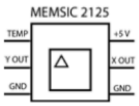
PLTW Engineering **Digital Electronics Manufacturing**

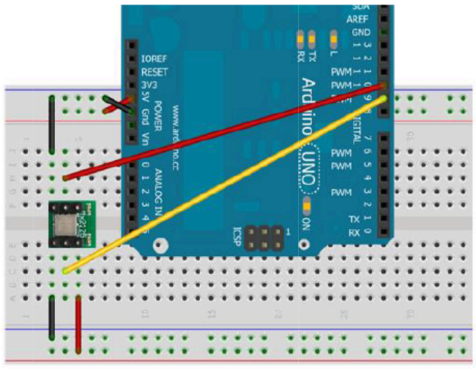
Jeide, Matthew 3/31/2025 Period 2

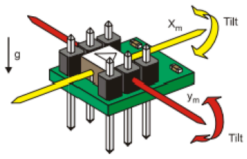
W11B: DE [Memsic 2125: Two-Axis Accelerometer] Arduino Variables Tutorial

Introduction

Programming languages have their own grammar called "syntax". Programs written with the Ardiuno software are called Sketches. A **Sketch** (program written with **Arduino**) will contain: a title, **constants**, **variables**, setup() functions, and loop() functions.







If the syntax of a language is not followed, the program will not compile correctly. This means that no executable code will be produced. Fortunately, the **Arduino** integrated development environment (IDE) will provide error messages that will help you fix your "bad grammar"… called "syntax errors". One of the most common syntax errors that students make is forgetting that lines of code need to end with a semicolon.

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 speed.

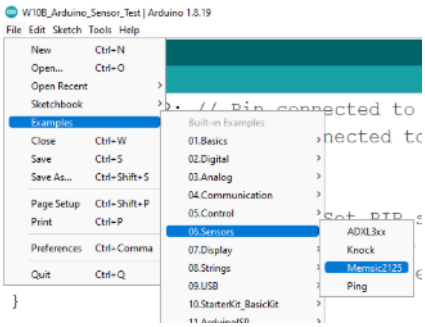
Servosare designed to receive PWM signal and translate the signal into a speed or a position. In this activity you will explore controlling motors and servos with PWM signals. We will also take a look at **accelerometers** that create PWM signals that a microcontroller can interpret into acceleration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

Equipment

- https://my.pltw.org
- Computer with Arduino Software
- **Use the Arduino Examples→ 06 Sensors→Memsic2125**
  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

- Use the **Arduino** Examples→ 06 Sensors→Memsic2125

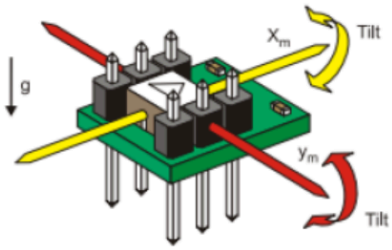


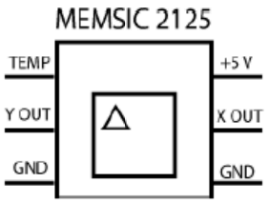
Procedure

Introduction: Create a "New Sketch" and enter the code from the Sensor Examples:Memsic2125.

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.

Type the code on the Arduino Software. Make sure to review the diagram above and connect correctly.

A. Use the link provided to understand the use of variables using C++ Code. Use the Arduino to program the code and the reference guide: **REFERENCE LINK:**

https://www.youtube.com/watch?v=HYUYbN2gRuQ&authuser=0

Copy and paste the code from the Arduino software → File—> Examples→ 06Sensor→Memsic2125

```
/*
  Memsic2125

  Read the Memsic 2125 two-axis accelerometer. Converts the pulses output by the
  2125 into milli-g's (1/1000 of Earth's gravity) and prints them over the
  serial connection to the computer.

  The circuit:
  - X output of accelerometer to digital pin 2
  - Y output of accelerometer to digital pin 3
  - +V of accelerometer to +5V
  - GND of accelerometer to ground

  created 6 Nov 2008
  by David A. Mellis
  modified 30 Aug 2011
  by Tom Igoe

  This example code is in the public domain.

  https://www.arduino.cc/en/Tutorial/BuiltInExamples/Memsic2125
*/

const int xPin = 2;
```

```
const int yPin = 3;

void setup() {
  Serial.begin(9600);
  pinMode(xPin, INPUT);
  pinMode(yPin, INPUT);
}

void loop() {
  int pulseX, pulseY;
  int accelerationX, accelerationY;

  pulseX = pulseIn(xPin, HIGH);
  pulseY = pulseIn(yPin, HIGH);

  accelerationX = ((pulseX / 10) - 500) * 8;
  accelerationY = ((pulseY / 10) - 500) * 8;

  Serial.print(accelerationX);
  Serial.print("\t");
  Serial.print(accelerationY);
  Serial.println();

  delay(100);
}
```

B. Using the code above. Explain what the numbers represent. Refer to the video for help.
https://www.youtube.com/watch?v=HYUYbN2gRuQ&authuser=0

Comment on the code. Make sure to review the video for help.

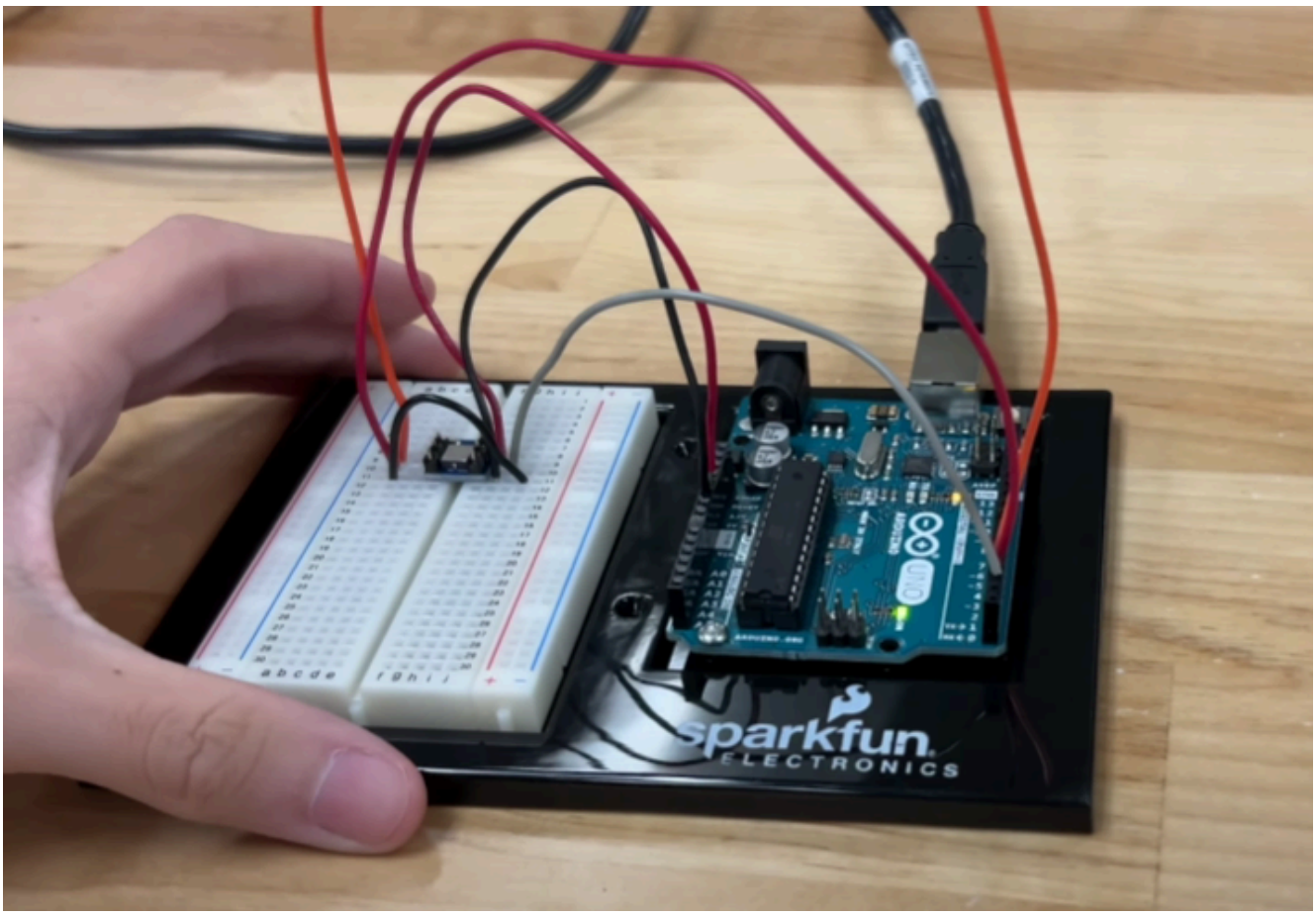
```
/*
  Memsic2125
```

```
  Read the Memsic 2125 two-axis accelerometer. Converts the pulses output by the
  2125 into milli-g's (1/1000 of Earth's gravity) and prints them over the
  serial connection to the computer.

  The circuit:
  - X output of accelerometer to digital pin 2
  - Y output of accelerometer to digital pin 3
  - +V of accelerometer to +5V
  - GND of accelerometer to ground

  created 6 Nov 2008
  by David A. Mellis
  modified 30 Aug 2011
  by Tom Igoe

  This example code is in the public domain.

  https://www.arduino.cc/en/Tutorial/BuiltInExamples/Memsic2125
*/

// these constants won't change:
const int xPin = 2;  // X output of the accelerometer
const int yPin = 3;  // Y output of the accelerometer

void setup() {
  // initialize serial communications:
  Serial.begin(9600);
  // initialize the pins connected to the accelerometer as inputs:
  pinMode(xPin, INPUT);
  pinMode(yPin, INPUT);
}

void loop() {
  // variables to read the pulse widths:
  int pulseX, pulseY;
```

```
  // variables to contain the resulting accelerations
  int accelerationX, accelerationY;

  // read pulse from x- and y-axes:
  pulseX = pulseIn(xPin, HIGH);
  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 1 g.
  accelerationX = ((pulseX / 10) - 500) * 8;
  accelerationY = ((pulseY / 10) - 500) * 8;

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

  delay(100);
}
```

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

Insert a photograph of the completed 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. Make sure to write comments on the code.

Enter the Arduino Examples→ 06 Sensors→Memsic2125 Sample Code

```
const int servoPin = 6;   // Servo connected to digital pin (PWM)

/*
 * Memsic 28017 Dual-Axis Accelerometer Serial Output
 *
 * This sketch reads the PWM outputs from a Memsic 28017 accelerometer
 * and calculates acceleration based on duty cycle.
```

```
 */

// Define pins connected to the Memsic accelerometer
const int xoutPin = 2;  // X-axis output connected to digital pin 2
const int youtPin = 3;  // Y-axis output connected to digital pin 3
const int toutPin = 4;  // Temperature output connected to digital pin 4 (optional)

// Variables for pulse measurements
unsigned long xPulseHigh;  // X-axis HIGH pulse duration (tHx) in microseconds
unsigned long xPulsePeriod; // X-axis total period (Tx) in microseconds
unsigned long yPulseHigh;  // Y-axis HIGH pulse duration (tHy) in microseconds
unsigned long yPulsePeriod; // Y-axis total period (Ty) in microseconds
unsigned long tPulseHigh;  // Temperature HIGH pulse duration in microseconds
unsigned long tPulsePeriod; // Temperature total period in microseconds

// Variables to store calculated values
float xDutyCycle;  // X-axis duty cycle (tHx/Tx)
float yDutyCycle;  // Y-axis duty cycle (tHy/Ty)
float tDutyCycle;  // Temperature duty cycle
float xAccel;      // X-axis acceleration in g
float yAccel;      // Y-axis acceleration in g
float tempC;       // Temperature in Celsius

// Calibration values (you may need to adjust these based on your specific unit)
float xZeroGDutyCycle = 0.5;  // Duty cycle at 0g for X-axis (ideally 50%)
float yZeroGDutyCycle = 0.5;  // Duty cycle at 0g for Y-axis (ideally 50%)
float tRefDutyCycle = 0.5;    // Duty cycle at reference temperature
float accelSensitivity = 0.2; // Sensitivity in duty cycle change per g

void setup() {
  // Set pin modes
  pinMode(xoutPin, INPUT);
  pinMode(youtPin, INPUT);
  pinMode(toutPin, INPUT);

  // Initialize serial communication
```

```
  Serial.begin(9600);
  Serial.println("Memsic 28017 Accelerometer Data");
  Serial.println("X-Accel (g), Y-Accel (g), Temp (C)");

  // Calibrate the zero g offset by taking initial readings
  calibrateZeroOffset();
}

void loop() {
  // Measure X-axis pulse
  xPulseHigh = pulseIn(xoutPin, HIGH);
  xPulsePeriod = xPulseHigh + pulseIn(xoutPin, LOW);

  // Measure Y-axis pulse
  yPulseHigh = pulseIn(youtPin, HIGH);
  yPulsePeriod = yPulseHigh + pulseIn(youtPin, LOW);

  // Measure temperature pulse (if connected)
  tPulseHigh = pulseIn(toutPin, HIGH);
  tPulsePeriod = tPulseHigh + pulseIn(toutPin, LOW);

  // Calculate duty cycles
  xDutyCycle = (float)xPulseHigh / (float)xPulsePeriod;
  yDutyCycle = (float)yPulseHigh / (float)yPulsePeriod;
  tDutyCycle = (float)tPulseHigh / (float)tPulsePeriod;

  // Calculate acceleration in g (based on duty cycle)
  // According to the documentation, acceleration is proportional to the duty cycle
  // At 0g, duty cycle is approximately 50%
  xAccel = (xDutyCycle - xZeroGDutyCycle) / accelSensitivity;
  yAccel = (yDutyCycle - yZeroGDutyCycle) / accelSensitivity;

  // Temperature calculation (approximate, may need calibration)
  tempC = 25.0 + (tDutyCycle - tRefDutyCycle) / 0.0037; // ~0.37% duty cycle change
per °C
```

```
  // it’s meant to be able to do this but this temperature is wildly inaccurate

  // Output to serial monitor
  Serial.print("X Duty Cycle: ");
  Serial.print(xDutyCycle * 100, 2);
  Serial.print("%, Y Duty Cycle: ");
  Serial.print(yDutyCycle * 100, 2);
  Serial.print("%, ");

  Serial.print("X Accel: ");
  Serial.print(xAccel, 2);
  Serial.print("g, Y Accel: ");
  Serial.print(yAccel, 2);
  Serial.print("g, Temp: ");
  Serial.print(tempC, 1);
  Serial.println("°C");

  // Short delay before next reading
  delay(200);
}

void calibrateZeroOffset() {
  // Take multiple readings to establish the zero g offset
  const int numSamples = 5;
  float xSum = 0, ySum = 0;

  Serial.println("Calibrating zero g offset...");
  Serial.println("Keep the sensor level and still");

  // Short delay to position the sensor
  delay(1000);

  for (int i = 0; i < numSamples; i++) {
    // Measure X-axis pulse
    xPulseHigh = pulseIn(xoutPin, HIGH);
```

```
    xPulsePeriod = xPulseHigh + pulseIn(xoutPin, LOW);

    // Measure Y-axis pulse
    yPulseHigh = pulseIn(youtPin, HIGH);
    yPulsePeriod = yPulseHigh + pulseIn(youtPin, LOW);

    // Calculate duty cycles
    xSum += (float)xPulseHigh / (float)xPulsePeriod;
    ySum += (float)yPulseHigh / (float)yPulsePeriod;

    delay(100);
  }

  // Set the zero g duty cycle values based on the average readings
  xZeroGDutyCycle = xSum / numSamples;
  yZeroGDutyCycle = ySum / numSamples;

  Serial.print("Calibration complete. X zero: ");
  Serial.print(xZeroGDutyCycle * 100, 2);
  Serial.print("%, Y zero: ");
  Serial.print(yZeroGDutyCycle * 100, 2);
  Serial.println("%");
  delay(1000);
}
```

4. Upload the **program** and open the serial monitor.

5. Place the breadboard on a flat surface and document the x-axis and y-axis inputs.

6. Pick up the breadboard while keeping it parallel to the floor. Slowly rotate the board around 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.

In the space below, describe how the motion relates to the values you see on the Arduino serial monitor.

Tilting the circuit in various directions results in different measurements from the Memsic sensor, aka,

the memsic sensor is able to detect movement in the form of tilt.

7. Slowly rotate the board around 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.

8. Verify with your instructor that the circuit works as expected.

9. E-Portfolio video with updated code.

E-Portfolio Published link with video file. Upload the file to your Google Drive to upload on your Portfolio. YouTube Videos are preferred.
https://sites.google.com/riversideunified.org/matthewjeide/notes/w11b-de-memsic2125-two-axis-accelerometer-arduino-variables-tutorial

Conclusion

Answer in complete sentences each of the questions below.

1. How can this sensor be used to design a project?

This sensor would be useful in a project that required information such as tilt, such as a navigation system for a drone, robot, or plane. The data from this device could be used to make real-time adjustments to avoid failures such as crashes.