NAZARBAYEV UNIVERSITY
SCHOOL OF SCIENCE AND TECHNOLOGY

Homework Assignment #4

ROBT305 – EMBEDDED SYSTEMS

Fall Semester 2018

MOBILE ROBOT DESIGN WITH PWM CONTROL OF DC MOTOR ACTUATORS AND ADDITIONAL FUNCTIONS IMPLEMENTED ON A BBB BOARD

DUE TIME AND DATE

- **Class presentation and report uploaded to Moodle on Tuesday 20 November.**

LEVEL OF COLLABORATION ALLOWED

You will be working in groups of four students

DELIVERABLES REQUIRED

You group is required to demonstrate in class the implemented PWM control of DC motors on the provided Dagu Rover 5 tracked chassis equipped with additional hardware for mobile robot design.

REFERENCES

Derek Molloy,"Exploring BeagleBone. Tools and Techniques for Building with Embedded Linux", Wiley, 2015
Lecture slides
Thomas Bräunl, Embedded Robotics: Mobile Robot Design and Applications with Embedded Systems, Springer, 2008 (available in library and Moodle)

ASSIGNMENT DETAILS

INTRODUCTION

In this project you will implement PWM control of the DC motor actuators of the Dagu Rover 5 tracked chassis (https://www.pololu.com/product/1551). The robot DC motors will be controlled through the DRV8833 Dual Motor Driver Carrier (https://www.pololu.com/product/2130) and powered from a 7.2 V Vex battery. Please refer to last lecture slides and Chapter 6 and 9 of the "Exploring BeagleBone" textbook (available in Moodle) regarding the basics of the BBB GPIO and PWM control of a simple DC motor using a driver board and PWM implemented in the BBB. Each group will also implement additional robot functional with the hardware used in the class and/or available in library, i.e. joystick control, webcam for robot vision transfer to a control PC, IMU based robot orientation control or other, in a threaded way.

TASK 1: MANUAL BBB PWM CONFIGURATION (CLASS ACTIVITIES – 20%)

All instruction are from the command line of the SSH session.

I found the way the PWM is setup in Exploring BeagleBone book is bit complicated. Below is the alternative approach using the universal device tree overlays that can you use in the project. Study Chapter 6 of the BBB textbook regarding GPIO arrangements, Linux device tree overlays and PWM.

Chapter 4 of the Embedded Robotics book (available in library and Moodle) also describes basics of PWM motor control.

1. You will need to start installing **the universal io device tree overlays**, whose source is in the folder **/opt/source/beaglebone-universal-io**. Login to BBB through ssh protocol and "**cd**" to this folder.

2. Type next command in terminal to install it:

```
make install
```

3. To load the universal IO device tree overlay, type:

```
echo cape-universaln > /sys/devices/bone_capemgr.*/slots
```

   This created files in the **/sys/devices/ocp.*** folder for the expansion header pins (the files have a "**pinmux**" extension), which can be used to for pin multiplexing. Since we use the double channel driver board we can control 2 motors, i.e. 1 left and 1 right motor, in the robot chassis using the following BBB PWM outputs: **P.9.14, P.9.16 and P9.22 and P9.21**. Please refer to Fig. 6-7 on page 212 of the BBB textbook.

4. To look at the multiplexing modes for each pin, type as below:

```
config-pin -l P9.14
```

   This will give you the following options: *default gpio gpio_pu gpio_pd pwm*

5. Configure the pins to pwm like this

```
config-pin P9.14 pwm
```

6. You can check the pinmux status using the pinmux file in the ocp folder like this

```
cat /sys/devices/ocp.*/P9_14_pinmux.*/state
```

   This should give you *pwm*

   Alternatively, the state can be set like this

```
echo pwm > /sys/devices/ocp.*/P9_14_pinmux.*/state
```

7. BBB has 8 PWM channels denotes as export numbers in the table below describing the pin mapping.

EXPORT NUMBER	PIN NAME	PINS
0	EHRPWM0A	P9.22,P9.31
1	EHRPWM0B	P9.21,P9.29
2	ECAPPWM0	P9.42
3	EHRPWM1A	P9.14,P8.36
4	EHRPWM1B	P9.16,P8.34
5	EHRPWM2A	P8.19,P8.45
6	EHRPWM2B	P8.13,P8.46
7	ECAPPWM2	P9.28

As we use P9.14 to control one DC motor you should use EHRPWM1A with the export number 3. To enable the peripheral you do the following:

```
echo 3 > /sys/class/pwm/export
```

8. Listing the contents of the pwm folder:

```
ls /sys/class/pwm/
```

This should give you: *export pwm3 pwmchip0 pwmchip2 pwmchip3 pwmchip5 pwmchip7 unexport*
Since the folder **pwm3** is shown this means that the export worked.

9. Listing the contents of the **pwm3** folder:

```
ls /sys/class/pwm/pwm3/
```

gives: *device duty_ns period_ns polarity power run subsystem uevent*
Refer to slides and BB textbook regarding meaning of these PWM parameters, i.e. duty, period, polarity.

10. To check the PWM settings write:

```
cat /sys/class/pwm/pwm3/period_ns
```

Default value: 0

11. To set the PWM frequency to 200kHz for PWM control of the DC motor set the period to 5000 ns:

```
echo 5000 > /sys/class/pwm/pwm3/period_ns
```

12. Write the previous command to see changes:

```
cat /sys/class/pwm/pwm3/period_ns
```

5000

13. Use the duty cycle settings to vary the output BBB analog voltage as the fraction of the max 3.3 V BBB pin output voltage, i.e. setting the duty cycle to 50% of the period will result to 1.65 V output:

```
        echo 2500 > /sys/class/pwm/pwm3/duty_ns
```

14. To run the PWM:

```
        echo 1 > run
```

15. To stop the PWM:

```
        echo 0 > run
```

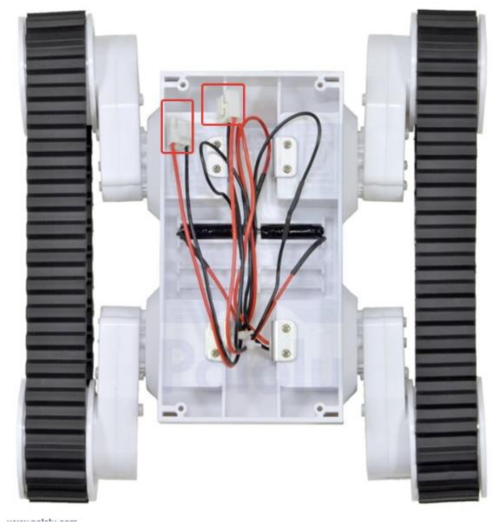
You can use a multimeter to observe average voltage difference from the P9.14 pin when varying the duty cycle parameters.

To configure another PWM channels, you don't need to "make install" again. Just start from step 4.

TASK 2: ROBOT HARDWARE CONNECTION (20%)

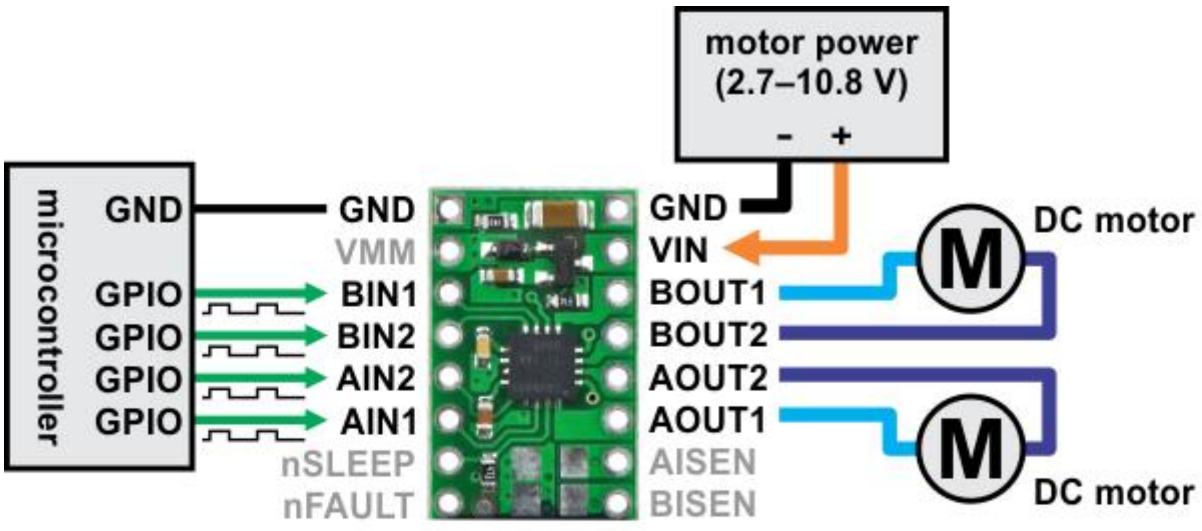
1. Study Chapter 10 of the BBB textbook and Chapter 4 pf the Embedded Robotics textbook for the principle of the DC motor control. Note that the book describes connection of another model of the driver board (DRV8535) that has the polarity input.

2. Study the Dagu Rover chassis description using the link https://www.pololu.com/product/1551 Identify the motor wires of Dagu Rover, you can see them in the picture below in red rectangles:



There are 4 motors in the chassis, so be sure to select only two of them, i.e. one from different sides of the robot. Use the red wire of Dagu Motor to connect it with AOUT1 and the black one with AOUT2, do the same connection of the second motor wires with BOUT1 and BOUT2 of the DRV8833 Dual Motor Driver Carrier board https://www.pololu.com/product/2130 or the corresponding pins of the L293D motor driver chip as described below. The chassis are given to you mainly preassembled, so just check the connections.

3. The motors are powered from the 7.2V Vex battery. Connect the battery to the GND and VIN pins of this driver chip. The input pins of the driver board should be connected to the corresponding PWM and GND pins on the BBB as in the Figure below.
Note that the batteries are old and do discharge fast even if not in use. Please ask for the charger from TAs and charge the battery yourself if needed. Please do not worry about the motor encoder connections at this point.



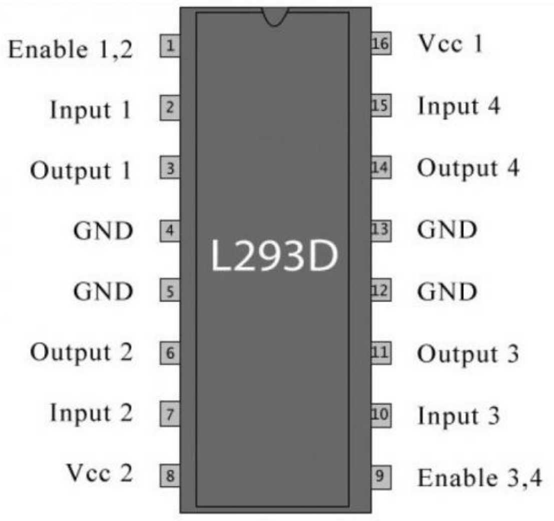
4. Study the DRV8833 Driver datasheet and Chapter 4 of the Embedded Robotics book (provided in Moodle). Note that the control of the motors can be done setting the PWM inputs as follows (Page 9, Table 2 in the datasheet doc):
   Forward rotation: AIN1 – PWM and AIN2 – 0;
   Reverse rotation: AIN1 – 0 and AIN2 – PWM.

5. Some groups will use the L293D motor driver chip instead of the DRV8833 Driver board shown in the Figure on the right.

   BIN1, BIN2, AIN1, AIN2 of DRV8833 correspond to IN1, IN2, IN3, IN4 in L293D.

   BOUT1, BOUT2, AOUT1, AOUT2 of DRV8833 correspond to OUT1, OUT2, OUT3, OUT4 in in L293D.

   For detailed information please refer to the provided datasheet.



6. Enable the PWM from BBB and verify the DC motor working at different velocities. You may put the robot chassis upside down (wheels up) for convenience. Note the setting the duty cycle to 0 results in 0 output voltage at that pin.

7. Control the DC motors using the C++. Study and test the provided template **BBB_PWM_motor_control_tempale.cpp** file. The file is taken from http://www.nagavenkat.adurthi.com/?p=304

8. Modify the C++ example code to run your motors at different velocities, i.e. 80% max speed for 3 sec, 40% speed at 3 sec, reverse max speed at 3 sec, reverse 50%speed at 3 sec.

TASK 3: MOTOR ENCODER CONNECTION (OPEN INSTRUCTION ACTIVITY - 30%)

1. The robot chassis has the quadrature encoders. Please refer to the datasheet for technical specifications. Connect the controlled motor encoders to the BBB board and verify its correct wheel position/speed measurements. You may try first the simple instructions using the same universal cape on the BBB board that we used for PWM setup as described at https://gist.github.com/pdp7/b0ae6b99348103eb60dd9e6b6ecb91f8

   More info you may find in the older example instructions provided in Moodle made by students from previous years based on instruction from https://www.youtube.com/watch?v=gEnrsIS17PM https://github.com/Teknoman117/beaglebot/tree/master/encoders

   If the instruction are not working on your BBB image version, please try to more recent library available at https://github.com/rosmod/lib-bbbeqep

2. Once you successfully connected the motor encoders, please calibrate the sensor reading to map actual wheel rotations.

3. Write the C++ code to read the sensor values.

OPTIONAL OPEN INSTRUCTION TASK 4

VARIANT 1. DIGITAL PID WHEEL POSITION CONTROL IMPLEMENTION (20%)

1. You will implement the PID position control of one DC motor using the encoder feedback to convert it to a servomotor. Please study Chapter 5 of the Embedded Robotics book (available in library and provided in Moodle) and the provided in Moodle extract on Digital Control (supplementary appendices to Franklin's Feedback Control of Dynamic Systems textbook, 7th edition).

2. Following examples in the Embedded Robotics book (Chapter 5) write the PID control code in C++ program taking BBB encoder measurements as feedback and calculating reference control signal to PWM pints to control the wheel positions.

3. Manually adjust the PID gains and achieve more or less suitable control behavior using Table below

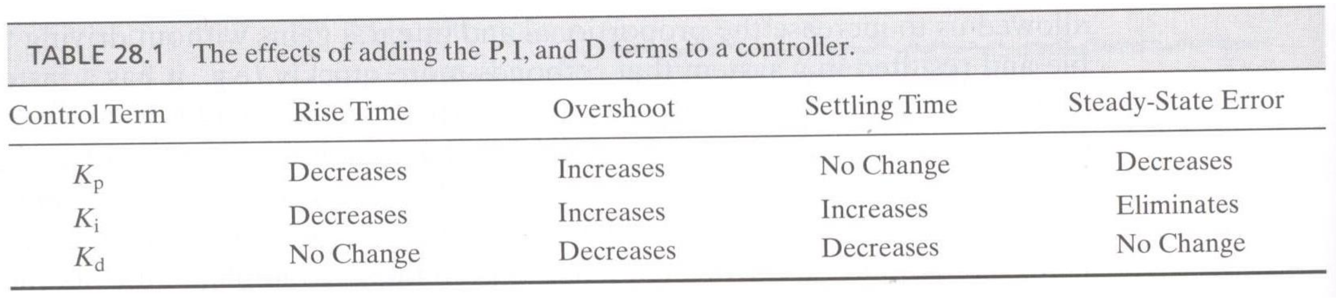


TABLE 28.1 The effects of adding the P, I, and D terms to a controller.

Control Term	Rise Time	Overshoot	Settling Time	Steady-State Error
K_p	Decreases	Increases	No Change	Decreases
K_i	Decreases	Increases	Increases	Eliminates
K_d	No Change	Decreases	Decreases	No Change

4. Test if you can control a robot wheel rotation to 90, 180, 360 degrees. Write the C++ code.

VARIANT 2. JOYSTICK CONTROL OF THE MOBILE ROBOT (15%)

We have limited number of game joysticks in 7.321 or 7.405 labs (ask TAs for assistance). You may try to implement the robot joystick control similar to a previous master student project https://youtu.be/g6enhc70i5E following the instruction on connecting the joystick from https://github.com/roboticsNU/Dagu-Rover-5-with-Joystick

You can take a power supply for the BBB board from TAs if needed. You may try to search for other info sources in Internet.

ADDITIONAL FEATURES TO REACH 100% GRADE

You may install additional features to your robot to get the maximum project grade. Options include installing USB webcam to your mobile robot (connecting to BBB) and implementing video streaming to a control PC from the robot, implementing precise IMU based rotation control of the robot, other options. ALL FEATURES SHOULD BE IMPLEMENTED AS SEPARATE THREADS OR PROCESSES RUNNING ON BBB.

Please inform the instructor if you spot any errors in this lab manual during experimentation. Any suggestions for improvement are also welcome.