Controlling VPython Simulations via an Accelerometer

Laura Lee and Aaron Titus
Department of Chemistry and Physics, High Point University, High Point, NC

Abstract

This experiment involved using an Arduino microcontroller to facilitate input from an accelerometer in order to control a VPython simulation. A Memsic 2125 dual-axis accelerometer was connected to an Arduino microcontroller, where the X-axis and Y-axis pulse width modulation signals were sent to. Serial communication was used to record these values and calculate the acceleration on the X-axis/Y-axis in an Arduino program. This program then used linear approximation to calculate the tilt in degrees on both axes. A VPython simulation then used these tilt values to rotate a virtual board in order to move a ball into a target ring.

Background

Pulse-width modulation (PWM) works digitally to elicit an analog result. Using digital controls, one can allow full DC supply to be either fully on or fully off, generating a square wave. This on-off pattern can be supplied as a repeating series of on and off pulses. The ”pulse width” therefore would be the time during which the DC supply is fully on. By fluctuating the pulse width, one can simulate various voltages between 5V (full on) and 0V (off).

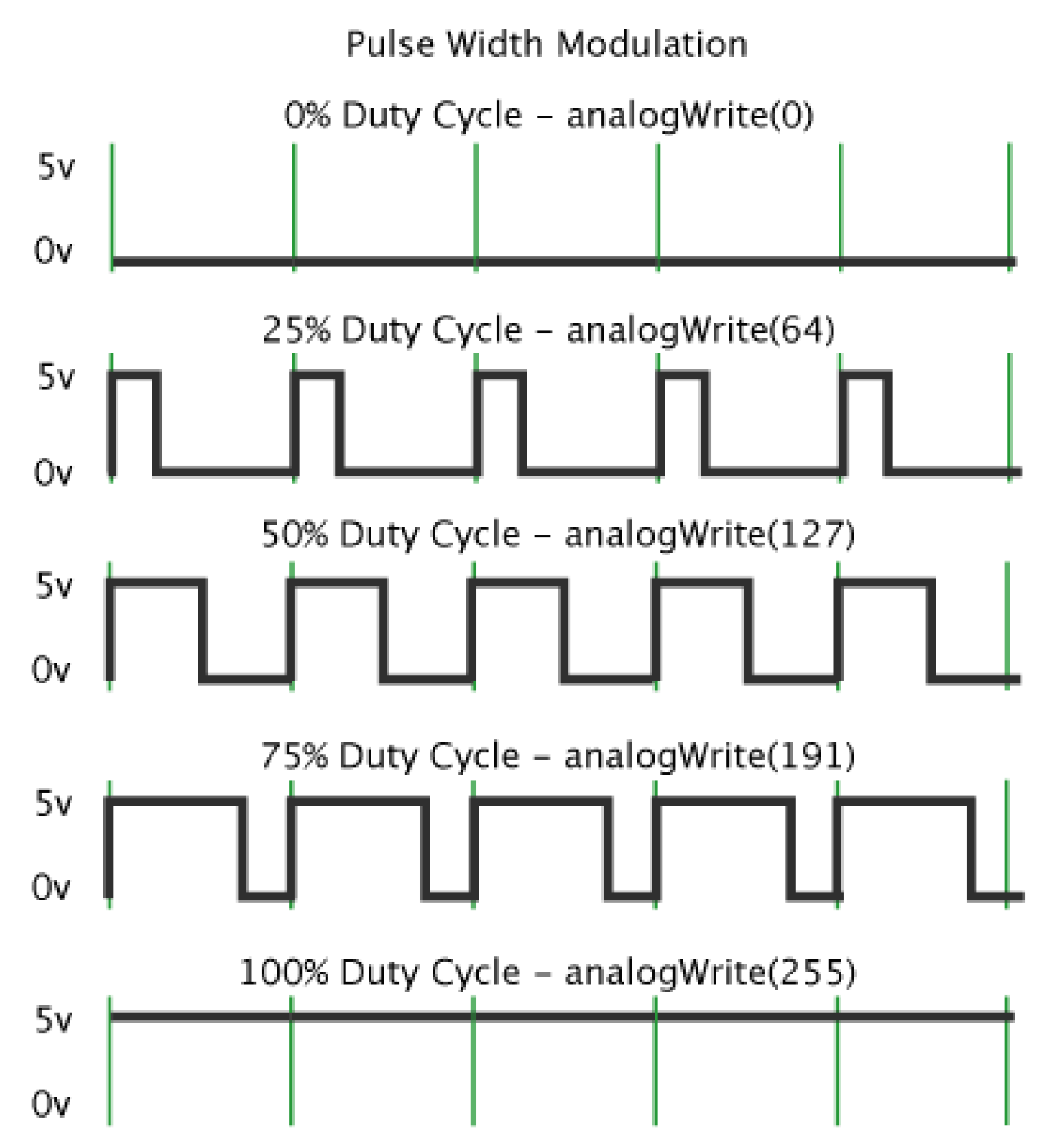
Pulse Width Modulation
0% Duty Cycle - analogWrite(0)
25% Duty Cycle - analogWrite(64)
50% Duty Cycle - analogWrite(127)
75% Duty Cycle - analogWrite(191)
100% Duty Cycle - analogWrite(255)
5v
0v

Fig. 1: PWM using Arduino functions

In the figure above, the Arduino microcontroller is using the PWD technique by calling analogWrite() to control the duty cycle. For example, analogWrite(255) generates a signal that is always on, thus a 100% duty cycle. Furthermore, a 50% duty cycle would be a signal that was on half the time.

Acceleration/Tilt Calculation

The Memsic 2125 dual-axis accelerometer works through a small heater, temperature sensors and a chamber of gas, all built into the device. While the device is level, the warm bubble of air shifts to the middle of the chamber, causing all surrounding sensors to read the same temperature level. However, when the accelerometer is tilted, this gas will shift closer to one particular temperature probe compared to the others. Thus, the Memsic accelerometer can analyze the various temperatures measured and convert them into PWD signals which a microcontroller can use.

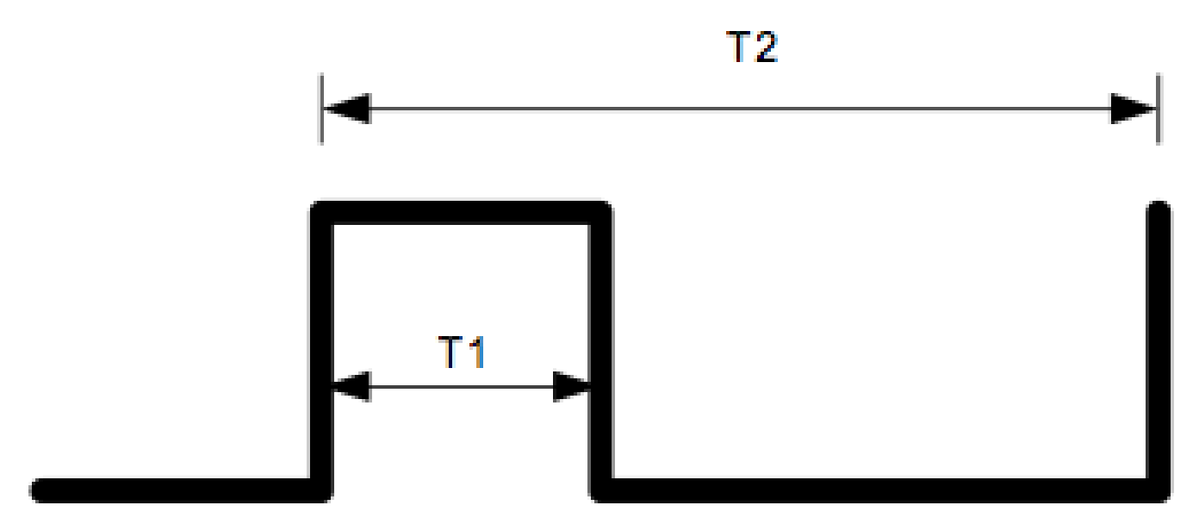

Fig. 2: Memsic output duty cycle

At 0 g, the Memsic 2125 is set to output a pulse of a 50% duty cycle, which will change in proportion to the acceleration. The formula (1) for calculating the specific acceleration at a particular tilt is listed below. T1 is the pulse width value. T2 is the length of the total cycle, factory calibrated to be 10 milliseconds at 25°. The 12.5% relates to the sensitivity scale factor (duty cycle change per g).

$$A(g) = (T1/T2 - 0.5)/12.5\% \tag{1}$$

Once the acceleration of both axes is known, the tilt angles (α and β) of these axes can be calculated by looking at the relationships between accelerometer outputs (Ax and Ay) and gravity (g).

$$\alpha = sin^{-1}(Ax/g) \tag{2}$$

$$\beta = sin^{-1}(Ay/g) \tag{3}$$

For very small inclination angles, this sine function is very close to a linear function. For applications such as this one, a linear function can actually act a relatively accurate approximation. In this experiment, a k value of 57.50 (°arc/g) was used.

$$\alpha = k \cdot Ax \tag{4}$$

$$\beta = k \cdot Ay \tag{5}$$

Experiment

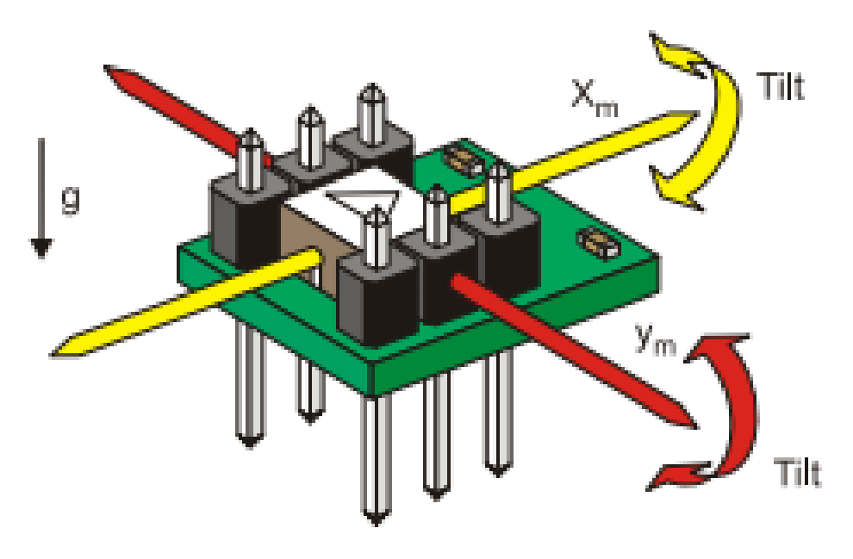

Fig. 3: Memsic Dual-Axis Accelerometer

Both the X-axis and Y-axis outputs were connected the Arduino Uno microcontroller. The Arduino used serial communication to record the PWD signals coming in as inputs. Employing the Arduino's pulseIn() function, the pulse width was measured and implemented into the acceleration equation (1). Once the acceleration for both axes were calculated, the approximate tilt was sent to a VPython program.

The VPython simulation was designed to display a virtual board, ball and ring target located at center of the board. It was set up to then serially ask for a set of values: the tilt angles of the x-axis and y-axis of the accelerometer. Defining the initial angles to be 0 radians, the simulation calculated the difference of the measured angles and the initial angles. It then would rotate the board along these axes by the calculated increment. These steps were looped to allow fluid rotation of the board.

```
Fgrav = -ball.m*g*yaxis
Fnet = Fgrav - dot(Fgrav,normal)*normal
ball.pi=ball.p
ball.p = ball.p + Fnet*deltat
ball.pos = ball.pos + (ball.p+ball.pi)/2/ball.m*deltat
```

Fig. 4: Portion of VPython Code

In order to manipulate the movement of the ball, a normal vector was initially defined as <0, 1, 0>. Once data from the accelerometer was beginning to be gathered, the simulation changed the vector to match the rotation vector on the board. The net force on the ball was then calculated by:

$$F_{net} = F_{grav} - (F_{grav} \cdot \vec{normal}) \times \vec{normal} \tag{6}$$

Using this net force, the momentum and position of the ball could be calculated. This entire process was looped along with the rotation calculations.

Results

A Memsic dual-axis accelerometer was successful in inputting PWM data serially through the Arduino Uno Microcontroller (see Figure 5).

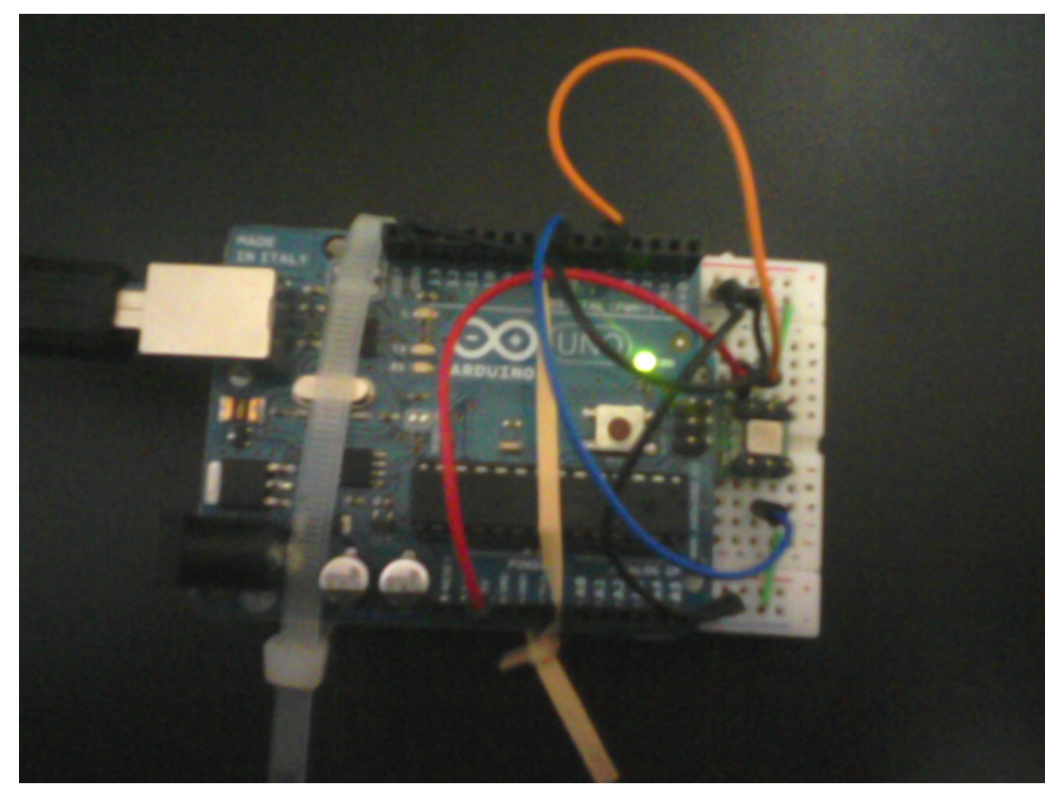

Fig. 5: Memsic 2125 Connected to Arduino

By tilting the completed device along the X-axis or Y-axis, the virtual board on the VPython simulation would tilt/rotate in a similar manner. Once the user successfully tilted the device such that the virtual ball would be forced into the target ring, the simulation would stop.

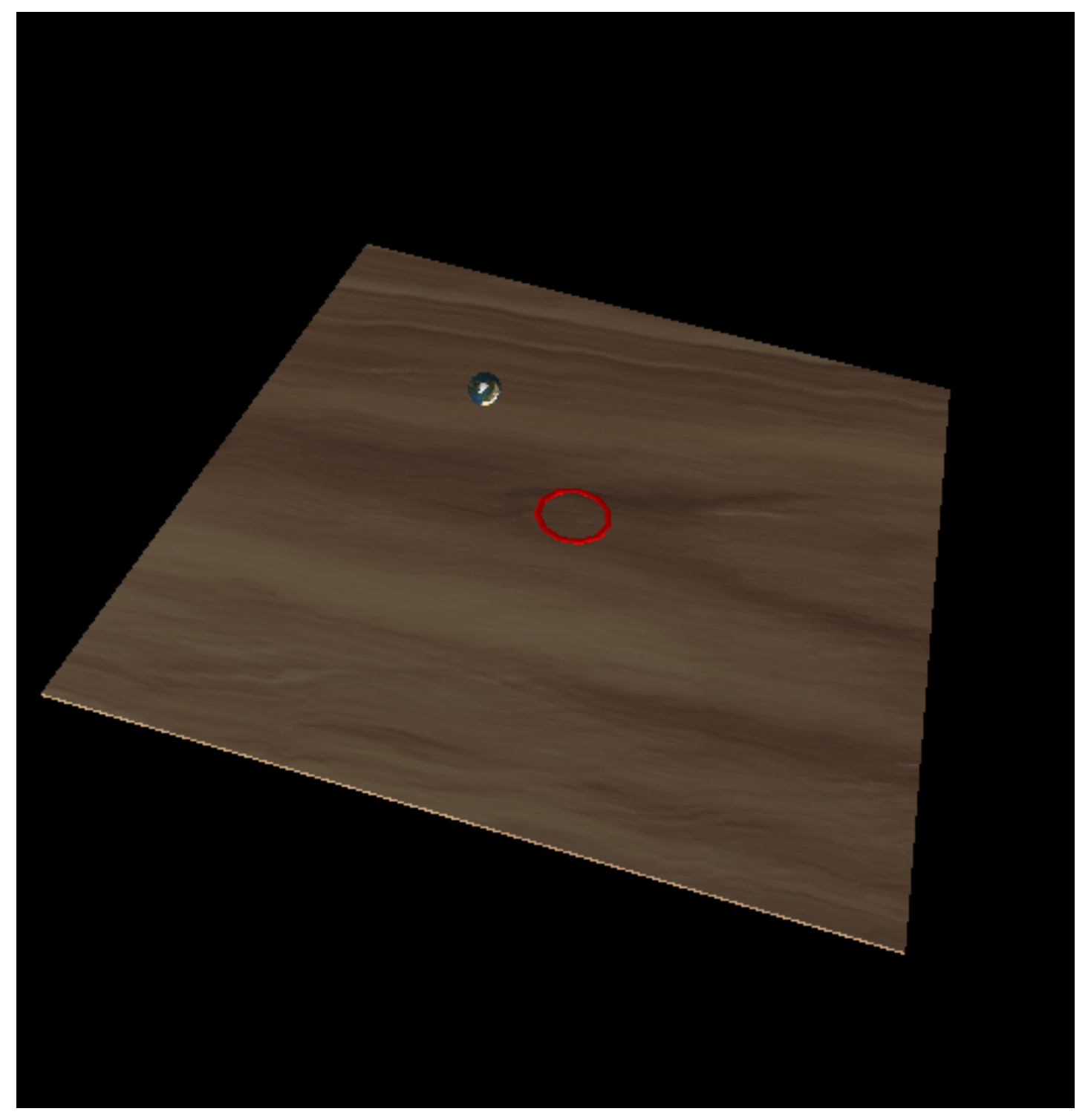

Fig. 6: VPython Simulation - Ball on Rotating Board

References

• Parallax Inc - Memsic Accelerometer Documentation (http://www.parallax.com)
• Arduino Documentation (arduino.cc/en/)

Acknowledgements

• The Department of Chemistry and Physics and the Department of Mathematics and Computer Science
• Dr. Gray Bowman, Chair of Department of Chemistry and Physics