Activity 4.2.3

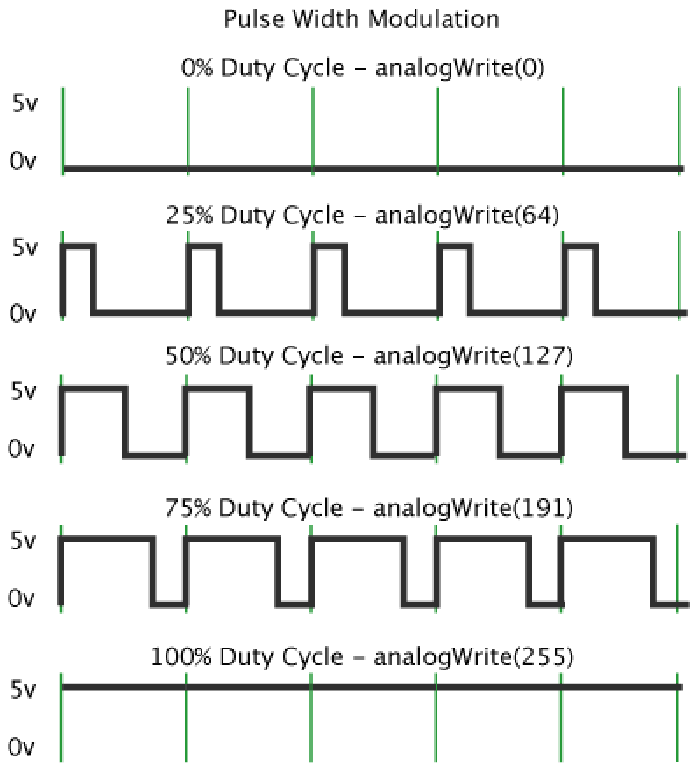
Pulse Width Modulation (PWM)

INTRODUCTION

Pulse Width Modulations (PWM) is a way of digitally encoding analog signal levels by controlling the duty cycle. In this way, a digital **microcontroller** can send varying voltages to simulate an analog signal to components such as a motor. This allows the motor to change speed smoothly instead of only operating a set speeds.

Servos are designed to receive PWM signal and translate the signal into a speed or a position. In this activity you will explore controlling motors and servos with PWM signals. We will also take a look at **accelerometers** that create PWM signals that a microcontroller can interpret into acceleration along an axis.

Microcontrollers can also receive PWM signals from input devices such as accelerometers.

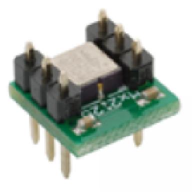

Pulse Width Modulation
0% Duty Cycle – analogWrite(0)
25% Duty Cycle – analogWrite(64)
50% Duty Cycle – analogWrite(127)
75% Duty Cycle – analogWrite(191)
100% Duty Cycle – analogWrite(255)
5v
0v

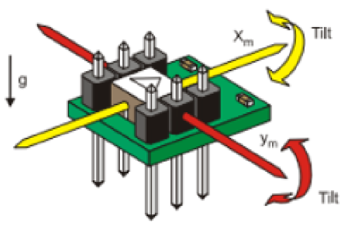
EQUIPMENT

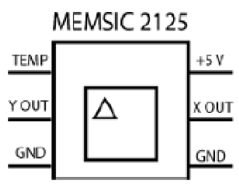
- Parallax® student DE bundle with Arduino
  - Arduino™ UNO Microcontroller Board
  - Memsic® 2125 Dual-axis Accelerometer
  - Parallax® 2-Axis Joystick
- Arduino™ IDE Software
- Breadboard
- #22-gauge solid wire
- VEX® potentiometer
- VEX® 393 Motor with Motor Controller 29

Input Device: Dual-axis Accelerometer

The Memsic 2125 is a two-axis accelerometer capable of measuring acceleration up to plus or minus 2 g. It has a simple digital interface: two pins (one for each axis) emit pulses whose duration corresponds to the acceleration of that axis. By using the Arduino's pulseIn() **function** to measure the length of that pulse in microseconds, you can determine the rate of acceleration and use that data for your purposes.







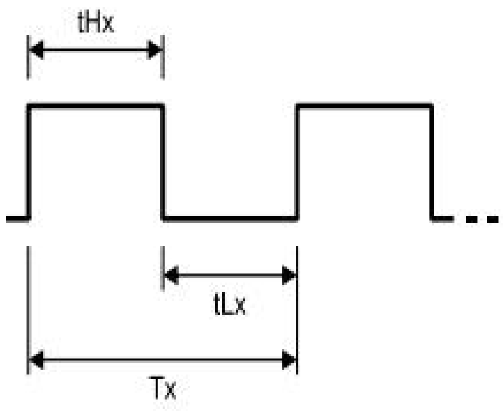
Accelerometer Features

- Measures ±3 g on each axis.
- Simple pulse output of g-force for each axis is easy for almost any microcontroller to read.

- Analog output of temperature (Tout pin) allows for fine-tuning of advanced applications.
- Low current at 3.3 or 5 V operation (less than 4 mA at 5 VDC) puts low demand on your system.

Communication Protocol

Each axis has a 100 Hz Pulse Width Modulated (PWM) duty cycle output in which acceleration is proportional to the ratio tHx/Tx.



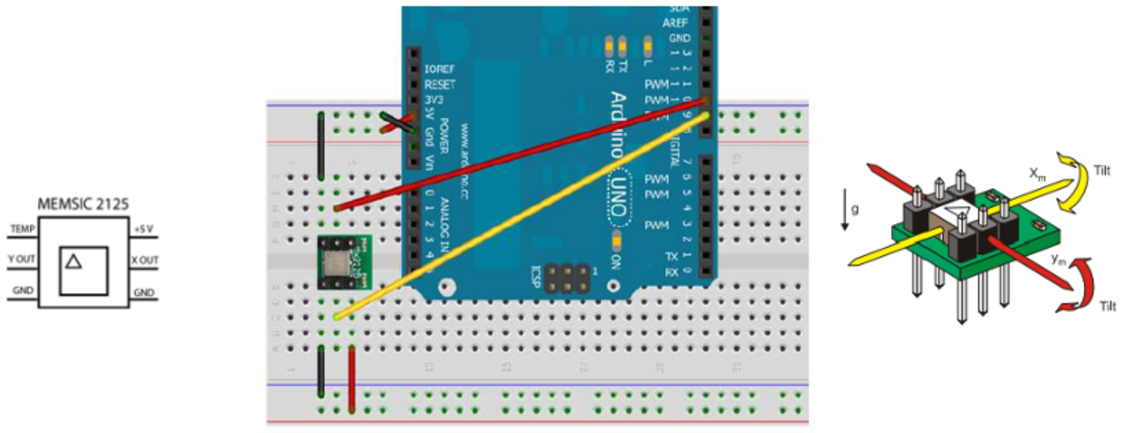
In practice, we have found that Tx is consistent, so reliable results can be achieved by measuring only the duration of tHx. With 5 V, 50% duty cycle corresponds to 0 g, but this will vary with each individual unit within a range of 48.7% to 51.3%.

RESOURCES

PWM: Pulse Width Modulation

Procedure

Sketch 1: *“Two Axis Accelerometer”*



Wiring Diagram: Memsic 2125 Dual-axis Accelerometer

In the following example code you will see:

Constants

- The X-axis assigned to pin 9 and Y-Axis assigned to pin 10.

Setup()

- The serial communication is initialized at 9600 b/s.
- The functions of the XPin and Ypin are defined as INPUTS.

Loop()

- **Variables** are defined as pulseX and pulseY. These will be the values of the pulse widths tHx.
- Variables are defined as accelerationX and accelerationY. These will be the resulting calculated accelerations.
- The pulseIn HIGH for both PulseX and PulseY will be read.
- The pulseIn will be translated into accelerations according to the following algorithms:

acceleration X = ((pulseX / 10) - 500) * 8;
acceleration Y = ((pulseY / 10) - 500) * 8;

- AccelerationX and AccelerationY are printed to the serial monitor separated by a tab character.

Sketch 1: *"Two Axis Accelerometer"*

Example Code

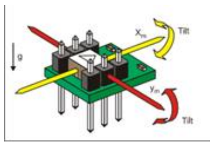
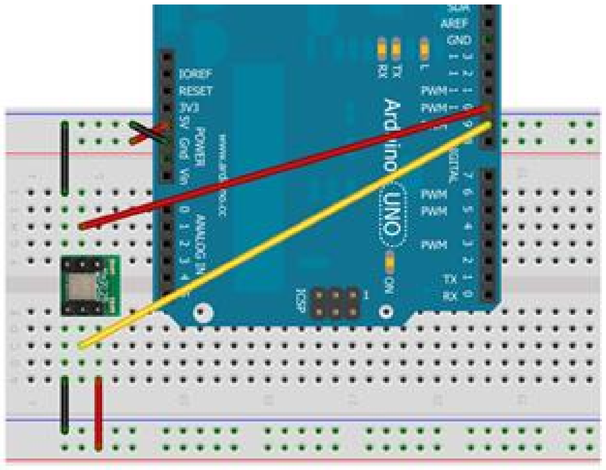
```
/*
  Memsic2125 - Read the Memsic 2125 two-axis accelerometer. Pulses output
  (1/1000) of earth's gravity) and prints them over the serial connection
  http://www.arduino.cc/en/Tutorial/Memsic2125
   created 6 Nov 2008 by David A. Mellis modified 30 Aug 2011 by Tom Igoe
  This example code is in the public domain.
 */

// Constants: Constants won't change. They're used here to set the pin nur
const int xPin = 9;                               // X output of the acceler
const int yPin = 10;                                      // Y output of the

// Variables: Variables will change. They're used do assign variable names
                                        // the variables are defined in the
// Setup: The setup routine runs once when you start or press reset:
void setup() {
  Serial.begin(9600);                                        // initialize s
  pinMode(xPin, INPUT);                           // initialize the pins cor
  pinMode(yPin, INPUT);
}

// Loop: The loop routine runs over and over again forever:
void loop() {
  int pulseX, pulseY;                                     // variables to re
  int accelerationX, accelerationY;               // variables to contain th

  pulseX = pulseIn(xPin,HIGH);                    // read pulse from x- and
  pulseY = pulseIn(yPin,HIGH);
  // convert the pulse width into acceleration
  // accelerationX and accelerationY are in milli-g's
  // earth's gravity is 1000 milli-g's, or 1g
  accelerationX = ((pulseX / 10) - 500) * 8;
  accelerationY = ((pulseY / 10) - 500) * 8;

  Serial.print(accelerationX);                    // print the acceleration
  Serial.print("\t");                                     // print a tab cha
  Serial.print(accelerationY);
  Serial.println();

  delay(100);
 }
```

1 Using the Arduino and a Memsic 2125 Dual-axis Accelerometer, create the circuit.

2 Open the Arduino software and copy/paste the example ***"Two Axis Accelerometer"*** into a new sketch. Save.

3 Title the project and add a description.

4 Upload the **program** and open the serial monitor. With the breadboard on a flat surface, determine:
   a. What values to you read for the x-axis?
   b. What values to you read for the y-axis?

Two Axis Accelerometer

5 Pick up the breadboard while keeping it parallel to the floor. Slowly rotate the board on the y-axis (part of the board farthest from you is down; part of the board from nearest you is up). Then slowly rotate the board in the opposite direction around the y-axis. Describe how the motion relates to the values you see on the Arduino serial monitor.

6 Slowly rotate the board on the x-axis (left part of the board is down; right part of the board is up). Then slowly rotate the board in the opposite direction around the x-axis. Describe how the motion relates to the values you see on the Arduino serial monitor.

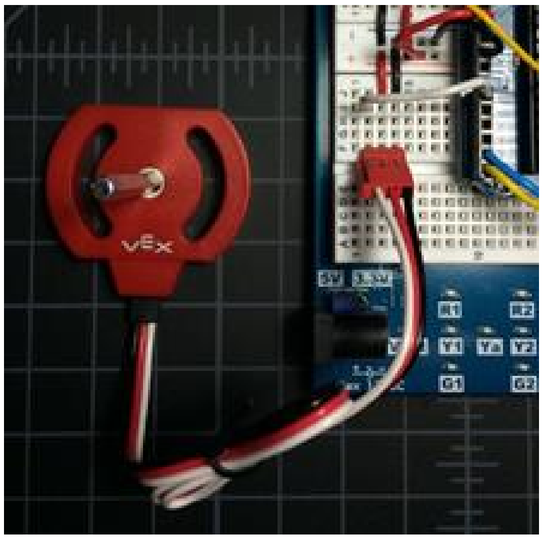
7 Have your instructor verify that the circuit works as expected.

Output Device: VEX 393 Motor with Motor Controller 29

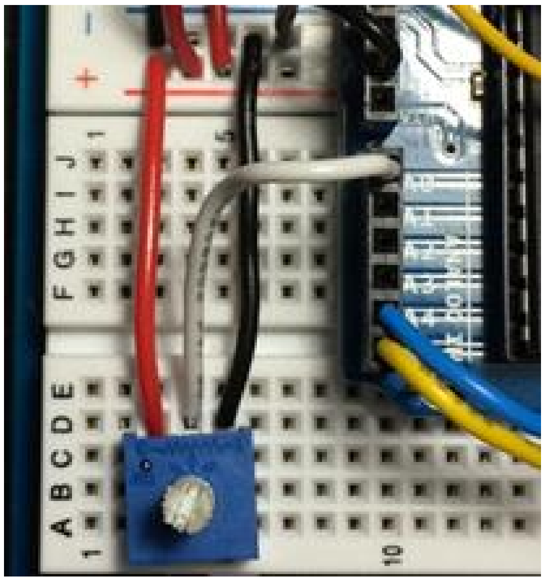
With the VEX 393 Motor with Motor Controller 29, you can control the position of the motor or the speed of the motor by sending a PWM signal to the Motor Controller 29 (which controls the motor). There are a few ways to do this:

- Non-continuous rotation servos are usually looking for values that correspond to an angle (0–180). This makes it easy to program a servo to move to a position.
- For continuous rotation servos (VEX 393 Motor with Motor Controller 29), the Arduino libraries allow you to translate this range of (0–180) into directional motion.
  - 90 degrees: stop motor
  - <90 degrees: motor forward
  - >90 degrees: motor reverse

Sketch 2 *"ServoPot"*



VEX Potentiometer with PWM



Potentiometer

8 Using the Arduino and a VEX 393 Motor with Motor Controller 29, create the circuit.

9 Open the Arduino software and copy/paste the example **ServoPot** into a new sketch. Save.

10 Title the project and add a description.

```
/*
 ServoPot: Controlling a servo motor using a potentiometer (variable resis
 Reads an analog input on pin A0 and moves a VEX 393 2-wire motor with a M
 With the sketch uploaded, open the serial monitor to see the values. (Too
 This example code is in the public domain.
 */

#include <Servo.h>
Servo myservo;                          // create servo object to control
// Constants: Constants won't change. They're used here to set pin numbers
int potpin = A0;                        // analog pin used to connect the

                                       // the servo is assigned to pin (6)

// Variables: variables will change:
int val;                               // variable to read the value from t

void setup()

{
  myservo.attach(6);                   // attaches the servo on pin 6 to t
  Serial.begin(9600);                  // initialize serial communication
}

void loop()
{
val = analogRead(potpin);             // reads the value of the potentiomet
val = map(val, 0, 1023, 40, 130);      // scale it to use it with the servo
myservo.write(val);                   // sets the servo position according
Serial.println(val);                  // print out the value between 0 and
delay(15);                            // waits for the servo to get there
}
```

Map

Map allows you to translate one scale into another. Examples:

```
val = map(val, 0, 255, 0, 180);                 // converts (0-255) scale
val = map(val, 0, 1023, 0, 180);                // converts (0-1023) scale
```

11 Upload the program and open the serial monitor. Rotate the potentiometer and note how the motor reacts.

12 The motor most likely does not respond the way that you want. The range that translates into the motor speed is actually a much smaller range than (0–180). Modify the code so that the potentiometer controls the speed and direction over a much wider range of motion for the potentiometer.

Pulse Width Modulation

Possibly a better way to control the motor would be to abandon the (0–180) scale (since we are not trying to use the servo to set a position anyway). Remember that the motor controller 29 is translating a motor speed and direction based on the pulse duration coming from the microcontroller.

- PWM Input of 1.5 ms (1500 µs): stop motor
- PWM Input of 1 ms (1000 µs): motor forward
- PWM Input of 2 ms (2000 µs): motor reverse

Sketch 3 *"microServoPot"*

Example Code

```
/*
microServoPot: Controlling a servo motor using a potentiometer (variable r
Reads an analog input on pin A0 and moves a VEX 393 2-wire motor with a Mo
With the sketch uploaded, open the serial monitor to see the values. (Too
This example code is in the public domain.
*/
#include <Servo.h>
Servo myservo;                                // create servo object to co
int potpin = A0;                                // analog pin used to conn
int val;                                        // variable to read the va
void setup()
{
 myservo.attach(6);                             // attaches the servo on p
 myservo.writeMicroseconds(1500);       // set servo to mid-point
 Serial.begin(9600);                            // initialize serial commu
}
void loop()
{
 val = analogRead(potpin);              // reads the value of the potentio
 val = map(val, 0, 1023, 1000, 2000);   // scale it to use it with the ser
 myservo.writeMicroseconds(val);        // sets the servo position accordi
 Serial.println(val);                           // print out the value bet
 delay(15);                                     // waits for the servo to
}
```

13 Open the Arduino software and copy/paste the example ***"microServoPot"*** into a new sketch.

14 Title the project and add a description. Save.

15 Upload the program and open the serial monitor. Rotate the potentiometer and note how the motor reacts.

16 Have your instructor verify that the circuit works as expected.

CONCLUSION

Imagine a transportation device that consists of two motors and an accelerometer. By leaning forward on the platform, the vehicle moves forward. By leaning back, the vehicle move backwards.

1. Create a demonstration using the accelerometer and a motor to demonstrate how this might work.
2. Have your instructor verify that the circuit works as expected and submit a copy of your program.

Proceed to problem