The Mechatronics Revolution: Fundamentals and Core Concepts
Lab Assignment 4

DC Motor Speed Control via PWM Signals

4.1 Objective

The main objective of this lab is to interface and control the two brushed DC motors of the TI-RSLK-Mechkit robot with the MSP432 LaunchPad, using PWM signals and an H-bridge motor driver. You will control both the *speed* and *direction* of the DC motors using the MSP432 LaunchPad.

In Module 4, we studied different types of actuators for mechatronic systems, and how to control them via MSP432. We also learned how to generate PWM signals using the MSP432 LaunchPad. Specifically, the fundamental knowledge required for this lab assignment were provided in the following topics of Module 4:

Module 4, Topic 2, Lesson 1	Introduction to Brushed DC Motors
Module 4, Topic 2, Lesson 4	Specifications of DC Motors
Module 4, Topic 3, Lesson 1	PWM Waveform
Module 4, Topic 3, Lesson 2	PWM Control of Mechanical Devices
Module 4, Topic 3, Lesson 3	Generating PWM Signals on the MSP432

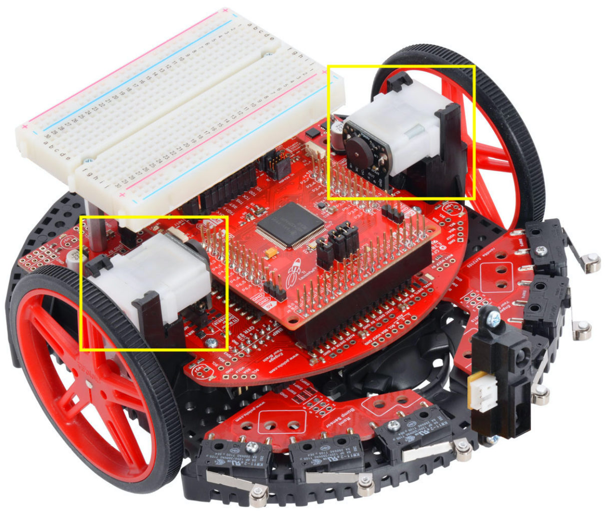
Figure 1: The TI-RSLK-Mechkit robotics system learning kit includes two brushed DC motors that are attached to the wheels of the robot.

4.2 Setup

This lab requires the *Code Composer Studio*, the *TI MSP432P401R LaunchPad*, and the *TI-RSLK-Mechkit* along with the *H-bridge motor driver circuit* which we built on a breadboard in Lab Assignment 2. The lab uses onboard features of the MSP432 LaunchPad such as *Timer A*, *GPIOs* and *switches*.

All components needed for this lab are included in the TI-RSLK-Mechkit. Details on how to assemble the TI-RSLK-Mechkit robot are presented in the Robot Assembly Manual, available under Course Resources on edX (Robot-Assembly-Manual.pdf). Six AA batteries (not included) will be needed to power your robot. You can use disposable 1.5V alkaline batteries or rechargeable 1.2V NiMH batteries.

Two predefined grading macros and a setup function are declared in the `mechrev.h` header file, available under Course Resources on edX (mechrev-header.zip). The output text file generated by the macros will be used to grade the assignment. The motors' encoders data which will be collected automatically will also be used for grading purposes.

The output of the grading macros will be printed to the Code Composer Studio console under the debug mode. Therefore, the LaunchPad USB cable must be connected from robot to PC when collecting the grading outputs.

Warning: Make sure that you have removed the +5V jumper on the MSP432 LaunchPad. Not removing this jumper will cause permanent damage to the LaunchPad and the TI-RSLK chassis board.

4.3 Problem Statement

Use the H-bridge motor driver circuit built in Lab 2 and the MSP432 LaunchPad to control the *speed* and *direction* of the two DC motors attached to the wheels of the TI-RSLK-Mechkit robot. The main component used in the motor driver circuit is the H-bridge IC (SN754410), implemented in Lab 2. The direction and speed control is performed through PWM signals, generated by your software program. The motors' speed is varied in three steps, based on the digital inputs provided by the *S1 and S2 switches* (pushbuttons) on the MSP432 LaunchPad.

Your software control program should operate as follows: Initiate S1 and S2 switches as digital inputs. The motors should start spinning in one direction when S1 is *pressed and held down*, and stop spinning when S1 is *released*. Same for S2, but in the opposite direction if S2 is pressed. The status of both motors is always the same.

All motor speed control should be accomplished using *PWM signals* generated by `Timer_A0`. The speed of the motors should be increased by 33.3% each time a switch is pressed. That is, the duty cycle of the PWM signals should be varied from 33.3% to 100% in 3 steps (33.3%, 66.6% and 100%) to control the motors' speeds while the motors are spinning in either direction. The speed/duty cycle should roll over by setting back to 33% after the 100% level. The motors speeds are consistent in both directions, that is if the motors are spinning with 33% speed in one direction, when the switch is released and the other switch is pressed, they should start spinning with 66% speed in the other direction.

More details about hardware and software requirements will be discussed in the next sections.

4.4 Hardware

4.4.1 Brushed DC Motor

The brushed DC motor is a type of motor that operates on DC voltage and current. It is bidirectional, meaning if the supply terminals are swapped the motor turns in the opposite direction. An H-bridge helps in achieving this functionality without physically having to swap the terminals.

The DC motor assemblies (with gearbox and encoder already installed) used by the TI-RSLK-Mechkit are manufactured by Pololu. Specifications for this motor can be found at: https://www.pololu.com/product/1520. The motor is capable of high speeds, with a rated output speed of 150 rpm.

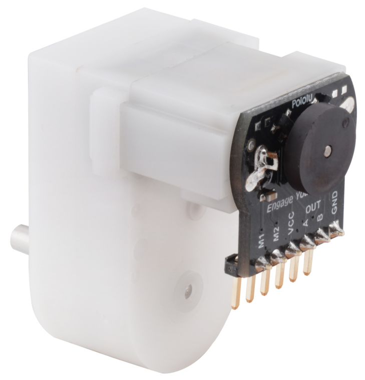

Figure 2: DC motor with encoder assembly (Image courtesy of Pololu).

For this lab assignment, you will manually connect the two terminals (labeled as M1 and M2) of each DC motor to the H-bridge motor driver circuit. The other 4 pins of the motor are encoder signals, which are directly coonected to the LaunchPad through the TI-RSLK chassis board and will be used for grading purposes.

4.4.2 H-bridge IC

The H-bridge IC that we use for this lab is the TI SN754410, the same as Lab 2. SN754410 is a quadruple half H-bridge driver IC. It has four half H-bridges that can be used either independently for motors with unidirectional applications, or in pairs for motors with bidirectional applications. The datasheet for SN754410 can be found at: http://www.ti.com/lit/ds/symlink/sn754410.pdf

We will be using all four channels of the H-bridge driver IC, channels 1 and 2 as a pair to control the right motor (MR), and channels 3 and 4 to control the left motor (ML) of the

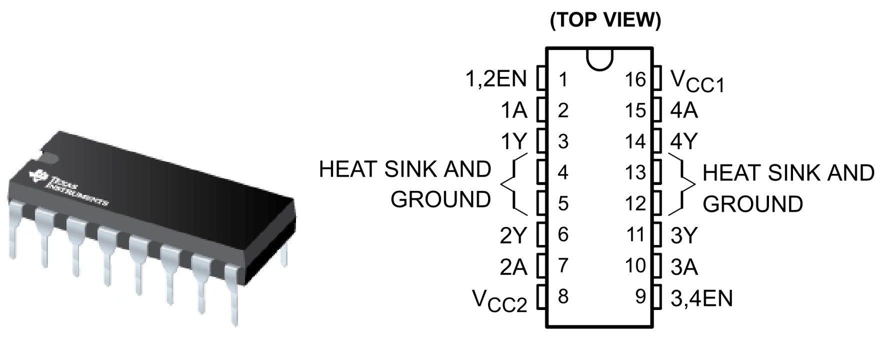

Figure 3: SN754410 pinout (Image courtesy of Texas Instruments).

TI-RSLK-Mechkit robot. This setup forms two full H-bridges which enables bidirectional operation of two DC motors independent from each other.

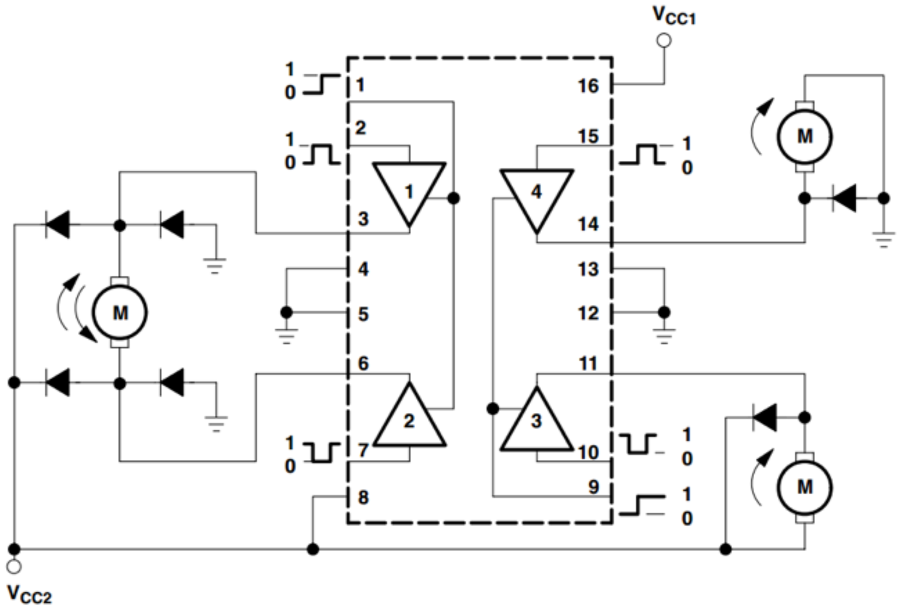

Figure 4: SN754410 functional block diagram (Image courtesy of Texas Instruments).

Figure 4 shows the possible wiring configurations that can be used with the SN754410 H-bridge IC. Note that in the figure, the interfaces at ports 3 and 4 are shown for unidirectional operation of two motors. The left side of the diagram (ports 1 and 2) shows the wiring schematic for bidirectional operation of a motor.

Note: For this lab assignment, we will be using **bidirectional configuration** for both sides of SN754410 in order to have bidirectional control of two DC motors.

Note that the SN754410 IC comes in a PDIP package, and has an identifier to help you locate Pin 1 on the chip: there is a Half Circle or Notch on the end of the chip, Pin 1 is to the left of the notch (Figure 3). IC pins are numbered in a counter clockwise fashion from Pin 1.

The pins of the SN754410 H-bridge IC are described in Table 1.

Pin Number	**Pin Name**	**Pin Function**	**MSP432 Connection**
1	1,2EN	Enable signal for driver channels 1 and 2 (active high input)	P1.6
2	1A	Input PWM signal for driver channel 1	P2.4
3	1Y	Output signal to the right motor for driver channel 1	
4	Heat Sink/GND	Connect to Ground	
5	Heat Sink/GND	Connect to Ground	
6	2Y	Output signal to the right motor for driver channel 2	
7	2A	Input PWM signal for driver channel 2	P2.5
8	VCC2	Power supply for motors (connect to VSW = 7.2V to 9V)	
9	3,4EN	Enable signal for driver channels 3 and 4 (active high input)	P1.7
10	3A	Input PWM signal for driver channel 3	P2.6
11	3Y	Output signal to the left motor for driver channel 3	
12	Heat Sink/GND	Connect to Ground	
13	Heat Sink/GND	Connect to Ground	
14	4Y	Output signal to the left motor for driver channel 4	
15	4A	Input PWM signal for driver channel 4	P2.7
16	VCC1	Power supply for internal logic (connect to 5V)	

Table 1: SN754410 Pins Description

To understand exactly what the input control PWM signals do, refer to Figure 4. Setting pin 2 of SN754410 high and simultaneously setting pin 7 of SN754410 low will turn the motor (attached to the output pins 3 and 6) in one direction. To turn the motor in the other direction, pin 2 should be low and pin 7 of SN754410 should be high. Pins 10 and 15 have the same function for the second DC motor attached to the output pins 11 and 14. Note that these input pins can also accept PWM signals from the MSP432. In order to operate the motor in either direction, a PWM signal should be applied to one of these two pins while maintaining the other pin low.

The enable signals (pin 1 and pin 9) are controlled via MSP432 GPIOs P1.6 and P1.7. Setting each enable pin to high will enable the associated driver channels. To make sure that the driver channels are disabled when the MSP432 is off or not running, we connect the enable pins to GND via pull-down resistors, as will be shown in the circuit schematic in Figure 7.

4.4.3 Diodes

Diodes are semiconductor devices used to control the direction of current flow. They are used often in mechatronic devices. The 1N5819 diodes provided with the "TI-RSLK-Mechkit" will be used as flyback diodes for the motor driver. Flyback diodes provide alternative path for current in motor coil and mitigates arcing/excessive current through transistors, to avoid damaging the transistors.

As shown in Figure 5, the silver line on 1N5819 diode indicates the cathode pin, which matches the vertical line in the diode circuit symbol.

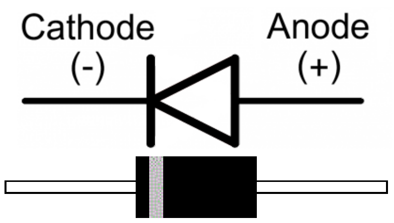

Figure 5: The diode circuit symbol, and the physical shape.

4.4.4 400-Point Breadboard

A solderless breadboard is provided with the TI-RSLK-Mechkit. Solderless breadboards are commonly used for prototyping because they allow you to quickly build temporary circuits without soldering.

The pins are arranged in groups of 5. The 5 pins in each group are electrically connected to each other at the back of the board. You connect leads and cables together by inserting them into pins within the same group.

Power rails at the top and bottom are marked with red (+) and blue (−) stripes. The groups in each rail are electrically connected along the entire length of the stripe. The remaining 5-pin groups on the board are labeled with numbers and letters. Each group is electrically isolated from the others.

The gap at the center of the breadboard allows easy connection of electronic components provided as dual-inline packages, such as the SN754410 IC.

For best results, use solid wires when breadboarding; you will find a pre-cut jumper wire kit in the TI-RSLK-Mechkit.

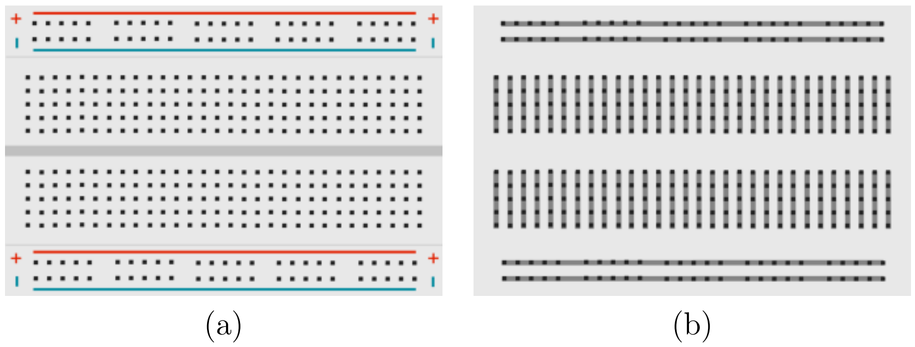

Figure 6: (a) Front of solderless breadboard showing power rails and connection pins. (b) Interconnections at back of board (normally hidden). The 5-pin groups in each power rail are interconnected. All other 5-pin groups are isolated (Image courtesy of Texas Instruments).

4.5 Circuit Schematic

Figure 7 shows the circuit we use to interface the two brushed DC motors of the TI-RSLK-Mechkit robot with the MSP432 LaunchPad. You have already assembled part of this circuit on the breadboard in Lab Assignment 2.

The motors are powered via the six AA batteries installed on the robot, which provide a voltage in the range of 7.2V to 9V through the VSW pin of the TI-RSLK chassis board. We recommend to connect the VSW test point on the TI-RSLK chassis board to the (+) power rail of the breadboard using an alligator clip, and then use the power rail for all the pins connected to VSW. Similarly, the GND pin of the LaunchPad can be connected to the (−) power rail of the breadboard using a Male-Female jumper wire, so that all the pins connected to GND can share that power rail. The +5V required for the VCC1 pin of SN754410 (pin 16) can be directly taken from the 5V pin of the LaunchPad using another Male-Female jumper wire.

The flyback diodes provide paths for current to flow when the drivers are switched off. Pay extra attention to the direction of the diodes when implementing them on the breadboard.

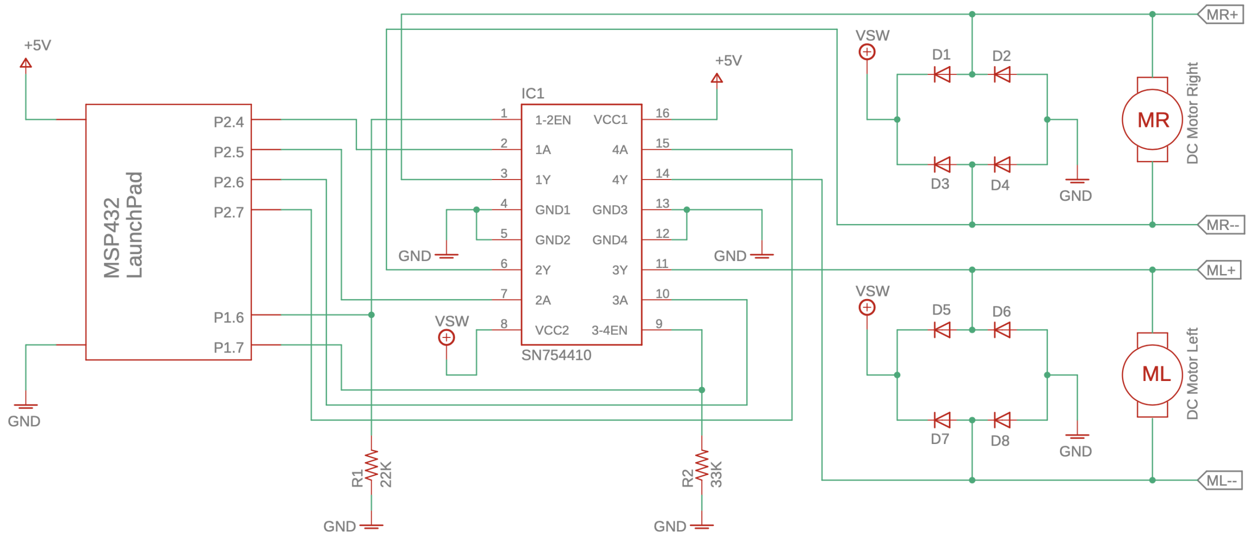

Figure 7: Circuit diagram for the H-bridge interface with motor.

As discussed in the previous section, the DC motors are connected to the SN754410 motor driver via the output pins 3, 6 and 11, 14. The PWM signals should be generated by `Timer_A0` channels of MSP432, specifically P2.4, P2.5, P2.6 and P2.7.

Two pull-down resistors ($22K\Omega$ and $33K\Omega$[1]) are connected to the SN754410 enable pins (pin 1 and pin 9), in order to keep the logic level low until the MSP432 LaunchPad is up and running These enable signals are active-high, and are controlled through software via P1.6 and P1.7 GPIOs of the MSP432 LaunchPad.

Note: The circuit shown in Figure 7 is the same as part of the circuit built in Lab 2.

[1]Resistor color codes: red-red-orange ($22K\Omega$), orange-orange-orange ($33K\Omega$)

Note: Remember to connect a common ground between the LaunchPad and the driver circuit.

Note: To use the MR+, MR–, ML+, ML– and VSW signals, you need to use the test points and the alligator clips provided with the "TI-RSLK-Mechkit". For more details on how to connect the test points and alligator clips, please see the Robot Assembly Manual, available under Course Resources on edX (Robot-Assembly-Manual.pdf).

Warning: While driving the motors using the H-bridge circuit, continue to check the temperature of the MSP432. If it is getting hot, disconnect the LaunchPad immediately and debug the circuit.

4.6 Software Architecture

Note: To create a new CCS project for the MSP432P401R target device, and include the `mechrev.h` header file and the DriverLib library in the project, follow the procedure described in Lab Assignment 1. The `mechrev.h` file and the `driverlib` folder are available under Course Resources on edX (mechrev-header.zip and driverlib.zip).
To get you started with designing the program architecture, a code template for Lab 4 is available under Course Resources on edX (lab-code-templates.zip).

After disabling the watchdog timer by calling the `WDT_A_holdTimer()` function to avoid unnecessary watchdog timer time-out interrupts, call the `mechrev_setup()` function provided by the `mechrev.h` header file to bypass the motor drivers of the TI-RSLK chassis board. Calling this function will also enable the encoder readings, which will be used only for grading purposes.

Warning: Make sure that you have called `mechrev_setup()` function to disable the built-in motor drivers on the TI-RSLK chassis board. Not disabling the default drivers will cause a conflict with the SN754410 motor driver on the breadboard and will potentially damage the chassis board.

Initialize the P1.1 and P1.4 GPIOs (connected to the S1 and S2 switches of the LaunchPad) as inputs with pull-up resistors.

Initialize the P1.6 and P1.7 GPIOs as output pins, used for enabling the H-bridge motor driver IC. We recommend to initialize these outputs to HIGH, in order to enable the driver channels while the program is running.

Initialize all four channels of `Timer_A0` module to generate the four PWM signals required for two motors: two PWM signal for each motor, one for each direction. That is, since `Timer_A0` is used for PWM generation, the channels `A0.1` (P2.4), `A0.2` (P2.5), `A0.3` (P2.6) and `A0.4` (P2.7) should be used to generate four PWM signals with different duty cycles. Make sure that `Timer_A0` channels are initialized to 0% duty cycle at the beginning.

As you learned in Module 4, after initializing the `Timer_A0` to generate the PWM signals, the duty cycle of a PWM signal can be changed by only changing the value of the `TAxCCRn` register associated to that timer channel. For example, to change the duty cycle of channel `A0.1`, we only need to change the value of `TA0CCR1` register (i.e. `x=0, n=1`).

Note: You can use a voltmeter, an oscilloscope or a logic analyzer to verify PWM signals and H-bridge driver circuit outputs.

Note: You can use the multi-colored LED on the MSP432 LaunchPad for debugging purposes, and to indicate and differentiate the three different levels of motors speed.

Design your code inside the main `while(1)` loop to detect whether the S1 and S2 switches are pressed or not. According to the status of the switches, adjust the duty cycle of the `Timer_A0` channels to change the motors' speed and direction, as explained in the Problem Statement in Section 4.3.

Note: You could use either "polling" or "interrupts" for handling S1 and S2 pushbuttons.

To turn a motor in one direction, set the duty cycle of one of its input channels to a level higher than 0%, and set the other duty cycle to 0%. Make sure that the SN754410 enable pins are set to high. To change the motor direction, set the duty cycle of its first input channel to 0% and the other channel to a level higher than 0%. To stop a motor, set both duty cyles to 0%. An example code snippet is provided below.

```
/* Turn on left and right motors with half speed */
/* (assuming the Timer A0 period is set to 3000 clock cycles during the
   initialization) */
TA0CCR1 = 1500; // set duty cycle for TA0.1 (P2.4) to 50%
TA0CCR2 = 0;    // set duty cycle for TA0.2 (P2.5) to 0%
TA0CCR3 = 1500; // set duty cycle for TA0.3 (P2.6) to 50%
TA0CCR4 = 0;    // set duty cycle for TA0.4 (P2.7) to 0%
// call the grading macro here when generating the grading outputs:
MACRO_LAB4_EVENT(); // see Section 4.9
```

Never set both PWM channels connected to a motor to duty cycles higher than 0% at the same time, as putting two simultaneous PWM signals through the motor in opposite directions may damage it.

Warning: A DC motor should never receive PWM signals of opposite spin directions at the same time. This could potentially cause the motor to burn out. Therefore, before you connect the motor terminals to the circuit, verify the functionality of using S1 and S2 as ON/OFF switches to ensure that each motor receives only one PWM signal at a time. Only when you are confident of the functionality of your code should you connect the terminals of the DC motor.

Note: You do not need to write software for processing the rotary encoders data. The encoders are used only for grading purposes, and they are handled via the `mechrev.h` header file provided with the lab assignment.

4.7 Expected Performance

Press and hold the switch S1 of the LaunchPad to start spinning both motors in one direction with 33.3% speed level. Then release S1 to stop spinning the motors. Now, press and hold S1 again to start spinning the motors with 66.6% speed level. Release S1 to stop. Continue this procedure for a few times to run the motors with full speed (100%), then back to 33.3%, and so on.

Now, press S2 to start spinning the motors in the opposite direction. Once again, releasing and holding S2 should allow you to vary the motors speed throughout their speed range from 33.3% to maximum. Continue this procedure to go through the speed cycles for a few times. Release S2 to stop driving the motors.

4.8 Troubleshooting

The chassis board or the LaunchPad or the SN754410 IC get hot:

– Double check the power (VSW and 5V) and GND wires connected to the breadboard.
– Verify the direction of flyback diodes.

Motors not do spin or get hot:

– Remove power and double check the connections.
– Recharge the batteries.
– Verify the six signals from the LaunchPad to the motor driver breadboard using a voltmeter, an oscilloscope or a logic analyzer.

One motor spins faster than the other:

– It is normal for the motor speeds to be ±20% of each other.
– Check for friction on the slower motor.

A pushbutton switch does not work:

– Verify that you have initialized the GPIOs and their internal pull-up resistors correctly.
– Set breakpoints using the CCS debugger and verify that the application goes into the polling `if()` statements when pressing a button.

4.9 Grading

An output text file with the file extension `.txt` should be uploaded to Vocareum in order for your lab assignment to be graded.

You can generate the grading output text file by the following procedure:

- Make sure that the `mechrev.h` header file is included in your main code.
- You must call `mechrev_setup()` during the initialization of your code (before the main `while(1)` loop). This will disable the chassis board motor drivers, and set up the encoders for the DC motors which are used for grading the lab assignment.
- You must also call `MACRO_LAB4_INIT()` after initializing all the GPIOs, Timers and other peripherals in your code (right before the main `while(1)` loop).
- You must call `MACRO_LAB4_EVENT()` right after detecting a switch press and adjusting the associated PWM signals. For example, if using `TA0CCRx` registers to change the PWM duty cycles, you need to call the macro after setting all four `TA0CCRx` registers for the motors. Note that if you are changing more than one PWM duty cycle at some part of the code, you should call the macro only once after changing all the PWMs. You can keep calling the macro while the switch is pressed. DO NOT call the macro when detecting a switch release and setting the PWM duty cycles to 0%.
- After finalizing your program and placing the provided macros in the code, generate the grading output file by following the steps below:
  - Connect the USB debug cable to the MSP432 LaunchPad.
  - Turn the robot on by pressing the Power button on the TI-RSLK chassis board. Make sure that the blue LED on the chassis board is turned on.
  - Start the debug mode in the Code Composer Studio. The code will be paused at the beginning of the main loop.
  - Press the Resume (F8) button in CCS to start debugging the code.
  - Test your code by pressing, holding and releasing the switch S1 for at least 15 times (make sure to hold the button for at least 3 seconds each time). The grading outputs should start printing in the CCS console. Repeat the same process for the switch S2 to collect more output data: press, hold and release the switch S2 for at least 15 times (make sure to hold the button for at least 3 seconds each time).
  - Stop the debug mode. Select all the outputs from the CCS console, copy them and paste them into an empty text file. Save the text file with an optional name (with file extension `.txt`) and submit it to Vocareum.

Warning: Make sure that the extension of your output text file is "`.txt`". Vocareum cannot open other file types, and would assign a zero grade to your work if the extension is other than `.txt`.

Warning: You can submit the lab assignment to Vocareum for up to 5 times. When *resubmitting* an assignment, make sure to first delete the old output file in the workspace (listed under the "work" directory) and then upload the new file, or use the same name for the new output file so it replaces the old one in the workspace.
DO NOT press the Submit button if there are more than 1 text file uploaded to the workspace, otherwise Vocareum would not be able to open the text file and would assign a zero grade to your work.

For more details on how to include header files, save data as text files, and submit the files to Vocareum, please refer to the Labs Helper Guide, available under Course Resources on edX (Labs-Helper-Guide.pdf).

4.10 Optional Module (Non-graded)

Add a servo motor to your design and control the direction of the servo motor with the LaunchPad switches, similar to the DC motors. The servo motor can be found at:
`https://www.pololu.com/product/2818`

Servo motors are controlled by sending a PWM signal through the signal wire. The frequency of the control signal should be 50Hz (pulse period of 20ms). The servos can usually rotate 180 degrees (they have a physical limits of travel), and the width of pulse determines angular position of the servo. Generally pulses with 1ms duration correspond to 0 degrees position, 1.5ms duration to 90 degrees and 2ms to 180 degrees. Though the minimum and maximum duration of the pulses can sometimes vary with different brands and they can be 0.5ms for 0 degrees and 2.5ms for 180 degrees position.

You can use another Timer A module (e.g. Timer `A2`) to generate an additional PWM signal. The PWM signal should have frequency of 50Hz , and the duty cycle should be varied from 5% to 10% (1ms to 2ms). The change in the duty cycle and the direction can be based on S1 and S2 switches.

Figure 8: Servo motor (Image courtesy of Pololu).