Lab Assignment 7
GPIO Programming to control robot arm in SW

The goal of this lab is to control the Robotic arm using software running on the ARM microprocessor on the ZedBoard. You will be sending Pulse Width Modulation (PWM) signals to the Robotic arm using Pmod connector on the ZedBoard.

Instructions

- Each lab assignment consists of a set of pre-lab questions, the actual time in the lab, and a lab report.
- The pre-lab assignments prepare you for the challenges you will be facing in the actual lab. So that each lab member gets the same benefits, pre-lab assignments have to be solved individually. Feel free to discuss problems with class and lab mates, but complete the assignment on your own.

Reading List

The following reading list will help you to complete the pre-lab assignment. The readings will also help you in subsequent lab assignments. Please complete the readings that are marked ***[Required]*** before attempting the pre-lab assignments.

[1] Review general materials on General Purpose I/O (GPIO) interfacing and PWM signal
- http://en.wikipedia.org/wiki/General-purpose_input/output
- http://en.wikipedia.org/wiki/Pulse-width_modulation

[2] Tutorial_PWM.pdf ***[Required]***

Pre-Lab Assignment

We have only very limited time for the lab available. To make efficient use of the lab time, you will need to prepare for it. First, go through the assigned reading above and get an overview of the schematics and the manuals mentioned. Second, approach the questions below to apply your knowledge. Please submit the solution to the pre-lab assignments in PDF format via blackboard.

PWM basics

Pre 8.1) Define what is a PWM signal. Define the PWM duty cycle.
Pre 8.2) Assume we are working with a PWM signal with a 50Hz frequency. Answer the following questions:

a. What is the clock period of the PWM signal with frequency 50Hz? (show your calculations)
b. Draw the PWM signal in a timing diagram (x-axis is time) with a duty cycle of 20% and another with 60% duty cycle. Clearly mark the time instance when the PWM signal changes. In each case, how long is the on duration (time during which the signal is high), and the off duration (time during which the signal is low)? Show your calculations.

GPIO Programming

Pre 8.3) Use the information in document [2] to write a C function (*degreeToOnDelay()*) to calculate the on period of a PWM signal for controlling a servomotor. The C function receives the servo position (0 to 180 degrees) and returns the time in microseconds that PWM signal should be on during each period so that the RC servo moves to the specified servo position.

Rotational Speed

Pre 8.4) Speed can be defined for linear motion, as well as for rotational motion. In the context of a rotational motion, the speed is often defined in degrees per second. A rotational speed is suitable for defining speed of a RC servo. As an example, with a speed of 18 degrees/s, in each second the servo moves by 18 degrees. Consequently, it takes 10 seconds for traveling from 0 angle to 180 angle.

a. Given a rotational speed of *Rot* [degree/sec], how much will occur within 20ms? Write the formula and show your reasoning.
b. Assuming *Rot* = 36 degrees/sec, how much rotation will occur within 20ms?

Lab Assignments

This lab introduces controlling robotic arm with GPIO. The Robotic arm has 5 servo motors named "Base", "Bicep", "Elbow", Wrist", and "Gripper". Each servo can rotate through 180° when sent the associated PWM signal. For your robot, the frequency of the PWM signal should be set at 50 Hz. The duty cycle (the ratio of the pulse width to total signal period) will determine the rotation angle. The duty cycle range can vary from 3% (a 600-microsecond pulse width) to 12% (a 2400 microsecond pulse width). Figure 1 below shows how the servo motor operates for different PWM signal duty cycle settings.

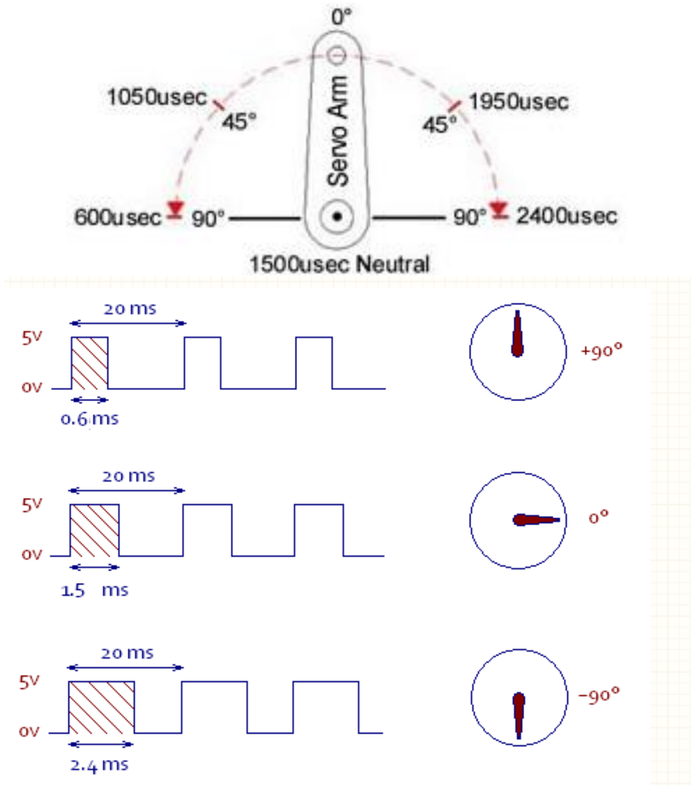


Figure 1. Servo motor operation for different PWM signal duty

You will be provided with a C program (PWM-thru-GPIO.c) to generate a 2000 usec wide pulse. In order to send the generated PWM signals to the Robotic arm, you will use a Pmod connector. The ZedBoard has 5 Pmod connectors (JA, JB, JC, JD, and JE). Each Pmod connector has 12 pins (in two rows), as shown in Figure 2 below.

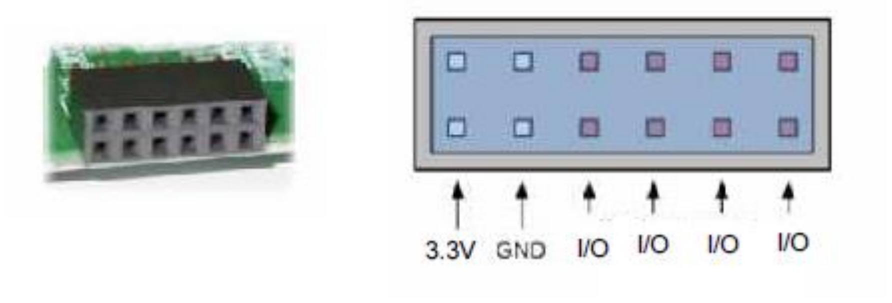


Figure 2. Pmod connector

Four Pmod connectors (JA, JB, JC, and JD) interface to the programmable logic (FPGA) side of the ZedBoard. If we want to connect the robotic arm to the ZedBoard through one of these 4 connectors, we will need to program the FPGA (this will be done in a later lab).
One of the Pmod connectors (JE) interfaces to the processing system (ARM microprocessor) and the programmable logic side of the ZedBoard. In this lab you will control the robotic arm using the JE connector on the ZedBoard. To connect the JE Pmod interface to the servo motors easily, we have provided an interface board.

- **Note -** all Pmod connectors are referred to as GPIO interfaces by the operating system. We will use these terms interchangeably.

Lab 8.1 Controlling GPIO in SW

1. We will use GPIO (General Purpose I/O) pins to generate PWM signals that control the robot servos. The servos will be connected through the JE Pmod connector on the ZedBoard via an interface board (mounted on the robot base).
2. Connecting to ZedBoard
    a. Login into the Windows desktop PC (we will refer to this system as the host) using your myneu credentials.
    b. Make sure your Zedboard is powered (plugged in and the power switch on the board is in the ON position), and has been given time to boot up (generally takes a few minutes). Then connect the Zedboard to the PC host via Ethernet to the RJ45 connector.
    c. Make an SSH connection to the Zedboard with root account, where the IP address is 192.168.1.10, and port number of 22.
    d. In this lab, you need to login with root
3. Before using the GPIOs to control the robotic arm servos, you must tell the operating system that you are going to assign and configure the connector, defining each pin as either input or output.
   You will need to do this logged in as root. Once you are done configuring the GPIO pins and performing the desired action, you will need to free the connector to allow other modules or processes to use them. Again, you will need to be logged in as root.
    - To reserve a Pmod pin, you should issue the following command:

echo XX > /sys/class/gpio/export

where XX is the GPIO pin number. **Note:** There is a space before and after ">".

- To free the reserved pin when you finish using it, issue:

  echo XX > /sys/class/gpio/unexport

4. On the ZedBoard, to reserve pin 13, issue:

   echo 13 > /sys/class/gpio/export

   After exporting pin 13 a new set of device files are created. List the directory "**/sys/class/gpio**" and report what you see. Next, change directory to the gpio13 subdirectory. Record what you see.

   Note: 13 is a pin number that is tied to JE1 on the JE GPIO connector on the ZedBoard (these assignments are configured by the operating system)

5. To set the direction of the reserved pin, you would use:

   echo in > /sys/class/gpio/gpioXX/direction

   for an input port, or

   echo out > /sys/class/gpio/gpioXX/direction

   for an output port. Set JE1 as an output by issuing:

   echo out > /sys/class/gpio/gpio13/direction
6. Now you can read the value on the GPIO pin (a 1 or 0) by issuing:

   cat /sys/class/gpio/gpio13/value

   or you can write a value of 1 by issuing:

   echo 1 > /sys/class/gpio/gpio13/value

   or a value of 0 by issuing

   echo 0 > /sys/class/gpio/gpio13/value

7. To control the Robotic arm, you will need to assign and configure 5 GPIO pins on the JE PMOD connector. The port number map is shown Table 1. Use Steps 2 and 3 above to export each of the first 5 GPIO ports listed below and then set their direction to 'out'.

Table 1. PMOD JE (on ZedBoard) Mapping of robotic arm

PIN Name	PIN Location	Port Number in Linux	PWM Signal for Servo
JE1	Upper JE1	13	Base
JE2	Upper JE2	10	Bicep
JE3	Upper JE3	11	Elbow
JE4	Upper JE4	12	Wrist
JE7	Lower JE1	0	Gripper
JE8	Lower JE2	9	
JE9	Lower JE3	14	
JE10	Lower JE4	15	

Lab 8.2 Connecting robotic arm to the ZedBoard

IMPORTANT

- Secure the robot to the table using clamps.
- Turn off the robot while writing your program.
- Be careful when the robot moves, it can hurt you.
- Keep the ZedBoards and other equipment far from the robots.
- Review connections carefully, the robot can be damaged.
- If a servo gets hot, stop your program right away with Control+C.

In this part of the lab, you will be guided through a set of steps to connect the robotic arm to the ZedBoard.

8. Connect the interface board to the PMOD connector (JE) on the ZedBoard (see Figure 4Figure 4). Note the white wire on the cable should be on the top left. There is also a small imprinted arrow on the connector aligned with the white cable.

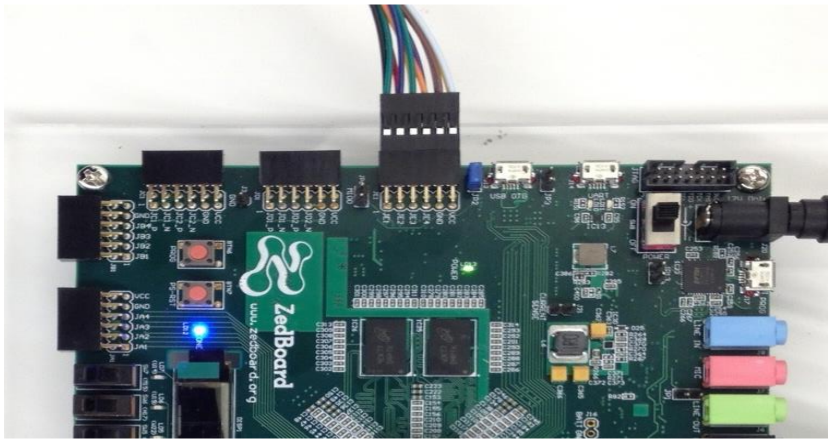
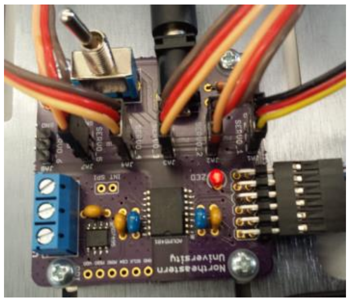


Figure 3 PMOD Connection on ZedBoard

9. Next, validate the connections on the interface board mounted on the base of the robotic arm. They should be already connected from a previous lab, but double checking avoids mistakes. See the table below for connectivity and the image for reference.

Interface Board Connector	Servo
Servo_1	Base
Servo_2	Bicep
Servo_3	Elbow
Servo_4	Wrist
Servo_5	Gripper



10. Before moving on, make sure your robotic arm is moved as far away from the ZedBoard so that it will not crash into the board. Also, make sure that the arm will move without unplugging from the ZedBoard. Also, secure your robots base using the clamp (provided in the cabinet) to clamp it to the table as shown in Figure 4. Have one of the lab supervisors check your setup before moving on.

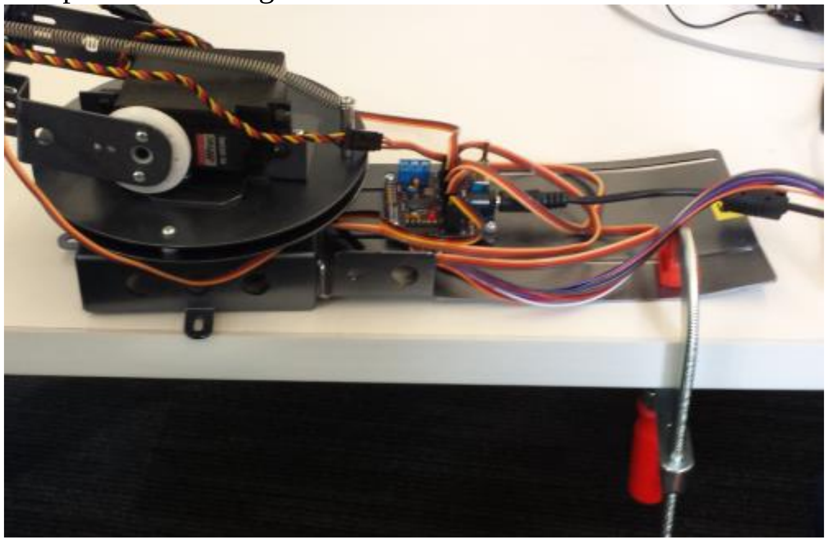


Figure 4 Clamping Robotic Arm to Table

11. Connect the robotic arm power supply to the interface board. Turn on the interface board by flipping the switch – the light should turn on.

Lab 8.3 Programming GPIO to control the robotic arm position

In this part, you will manipulate the provided C program to control the robotic arm. ***Make sure that your robot is switched OFF when you are programming, or else you can burn up a servo motor.***

12. The ZedBoard loses the GPIO configuration on each startup and starts with a default configuration. Since it was powered down when reconnecting, it has lost the configuration changes, we made in steps 3 – 6. To quickly reconfigure the settings, download the shell script "GPIO_init.sh" from Blackboard, change the permissions to make the script executable (hint use chmod) and then run it as root.
13. Inspect the shell script and report its functionality in your lab report.
14. Compile and run the PWM-thru-GPIO.c program, control the robotic arm and record the results.
15. Now write a program (servoPos.c) that takes two integers (one will be used for the servo number and the other for the position) that moves the selected servo to the desired position. Note, each servo will have different ranges of motion – welcome to the real world! You should find that range for each servo.
    Start with controlling the wrist, it is the safest motion on the robot arm.

Lab 8.4 Programming GPIO to control the robotic arm position and speed

The servo only accepts position information encoded as PWM. Given a position, the servo moves to the specified position as fast as possible regardless of the initial state. To slow down the motion, intermediate steps can be given. Then, the servo moves to each step individually. If the intermediate steps are reached with some delay in between the overall speed at which the servo moves seems to have slowed down.

16. Expand the previous program (rename to servoPosSpeed.c) such that it controls the speed of the movements. The program takes four integer numbers:
    a. servo number
    b. start position [angle in degree 0 … 180]
    c. end position [angle in degree 0 … 180]
    d. rotational speed [degree/second]

    The program should control the servo to do the following 3 steps:
    - Stay in start position for 4 seconds (i.e. generate a PWM with constant duty cycle),
    - Move from start position to end position with the specified speed.
    - Stay in end position for another 4 seconds.

    Speed is defined as the rotation speed in degree angle per second. As an example, with a speed of 18 degrees/s, in each second the servo moves by 18 degrees. Consequently, it takes 10 seconds for traveling from 0 angle to 180 angle.

Hint: Each period of PWM defines a new position for the servo. By incrementally changing the position with each new PWM period, the speed of the servo can be controlled. Use the function from Pre 8.3 to compute an updated duty cycle (on duration) for each new PWM period. Update the angle according to the speed definition. Pay attention: a new position is given with each PWM period meaning every 20ms, conversely the speed is given in degree/second (see Pre 8.4 to help you).

✓ Have some fun, but don't "hurt" your robot.

Laboratory Report

You should follow the lab report outline provided on Blackboard. Your report should be developed on a word processor (e.g., OpenOffice or LaTeX), and should include graphics when trying to present a large amount of data. Include the output of compiling and running your programs. Upload the lab report on blackboard and include the programs you wrote in the appendix of your report.