CPEN 442
Fall 2023

Project Assignment #3: DC Motor Speed Control

Allowed 4 Students Per Team
Check-off November 30th, 2023
Presentations:
December 5th, three groups
December 7th, two groups

Objectives:

- Design and Implement a robotic control system on a microcontroller with an operating system support.
- Use of periodic interrupts to prioritize different events.
- Realize complex robotic systems using an engineering design approach.
- Present a technical design in a clear and informative manner.

Project Summary:

You are required to design and implement a DC motor speed control system that utilizes an operating system to function and provide real-time response guarantees. The system continually accepts an input target speed in RPM through a keypad and applies necessary control to adjust the actual motor speed to match the target speed. The target speed and the actual (measured) speed of the motor are to be displayed on an LCD.

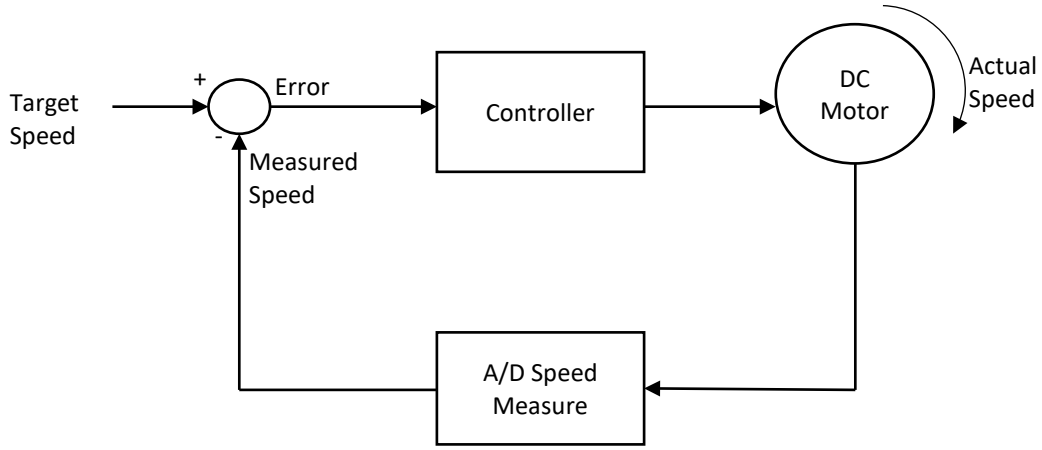


Figure 1 Block diagram of the DC Motor Speed Control System.

The project assumes a feedback control system where the actual speed of the motor is sampled periodically via an analog-to-digital (ADC) converter to determine the error. That error is then fed into your designed controller to update the voltage applied to the DC motor through pulse-width-modulation (PWM). Figure 1 shows a block diagram for the required control system. A picture of the actual project running on the Edubase evaluation board is provided in Figure 2.

Project Specifications and Description:

The system should run on the real-time operating system (RTOS-v2) platform developed in Project 1 and also included with the provided files for this project. RTOS-v2 runs at a thread switching time **TIMESLICE** of 2 ms. The TM4C123GH6PM should be configured to operate at a clock frequency of 16 MHz.
Below is a guide to the relevant components required by the system.

1) <u>The DC Motor:</u>

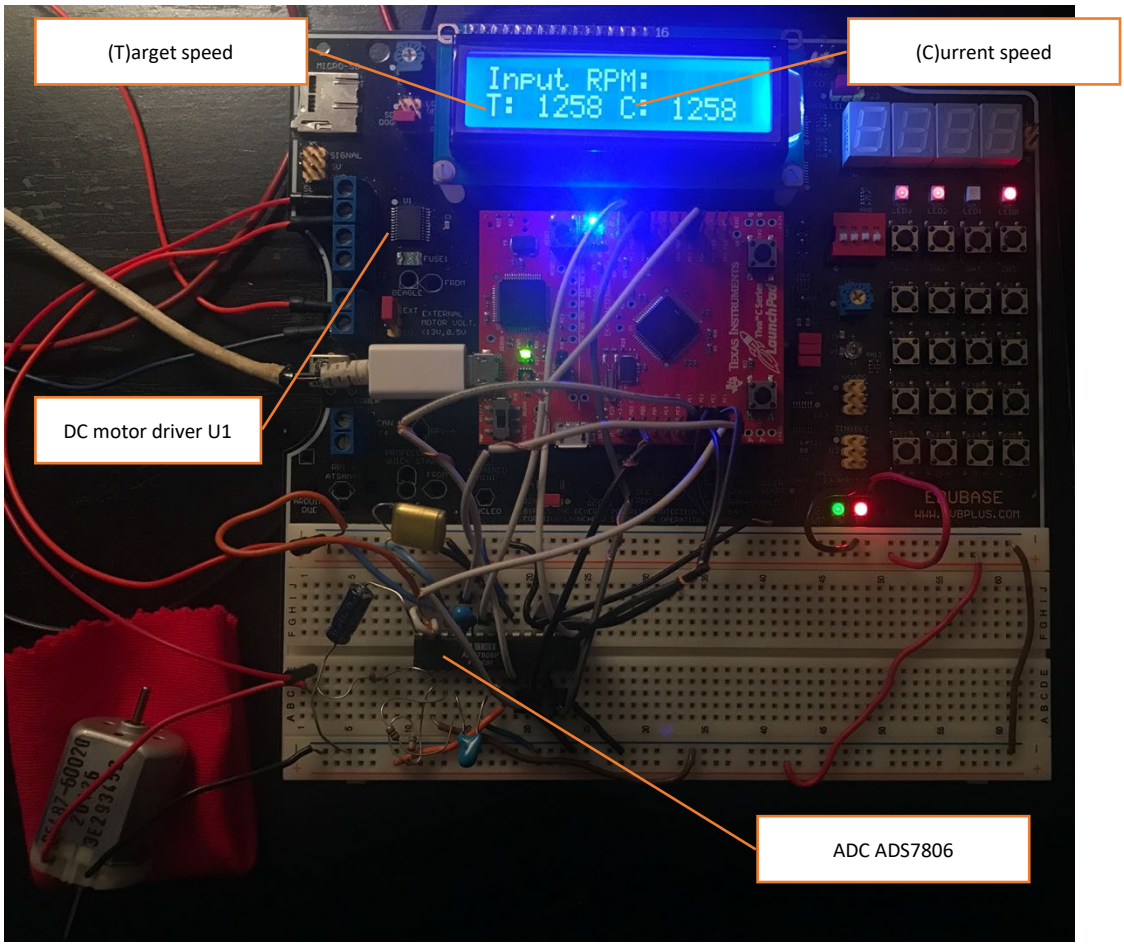
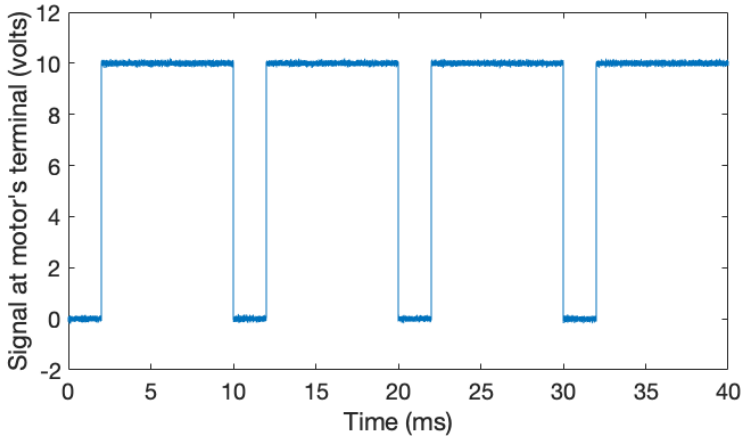


Figure 2 System running on Edubase evaluation board with external ADC. The DC motor is driven by a 10 v power source connected to T3 on the board.

- You may use a basic DC motor similar to the one shown in Figure 2. It should be powered by an external power supply at 10 volts DC connected to terminal T3 on the board. The acceptable motor speed range is: 0, and from 400 to 2400 rpm. The voltage supplied to the DC motor comes from the driver U1 (see Figure 2) which receives a PWM signal from the microcontroller through pin PF2 and the direction of rotation through pins PB1, PB0. The chip U1 boosts these signals' power and drives the motor.
- The PWM should be generated through `M1PWM6` on the TM4C123GH6PM microcontroller. The PWM frequency should be set at 100 Hz.
- As the DC motor rotates, it induces a back e.m.f. proportional to its angular speed. The back e.m.f. opposes the effect of the applied PWM signal to the motor resulting in a

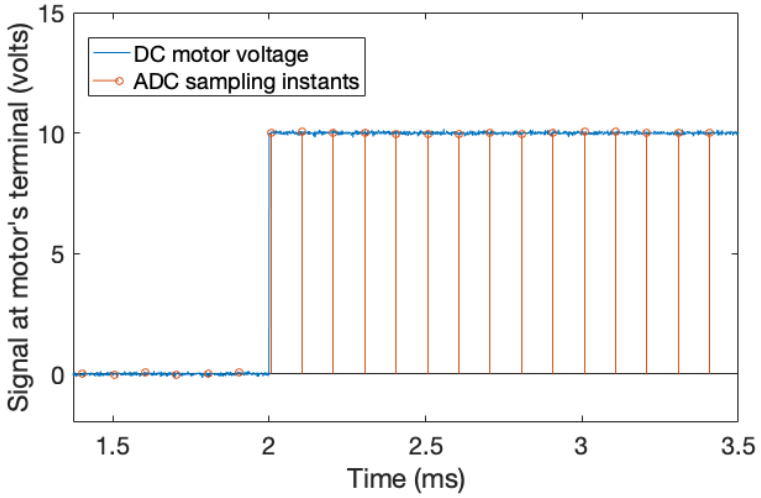
different speed than the expected. Accordingly, the applied PWM has to be continuously adjusted to compensate for the back e.m.f. as well as any voltage inaccuracies (disturbance) caused by the motor driver U1. Figure 3 shows an example of the noisy voltage signal present at the DC motor's terminal for 4 cycles of the PWM control signal. Note that the PWM cycle time is 10 ms which corresponds to the 100 Hz frequency.



- By measuring the actual voltage across the terminal of the motor in *milli-volts*, the given C function `Current_speed` (in provided file **Voltage2RPM.c**) returns the corresponding motor speed in rpm.

2) <u>The ADC:</u>

- In order to measure the actual voltage across the motor terminal (and accordingly get its speed from the function `Current_speed`), an analog to digital converter (ADC) has to be employed in order to digitally provide the analog voltage value to the control system for computations.



- In the system shown in Figure 2, the 12-bit low power ADC: ADS7806 is used. It is connected and configured for ±10 volts analog input and 12-bit parallel output as shown on Page 7 of the datasheet file: **ADS7806 Datasheet.pdf** included with the provided files.
- Since the PWM generated signal value is ideally between 0 and 10 volts, the accuracy (resolution) of ADC need not be exactly all the 12 bits. Instead, the most significant 6 to 8 bits are enough to reasonably capture the instantaneous voltage across the motor.
- Note that in the ADC configuration above, the input voltage ranges from -10 volts to +10 volts, so there are cases when the instantaneous motor voltage drops slightly below 0 due to noise and back e.m.f. (see Figure 3). The ADC may thus output the negative voltage

Figure 4 ADC sampling the DC motor voltage at a rate of 10^4 samples/second.

represented in the 2's complement.

- In order to measure the DC motor's voltage and accordingly determine the measured speed, the ADC should sample the DC motor signal fast enough to track the Hi-Lo variations of the PWM across the motor and efficiently estimate the average driving voltage. As such, you are required to configure your control system, using a periodic interrupt, to take an ADC sample once every 100 μs, or at a sampling rate of 10^4 samples/second. Figure 4 depicts the ADC sampling instants on a snapshot of the noisy PWM signal at the DC motor terminals. The sampling rate of 10^4 samples/second guarantees that every noisy PWM cycle will get 100 samples to measure the actual voltage applied to the motor by this noisy cycle.
- The average voltage of the DC motor signal is to be evaluated every 10 ms, i.e., over the past 100 samples. That average voltage, after being converted to milli-volts, is then fed into the given function `Current_speed` to obtain the measured DC motor speed in rpm.

3) The Controller:
   - Periodically, every 10 ms, the error is computed and accordingly the control PWM duty cycle is adjusted to eliminate the error. For r_T being the target speed in rpm, and $\hat{r}(t)$ being the measured speed at time t, then you should compute the error, $e(t)$, as
   $$e(t) = r_T - \hat{r}(t),$$
   and accordingly determine the controller PWM duty cycle control $u(t)$.
   - The type of control $u(t)$ is optional. You may choose an incremental controller, a PID controller or a fuzzy logic controller.
   - The controller performance is considered acceptable as long as the steady state error is within $\pm$15 rpm.
   - Make sure to avoid overflow in the PWM duty cycle. In particular, the maximum duty cycle you can apply is 99.5%. The minimum duty cycle you can apply is either 0 if the target speed is 0 rpm, or 18% otherwise. The allowed target speed range (0 or from 400 rpm to 2400 rpm) is chosen to guarantee proper operation within the PWM duty cycle boundaries, and the power supply voltage of 10 volts.

4) The Keypad:
   - The Keypad is used to input the target DC motor speed. You may utilize a keypad interfacing code from a previous lab, or the provided functions in the provided **Keypad.s** file, instead of developing a new interface from scratch.
   - You are provided the assembly file **Keypad.s** which includes the void function `Scan_Keypad` which, when called, keeps on scanning the keypad until it detects a key-press. Then it stores the ASCII value of the pressed key in an 8-bit unsigned integer **Key_ASCII**. You should declare **Key_ASCII** as a `uint8_t` variable in your main project file.
   - A separate OS thread should be allocated to inputting the target speed via the keypad.
   - The target speed is limited to a 4-digit decimal number. During the input process, any keypress on 'A', 'B', 'D', or '*' is ignored. A keypress on 'C' can either be ignored or it can

simply clear the input digits so far to enable re-entry of target speed. A keypress on '#' should apply the input target speed to the system.

- During the target speed entry process, any valid keypress should be reflected on the LCD screen, on the top line, as shown in Figure 2.
- While keypresses are returned to the microcontroller in the ASCII format, the system should convert the resulting array of ASCII characters into the corresponding number in hexadecimal to be processed to generate the target speed. The provided assembly file: **ASCII_Conversions.s** includes a function named `ASCII2Hex` that takes a pointer to an array of ASCII characters input through the keypad and returns the equivalent hexadecimal value.
- To adjust the input value for the allowed rpm target range, apply the following truncation. If the returned hexadecimal value by the `ASCII2Hex` function is greater than 2400 rpm, it should be truncated to 2400 rpm. If the returned value is positive (>0) but smaller than 400 rpm, it should be truncated to 400 rpm. Finally, if the hexadecimal value is 0 or falls in the range of 400 to 2400 rpm, set it directly as the target speed value, r_T.

5) <u>The LCD:</u>
    - The first line of the LCD should be interactively demonstrating the keypresses from the keypad as the user inputs the target speed digits. When an acceptable key is pressed (please refer to The Keypad section above for the specification of acceptable keypresses), it is automatically displayed in its respective position on the LCD. When '#' is pressed or 4 acceptable digits have been entered the LCD should blank the input RPM value on the first line meaning that it is ready to receive a new input.
    - The second line of the LCD should display the target speed as well as the current speed which is changing dynamically. Since the DC motor speed is sampled every 10 ms for control, updating the current speed on the LCD every 10 ms is inefficient and will not allow the user's eye to catch a steady value for it. Instead, you are required to display the ***<u>average</u>*** of the measured speeds and update it every whole second.
    - In order to convert the hexadecimal values of the target speed and of the average speed into arrays of ASCII digits to be displayed on the LCD, use the given function `Hex2ASCII` which is included in the provided file **ASCII_Conversions.s**. In particular, the function takes two inputs: 1) A pointer to an array that will contain the output number in ASCII, and 2) the hexadecimal number to be converted to ASCII. The function modifies the array whose reference pointer is passed as the first input to contain the ASCII representation of the number passed as the second input. The function is operational as long as the input hexadecimal number is between 0 and 0x270F (which is 9999 in decimal).
    - The display of the target speed as well as the current speed should be realized as a separate thread on your RTOS-v2 platform. Make sure to coordinate the use of the LCD over different threads using blocking semaphores.

Deliverables

1. A working project will be checked off in the lab on the due date.

2. A lab report containing a discussion of the system design you followed including the system threads, the interrupt mechanism used to sample the motor's voltage, and the type of controller used together with your choice of the controller parameters. The lab report should include screenshots and pictures of different phases of progress through the project and a discussion of how the project work has been allocated to the team members. The lab report is due two days after the project has been checked off, and will be submitted through the dedicated link on Canvas.
3. A 15-minute in-class presentation summarizing your development of the project and discussing the main challenges the team has been through and the efforts done to overcome those challenges. The presentation should include an argument on the particular choice of controller and how its parameters have been set. The presentation should provide screenshots and pictures of the project – all students must be attending all presentations.