SY202 – Cyber Systems Engineering

Due Date: 02 March 2017

Intro CSE

LABORATORY INVESTIGATION #05: Actuation

Warning: Do not unplug and do not re-wire components of the testbed. If you connect something incorrectly, you may damage the mbed, the mbed application board, the h-bridge, or the motors.

Objectives:

- To reinforce basic C programming concepts and familiarization with the mbed microcontroller.
- To introduce concepts of hardware interfacing for actuation.
- To introduce the use and manipulation of DC motors and RC Servo motors, two common actuators to be used later in the semester.
- To introduce the importance time-scheduling control tasks within a cyber-physical system.

Introduction:

In this lab, the student will be connecting the mbed microcontroller with an application board and will use elementary logic to control the speed and direction of a DC motor using an H-bridge as well as two small RC (servo) motor. Both DC- and servo- motors are widely used actuators. The student will develop code that will actuate each motor using timing and system dependencies. This lab serves as an introduction to your final project, in which you will need to control the position of an "elevator" by controlling a DC motor.

Equipment:

Review equipment (see Figure 1) in Appendix for more details:

- Lab PC with internet connection
  - TeraTerm Software
  - Windows Serial Port Driver
  - Internet Connection (Online Compiler)
- Mbed NXP LPC1768 microcontroller
- Mbed Application Board (w/ 6V-2A power, USB cable)
- DC Motor (w/ H-bridge, 12V power)
- (2) RC Servo Motor

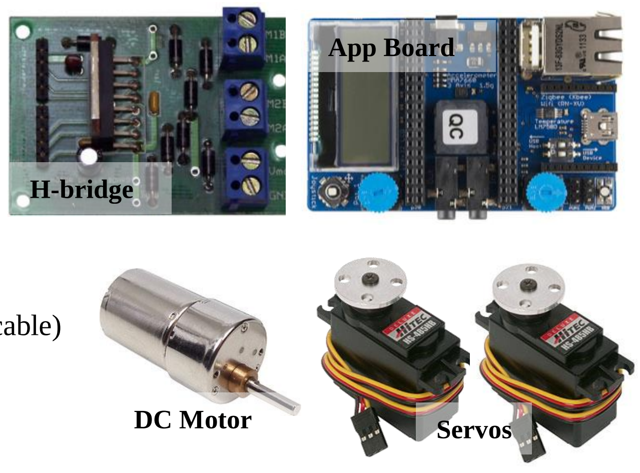


Figure 1. Equipment - Components

Exercise #1: Control of Brushed DC Motor

Brushed DC motors are current controlled devices. They typically require much greater current than can be supplied from a microprocessor. As such, we typically use motor drivers as interfaces between low-power circuits (such as microprocessors) and high power loads (such as DC motors). In this case, we use a L298 motor driver board, interfaced to the PWM outputs of the mbed processor. The L298 is connected directly to a 12V power supply, and as such can supply large current to the motor.

Your workstation should have a DC motor connected to an H-bridge along with 12 V power supply, as illustrated in Figure 2 (if not, ask your instructor). An mbed should also be connected to the motor driver following the convention: GND to GND, Vout to VCC, p# (from DigitalOut) to IN1, p# (from DigitalOut) to IN2, p# (from PwmOut) to ME1.

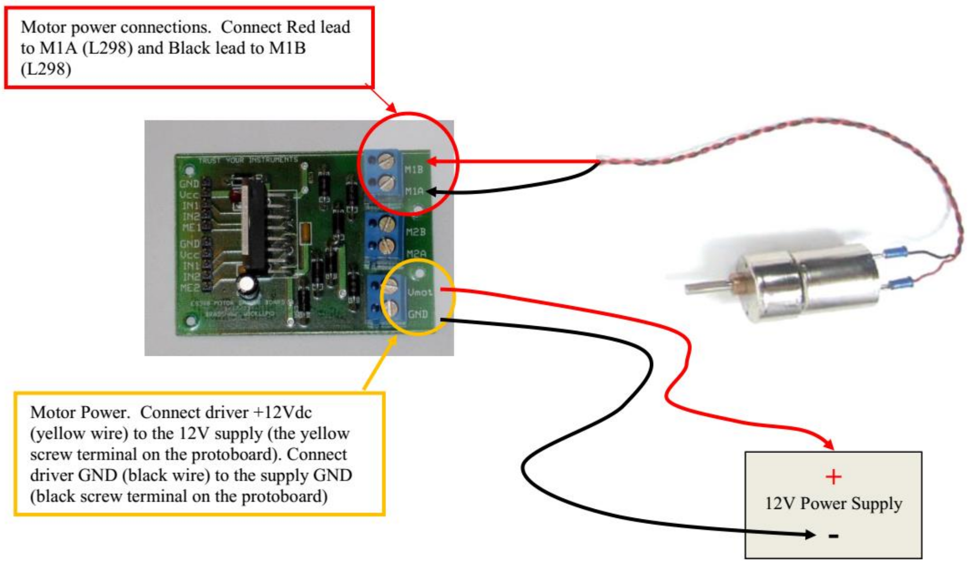

Motor power connections. Connect Red lead to M1A (L298) and Black lead to M1B (L298)
M1B
M1A
M2B
M2A
Vmot
GND
GND Vcc IN1 IN2 ME1
GND Vcc IN1 IN2 ME2
Motor Power. Connect driver +12Vdc (yellow wire) to the 12V supply (the yellow screw terminal on the protoboard). Connect driver GND (black wire) to the supply GND (black screw terminal on the protoboard)
+
12V Power Supply
-

Figure 2. DC motor and H-bridge

As discussed in class, you will use a technique called pulse-width modulation to control the motor speed. Control of DC motor with the mbed processor is straightforward. It requires a library “Motor.h”, an initialization command, and a direction/speed command function call.

```
      // Before main function
#include "Motor.h"         // allows your code to use the motor library
Motor m(pwm, fwd, rev);    // tells the mbed which pins to use for PWM motor control
      // After main function
m.speed(value);            // where value is replaced with a variable containing
                           // a PWM value between -1.0 and 1.0 (sign dictates direction)
```

Note #1: Check the mbed-hbridge connection of your testbed. Very likely, the pin numbers to use for the speed command are p23,p24,p25 (in that order). The first pin number (the pwm signal) should correspond to the pin connected to the ME1 in the H-bridge. The other two pin numbers are direction and should be connected to digital output ports. Interchanging them only change the direction at which the motor spins.

Note #2: You must add the Motor.h library (in addition to any other required library like mbed.h) to your program to compile this code.

- After you create a new program, right click on the program name (in the Program Workspace window) and select “Import Library > From Import Wizard …”
- Page down until you see the “Motor” library by Simon Ford.
- Double-click on the “Motor” library and it will be added to your current program.

Task:

Design an algorithm that accepts an initial PWM value from a user (use TeraTerm) between -1.0 and 1.0 (note that the speed command (e.g., `motor.speed()`) accepts values between -1.0 to 1.0) to set the initial speed of the DC motor. Then, allow the user to increment, decrease, or stop the speed of the motor in steps of 0.05 by using ‘u’, ‘d’, or ‘s’, respectively. Print to TeraTerm the actual PWM value being used to set the speed of the motor. Make sure to always send a PWM command between -1.0 and 1.0. Saturate the PWM command at the limit values. Always set the motor's speed to zero whenever you exit your main program (before closing the main function).

Exercise #2: Control of RC Servo Motor

RC servomotors are pulse-code controlled devices. Still, they too typically require much greater current than can be supplied from a microprocessor. Because all of the circuitry needed to drive the motor is internal to the device itself, we merely need to supply an external power source capable of providing the proper current and voltage (5 volts).

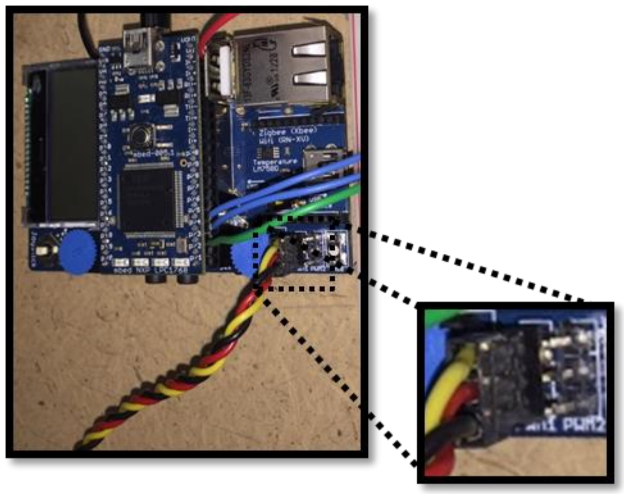


Figure 3. Servo Motor Connection

Your workstation should have two Servo Motors connected to the mbed application board. The Black Lead should be connected to the GND pin of the 3 pin connector. The application board has four 3 pin connectors hard wired to pins 21, 22, 23 and 24 for PWM, 6V power from input power (6V – 2A power supply plugged into the top of the application board), and ground.

The yellow cable is connected to pin #21 if using the PWM 1 slot (3 cable connector) or to pin #22 if using PWM 2. The red cable should be connected to the power supply (6V -2A) and the black to ground.

As mentioned in class, RC servos are controlled using a pulse coded signal. Your mbed processor has a library function that will take a desired position (0.0 to 1.0) and compute and output an appropriate pulse train to achieve position control. You will need the following commands to control RC servos:

```
      // Before main function
#include "Servo.h"          // allows your code to use the servo library
Servo myservo(p#);          // tells the mbed which pin to use for the pulse code signal
                            // either p21 or p22 if using the 3 connectors of app board
      // After main function
myservo.calibrate(PW/2 (s), MaxDegrees/2);
                            // calibrates servo's range of motion with pulse width (ms)
                            // Do this only once after main
myservo = value;            // set the angular position of servo to value, where value is
                            // float between 0.0 and 1.0 (scaled between 0° and MaxDegrees)
```

Note #1: You must add the "Servo.h" library from Simon Ford to your program to compile this code. Follow the same steps you did for the inclusion of Motor.h.

Note #2: You have two servo motors connected to the application board. Pick one of them for the next experiment.

Note #3: You also need to calibrate the servo motor, according to the manufacturer's settings. Use the command `myservo.calibrate(0.0009,90)` at the beginning of your main function to set a range of 180 degrees.

Task:

Design an algorithm that accepts an initial angular value from a user (use TeraTerm) between 0.0 and 180 degrees (note that the value passed to the servo must be normalized between 0.0 and 1.0, where 1.0 should represent 180 degrees). Then, allow the user to increment, decrease, or return to position zero in increments of 10 degrees by using ‘u’, ‘d’, or ‘r’, respectively. Print to TeraTerm the actual position in degrees of the servo motor. Saturate the commands between 0.0 and 1.0 if the user exceeds the limits.

Exercise #3: Control of Brushed DC Motor and RC Servo Motors

Most control systems and applications are time-critical. Tasks within a cyber-physical system need to be scheduled and executed under time constraints. On one hand, you want your control system to react quickly to disturbances and unexpected events. On the other hand, fast motion of actuators can produce physical damage to the structure they are aiming to control. In the next exercise, you will control multiple actuators while regulating time.

Your work station should have a DC motor and two servo motors. Now, you will control all three simultaneously. Take a moment to think about how to implement the following code.

Task:

Design an algorithm that simultaneously does the following:

- Spin the DC motor forward using a PWM signal of 0.5 for 5 seconds; then, stop the motor (PWM = 0.0) for another 5 seconds; then, change to reverse motion with a PWM of -0.5 for another 5 seconds; and; finally, stop the motor for 5 seconds. Repeat the 20 seconds sequence as many times (1-9 times) as specified by the user.
- Gradually moves Servo Motor #1 from 0 to 180 degrees in 10 seconds, then move from 180 to 0 degrees in the next 10 seconds. Make motion as smooth as possible and repeat as many times as specified by the user.
- Gradually moves Servo Motor #2 from 180 to 0 degrees in 10 seconds, then move from 0 to 180 degrees in the next 10 seconds. Make motion as smooth as possible and repeat as many times as specified by the user.

Basically, your program will move two servo motors simultaneously back and forth in opposite directions like windshield wipers, completing one full cycle in 20 seconds. Meanwhile, a DC motor will spin back and forth a constant speed with stops of 5 seconds in between, completing one full cycle also in 20 seconds. The whole process should repeat for as many times as specified by the user.

Note #1: Declare/initialize the **motor before the servos** (in the preamble).

Note #2: Make sure that the motor speed is always set to zero before you culminate your program.

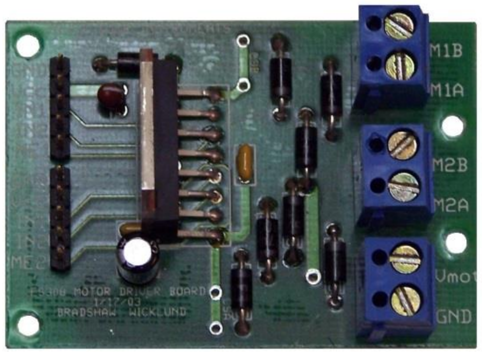
Note #3: You will need to time the execution of your program. Later in the semester you will learn about other timing techniques that will allow you to control the sampling period of your program with better accuracy and precision. For the moment, simply use `wait()` or `wait_ms()` commands to time your program. Time of cycles does not have to be 100% accurate but close to 20 seconds.

Deliverables:

1. Demonstrate program operation to the instructor. This may be accomplished in person or by providing your functioning program (ie a compiled ".bin" file) that can be directly loaded onto the instructor's mbed to verify functionality.
2. Follow the lab report template and the general lab guidelines for SY202 lab reports. Refer to the lab rubric for the grading of the lab.

Appendix: Equipment Details

H-Bridge Quad Power MOSFET Driver for DC Motor Control



- The L298 Motor Driver Board is capable of driving DC motors and inductive loads at 2A of peak output current on each channel. The board has two 5-pin single in-line headers for single board computer or microcontroller interfacing of motor control signals. Motor leads are attached through screw terminals as well as the motor power source (up to 46V). The IC has over-temperature protection and the circuit board allows current measurement capability through the application of an external resistor.
- Allows N-channel power MOSFETS driving in a full H-bridge configuration and is best suited for DC Motor Control Applications. The four driver's outputs are designed to allow 25kHz MOSFET switching.
- The speed and direction of the motor are to be set by two pins (P26 & P30 or P25 & P27).
- The H-bridge is capable of driving two motors, however, in this lab only one motor is required.
- The H-bridge is powered by a 12-volt source, denoted by reference voltages V+ and V-.
- The H-bridge takes a 3.3V PWM signal as input and produces a $\pm$12V output PWM signal.
- Voltage output is controlled by low side Pulse Width Modulation (PWM). This PWM feature can be made internally when the input pin is connected to an analog signal, or it can be given directly from a digital source.

Jameco Reliapro 12V DC motor: HN-GH12-1634T-R

- The DC motor produces angular velocity proportional to the voltage difference applied at the input terminals
- 4.5-12 Volt operating range
- 293 mA current at max. efficiency
- 850 g-cm torque at max. efficiency

HS-422 Deluxe Standard Servo

Consumer grade servo is able to take in 6 volts and deliver 57 oz-in. of maximum torque at 0.16 sec/60° The HS-422 servo comes standard with a 3-pin power

- Voltage: 4.8-6.0 Volts
- Torque: 45.82/56.93 oz-in. (4.8/6.0V)
- Speed: 0.21/0.16 sec/60° (4.8/6.0V)
- Direction: Clockwise/ Pulse Traveling 1500-1900usec
- Rotation: 180°

mbed microcontroller

- In this application the mbed microcontroller produces a PWM signal whose duty cycle is user-specified. The duty cycle corresponds to an average applied voltage to the DC motor.