TI-RSLK MAX

Texas Instruments Robotics System Learning Kit

TEXAS INSTRUMENTS

Module 12

Lab 12: DC motors

Lab: DC motors

12.0 Objectives

The purpose of this lab is to build the electronics needed to spin the motors. The TI-RSLK chassis board includes an H-bridge motor driver using the TI DRV8838 driver that allows the software to spin each motor forward or backward. The software can vary the **electrical power** delivered to each motor using **pulse width modulation** (PWM). In this module,

1. You will learn the electromagnetic aspects of the motor.
2. You will use the TI-RSLK chassis board to interface the motors to the microcontroller.
3. You will measure the voltage and current to the motors.
4. You will perform an analysis of the behavior of the motor, plotting motor speed versus duty cycle.

Good to Know: Even though you will measure motor speed as a function of duty cycle, this relationship depends on many factors that can change over time, such as motor efficiency, battery voltage, voltage drop in the H-bridge, mechanical forces, and friction. For all practical purposes, without sensors, the software can only choose to go faster or to go slower, but it cannot set the motor speed. On this robot, there are two motors in differential drive configuration. This means even the simplest operation such as moving in a straight line will require sensor feedback. There are three such sensors available in this course: the line sensor (Module 6), the IR distance sensors (Module 15), and the tachometer (Module 16).

12.1 Getting Started

12.1.1 Software Starter Projects

Look at these two projects:
Lab09_SysTick (your solution to Lab 9)
Lab12_Motors (starter project for this lab)

Note: Please do not use the voltmeter, oscilloscope or logic analyzer created by TExaS for this lab. Voltages applied to inputs of the MSP432 must remain between 0 and 3.3V. Voltages outside this range will damage the MSP432.

12.1.2 Student Resources (in datasheet directory)

ti-rslk-chassis-board-v1.0-schematic.pdf	Circuit diagram for TI-RSLK board
drv8838.pdf	Data sheet for the H-bridge driver

12.1.3 Reading Materials

Chapter 12, “Embedded Systems: Introduction to Robotics"

12.1.4 Components needed for this lab

All components needed for this lab are included in the TI-RSLK Max kit (TIRSLK-EVM). Batteries will be needed to power your robot.

Quantity	Description	Manufacturer	Mfg P/N
1	TI-RSLK MAX kit	TI	TIRSLK-EVM

12.1.5 Lab equipment needed

Oscilloscope (one or two channels at least 10 kHz sampling)
Voltmeter, ohmmeter, and current meter

12.2 System Design Requirements

The goal of this lab is write software to activate the two motors on the robot. The TI-RSLK chassis board (**main board**) used in Module 5 lab also includes two H-bridge drivers (TI DRV8838) that provide the voltage and current needed to spin the motors.

First, you will use the PWM software from Lab 9 to adjust the delivered power to the two wheels.

The second part of this lab is to study the behavior of the motor. You will measure voltage (volts), current (amps), average power (watts), and rotational speed (rpm) of the DC motor as a function of duty cycle.

The outcome of this lab is to build a system that drives in more or less a straight line until one of the bump sensors detects a collision.

12.3 Experiment set-up

Details on how to connect the chassis, TI-RSLK chassis board, motors, and wheels were presented as part of Lab 5, see Figure 1. The first step is to review the circuit diagram for the TI-RSLK chassis board and compare it to the simplified version drawn as Figure 2. Look up the data sheet for the TI DRV8838 for operation and details, and determine current path from +BAT, through the

Lab: DC motors

DRV8838, across the motor, through the DRV8838 again, and then back to – BAT (ground). Note: If you stall your motor you will draw a large current which could damage your motor and TI-RSLK chassis board. See datasheet for the motor assembly on www.Pololu.com.

Warning: Please remove the +5V jumper on the MSP432 LaunchPad. Not removing this jumper will cause permanent damage to the LaunchPad and the TI-RSLK chassis board.

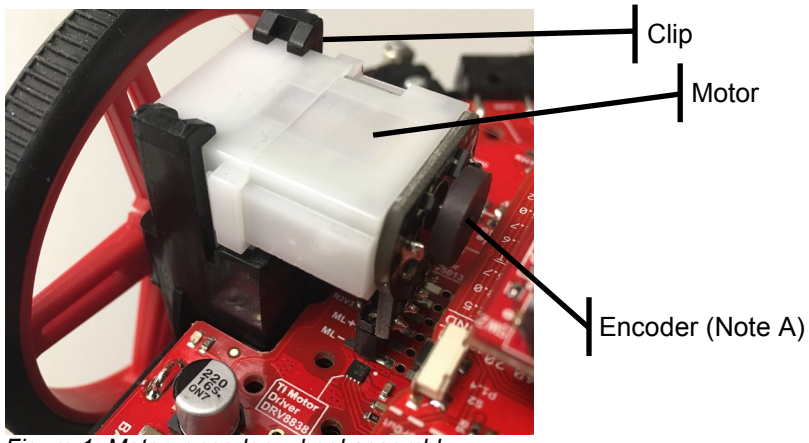

Figure 1. Motor, encoder, wheel assembly.

LaunchPad	TI-RSLK chassis board	DRV8838	Description
P5.5	DIRR	PH	Right Motor Direction
P3.6	nSLPR	nSLEEP	Right Motor Sleep
P2.6	PWMR	EN	Right Motor PWM
P5.4	DIRL	PH	Left Motor Direction
P3.7	nSLPL	nSLEEP	Left Motor Sleep
P2.7	PWML	EN	Left Motor PWM

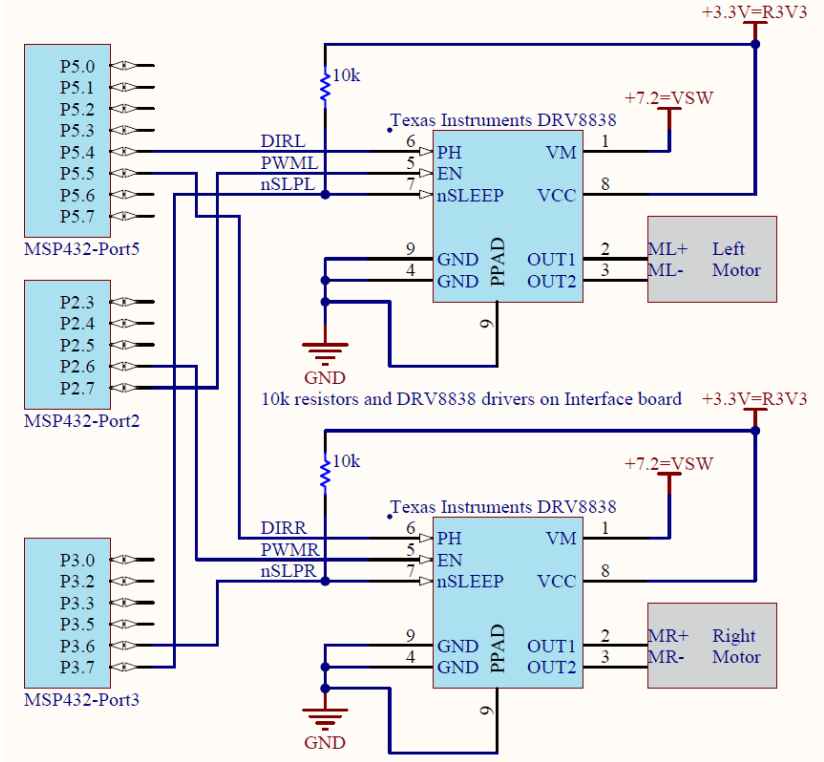

Figure 2. Motor interface circuit.

PH	EN	nSleep	Motor
0	0	1	Stop
1	0	1	Stop
0	1	1	Forward
1	1	1	Back

To go forward, set nSleep=1, PH=0, and activate EN with a PWM signal.

12.4 System Development Plan

12.4.1 Low-level software driver

You will start with creating a suite of software functions that control the two wheels on the robot. The frequency of the PWM signal sent to both motors should be 100 Hz (10ms). In this lab, we will keep the duty cycle the same for both motors as well. In the next module, we will use the hardware timer so each motor will have its own duty cycle. To stop the motors you will stop the PWM and set the **nSleep** signal to 0. Use the simple approach of Lab 9 to create the PWM signals. The prototypes for the driver are:

void Motor_InitSimple(void);
Initializes the 6 GPIO lines and puts DRV8838 drivers to sleep
Returns right away

void Motor_StopSimple(void);
Stops both motors, puts DRV8838 drivers to sleep
Returns right away

void Motor_ForwardSimple(uint16_t duty, uint32_t time)
Drives both motors forward at **duty** (100 to 9900)
Runs for **time** duration (units=10ms), and then stops
Stop the motors and return if any bumper switch is active
Returns after **time***10ms or if a bumper switch is hit

void Motor_BackwardSimple(uint16_t duty, uint32_t time)
Drives both motors backward at **duty** (100 to 9900)
Runs for **time** duration (units=10ms), and then stops
Stop the motors and return if any bumper switch is active
Returns after **time***10ms or if a bumper switch is hit

void Motor_LeftSimple(uint16_t duty, uint32_t time)
Drives just the left motor forward at **duty** (100 to 9900)
Right motor is stopped (sleeping)
Runs for **time** duration (units=10ms), and then stops
Stop the motor and return if any bumper switch is active
Returns after **time***10ms or if a bumper switch is hit

void Motor_RightSimple(uint16_t duty, uint32_t time)
Drives just the right motor forward at **duty** (100 to 9900)
Left motor is stopped (sleeping)
Runs for **time** duration (units=10ms), and then stops
Stop the motor and return if any bumper switch is active
Returns after **time***10ms or if a bumper switch is hit

12.4.2 Control of the motor

In this part of the lab you will implement the functions to test the motors. Place voltmeters on the VSW line (+7.2) and on 5V line (+5V) the first time you power up the wheeled robot. Place the robot on blocks, so the wheels do not touch the ground, and test the low-level motor functions, using a program like **Program12_1**. This allows the motors to spin without the robot moving. With the wheels off the ground, there will be minimal friction, the fastest rotation, and the smallest current.

```
// Driver test
void Pause(void){
  while(LaunchPad_Input()==0);  // wait for touch
  while(LaunchPad_Input());     // wait for release
}
int Program12_1(void){
  Clock_Init48MHz();
  LaunchPad_Init(); // built-in switches and LEDs
  Bump_Init();      // bump switches
  Motor_InitSimple();     // your function
  while(1){
    Pause();
    Motor_ForwardSimple(5000,2000);  // your function
    Pause();
    Motor_BackwardSimple(5000,2000); // your function
    Pause();
    Motor_LeftSimple(5000,2000);     // your function
    Pause();
    Motor_RightSimple(5000,2000);    // your function
  }
}
```

Use an oscilloscope to observe the motor signals motor board (MR+, MR-, ML+, ML-) during operation. *You should see voltage versus time. The voltage difference between MR+ and MR- is the applied voltage to the motor.*

Note: As mentioned in Lab 9, using software delays to create PWM consumes all of the processor time. In the next module, we will use the hardware timers on the microcontroller to create the two PWM outputs. In this way, software needs to execute only when it wishes to change the duty cycle or change direction.

Lab: DC motors

12.4.3 Behavior

From an electrical standpoint the motor has three components, resistance (caused by the long wires), inductance (caused by the coiled wires) and electro motive force (emf -voltage caused by the coupling between mechanical and electrical forces). Begin by measuring the resistance of the motor when all power is turned off and the motor is not spinning. Let **R** be this static resistance. Assuming a voltage of 7V, use Ohm's Law to calculate the expected current.

In this section, you will measure actual voltage (**V** in volts), current (**I** in amps), and speed (**s** in rpm) as a function of the **duty** parameter (2000 to 8000). If you place the robot on blocks and attach string/yard/tape to a wheel you can both see and hear the wheel turn. First you will use a stopwatch to count the number of rotations in a fixed time (e.g., 60 seconds).

There are two approaches to measuring motor **current** (**I**). One approach is to remove the batteries and connect a bench supply (which allows you to set the voltage to 7.2V and measure the current) to power the robot. The second approach is to place a current meter in the loop between the batteries and the robot. For example, you can tape one wire to the + side of a battery and tape another wire to the – side of a second battery, see Figure 3. You then place these taped ends together into the battery compartment of the TI-RSLK chassis.

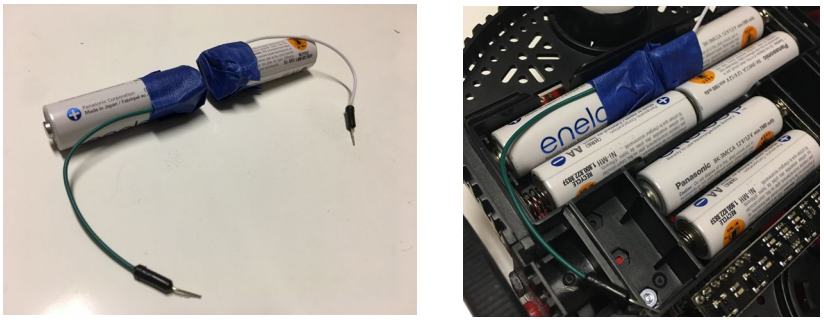

Figure 3. One mechanism to measure current.

Close the compartment. You then can place the current meter on the two wires. The current meter between the wires completes the battery circuit. You can measure motor **voltage** (**V**) with the oscilloscope and verify which duty cycle is active. You will first measure current to the robot with the motors stopped, and then you will measure voltage, current, speed required to spin one motor. The difference in these two current measurements is the current to the motor. You can use a program like **Program12_2** to collect data.

```
// Voltage current and speed as a function of duty cycle
int Program12_2(void){ uint16_t duty;
  Clock_Init48MHz();
  LaunchPad_Init();   // built-in switches and LEDs
  Bump_Init();        // bump switches
  Motor_InitSimple(); // initialization
  while(1){
    for(duty=2000; duty<=8000; duty=duty+2000){
      Motor_StopSimple();   // measure current
      Pause();
      Motor_LeftSimple(duty,6000);  // measure current
    }
  }
}
```

Make a table and graphs of voltage, current, power, emf, and speed as a function of duty cycle. Calculate **emf** as

emf = V – I*R

where **V** is the measured motor voltage, **I** is the measured motor current, and **R** is the static resistance of the motor. Under normal operating conditions, emf will be negative, meaning it draws more current than predicted using the static resistance. Calculate power as

P = V * I * duty/10000

Describe the general behavior of the motor.

Perform a maximum speed test using **Program12_3**. First measure the rotational speed of the motors when the robot is on blocks, and then repeat the measurement when the robot is on the ground.

```
int Program12_3(void){
  Clock_Init48MHz();
  LaunchPad_Init();   // built-in switches and LEDs
  Bump_Init();        // bump switches
  Motor_InitSimple(); // initialization
  while(1){
    Pause();
    Motor_ForwardSimple (9900,1500); // max speed 15 s
  }
}
```

12.5 Troubleshooting

Motors not do spin or gets hot:

- Remove power and double check the connections.
- Review steps in Lab 5.
- Recharge the batteries.
- Verify the six signals from the LaunchPad to the motor board 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

12.6 Things to think about

In this section, we list thought questions to consider after completing this lab. These questions are meant to test your understanding of the concepts in this lab. The goal of this module is for you to experience voltage, current, and power as they relate to DC motors.

- How does friction affect motor current?
- In this lab, we do not set the speed or the current. Rather, we set just the voltage and duty cycle. Why is it difficult in this lab for the robot to go straight?
- How does the two H-bridges allow the robot to turn, to back up?
- How does the software adjust power delivered to the motors?
- In what two ways could software cause the robot to turn?

12.7 Additional challenges

In this section, we list additional activities you could do to further explore the concepts of this module. For example,

- If you do not have the TI-RSLK main board, you could build your own H-bridge circuits to control the motors on the robot. In particular, you could build two H-bridges described in lecture using the L293. If you build your own H-bridge please test it before attaching the motors and before attaching the microcontroller.
- An impossible challenge would be to try to write software that makes the robot travel in a square pattern. Basically, repeat this two-step process: 1) go straight for a fixed amount of time; 2) turn left 90 degrees. It will not be possible. However, it will be instructive to determine why the effort fails.

12.8 Which modules are next?

There are two major limitations to the robot conceived in this lab. 1) the software consumes all the processor time, and 2) the speed of the motors depends on many factors most of which cannot be predicted in advance. Over the remaining labs we will solve these limitations.

Module 13) Use timers to create PWM signals, and use interrupts to manage multiple software tasks
Module 15) Use the ADC to interface distance sensors. Two distance sensors can be used to drive the robot at a fixed distance and fixed angle to the wall.
Module 16) Interface tachometers (Romi Encoder Pair Kit) and use timer capture to measure the speeds of each wheel directly.
Module 17) You can combine modules 12, 13, and 16 to create a control system that does spin the motors at a desired speed.

12.9 Things you should have learned

In this section, we review the important concepts you should have learned in this module:

- Understand voltage, current, and power to a motor.
- Be able to use PWM output to adjust power to the motors.
- Understand basic operation and purpose of an H-bridge.
- Know how to write and test a low-level software driver.

ti.com/rslk

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265