Lab 7: DC Motor Speed and Direction Control

Name: Syed Muhammad Ghufran Alvi & Zaid Bin Khalid ID: 08235; 08128 Section: L-1

Objective

To use TM4C123 ADC and PWM signal for motor speed control.

In-Lab

Task 1: Speed Control of DC Motor with PWM Signal Generation
Task 2: Speed Control of DC Motor with TM4C123 ADC
Task 3: Direction Control of DC Motor with Push Button
Task 4: DC Motor Speed and Direction Control on Keil
Task 5: Straight line motion for a differential drive robot.

1 Introduction to DC Motor and Motor Driver

In this lab, we will interface DC motor with the TivaC board. A DC motor (Direct Current motor) is the most common type of motor. DC motors normally have just two leads, one positive and one negative. If you connect these two leads directly to a battery, the motor will rotate. If you switch the leads, the motor will rotate in the opposite direction.

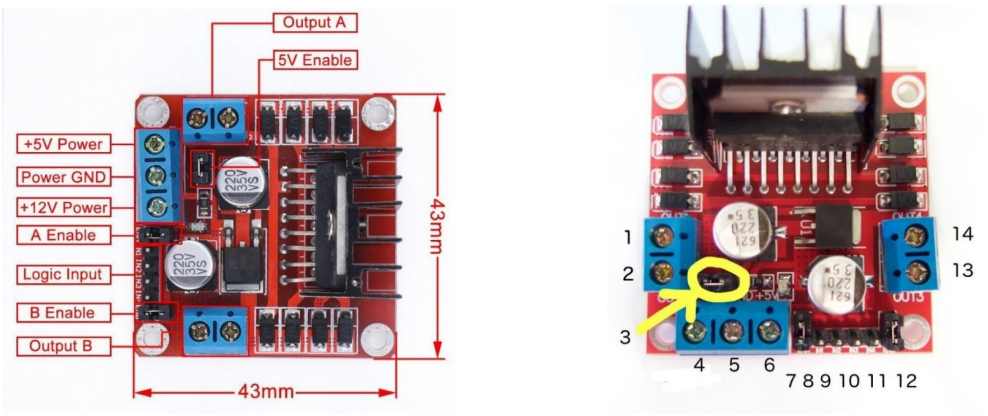
WARNING

Do NOT drive the motor directly from the microcontroller pins. This may damage the controller. Use driver circuit like L298N or its IC.

1.1 Driver Module L298N

The dual bidirectional motor driver module shown in Fig. 1a, is based on the very popular L298 Dual H-Bridge Motor Driver IC. This module will allow you to easily and independently control two motors of up to 2A each in both directions.
It is ideal for robotic applications and well suited for connection to a microcontroller requiring just a couple of control lines per motor. Fig. 1b shows the part description of L298N module with its description given next.

1. DC motor 1 (+)ve lead
2. DC motor 1 (-)ve lead



(a) (b)

Figure 1: L298N Motor Driver Configuration

3. 12V jumper – remove this if using a supply voltage greater than 12V DC. When the jumper is in place, the on board voltage regulator is active (12V max to 5V). Do not remove for now.
4. Connect your motor supply voltage (12V) here, maximum of 35V DC. Remove 12V jumper if Supply Voltage>12V DC.
5. Common GND
6. 5V output if the 12V jumper at No.3 is in place. This is ideal for powering your Controller, whenever we are not powering it with USB cable.
7. DC motor 1 enable jumper ENA, if jumper connected, then motor runs at full speed, otherwise PWM can be given at this pin.
8. IN1 for motor 1 to control direction
9. IN2 for motor 1 to control direction
10. IN3 for motor 2 to control direction
11. IN4 for motor 2 to control direction
12. DC motor 2 enable jumper ENB, if jumper connected, then motor runs at full speed, otherwise PWM can be given at this pin.
13. DC motor 2 (+)ve lead
14. DC motor 2 (-)ve lead

1.2 Speed Control with ENABLE (ENA/ENB) Pins

ENA controls the speed of motor A and ENB controls the speed of motor B. If both of the pins are in a logic HIGH (5V/3.3V) state, then both the motors are ON and spinning at maximum speed. If both of the pins are in a logic LOW (ground) state, then both the motors are OFF.

Through the PWM functionality, we can also control the speed of the motor as shown in Fig. 2. The dutycycle of PWM will determine the voltage supplied to DC Motor. Hence, by controlling

PWM through code at Pin ENA and ENB, we can control the speed of motor. By default, there is a jumper connected to these pins which keep these pins in a HIGH state. In order to control the speed, we need to remove the jumpers at ENA and ENB and connect these terminals with the PWM pins of TivaC and program them in code.

1.3 Direction Control with IN(X) Pins

The direction control pins are the four input pins (IN1, IN2, IN3, IN4) on the module. Through these input pins we can determine whether to move the dc motor forward or backwards. IN1 and IN2 control motor A's spinning direction whereas IN3 and IN4 control motor B's spinning direction. The table below shows the logic signals required for the appropriate spinning action for motor A with IN1 and IN2.

IN1	IN2	Motor Action
1 (HIGH)	1	OFF
1	0 (LOW)	Backward
0	1	Forward
0	0	OFF

Figure 3: Direction Control States

In Fig. 3, whenever one of the inputs is in a HIGH state (5V) then the motor will spin. Otherwise, when both the inputs are LOW (ground) state or both are in HIGH state then the motor stops. In order for motor A to spin forward, IN1 should be LOW and IN2 should be HIGH. For backwards motion, IN1 should be HIGH and IN2 should be LOW. Motor B is also controlled in a similar way with IN3 and IN4.

2 TivaC LaunchPad with Energia

In Energia IDE, unlike Kiel uVision, we can use TivaC LaunchPad pins for various peripherals without the need to activate Ports using registers or specifying function of the pin. But it also comes with limitations of usability and programmable scope of the board. In Energia, we can refer to Pins of TivaC directly using numeric digit like 1,2,3.. and so on. Pin map for the EK-TM4C123GXL LaunchPad is given in Fig. 4 with Black Columns under J1, J2, J3 and J4.

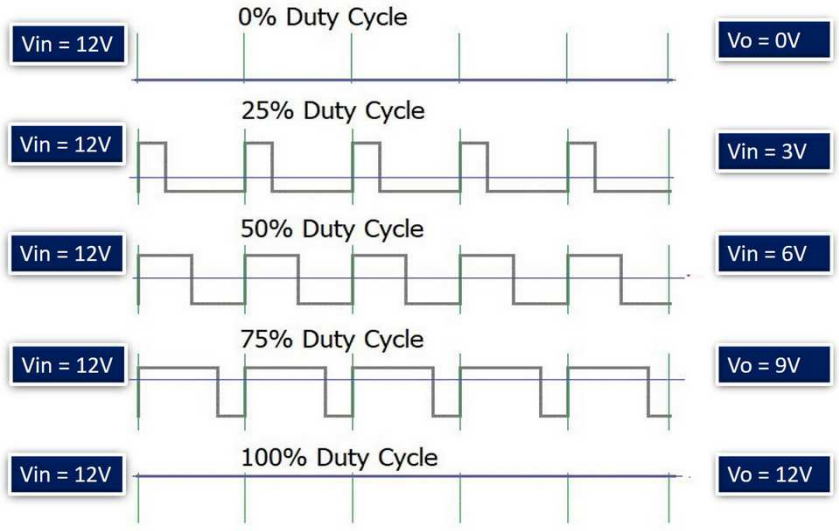


Figure 2: PWM Signal For Motor Speed Control

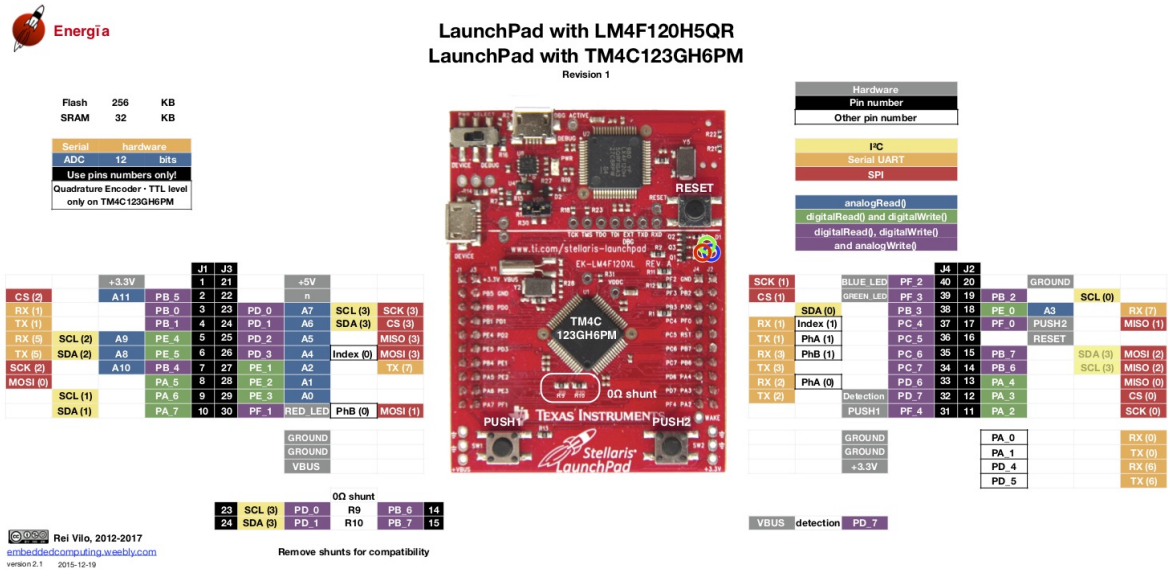


Figure 4: Pin Configuration of TivaC for Energia (Front)

3 In-Lab Tasks

Task 1: Speed Control of DC Motor with PWM Signal Generation

Controlling the speed of the DC motor essentially mean controlling the magnitude of the voltage supplied to it. Section 1.2 discuss how voltage magnitude can be controlled by supplying PWM signal to ENABLE pin at differing duty cycles. In this task, we will control the speed of DC motor using PWM signal for multiple dutycycles.

WARNING

Do NOT supply more than 12V to L298N from DC Power Supply when Regulator Jumper is placed at No.3 pins in Fig. 1b. Remove the +12V jumper if you are using powers higher than +12V.

Make connection between your DC motor and L298N as shown in Fig. 5 and Table 1. Instead of 9V Battery, **provide 12V** to L298N using DC Power Supply as DC motor can quickly drain the 9V Battery. Make sure to not exceed 12V from DC Power Supply as it may damage your driver and board.

Table 1: Circuit Connections

TivaC	**L298N**
PD3	ENA
PA2	IN1
PA3	IN2

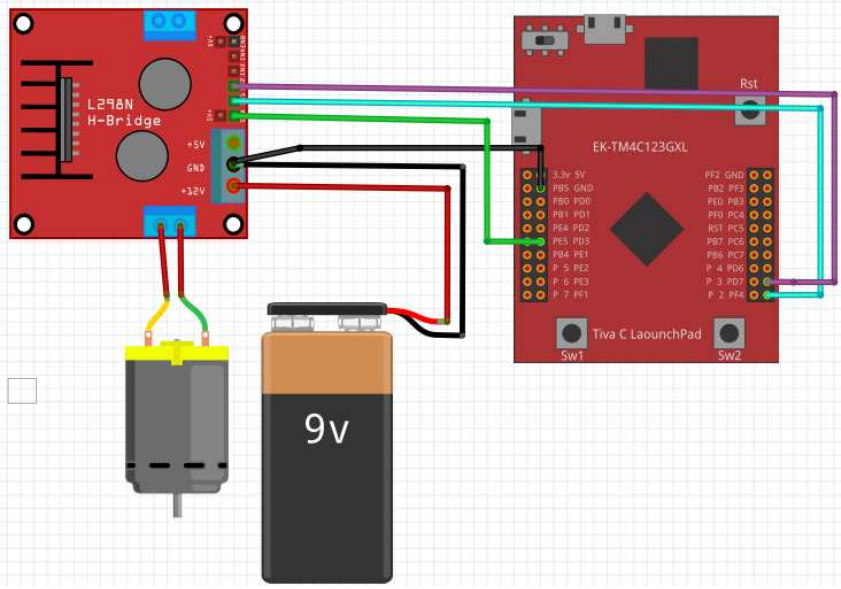


Figure 5: Schematics for Circuit

Open Energia and use the file "motor.ino" to generate 50HZ PWM signals at PD3 for 4 different duty cycles i.e. 0%, 25%, 75% and 100% using the demo 10% function provided and call the functions in main loop() with delay of 2 seconds to obtain varying speed on your DC Motor. Read the entire code and complete the missing commented out commands to make code functional.

Provide your clear circuit image below

Please refer to the last page for the picture required in this frame. (This block only allowed text as input.)

Provide your code here with appropriate comments below (Get the circuit demonstration checked with RA within Lab)

```
// Motor A, TivaC and L298N connections
int enA = 26;   //ENABLE pin (PD3) for Motor A to provide PWM
int in1 = 11;   //IN1 pin (PA2) for Motor A for direction
int in2 = 12;   //IN2 pin (PA3) for Motor A for direction
//const int SW2 = 17;   //Pin number for Switch SW2
bool direction = true;  //Flag for clockwise or anti-clockwise direction

void setup() {
  Serial.begin(115200); //Serial initializing for Debugging

  //Specify pinModes for enA, in1 and in2 below
  pinMode(enA, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  // Turn off motors - Initial state
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);

  //Initialize SW2 Switch as Input
  //pinMode(SW2, INPUT_PULLUP);  //State for SW2 with pullUp resistor
}

void loop() {
    // Make the motor run clockwise direction using in1 and in2 states below
    digitalWrite(in1, HIGH);
    digitalWrite(in2, LOW);
//    digitalWrite(enA, HIGH);

    //Varying Speed for motor below with delay of 2 seconds
    tenP_Speed(); //Run motor with 10% duty cycle
    delay(2000);
}

void tenP_Speed() {
    //Generates PWM of 50Hz (20ms total) with Duty Cycle of 10%
    for (int i=0; i<50; i++){
      digitalWrite(enA, HIGH);
      delayMicroseconds(2000); // High for 2ms, Duty Cycle of 10%
      digitalWrite(enA, LOW);
      delayMicroseconds(18000); // Off for remaining  18ms
  }
}

/*
void changeDirection(){
  //To be used for task03 only
```

Task 2: Speed Control of DC Motor with TM4C123 ADC

In the previous tasks, we achieved the desired speed through dutycycle of PWM signal at ENABLE pin, we can also use the TM4C123 Analog to Digital Converter (ADC) at pin PD3 to achieve the desired speed. ADC gives us range of outputs and inputs instead of just two level like in digital i.e. 0V and 3.3V. Similar to digitalRead(), we have analogRead() which gives us range of values between 0-4095 as we have 12-bit ADC. And like digitalWrite(), we have analogWrite() through which we can write value between 0-255 giving out range of 0-3.3V at Pin.

The desired analog value for different speeds can be calculated by formula **255*(dutyCycle)**, for example for 25% duty cycle we would have **255*0.25 = 64**, hence we would use analogWrite(pin,64) to achieve PWM with 25% dutycycle at PD3.

Complete the Table. 2 for various dutyCycles.

Table 2: ADC Bits Calculations

DutyCycle	ADC Value
0%	0
25%	64
50%	128
75%	192
100%	255

Does the pin PD3 allows analogWrite()? Refer to Energia TivaC Pin Modes with Color Coding in Fig. 4.

yes

Does the pin PD3 allows analogRead()? Refer to Energia TivaC Pin Modes with Color Coding in Fig. 4.

no

Now instead of using PWM functions, use analogWrite() function to change speed of DC motor to 0%, 25%, 75% and 100% with delay of 2 seconds in between. Provide your code below.

Provide your code here with appropriate comments below (Get the circuit demonstration checked with RA within Lab)

```
// Motor A, TivaC and L298N connections
int enA = 26;   //ENABLE pin (PD3) for Motor A to provide PWM
int in1 = 11;   //IN1 pin (PA2) for Motor A for direction
int in2 = 12;   //IN2 pin (PA3) for Motor A for direction
//const int SW2 = 17;   //Pin number for Switch SW2
bool direction = true;  //Flag for clockwise or anti-clockwise direction

void setup() {
  Serial.begin(115200); //Serial initializing for Debugging

  //Specify pinModes for enA, in1 and in2 below
  pinMode(enA, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  // Turn off motors - Initial state
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);

  //Initialize SW2 Switch as Input
  //pinMode(SW2, INPUT_PULLUP);  //State for SW2 with pullUp resistor
}

void loop() {
    // Make the motor run clockwise direction using in1 and in2 states below
    digitalWrite(in1, HIGH);
    digitalWrite(in2, LOW);
//    digitalWrite(enA, HIGH);

    //Varying Speed for motor below with delay of 2 seconds
    tenP_Speed(); //Run motor with 10% duty cycle
    delay(2000);
}

void tenP_Speed() {
    //Generates PWM of 50Hz (20ms total) with Duty Cycle of 10%
//    for (int i=0; i<50; i++){
//      digitalWrite(enA, HIGH);
//      delayMicroseconds(2000); // High for 2ms, Duty Cycle of 10%
//      digitalWrite(enA, LOW);
//      delayMicroseconds(18000); // Off for remaining  18ms
//    analogWrite(enA, 64);
//    delay(100);

//    analogWrite(enA, 128);
//    delay(500);
```

Task 3: Direction Control of DC Motor with Push Button

From Section 1.3, you know how to change direction of the DC motor. We can alternate the IN1 and IN2 value from HIGH to LOW and vice versa to change the direction of DC motor to clockwise or anti-clockwise.

For this tasks, you are required to use on-board push button SW2 to change both speed with analogWrite() and direction with IN1/IN2 of the DC motor. SW2 whenever pressed, should toggle between state 1 and state 2 below, remember to use delay() with SW2 to avoid debouncing:

State 1	50% DutyCycle	Clockwise
State 2	100% DutyCycle	Anti-Clockwise

Provide your code here with appropriate comments below (Get the circuit demonstration checked with RA within Lab)

```
// Motor A, TivaC and L298N connections
int enA = 26;   //ENABLE pin (PD3) for Motor A to provide PWM
int in1 = 11;   //IN1 pin (PA2) for Motor A for direction
int in2 = 12;   //IN2 pin (PA3) for Motor A for direction
const int SW2 = 17;   //Pin number for Switch SW2
bool direction = true;  //Flag for clockwise or anti-clockwise direction
bool lastButtonState = HIGH;
void setup() {
  Serial.begin(115200); //Serial initializing for Debugging

  //Specify pinModes for enA, in1 and in2 below
  pinMode(enA, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  // Turn off motors - Initial state
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);

  //Initialize SW2 Switch as Input
  pinMode(SW2, INPUT_PULLUP);  //State for SW2 with pullUp resistor
}

void loop() {
  // Read the state of the pushbutton
  bool buttonState = digitalRead(SW2);

  // Check if the button state has changed
  if (buttonState != lastButtonState && buttonState == LOW) {
    // Debounce delay
    delay(50);
    // Confirm button press
    if (digitalRead(SW2) == LOW) {
      // Toggle direction flag
      direction = !direction;
    }
    // Additional delay to avoid multiple toggles
    delay(500);
  }

  // Update the last button state
  lastButtonState = buttonState;

  // Set motor direction
  if (direction) {
    digitalWrite(in1, HIGH);
```

Task 4: DC Motor Speed and Direction Control on Keil

Objectives

- Learn how registers control a DC motor's speed and direction.
- Learn how to use the datasheet of the microcontroller.
- Use Keil uVision to write and upload the program to the TM4C123 Launchpad.

In this task, you will learn how to interface a DC motor with a TM4C123 Launchpad and control its speed and direction using Keil uVision. You will also learn how to use the datasheet of the microcontroller to understand how the registers control the motor.
As explained in task 1, we can use the PWM peripheral to control the speed and direction of the motor. However, since this task is on Keil instead of energia, it is important to understand how registers are used to accomplish this task. For this purpose, it is advised to view the PWM configuration example given in datasheet page 1239. You can also find it in the appendix section (3) at the end.

A skeleton code titled "Task4.c" for this task has been provided on Canvas. The code rotates the motor in both directions, starting from the lowest speed to the highest speed. That means first we apply the lowest possible duty cycle to EnA pin of L298N. After that gradually increase the duty cycle from low to high value. In a similar order, the speed of the DC motor also increases. The Code does this by implementing three functions:

1. `void Turn_Clockwise(void)`
2. `void Turn_AntiClockwise(void)`
3. `void PWM_init(void)`

Let's go over their function descriptions one by one.

void Turn_Clockwise(void) & void Turn_AntiClockwise(void)

This function and its anticlockwise counterpart are responsible for the motor direction. They change the pin value for pins corresponding to IN1 and IN2. Figure 3 shows how the pin value should be manipulated to achieve the desired outcome.

void PWM_init(void)

This function basically initializes the PWM peripheral. GPIO pins can be controlled by other peripherals on the microcontrollers, e.g. UART, I2C, PWM, etc. For this purpose the **GPIOAFSEL** register is used. GPIOAFSEL sets the corresponding pin of the port to an alternate function. The selection of the alternate function is dependent on **GPIO Port**

Control (GPIOPCTL); see page 688. The value of the register depends upon the table on page 651 in the datasheet; the table is also provided in the appendix below.

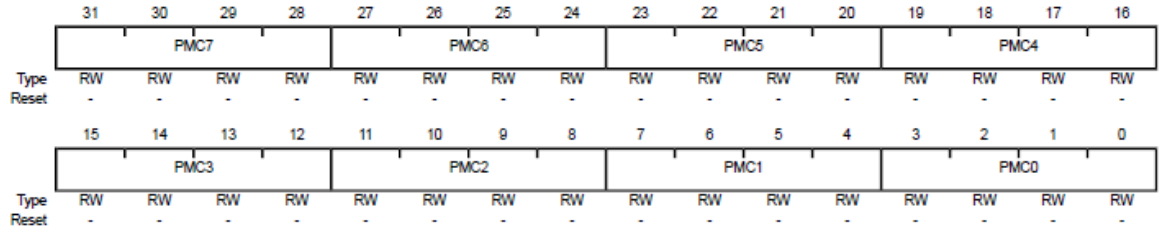


Figure 6: The GPIOPCTL register. PMCn is the port mux control for nth pin. The value of PMCn is determined by the table given in the appendix

For example if we wanted pin PE4 to be controlled by the CAN peripheral

```
    GPIOE -> AFSEL |= (1 << 4)
    GPIOE -> PCTL ~= 0x000F0000 // Making sure that the port mux control for
    pin 4 is at zero, i.e. functions as normal
    GPIOE -> PCTL |= 0x00080000
```

Listing 1: Example code of setting PE4 pin to be controlled by the CAN peripheral

Lets now take a look at what the `void PWM_init(void)` code is trying to do, particularly the snippet shown in Listing 2. We want the PWM pin, i.e. PF2 in our case, to be high when the PWM counter value is the same as `PWM1->_3_LOAD` register. And, the PWM pin to get low when PWM counter value is the same as `PWM1 - > _3_CMPA`. For this code we have chosen the PWM counter to count down.

```
    void PWM_init(void)
{
  ...
    PWM1->_3_CTL &=     /* Disable Generator 3 counter */
    PWM1->_3_CTL &=     /* select down count mode of counter 3*/
    PWM1->_3_GENA =      /* Set PWM output when counter reloaded and clear when
    matches PWMCMPA */
    PWM1->_3_LOAD = ;      /* set load value for 50Hz, note that you have
    divided the clock freq of 16MHz by 64 when accessing the reguster SYSCTL->
    RCC */
    PWM1->_3_CMPA = 4999;      /* set duty cyle to minumum value*/
    PWM1->_3_CTL = 1;           /* Enable Generator 3 counter */
    PWM1->ENABLE = 0x40;       /* Enable PWM1 channel 6 output See page Page
    1247*/
    ...
}
```

Listing 2: Explanation of void PWM_init(void)

Pin Mapping

L298N	Tiva C	Motor
ENA1	PF2	
IN1	PA3	
IN2	PA2	
OUT1		Terminal1
OUT2		Terminal2

Table 3: Circuit Connection

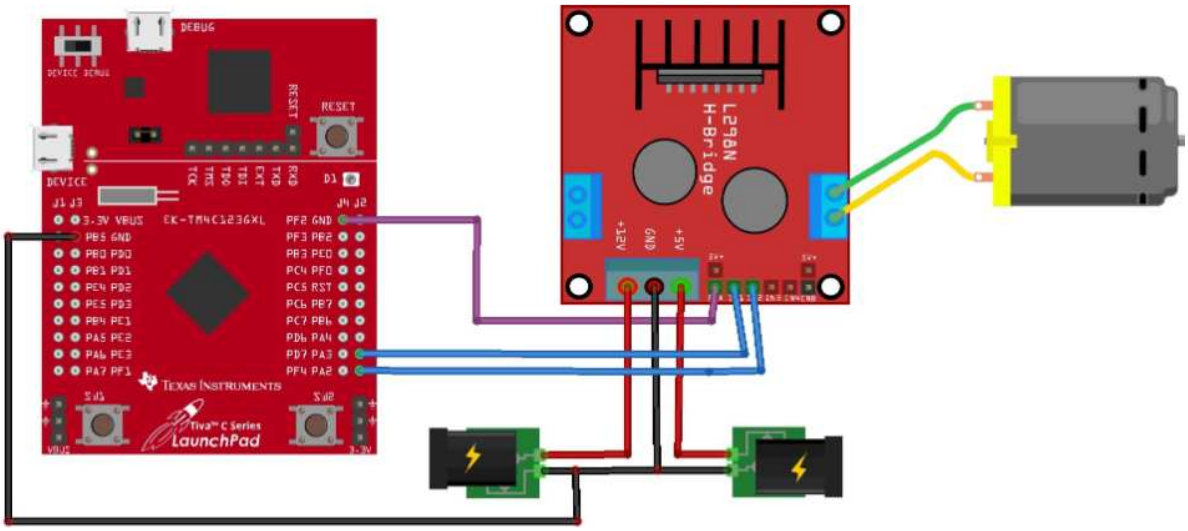


Figure 7: Connection Mapping

Keil Uvision V5 Instructions

The skeleton code provided results in a different system clock rate than the default 16 MHz frequency. But the code works on the default 16 MHz system clock. Therefore, it leads to timing issues with Keil v5. You may retain the default system clock rate in Keil v5, by the following steps:

1. Expand the Project→Device to show system_TM4C123.c (startup)
2. Double click to open the file in the editor window
3. Find the line `#define CLOCK\_SETUP 1` as the figure below
4. Comment out the line
5. Rebuild the project

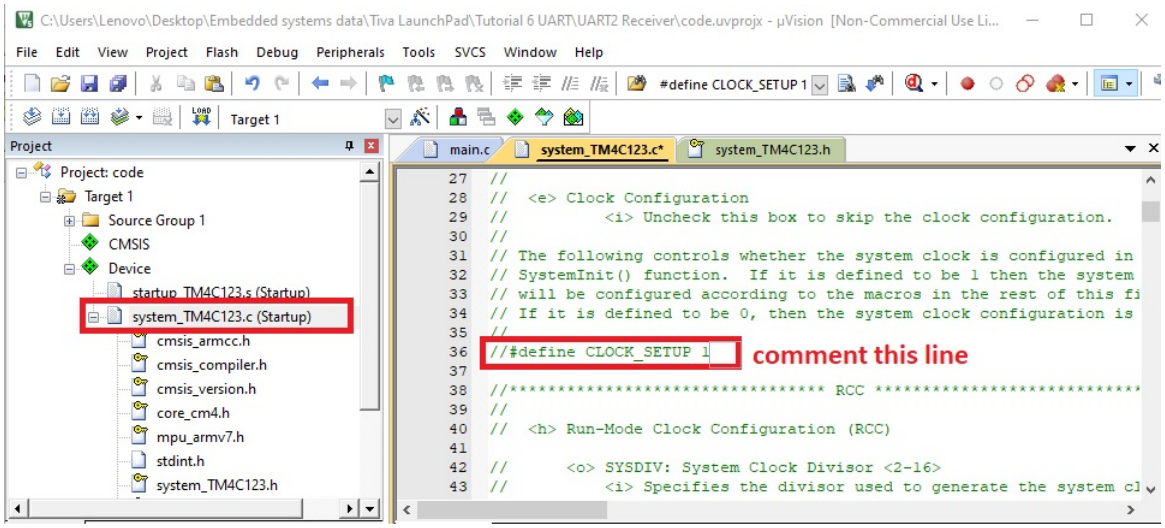


Figure 8: Instruction for Keil Timing Setup

Provide your code here with appropriate comments below (Get the circuit demonstration checked with RA within Lab)

```
#include "TM4C123GH6PM.h"
void Delay_ms(int time_ms);
int duty_cycle = 4999;
void PWM_init(void);
void Turn_AntiClockwise(void);
void Turn_Clockwise(void);
int main(void)
{

      PWM_init();
       Turn_Clockwise();
   for(;;)
   {
       duty_cycle = duty_cycle - 10;
       if (duty_cycle <= 0)
   duty_cycle = 5000;
       PWM1->_3_CMPA = duty_cycle;
       Delay_ms(100);
   }
}


//Spin motor in one direction by giving IN1 and IN2 signals to L298N
void Turn_Clockwise(void)
{
  SYSCTL->RCGCGPIO |=  0x1; /* enable clock to PORTF */
   GPIOA->DIR |=  (1 << 3) | (1 << 2);  /* Make PA3 and PA2 an output pin*/
    GPIOA->DEN |=  (1 << 3) | (1 << 2);/* Make PA3 and PA2 an digitally enabled*/
    GPIOA->DATA |=  (1 << 2);/* IN1 Should be HIGH */
  GPIOA->DATA &=  ~(1 << 3);/* IN2 Should be Low*/
}

//Spin motor in other direction by giving IN1 and IN2 signals to L298N

void Turn_AntiClockwise(void)
{
    SYSCTL->RCGCGPIO |=  0x1;/* enable clock to PORTF */
   GPIOA->DIR |= (1 << 3) | (1 << 2);   /* Make PA3 and PA2 an output pin*/
    GPIOA->DEN |=  (1 << 3) | (1 << 2); /* Make PA3 and PA2 an digitally enabled*/
    GPIOA->DATA |= ~(1 << 2); /* IN1 Should be LOW */
  GPIOA->DATA &= (1 << 3);/* IN2 Should be HIGH*/}

void PWM_init(void)
{
```

Task 5: Straight line motion for a differential drive robot

In the previous tasks we have learnt how to control the speed and direction of a motor. In this task, we will use the acquired knowledge to implement a simple **differential drive** robot.
A differential drive robot is a type of mobile robot that uses two independently-driven wheels to move. The wheels are typically located on either side of the robot, and their speeds are controlled to make the robot move in the desired direction. They have a simple design that makes them easy to build and control. The movement of such a robot can broadly be categorized into two types:

1. **Straight Line Motion**: Both wheels spin with the same speed.
2. **Rotational Motion**: The inner wheel spins slower than the outer, e.g. for a left turn the left wheel will be the inner wheel.

In theory, we assume the ideal condition of both the wheels spinning at the same rate for straight-line motion. However, in real life, a motor can spin at a higher velocity for the same voltage.
To mitigate the effect of this issue one can implement a controller that would minimize this error. A simple controller for this purpose could be the PID controller. However, the PID controller requires some form of feedback, i.e. the velocity at which the wheel is moving. This can be done through a wheel encoder, shown in Figure 9. This is something that you could implement in your project. For the purpose of this lab, it is sufficient to experimentally decide the values of voltage (through PWM or ADC) required for straight-line motion. The student is free to choose to implement this task on Keil or Energia.

Write code to implement a straight-line motion using speed and direction control of the left and right motors. To test your code, you will need an assembled chassis with motors and wheels attached on both sides. Feel free to use your own from your projects.

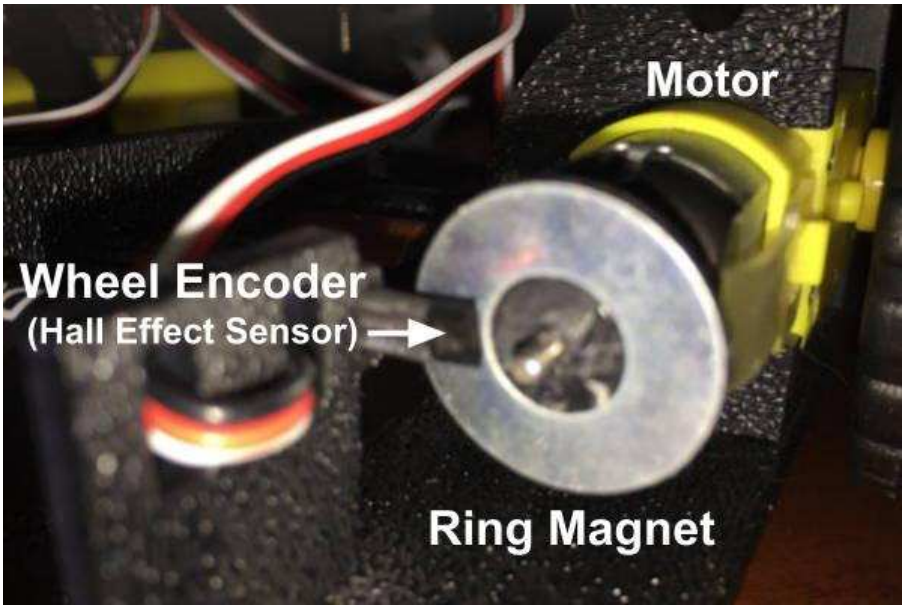


Figure 9: Wheel Encoder Sensor

Provide your code here with appropriate comments below (Get the circuit demonstration checked with RA within Lab)

for task 5, use below code

```
// Motor A, TivaC and L298N connections
int enA = 26;   //ENABLE pin (PD3) for Motor A to provide PWM
int in1 = 11;   //IN1 pin (PA2) for Motor A for direction
int in2 = 12;   //IN2 pin (PA3) for Motor A for direction

int enB = 25;
int in3 = 9;
int in4 = 10;
const int SW2 = 17;   //Pin number for Switch SW2
bool direction = true;  //Flag for clockwise or anti-clockwise direction

void setup() {
  Serial.begin(115200); //Serial initializing for Debugging

  //Specify pinModes for enA, in1 and in2 below
  pinMode(enA, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);

  pinMode(enB, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);

  // Turn off motors - Initial state
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);

  //Initialize SW2 Switch as Input
  pinMode(SW2, INPUT_PULLUP);  //State for SW2 with pullUp resistor
}

void loop() {
    // Make the motor run clockwise direction using in1 and in2 states below
     digitalWrite(in1, HIGH);
     digitalWrite(in3, HIGH);


      if (direction)
      {
      analogWrite(enA, 128); // State 1
      analogWrite(enB, 128);
```

Appendix A

PWM example given in Tiva C Launchpad Datasheet (Page 1239)
The following example shows how to initialize PWM Generator 0 with a 25-kHz frequency, a 25% duty cycle on the MnPWM0 pin, and a 75% duty cycle on the MnPWM1 pin. This example assumes the system clock is 20 MHz.

1. Enable the PWM clock by writing a value of 0x0010.0000 to the RCGC0 register in the System Control module (see page 456).
2. Enable the clock to the appropriate GPIO module via the RCGC2 register in the System Control module (see page 464).
3. In the GPIO module, enable the appropriate pins for their alternate function using the GPIOAFSEL register. To determine which GPIOs to configure, see Table 23-4 on page 1344.
4. Configure the PMCn fields in the GPIOPCTL register to assign the PWM signals to the appropriate pins (see page 688 and Table 23-5 on page 1351).
5. Configure the Run-Mode Clock Configuration (RCC) register in the System Control module to use the PWM divide (USEPWMDIV) and set the divider (PWMDIV) to divide by 2 (000).
6. Configure the PWM generator for countdown mode with immediate updates to the parameters.
   - Write the PWM0CTL register with a value of 0x0000.0000.
   - Write the PWM0GENA register with a value of 0x0000.008C.
   - Write the PWM0GENB register with a value of 0x0000.080C.
7. Set the period. For a 25-KHz frequency, the period = 1/25,000, or 40 microseconds. The PWM clock source is 10 MHz; the system clock is divided by 2. Thus there are 400 clock ticks per period. Use this value to set the PWM0LOAD register. In Count-Down mode, set the LOAD field in the PWM0LOAD register to the requested period minus one.
8. Set the pulse width of the MnPWM0 pin for a 25% duty cycle.
   - Write the PWM0CMPA register with a value of 0x0000.012B.
9. Set the pulse width of the MnPWM1 pin for a 75% duty cycle.
   - Write the PWM0CMPB register with a value of 0x0000.0063.
10. Start the timers in PWM generator 0.
    - Write the PWM0CTL register with a value of 0x0000.0001.
11. Enable PWM outputs.
    - Write the PWMENABLE register with a value of 0x0000.0003.

AFSEL Table

IO	Pin	Analog Function	Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a]										
			1	2	3	4	5	6	7	8	9	14	15
PB1	46	USB0VBUS	U1Tx	-	-	-	-	-	T2CCP1	-	-	-	-
PB2	47	-	-	-	I2C0SCL	-	-	-	T3CCP0	-	-	-	-
PB3	48	-	-	-	I2C0SDA	-	-	-	T3CCP1	-	-	-	-
PB4	58	AIN10	-	SSI2Clk	-	M0PWM2	-	-	T1CCP0	CAN0Rx	-	-	-
PB5	57	AIN11	-	SSI2Fss	-	M0PWM3	-	-	T1CCP1	CAN0Tx	-	-	-
PB6	1	-	-	SSI2Rx	-	M0PWM0	-	-	T0CCP0	-	-	-	-
PB7	4	-	-	SSI2Tx	-	M0PWM1	-	-	T0CCP1	-	-	-	-
PC0	52	-	TCK SWCLK	-	-	-	-	-	T4CCP0	-	-	-	-
PC1	51	-	TMS SWDIO	-	-	-	-	-	T4CCP1	-	-	-	-
PC2	50	-	TDI	-	-	-	-	-	T5CCP0	-	-	-	-
PC3	49	-	TDO SWO	-	-	-	-	-	T5CCP1	-	-	-	-
PC4	16	C1-	U4Rx	U1Rx	-	M0PWM6	-	IDX1	WT0CCP0	U1RTS	-	-	-
PC5	15	C1+	U4Tx	U1Tx	-	M0PWM7	-	PhA1	WT0CCP1	U1CTS	-	-	-
PC6	14	C0+	U3Rx	-	-	-	-	PhB1	WT1CCP0	USB0EPEN	-	-	-
PC7	13	C0-	U3Tx	-	-	-	-	-	WT1CCP1	USB0PFLT	-	-	-
PD0	61	AIN7	SSI3Clk	SSI1Clk	I2C3SCL	M0PWM6	M1PWM0	-	WT2CCP0	-	-	-	-
PD1	62	AIN6	SSI3Fss	SSI1Fss	I2C3SDA	M0PWM7	M1PWM1	-	WT2CCP1	-	-	-	-
PD2	63	AIN5	SSI3Rx	SSI1Rx	-	M0FAULT0	-	-	WT3CCP0	USB0EPEN	-	-	-
PD3	64	AIN4	SSI3Tx	SSI1Tx	-	-	-	IDX0	WT3CCP1	USB0PFLT	-	-	-
PD4	43	USB0DM	U6Rx	-	-	-	-	-	WT4CCP0	-	-	-	-
PD5	44	USB0DP	U6Tx	-	-	-	-	-	WT4CCP1	-	-	-	-
PD6	53	-	U2Rx	-	-	M0FAULT0	-	PhA0	WT5CCP0	-	-	-	-
PD7	10	-	U2Tx	-	-	-	-	PhB0	WT5CCP1	NMI	-	-	-
PE0	9	AIN3	U7Rx	-	-	-	-	-	-	-	-	-	-
PE1	8	AIN2	U7Tx	-	-	-	-	-	-	-	-	-	-
PE2	7	AIN1	-	-	-	-	-	-	-	-	-	-	-
PE3	6	AIN0	-	-	-	-	-	-	-	-	-	-	-
PE4	59	AIN9	U5Rx	-	I2C2SCL	M0PWM4	M1PWM2	-	-	CAN0Rx	-	-	-
PE5	60	AIN8	U5Tx	-	I2C2SDA	M0PWM5	M1PWM3	-	-	CAN0Tx	-	-	-
PF0	28	-	U1RTS	SSI1Rx	CAN0Rx	-	M1PWM4	PhA0	T0CCP0	NMI	C0o	-	-
PF1	29	-	U1CTS	SSI1Tx	-	-	M1PWM5	PhB0	T0CCP1	-	C1o	TRD1	-
PF2	30	-	-	SSI1Clk	-	M0FAULT0	M1PWM6	-	T1CCP0	-	-	TRD0	-
PF3	31	-	-	SSI1Fss	CAN0Tx	-	M1PWM7	-	T1CCP1	-	-	TRCLK	-
PF4	5	-	-	-	-	-	M1FAULT0	IDX0	T2CCP0	USB0EPEN	-	-	-

Figure 10: Alternate Function table; See page 651

4 Assessment Rubrics

Marks distribution

		LR2	LR4	LR5	LR9
In-lab	Task 1	2.5 points	2.5 points	2.5 points	10 points
	Task 2	2.5 points	2.5 points	2.5 points	
	Task 3	5 points	5 points	5 points	
	Task 4	10 points	10 points	10 points	
	Task 5	10 points	10 points	10 points	
Total Marks 100					

Marks obtained

		LR2	LR4	LR5	LR9
In-lab	Task 1				
	Task 2				
	Task 3				
	Task 4				
	Task 5				
Obt. Marks out of 100					