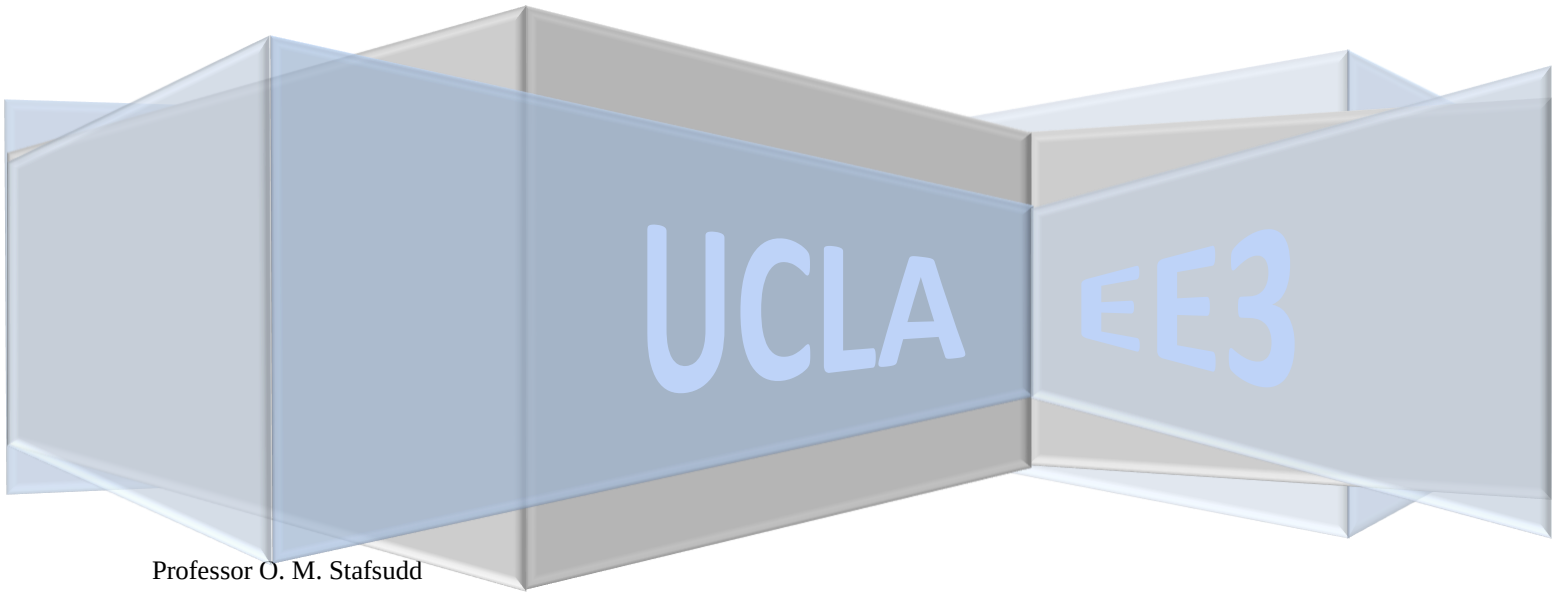
EE3 Introduction to Electrical Engineering

LABORATORY MANUAL



Professor O. M. Stafsudd
Department of Electrical Engineering
October 2013
Rev. September 2016 (version 1.9.3) Rev. October 2019 (version 1.10; Dr. Dennis M. Briggs)
Rev. September 2020 (version 1.11; Dr. Dennis M. Briggs)

Acknowledgements

I would like to acknowledge and thank the many people who have contributed to the re-design and development of EE3. In particular, the Chair of the Department of Engineering, Professor Frank Chang was instrumental in providing the resources of staff and funding necessary to develop the new course.

Professor Greg Pottie developed the new lecture format and content of the course and greatly assisted in the development of the laboratory part.

Professor William Kaiser provided valuable suggestions on the use of the NI myDAQ device that has been included in support of the laboratory portion of this course.

Dr. Michael Briggs is thanked for his generous insight into the changes that needed to be made to improve the program based on his experience with the previous EE3.

Previous students of EE3 gave us many suggestions that we have attempted to include in the re-design of this important "Introduction to Electrical Engineering."

Finally, I thank the Elenco Corporation for the generous use of their copyrighted materials.

-Prof Oscar Stafsudd, 2012

Student Albert Liu contributed much to Version 1.9, both in providing constructive ideas and in seeing that the changes were implemented.

-Dr. Michael Briggs, 2015

TA Xin Li has contributed much to Version 1.10.1, especially in Lab Experiments 2 and 3.

-Dr. Michael Briggs, 2019

Student Arhison Bharathan contributed much to Version 1.11.

-Dr. Michael Briggs, 2020

Week 4: RSLK Setup and Introduction

Week 4 Prelab

1. **Introduction:** This lab introduces you to the RSLK, the Texas Instruments Robot System Learning Kit. You will be looking it over and performing some simple but revealing tasks to familiarize you to the system and give you a head start on the project.

2. **Arduino Installation:**

You will use a software product called Arduino. Arduino development environment works with the TI MSP432 microcontroller. The MSP432 LaunchPad is a development kit including the MSP432. It is mounted to the RSLK that you will be using for the project. Download and install the required Arduino IDE software before the lab. You will be submitting to CCLE the proof of your installation. The latest software version of Arduino can be found online at:

https://www.arduino.cc/en/software

FOR WINDOWS USERS: After you downloaded the Arduino IDE (should appear as a zip file and after you unzip the file you can directly use the software interface without any further installation steps), please go to CCLE under "Site info > Project Related Materials" and download the file "energia-drivers.zip". Unzip the file and install these additional drivers using either the 32 bit/64 bit installation wizard.

You will also need to update the drivers to include the MSP432. This update will take about 15 minutes. Here is how to do that: (reference: https://www.hackster.io/measley2/robotics-system-workshop-arduino-programming-on-ti-rslk-max-d33faa)

(1) Add the URL http://s3.amazonaws.com/energiaUS/packages/package_energia_index.json to your File > Preferences > Additional Board Manager URLs. Also check the box for show line numbers if you like that and adjust your font size if you would like it bigger.
(2) Go to Tools > Boards > Boards Manager and scroll to the bottom of the list with Energia MSP432 and install 5.29.1. Note: this step takes a few minutes, leave time for the install between 10-20 minutes.
(3) After the install completes, You need to select the board and COM port in Arduino IDE. Go to Tool > Boards and now you should see the "Energia MSP432 Red Boards" > "Red LaunchPad MSP432P401R EMT" and make sure this is selected.

3. Take a screen shot of your installation and submit it to the proper Pre-Lab slot on CCLE.

4. In lab section this week, you will use Energia to upload your first sketches (i.e. programs) to the RSLK. Read through them to get an idea of what you'll be doing:

Week 4 Section – First Project Day Instructions

1. Start with no batteries in RSLK.
2. Launch Energia.
3. Tools | Board | Boards Manager, scroll down to Energia MSP432 EMT RED boards. Check version #; should be 5.6.3 and have "INSTALLED" next to the version number.
4. insert USB cable, check for green LED.
5. Remove USB cable.
6. Push the slide switch on the rear edge of the Chassis Board to OFF. It will stay in that position for the remainder of the course.
7. (VERY IMPORTANT: YOU WILL *KILL YOUR CAR* IF THIS JUMPER IS *NOT* REMOVED!) Be sure the 5 V jumper is missing; that is, there is a gap at the 5 V position. The gap may be shrouded under a piece of tape. Check to be sure that a gap exists at the 5 V position. If there is no tape, ask your TA to help you find the 5 V position.
8. Insert batteries.
9. Push the POWER pushbutton switch. Both the green and a blue LED should light up.
10. Push the POWER pushbutton switch again. Both LEDs should turn off.
11. insert USB cable.
12. File | Examples | 01.Basics | Blink
13. Click on right-facing arrow in upper left corner of window.
14. Blinky should compile and load successfully, red LED should start blinking.
15. In lines containing the `delay()` function, change the numbers in the arguments. NOTE: if Blink does not compile and shows an error message stating that `LED_BUILTIN` is not recognized, consult the LaunchPad with MSP432P401R Pin Map (scroll down in web page). Look for the "RED LED" entry in the table close to the lower left corner of the circuit board picture. Replace `LED_BUILTIN` with the corresponding pin number.
16. Click on right-facing arrow in upper left corner of window.
17. The red LED should blink at a different rate.
18. Make sure the Off-On slide switch in the lower left corner of the Chassis Board is in the "Off" position.
19. Push the Power pushbutton switch in the lower left corner of the Chassis Board. Check for both blue and green LEDs.
20. Download Basic Code text file from CCLE, Site Info | Project Materials, store in some appropriate place.
21. In Energia, File | New; Delete existing code; Paste Basic Code into the sketch.
22. Pick up the RSLK so the wheels are off the tabletop. Click on the right-pointing arrow. After compile and download, one wheel should turn, one yellow LED on right front of Chassis Board should flash.
23. Power down the Chassis Board by pushing the Power button again.
24. Download the MSP432 Pinchart on CCLE, Site Info | Project Materials.
25. Note the pin numbers for direction, non-sleep (nslp), and PWM.
26. Modify the Basic Code to make the RSLK do a doughnut (spin like a top). To do this, the wheels must spin in opposite directions.
27. Download ECE3.zip from CCLE, Site Info | Project Materials. Do NOT unzip the zip file.
28. Energia | Sketch | Include Library | Add .ZIP Library … navigate to ECE3.zip and select.
29. Sketch | Include Library; ECE3 Library should appear on pulldown list
30. Navigate to ~/Documents/Energia/libraries/ECE3/examples/IR_Sensor_Example; double-click IR_Sensor_Example.ino. This opens IR_Sensor_Example in Energia.
31. Navigate to ~/Documents/Energia/libraries/ECE3/src/ECE3.cpp. Open ECE3.cpp with a text editor.
32. After `QTRSensors IR` (**IF NOT ALREADY 13 & 12**), paste
```
#define P5_0 13
#define P5_2 12
```
33. Save
34. Power up the Chassis Board.
35. Compile and download to the RSLK.
36. Tools | Serial Monitor should show 8 columns of numbers from ~500 to ~2000, depending on amount of reflection from surface. If you place the car on a white sheet of paper, all 8 columns should be in the

500-800 range. If you pick up the car and hold it over the floor at desktop height, all 8 columns should read ~2500.

Week 4 Prelab End

RSLK Inspection

You will be led on a walk-around of the RSLK in order to familiarize you with its components and their locations. The components and procedures to be covered in this section are:

- The MSP432 LaunchPad
- Wheels and motors
- Speed sensors
- Path sensors
- Battery compartment and battery insertion
- Four programmable LEDs
- Power on and off procedure
- Connecting to your laptop

Introduction to Energia

Energia is an Arduino-like development environment that enables one to create control and monitoring programs and download them to the LaunchPad. You will first execute the "Hello World" for a real time embedded microcontroller: blinking an LED. Then you will learn about:

- The Pin Chart
- The motor control commands
- The path sensing commands
- Serial print and plot

The Pin Chart

Below is the pin chart for the MSP432 microcontroller. When you wish to control a pin, or read the value of a pin, in your Energia code, use the pin numbers in the "Energia pin #" columns.

- Motor control pins:
    - Pins 29 & 30 control the rotation directions.
    - Pins 39 & 40 control the motor speeds.
    - Pins 11 & 31 turn the motors on and off (nSLP stands for "not Sleep").
- Chassis Board LED Control Pins (41, 51, 57, 58)
- Chassis Board Bump Switch Pins (6, 8, 24, 25, 27, 28)

Main headers J1-J4:

		Energia pin #	J1 1	J3 21	Energia pin #			Energia pin #	J4 40	J2 20	Energia pin #	
	CC2650/CC3100	1	3.3V	5V	21	CC2650/CC3100	PWML, Left Motor PWM	40	P2.7	GND	20	CC2650/CC3100
	CC2650	2	P6.0	GND	22	CC2650/CC3100	PWMR, Right Motor PWM	39	P2.6	P2.5	19	CC2650/CC3100
	CC2650/CC3100	3	P3.2	P6.1	23	Center IR Distance / OPT3101	PWM Arm Height Servo	38	P2.4	P3.0	18	CC3100, SPI_CS, GPIO
	CC2650/CC3100	4	P3.3	P4.0	24	Bump 0 [3]	CC3100, UART1_CTS	37	P5.6	P5.7	17	available GPIO? / OPT3101 RST?
	nHIB	5	P4.1	P4.2	25	Bump 1 [3]	CC3100, UART1_RTS	36	P6.6	IRST	16	CC2650/CC3100
	Bump 2 [3]	6	P4.3	P4.4	26	TExaS scope input	CC2650	35	P6.7	P1.6	15	CC3100 SPI MOSI
	CC3100, SPI_CLK	7	P1.5	P4.5	27	Bump 3 [3]	CC3100, NWP_LOG_TX	34	P2.3	P1.7	14	CC3100 SPI MISO
	Bump 4 [3]	8	P4.6	P4.7	28	Bump 5 [3]	CC3100, WLAN_LOG_TX	33	P5.1	P5.0	13	ERB (3.3V) [1]
	UCB1SCL [4]	9	P6.5	P5.4	29	DIR_L	PWM Arm Tilt Servo	32	P3.5	P5.2	12	ELB (3.3V)[1]
	UCB1SDA [4]	10	P6.4	P5.5	30	DIR_R	nSLPL [2] / nSLPR [2]	31	P3.7	P3.6	11	PWM Gripper Servo

Notes:

[1] This is encoder output. Sever VPU=VREG jumper and connect VPU to 3.3V

[2] This disables a motor driver. 0 to sleep/stop. Sever VCCMD=VREG jumper and connect VCCMD to 3.3V. Consider severing nSLPL=nSLPR jumper.

[3] Use Port 4 for edge-triggered interrupts

[4] Primary I2C channel supported by Energia

Bump 0 is right side of robot, Bump 5 is left side

CTRL on the motor board is a power switch. A high pulse (>1V) turns on the switch; a low pulse turns off the switch and power to the microcontroller. Leave this pin floating (an input) for normal operation.

Yellow highlights changes from previous pin assignments

Red highlights changes from version 4

Grey is changes from version 5

Orange needs to verify with Jan if routing possible to combine nSLP to free up an additional PWM pin

J5:

					no energia														
	Yellow Front Right LED	Right IR Distance / OPT3101	Analog Arm Height Servo	Analog Gripper Servo	Reflectance LED illuminate (odd)	AUXR IR Distance / OPT3101	Reflectance 3	Reflectance 1	Nokia5110 CS	Nokia5110 CD	Yellow Front Left LED	Reflectance 4	Reflectance 6	UCB3CLK (not available in	UCB3SDA (not available in	ERA (3.3V)[1]			
Energia #	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	5V	3.3V	GND
	P8.5	P9.0	P8.4	P8.2	P9.2	P6.2	P7.3	P7.1	P9.4	P9.6	P8.0	P7.4	P7.6	P10.0	P10.2	P10.4	5V	3.3V	GND
	P8.6	P8.7	P9.1	P8.3	P5.3	P9.3	P6.3	P7.2	P7.0	P9.5	P9.7	P7.5	P7.7	P10.1	P10.3	P10.5	5V	3.3V	GND
Energia #	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	5V	3.3V	GND
	Red Back Left LED	Red Back Right LED	Left IR Distance / OPT3101	Analog Arm Tilt Servo	Reflectance LED illuminate (even)	Nokia5110 RST	AUXL IR Distance / OPT3101	Reflectance 2	Reflectance 0	Nokia5110 Clock	Nokia5110 MOSI	Reflectance 5	Reflectance 7	AUXC IR Distance / OPT3101	UCB3SCL (not available in	ELA (3.3V) [1]			

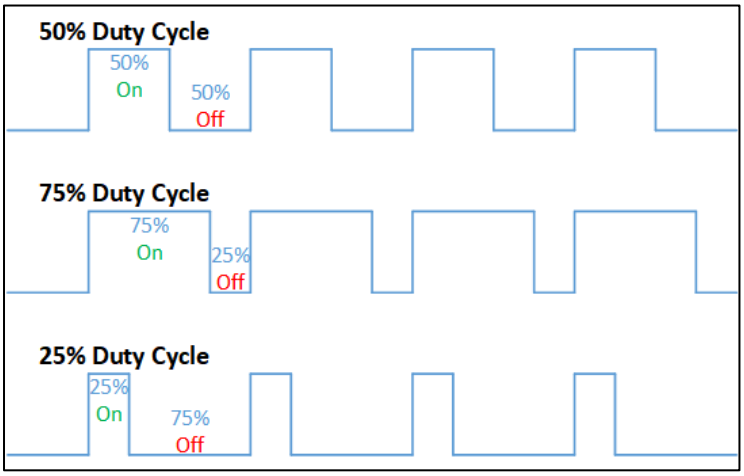
11/19/2018 changes from version 6, jan@pololu.com

2/11/2019 changes from rom05a02

PWM Based DC Motor Control

The RSLK uses a circuit called an H-bridge to control the direction of each of its two DC motors. View the **Lab 4 Appendix** for a detailed explanation. As said in that discussion, the H-Bridge can only dictate the direction the motor turns and whether it is on or off. To control the speed of a DC motor, the RSLK must also drive the H-bridge using a pulse-width modulated (PWM) signal.

A PWM signal has two states: high and low. The signal is also periodic, having the properties of period (measured in seconds) and duty cycle (a percentage from 0% to 100% inclusive). The duty cycle dictates for how much of a period the signal is high ((period * duty cycle) seconds) while for the rest of the period, the signal will be set low ((period * (1 - duty cycle)) seconds). Below we illustrate possible duty cycles:



https://en.wikipedia.org/wiki/Pulse-width_modulation

PWM signals are useful in cases where the average value of signal matters more than the instantaneous value. This is because you can easily vary a PWM signal's average value by adjusting the duty cycle. For example, one application of PWM is a dimmable LED. With a short enough period, the eye cannot distinguish when a PWM signal is on or off so a person only perceives the average intensity of the light. Therefore, a short duty cycle makes the LED appear dim and a long duty cycle makes the LED appear bright.

PWM – Duty Cycle

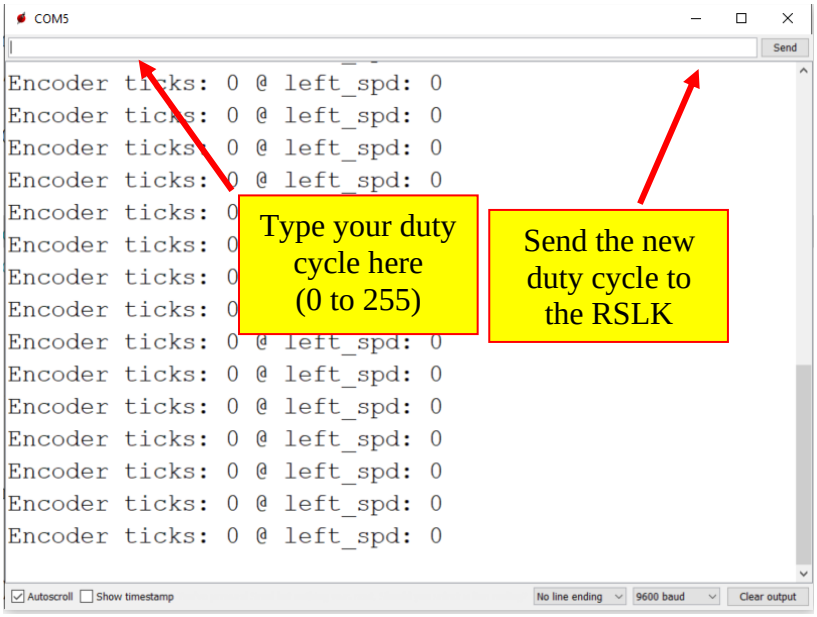
In this exercise, you will apply a PWM signal at varying duty cycles to the left DC motor. You will see how duty cycle changes the speed of the DC motor while also learning how to use the analogWrite function to create a PWM signal. You will be using this function in your own sketches.

The analogWrite(**pin**, **value**) function has two parameters: pin number (**pin**) and duty cycle (**value**). Use pin number to select the motor you would like to apply the PWM signal to (pins PWMR and PWML in

the chart above). Duty cycle accepts an integer from 0 to 255 where 0 implies a 0% duty cycle and 255 implies a 100% duty cycle.

Follow these steps to complete the exercise:

1. Download the sketch pwm_dutycycle.ino from CCLE. Launch Energia and open the sketch.
2. Read through the code to get an understanding of what it does. Ensure you understand how we used the encoder to calculate speed.
3. Hold the car in your hand or place it on a box so it does not move unexpectedly.
4. Plug in your car and open the **Serial Monitor** (in the top bar click Tools > Serial Monitor). **Set the Serial Monitor baud rate to 9600**.
5. Upload the sketch to your Launchpad. You should see "Encoder ticks: 0 @ left_spd: 0" being printed out continuously on the Serial Monitor.
6. Turn on your car (press the push button labeled "Power").
7. In the Serial Monitor, use the text box at the top to specify a duty cycle from 0 to 255 and click "Send". This will update the left_spd variable, turning the left DC motor. Use the following image:



Serial Monitor Window
(Note: your COM port may be different)

8. For each duty cycle listed below, calculate the correct **value** argument for the analogWrite(pin, value) function and apply the PWM signal to the left DC motor. Record the encoder counts per bin for each duty cycle:

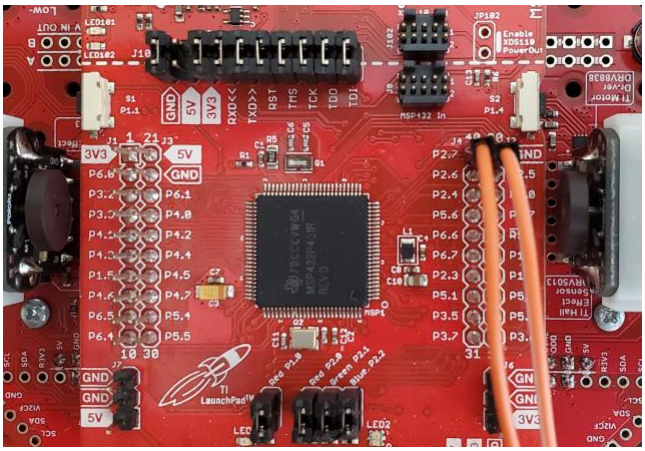
WORK SHEET HERE:

Duty Cycle %	analogWrite value (0 to 255)	Encoder Counts Per Bin	Bin Length: _____s (from code)
10	_____	_____	
20	_____	_____	
30	_____	_____	
40	_____	_____	
50	_____	_____	
60	_____	_____	
70	_____	_____	
80	_____	_____	
90	_____	_____	
100	_____	_____	

Plot your data in your favorite plotting app with Duty Cycle on the y-axis and Encoder Counts **per Second** on the x-axis:

Include Graph Here

9. Using the skills you've built from previous labs, we can probe the PWM signal the RSLK applies to the left DC motor. Using you AD2, connect Channel 1 to probe the pin PWML. To do this, using the chart above, you must attach 1+ to pin P2.7 and 1− to GND. Notice that you can attach the AD2 directly to the Launchpad (no breadboard necessary). Use the image below for guidance:



AD2 Channel 1 Connected to P2.7 and GND

With the left DC motor running (at a duty cycle of your choice (other than 0% and 100%)), use your AD2's Scope application to view the PWM waveform. Include a screenshot of the waveform here (you must set the time base and amplitude appropriately):

Oscilloscope Screenshot

Answer the following questions to verify your oscilloscope view is correct:

Duty Cycle Setting: _____%

Measured Signal On-Time (per period): _____

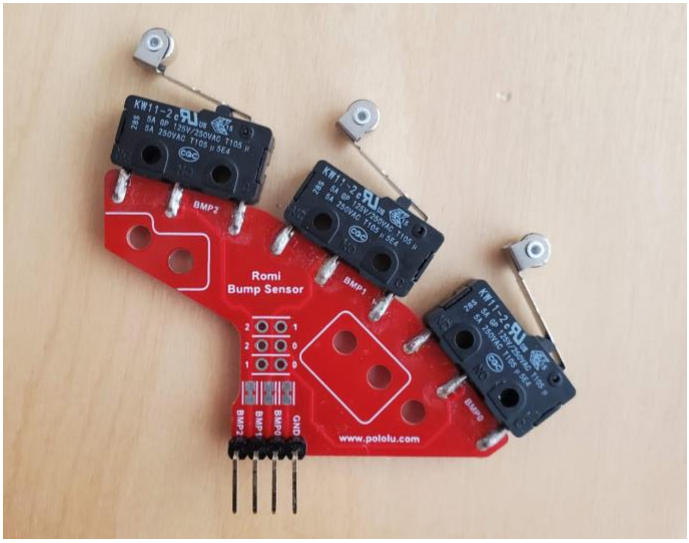
Measured Signal Off-Time (per period): _____

Measured Duty Cycle: _____

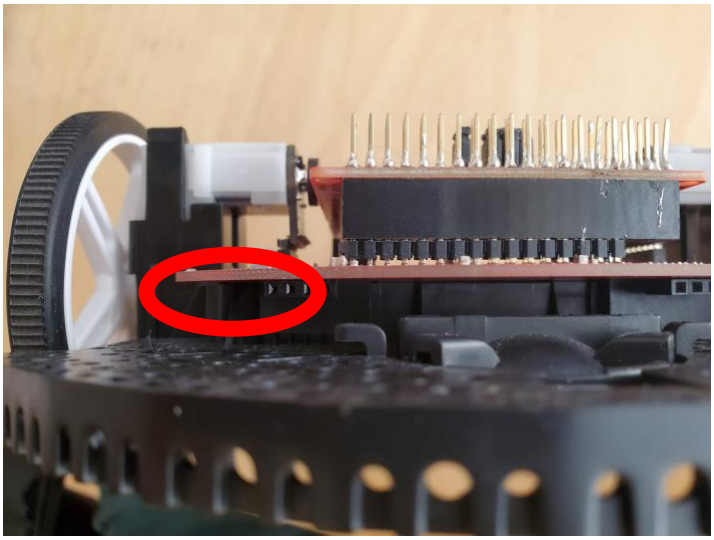
PWM – Period (EXTRA CREDIT)

On the RSLK, we use PWM to vary the speed of the DC motors. In this section, we will demonstrate the role of **period** in a PWM signal when running a motor.

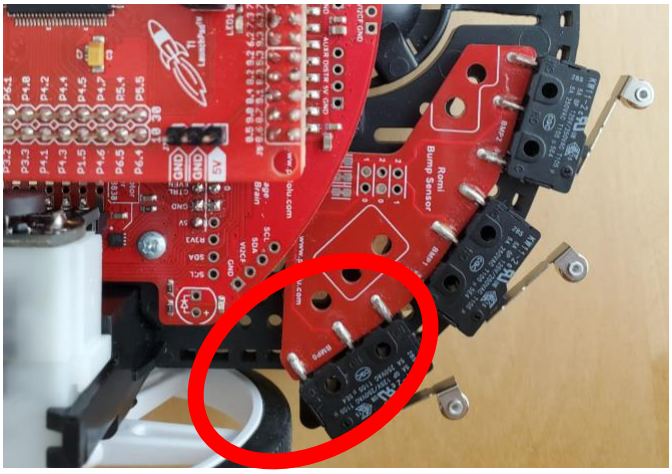
1. Download the sketch pwm_period.ino from CCLE. Launch Energia and open the sketch.
2. Read through the code to get an understanding of what it does. Some of the techniques used here (such as using micros() rather than delay() for time delays) may become useful to you later on.
3. Make sure you have the bumper attachment connected to your car. Look at the following images to ensure this is done correctly:



Bumper Attachment

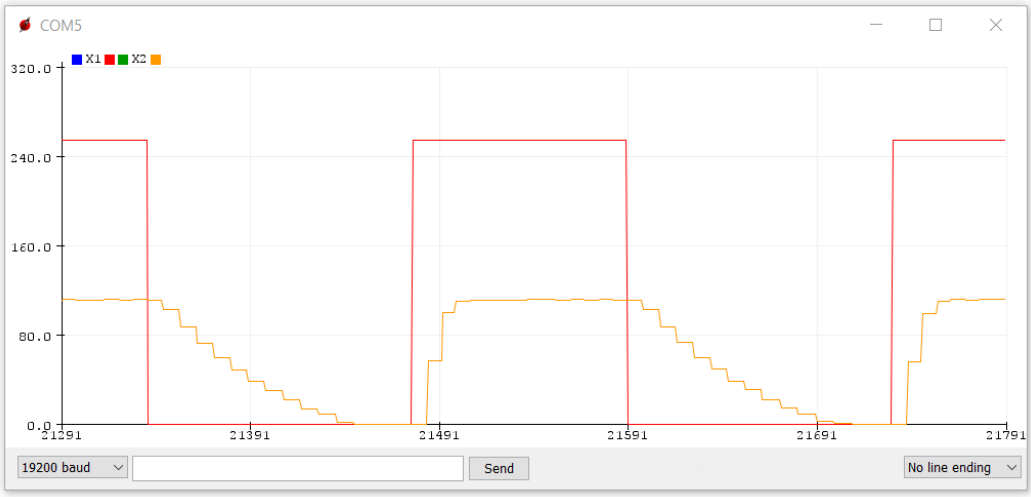


RSLK Front View (Bumper Installation Location Circled)



Complete Bumper Installation (Bumper 0 (BMP0) Circled)

4. Hold the car in your hand or place it on a box so it does not move unexpectedly.
5. Plug in your car and open the **Serial Plotter** (in the top bar click Tools > Serial Plotter). **Set the Serial Plotter baud rate to 19200**.
6. Upload the sketch to your Launchpad. You should see the Serial Plotter printing the PWM signal.
7. Turn on your car and you should now see a second curve tracing the left motor speed. The following image shows what you should see:



Serial Plotter Output After Turning the Car On
(Note: your COM port may be different)

8. Press BMP0 to vary the period. You will hear a high frequency warbling sound from the motor for the smaller period. This is because the pulse rate is 10 pulses per second, and the motors can partially follow the pulses at that rate. Compare this pulse rate to then end of the previous exercise, where you measured the PWM pin pulse rate when using analogWrite.

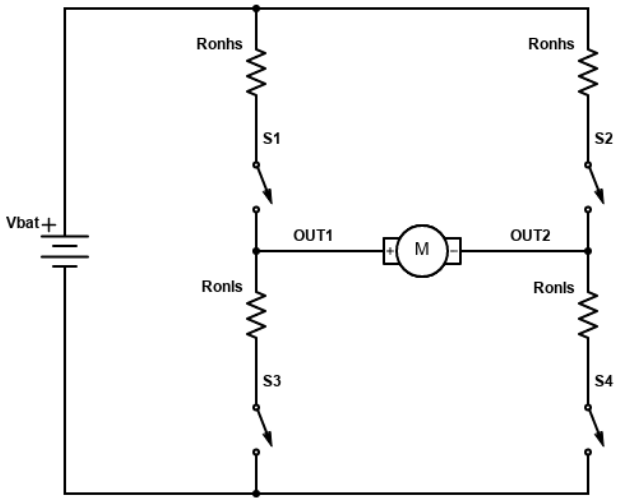
Questions:

1. If we look at this motor control scheme as a system, we'd say that the input is the PWM signal and the output (also called the response) is wheel speed. From this perspective, what kind of filter could you characterize this system as: low pass, high pass, band pass, or band stop? Explain you answer.

The pwm_period.ino sketch seen here is only for demonstration. As said earlier, when you create your own sketches for the RSLK, you will use the analogWrite(pin, value) function to create a PWM signal.

Lab 4 Appendix – H-Bridge Motor Control

The RSLK uses two DC motors to drive the left and right wheels. In order to control each DC motor, a basic circuit called the H-bridge is used. Below is a *simplified*[11] schematic of this circuit to help understand its working principles:



Simplified H-Bridge Schematic

In practice, H-bridge circuits use four MOSFETs (metal-oxide-semiconductor field-effect transistors) to control a DC motor. In this diagram, we use four SPST switches (S1, S2, S3, S4) in series with resistors (Ronhs, Ronls) in place of these transistors to show a simplified view of how these MOSFETs would behave in-circuit. Note that Ronhs and Ronls have very small values (on the RSLK, Ronhs + Ronls = 230 mΩ).

Terms:

- M – The DC motor.
- Ronhs – On-resistance of the high side MOSFETs.
- Ronls – On-resistance of the low side MOSFETs.

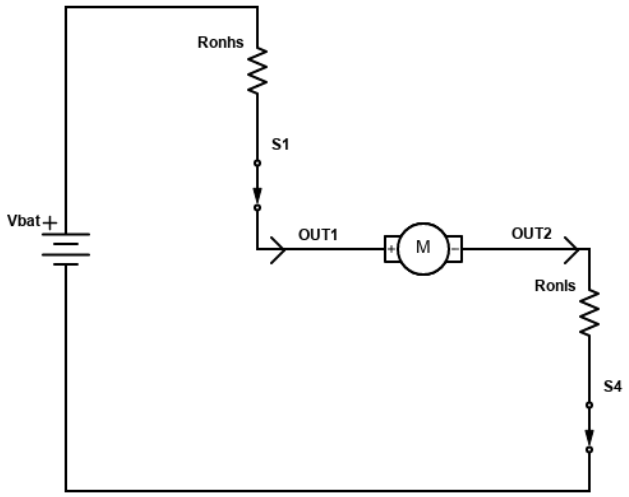
Operating Principle:

The purpose of the H-bridge is to give the ability to turn a DC motor in two directions rather than one. To do this, the circuit must be able to send current through the motor in two different directions. In

[11] Among other things, this circuit is missing *essential* voltage spike protections.

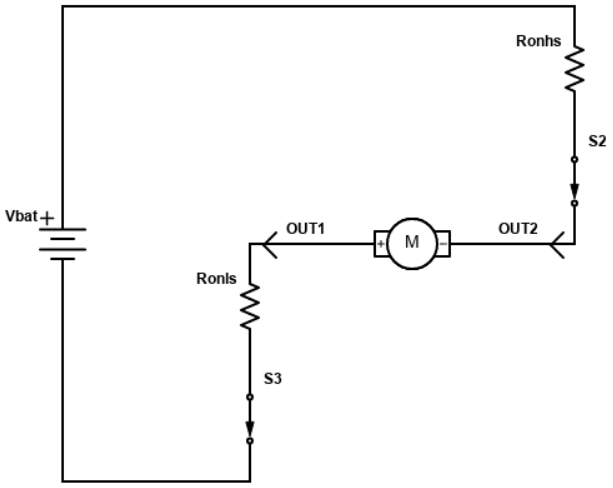
the schematic, this is done by individually pulling OUT1 and OUT2 either high (to Vbat) or low (to GND).

In this way, the circuit turns the motor forward by **closing switches S1 and S4** and **opening switches S2 and S3** to drive OUT1 high and OUT2 low:



Driving the DC Motor Forward

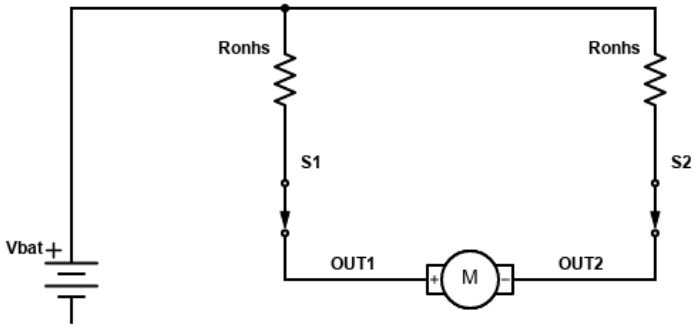
The circuit turns the motor backward by **opening switches S1 and S4** and **closing switches S2 and S3** to drive OUT1 low and OUT2 high:



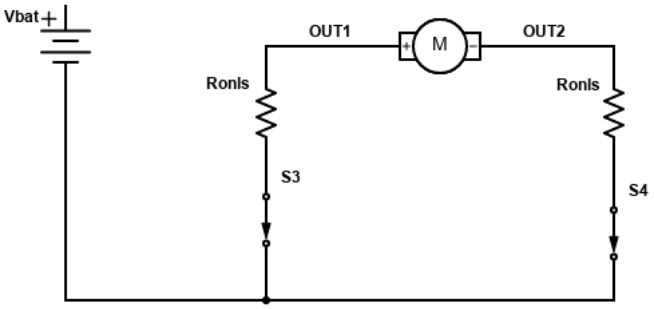
Driving the DC Motor Backward

Finally, an H-bridge can brake the DC motor by pulling OUT1 and OUT2 to the **same** potential. This can be realized in two ways. One is to **close switches S1 and S2** and **open switches S3 and S4** to drive

OUT1 and OUT2 high. The other is to **open switches S1 and S2** and **close switches S3 and S4** to drive OUT1 and OUT2 low:



Braking by Driving OUT1 and OUT2 High



Braking by Driving OUT1 and OUT2 Low

Driving the RSLK's DC Motors:

On the RSLK, each DC motor has its own H-bridge housed in an IC (integrated circuits) on the Chassis Board. These ICs have additional logic that simplifies how you interface with the H-bridge to three 3.3 V input signals: nSLEEP, PH (phase), and EN (enable) (meaning you only indirectly control the MOSFETs corresponding to switches S1, S2, S3, and S4). The table[12] below shows how these input signals drive the outputs OUT1 and OUT2:

nSLEEP	PH	EN	OUT1	OUT2	Motor State
0	X	X	Z	Z	COASTING
1	X	0	LOW	LOW	BRAKING
1	1	1	LOW	HIGH	BACKWARD

[12] This table is from the H-bridge IC's datasheet (DRV8838)

1	0	1	HIGH	LOW	FORWARD

Terms:

- X – This signal state has no effect on the output (often called a “don’t care”)
- Z – The signal is disconnected from both Vbat and GND (aka high impedance)

This discussion boils down to the following. To run a motor in the forward direction you will drive its corresponding nSLEEP high, PH low, and EN high. To run a motor in the reverse direction, you will drive its corresponding nSLEEP high, PH high, and EN high. When referencing the chart at the beginning of Lab 4, the left motor IC’s pins are labeled:

Left Motor H-Bridge IC Input Name	Energia Pin Chart Name
nSLEEP	nSLPL
PH	DIR_L
EN	PWML

Week 4 Lab End