
- Pololu Dual VN5019 Motor Driver Shield for Arduino
 - <https://www.pololu.com/product/2507>
- ACS711EX Current Sensor
 - <https://www.pololu.com/product/2452>
- 12V DC motor with 18.75:1 gearbox and 64 CPR encoder
 - <https://www.pololu.com/product/4751>
- Extech handheld DMM with grabber cables