EE 337: Speed Control of DC Motor
Lab 8

3^{rd} October, 2016

1 Interfacing a Motor

In the set of experiments for the week, we shall learn how to drive a DC motor using micro-controllers and receive data or command from PC using UART.

The output ports of a micro-controller cannot provide sufficient current to drive a motor. Therefore, we use an interface chip such as the H-bridge L293D, which receives microstandard digital signals from a micro-controller and provides large currents to an external device. Data sheet for L293D has been posted on the Moodle site.

For this experiment, you will be provided with a breadboard, the L293D IC, a DC motor, USB to serial converter and an external power supply. You have to set up the driver IC connections to the micro-controller and motor on the breadboard to drive the motor.

DC motors can be controlled by controlling the average DC voltage applied to the motor. Analog control of this voltage is not very easy for a micro-controller (though this can be done, as we shall learn later). However, a micro-controller can change the duty cycle of the voltage applied. This can be done by applying either a voltage V or 0 through an H bridge, as controlled by a port pin of the micro-controller. If the voltage is switched between V and 0 very rapidly, the motor cannot respond to the instantaneous voltages. It responds to the average value of voltage applied to it. We will control the speed of the DC motor by using pulse width modulation (PWM). If a voltage V is applied for time T_{ON} and 0 V is applied for time T_{OFF}, the average applied DC voltage is $V \cdot \frac{T_{ON}}{T_{ON}+T_{OFF}}$. This average value can be controlled by varying T_{ON} and T_{OFF}, typically keeping $(T_{ON} + T_{OFF})$ constant.

For communicating between the PC and micro-controller through UART we need a serial terminal software installed on the PC, e.g., RealTerm, Hyper terminal, etc.

Note: Install the following software in your laptop before coming to Lab.

- For *W*indows user: install realterm or any serial terminal.
- For *L*inux: Install gtkterm

```
sudo apt-get install gtkterm
```

Problem Description

1. The DC motor (in the given setup) drives a slotted wheel, which interrupts an infrared light beam from an LED to a photo detector. There are 30 slots in the wheel. A wheel that rotates at 1 rps (revolutions per second) would give 30 interruptions in 1 second. The circuit for this is given in the accompanying document `IR Encoder`. By counting the number of light interruptions per second, the speed of the DC motor can be estimated. The motor's rating is maximum of 300 to 350 RPM (revolutions per minute) at 12V DC.

   A block-diagram of the expected operation is shown in Fig 1. The flow for the problem is given below:

   (a) Receive PWM width as input from the user using Serial Terminal (UART interface).

   (b) The value of the PWM width dictates the ON time of a 30 ms pulse that is used to drive the motor driver.

   (c) The motor driver produces an appropriate voltage which is given to the motor.

   (d) The motor runs at a corresponding RPM.

   (e) This RPM is measured by the micro-controller. The user's input and the RPM measured is displayed on the Serial Terminal every 1 second (by sending data on UART interface). The micro-controller should count the interruptions from the IR sensor for every 1s and use that to calculate the RPM.

   (f) The display format on the Serial Terminal must be:

   ```
   INPUT:XX,RPM:XX
   ```

This problem requires 3 timed events. Event E1 is timed interrupt every 33 ms (count it 30 times to get 1 second event). Event E2 is to ensure that a single PWM cycle is of time period 30 ms. Event E3 is the interruption created by the UART interface. We suggest that E1 & E2 is done using a hardware timer. Event E3 will require interrupt to receive input from user on UART. Note that there are three timers in the device.

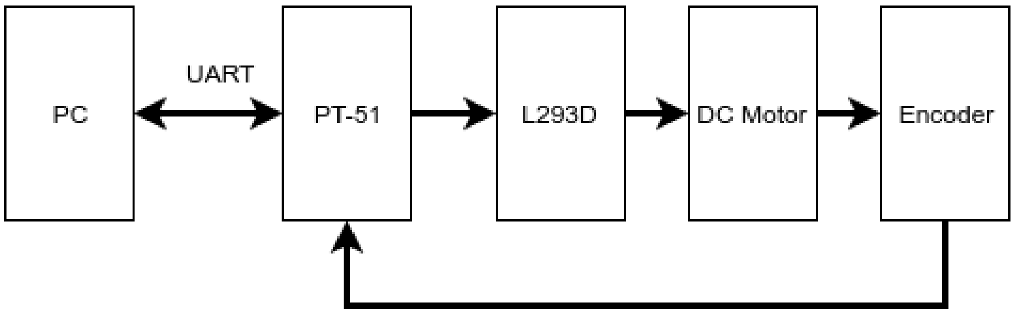

Figure 1: DC motor speed control and measurement using Pt-51 micro-controller and UART interface to a PC.

Lab Work

1. Before interfacing the DC motor, you will write a program to control the brightness of LEDs by controlling PWM width through user inputs obtained from the PC. To do this

   (a) Write a program that uses a hardware timer to control the duty cycle of ON and OFF time of any of the LEDs on the board.

   (b) You should be using Serial Terminal to receive 8-bit PWM width value from user using UART. If the input from user is k, the ON time

   $$T_{ON} = \frac{k}{100} * (T_{ON} + T_{OFF}).$$

   The sum of T_{ON} and T_{OFF} ($T_{ON} + T_{OFF}$) should always be 30 ms. For example, if the input is $k = 50$, the T_{ON} time must be 15ms and T_{OFF} time must be 15ms. PWM width of output should be changed immediately after receiving input from user.

   (c) The whole process has to run indefinitely and you should observe the LED's brightness change as per input.

   (d) Also, connect the port to an oscilloscope during the lab to observe the PWM waveform (that will be later fed to the motor controller).

2. Now you can interface the DC motor using the program in the previous part as the starting point.

   (a) Do all the calculations required for displaying the RPM.

   (b) Read the documentation of the L293D and of the IR circuitry.

   (c) Connect PWM output pin to the input of L293D as given in supported document.

Note: You need to write your program in C (a template to get started is given next). Convert your subroutines from Timer lab assignment to C functions.

Program template:

```
#include <at89c5131.h>
/* Global variables and definition */
void pwm_setup(int duty_cycle);
void serial_init(int baud_rate);//use 8 bit UART
void delay (int k);
void serial_send(char dat);
void serial_send_string(char *str);
int measure_rpm();

int pwm_width=50;//0 to 100

void main()
{
        int rpm=0;
        pwm_setup(pwm_width);//default 50% duty cycle
        serial_init(9600);//baud_rate=9600 default
        while(1)
        {
         rpm= measure_rpm();
        /*
         motor's RPM is more than 255, So we need to send 16 bits
        */
                serial_send_string("INPUT:");
                serial_send(pwm_width);
                serial_send_string(",RPM:");
                serial_send(rpm[Higher 8 bits]);//send Higher 8 bits of rpm
                // to Terminal/UART
                serial_send(rpm[Lower 8 bits]);//send Lower 8 bits of rpm
                //to Terminal/UART

        }
}

int measure_rpm()
{
        /*you can measure encoder pulses by
        polling or using external interrupt */
}
```

```
void serial_init(int baud_rate)
{

}

void serial_send_string(char *str)
{
        //Write code to send a string on UART
        //(You can take help of '\0' to get end of string )
}

//sending data via UART
void serial_send(char dat) //send 8 bit data on UART
{

}

//receiving data via UART
void serial_read(void) interrupt 4
{

        ES=0;
        if(RI); //Wait for Receive flag
        {
                RI = 0;
                pwm_width = SBUF;
                pwm_setup(pwm_width);


        }
        ES=1;
}
void delay (int k)
{
}
void pwm_setup(int duty_cycle)
{
//PWM generation Logic
//You can use timer interrupt to generate PWM
//Duty cycle from 0% to 100%
}
```