Mechatronics: Progress Report 2

ECOR1044 C: Mechatronics

Group 1044-C2-23

Group Member 1: 101241887 Uchenna Obikwelu

Performed on: 16/03/2022

Submitted on: 19/03/2022

a. *Pulse Width Modulation (PWM) and Duty Cycle.*

Pulse width modulation (PWM) which is also known as pulse-duration modulation (PDM), is a method of reducing or increasing the average power delivered by an electrical signal, by effectively breaking it into discrete parts of varying durations. The average value of voltage (and current) fed to the load is controlled by turning the (electronic) switch between supply and load on and off at a fast rate. The longer the switch is 'ON' as compared to the off periods, the higher the total power supplied to the load and vice versa.

A duty cycle, also known as a power cycle, is the fraction of one period in which the signal is "On". Duty cycle is commonly expressed as a percentage or a ratio. A period is the time it takes for a signal to complete an on-and-off cycle. Formula for a duty cycle may be expressed as: D=PW/T*100
Where P W is pulse width, and T is the period of the signal. For example, a 60% duty cycle means the signal is 'ON' 60% of the time but 'OFF' 40% of the time. Therefore, we can increase and decrease the average power delivered by a voltage source by increasing or decreasing the duty cycle of its output voltage signal.

b. *Describe the steps used to adjust the brightness of an LED (including a brief description of any functions used)*

During the procedure of the lab in order to adjust the brightness of the LED, you would need to alter the duty cycle within the given code.
pin.ChangeDutyCycle()
This is the given code you would need to change in order to alter the brightness of the LED. To change the Duty cycle, you would put a value from 0 to 100 in the parentheses, and then when that code was executed, the duty cycle of the LED was changed resulting in it's brightness to change accordingly.

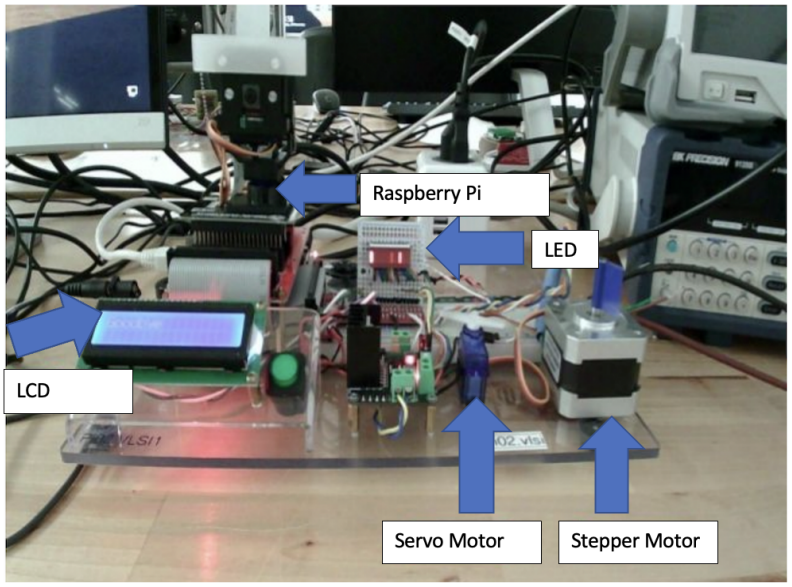
c. *If you were to use PWM on a speaker how would you expect changing the frequency of the signal to change the sound the speaker is making? How would you expect the volume of the sound to change if you changed the duty cycle from 25% to 50%?*

By changing the PWM frequency, you are able to make the buzzer produce sounds at different levels. This means that by altering the frequency you are able to make different pitches and tones. For example by setting the frequency of a piezo buzzer to 440Hz, you are able to produce an A3 note.Reference:http://justanotherlanguage.org/content/tutorial_pwm2
By changing the duty cycle of the buzzer, you are able to adjust the volume. The higher the duty cycle, the louder the buzzer will buzz. Therefore changing the duty cycle from 25% to 50 %will make the volume of the sound louder.

d. *What is the advantage of using Full Step Drive mode for a stepper motor over Wave Drive mode? Why does Full Step mode not provide a better resolution than the Wave Drive mode?*

Wave Drive Mode and Full Step Drive Mode have the same number of steps but the advantage of using FSDM is that the motor provides maximum rated torque as compared to significantly less torque than rated provided by the motor on WDM . Resolution depends on the number of steps per revolution of a stepper motor and its step angle size. So since WDM produces less torque as compared to FSDM, the torque has little effect on the accuracy of the motor provided a better resolution as compared to FSDM

e. *Attach an annotated hardware diagram of the lab setup, this should be referenced when discussing the system or components*



f. *Write a small conclusion section summarising your experience and learning from Lab 2.*

From this LAB 2, I learnt what Pulse width Modulation and Duty Cycle were and how to use them to program a stepper motor to vary the power delivered to an LED in order to change its brightness.I also learnt how to write a program to display text onto an LCD. I found this lab more fun, educative and easier than the first as I was able to apply knowledge I learnt from it. Looking forward to learning more about the raspberry pi in the next lab.

Appendix A-Experiment Scripts

6.2 Using an LED with PWM

```
import time
import RPi.GPIO as GPIO
import lcd_i2c

print("Setup")
#informs user setup has begun
stringvar1='Welcome'
stringvar2='Goodbye'

#General GPIO Setup
GPIO.setmode(GPIO.BCM) #sets how we reference GPIO pins
GPIO.setup(23, GPIO.OUT) #sets GPIO pin 23 as an output

#PWM Signal Setup
pin = GPIO.PWM(23,500) #set pin 23 as a PWM output, with a frequency of 500 Hz
pin.start(0) #sets the starting duty cycle of the PWM signal to 0% and initialises the signal
time.sleep(1) #sleep for a second to ensure signal is initialised properly

print("Begin") #informs the user the main function of the program is beginning

#Main portion of program
try:
   lcd_i2c.printer(stringvar1,") #prints to LCD
   time.sleep(2)#sleep for 2 seconds
except KeyboardInterrupt: #stops try if (ctrl+c) is pressed
   pass
lcd_i2c.cleanup()#LCD cleanup

pin.ChangeDutyCycle(50) #changes duty cycle to 50%
time.sleep(10) #sleeps for 5 seconds
try:
    lcd_i2c.printer(stringvar2,") #prints to LCD
   time.sleep(2)#sleep for 2 seconds
   lcd_i2c.cleanup()#LCD cleanup
   pin.stop() #stops the pin initialization
   GPIO.cleanup()#cleanup all the GPIO pins used within the script
except KeyboardInterrupt: #stops try if (ctrl+c) is pressed
   pin.stop() #stops the pin initialization
   GPIO.cleanup()#cleansup all the GPIO pins used within the script
   lcd_i2c.cleanup()#LCD cleanup
   print("failed")

print('Done')
```

6.3 Testing The Effect of PWM Duty Cycle with an LED

```
# Imports
import time
import RPi.GPIO as GPIO
import lcd_i2c
print("Setup") #informs user setup has begun
#General GPIO Setup
GPIO.setmode(GPIO.BCM) #sets how we reference GPIO pins
GPIO.setup(23, GPIO.OUT) #sets GPIO pin 23 as an output #PWM Signal Setup
pin = GPIO.PWM(23,500) #set pin 23 as a PWM output, with a frequency of 50 Hz
pin.start(0) #sets the starting duty cycle of the PWM signal to 0% and initialises the signal
time.sleep(1) #sleep for a second to ensure signal is initialised properly
print("Begin") #informs the user the main function of the program is beginning
# Main Script
try:
    while True:
        for dc in range(0,101,20):
            pin.ChangeDutyCycle(dc) #changes the duty cycle to dc
            time.sleep(2) #sleeps for 2 seconds
            lcd_i2c.lcd_string(str(dc),1) #Send duty cycle to display
            print(dc)

except KeyboardInterrupt: #stops try if (ctrl + c) is pressed
    pass
pin.stop() #stops the pin initialization
GPIO.cleanup() #cleansup all of the GPIO pins used within the script
lcd_i2c.cleanup()#LCD cleanup
```

6.6 Task 2

```
import RPi.GPIO as GPIO
import time import lcd_i2c

#Setup the RaspPi to use the numbering on the board
GPIO.setmode(GPIO.BCM)

controlPins = [21, 20, 12, 16]

# Setup the outputs Pins to control the stepper
GPIO.setup(controlPins[0], GPIO.OUT)
GPIO.setup(controlPins[1], GPIO.OUT)
GPIO.setup(controlPins[2], GPIO.OUT)
GPIO.setup(controlPins[3], GPIO.OUT)

# The following matrix should contain the coil energizing sequence for coils A, B, A', and B'
halfstepSequence = [ [1,0,0,0],                    [1,1,0,0],
                     [0,1,0,0],
                     [0,1,1,0],
                     [0,0,1,0],
                     [0,0,1,1],
                     [0,0,0,1],
                     [1,0,0,1],
                     ]

counter = 0
 try:        #We will now go one full rotation - notice you can see the pins turn ON through the LEDs
        for x in range(0,100,2): #Range is 50, as it needs 50 cycles through the halfstepSqeuence fora
full rotation (each sequence loop represents a rotation of 7.2 degrees)
        for halfstep in range (len(halfstepSequence)): #Loop from 0 to len (length) of the half step
sequence (8)
        for pin in range (len(controlPins)): #Loop from 0 to len
(length) of controlPins (4)
        GPIO.output(controlPins[pin - 1], halfstepSequence[halfstep][pin - 1])
        time.sleep(0.05) #Adjustable for faster or slower speed (too fast and the stepper will stop
working), if your connection is slow or you are debugging using the LEDs. Increase the value for
slower response.
        time.sleep(5)       print('Step:',counter)
counter +=1
        #We will now rotate back in the other direction          print('Returning')
        for x in range(50): #Range is 50, as it needs 50 cycles through the halfstepSqeuence for a full
rotation (each sequence loop represents a rotation of 7.2 degrees)
        for halfstep in reversed (range (len(halfstepSequence))):
#Loop from 0 to len (length) of the half step sequence (8)
        for pin in range (len(controlPins)): #Loop from 0 to len
(length) of controlPins (4)
        GPIO.output(controlPins[pin - 1], halfstepSequence[halfstep][pin - 1])
```

```
        time.sleep(0.05) #Adjustable for faster or slower speed (too fast and the stepper will stop working), if your connection is slow or you are debugging using the LEDs. Increase the value for slower response.

        lcd_i2c.string('Finished, Enjoy',1) except KeyboardInterrupt:
        pass


GPIO.cleanup()
print('Done')
```

References

1. 'ECOR1044 Lab 2 Instructions' *Bright Space ECOR 1044* [Online] Available: https://brightspace.carleton.ca/d2l/le/content/57180/viewContent/2550925/View.[Accessed: 19-Mar-2022].

2. Leonov, B., Kristoff, S., Brown, A., and Synthesis, P., 2020.PWM Sound Synthesis.[online] Hackster.io. Available at:<https://www.hackster.io/106958/pwm-sound-synthesis-9596f0> [Accessed: 19-Mar-2022].

3. Justanotherlanguage.org. 2020. PWM (Part 2) - Sound And Frequency With Piezo Buzzer| Just Another JAL Website. [online] Available at: <http://justanotherlanguage.org/content/tutorial_pwm2> Accessed: 19-Mar-2022].

.

4.]"Stepper motor," Wikipedia, 23-Nov-2020. [Online]. Available: https://en.wikipedia.org/wiki/Stepper_motor#:~:text=stepping and microstepping.-,Wave drive (one phase on),It is rarely used. Accessed: 19-Mar-2022].

5.]"Resolution of motors,"Granitedevices.com. [Online]. Available: https://granitedevices.com/wiki/Resolution_of_motors.Accessed: 19-Mar-2022].