University Physics 1
Lab 02
Linear Motion

1 Objective

Measure, display, analyze, and describe motion in one dimension.

2 Equipment List

Item	Quantity
Arduino mini computer	1
Apple Computer	1
usb cable	1
Wires	4 or 5
Vernier Motion Detector	1
Low Friction Cart	1
Cart Track	1
Track Angle Indicator	
Basketball	1

3 Theory

We have looked at 1-dimensional motion in class. The equations describing linear motion with a constant acceleration are

$$x_f = x_0 + v_{x0}\Delta t + \frac{1}{2}a_x(\Delta t)^2$$

$$v_{xf}^2 = v_{x0}^2 + 2a_x\Delta x$$

and

$$\Delta x_f = \frac{v_{avg}}{\Delta t}.$$

These are the equations that describe motion in a line with constant acceleration (keep in mind 0 m/s^2 is a constant acceleration). We will look at three situations with a constant acceleration: a cart moving on a level track, a cart moving on a tilted track, and a basketball rolling down a tilted track. We will measure the location of these objects as a function of time and use that data to describe the motion of each object.

4 Starter Code

We will use the Vernier Motion Detector to determine the location of an object along a line. The motion detector uses an ultrasonic pulse to provide the information needed to measure distance. The motion detector itself does not measure distance. It sends out a pulse and indicates when the echo comes back. That is all this device does. We will have to use this information to determine how far the pulse went. You have a starter code for this measurement on Black Board. Open this code up and lets go through it.

After a long comment at the top from Vernier Software, there are two lines of code before we get to the setup() function. These two lines define two variables that will be used to define the pin on the Arduino used to trigger a pulse from the detector and the pin used to note when the detector measures the echo coming back. You can simply use the numbers "3" and "2" in the code if you like, but using these descriptive variables makes your code easier to follow when reading it.

```
const int TriggerPin = 3; //trigger pin
const int EchoPin = 2;// echo pin
```

The two lines note that these are constant integers that will not change value as the code runs.

The setup() function follows and it contains a lot of code. The first three lines of code tell the Arduino to setup pins TriggerPin and EchoPin as OUTPUT and INPUT pins respectively. This is important as by default the pins are always OUTPUT pins. It is good form to make things obvious in the code when you can. The third line initiates the serial communications between the Arduino and the computer to allow us to send data back to the computer.

```
  // initialize the Ping pin as an output:
  pinMode(TriggerPin, OUTPUT);
  pinMode(EchoPin, INPUT); //this is the pin that goes high when an echo is
    received
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
```

The next lines of code set up headers for the columns of data that will be reported. The first three lines of the header will read:

Vernier Format 2
Motion Detector Readings taken using Ardunio
Data Set

The next three lines give a description of the quantities in each column. This is followed by three lines indicating the symbol used to represent each quantity. The last three lines give the units used in reporting each quantity (very, very, very important). Note that the print command on the third line in these sets is a Serial.println() which moves to the next line after the print is complete. The middle line in each set provides a tab between each of the columns. The tab divider will allow the data to be easily cut and pasted into a spread sheet. While the columns will look fine in a spreadsheet, they will not always lineup in the Monitor window.

```
  Serial.print("Time for Echo");//long name
  Serial.print("\t"); //tab character
  Serial.println ("Distance"); //long name
  Serial.print("delta t");//short name
```

```
Serial.print("\t"); //tab character
Serial.println ("D"); //short name
Serial.print("seconds");//units
Serial.print("\t"); // tab character
Serial.println ("centimeters"); //units
```

```
Time for Echo    Distance
delta t     D
seconds     centimeters
```

The loop() function comes next in the code. This function will carry out the measurement and report the results. Five new variables are defined in the first five lines of code. These variables will be used in the loop() function to determine the distance to an object.

```
long time; // clock reading in microseconds
long Duration; // time it take echo to return
const float SpeedOfSound = 340; //in m/s
float Distance;// in centimeters
int val = 0;
```

The long definitions for the fist two variables allow them to represent very large integers. An int can hold integers between -32,768 to 32,767 while a long can hold integers between -2,147,483,648 to 2,147,483,647. Measuring time in μs means you can have some very large numbers.

The next five lines of code will; set the TriggerPin to 0V (LOW), wait 4000 μs, set the TriggerPin to 5V (HIGH, sending a pulse from the motion detector), record the time that the pulse was triggered, and wait 900 μs before listening for the echo return. The micros() command returns the amount of time (in μs that has gone by since the program started running. It is like a stopwatch that starts running when the program does and you check the time with micros().

```
digitalWrite(TriggerPin, LOW);
delayMicroseconds(4000);
digitalWrite(TriggerPin, HIGH); // start the ultrasound pulse
time = micros(); //note time
delayMicroseconds(900); //delay during the blanking time
```

The do/while statement that follows will continuously check the EchoPin till it returns a HIGH value (5 V) indicating that the echo was received from the target object. When this occurs, the program will again note the time (in μs) on the internal Arduino clock using the micros() command, subtract from this the time that the pulse was sent and you know how long the acoustic pulse was in the air.

```
do
{
  val =digitalRead(EchoPin);
  // if no echo, repeat loop and wait:
}
while (val == LOW) ;
Duration =micros() - time;
```

The program then uses this information to determine how far away from the sensor the object is and reports the results.

5 Measurements

5.1 Constant Velocity Measurement

The first scenario is to document and analyze a cart rolling at a constant speed along the horizontal ramp away from the detector. Be sure that you track is level on your table. Give the cart a little push away from the motion detector and record its motion.

(a) Produce a graph of position vs time for your cart.

(b) Determine the speed of your cart using your graph.

(c) Report your cart's speed in the table on the white board up front.

(d) What would this graph look like if the cart was rolling toward the detector?

Once you have your cart's speed on the front board wait for the rest of the class to complete this portion of the laboratory.

5.2 Constant (non-zero) Acceleration

Use a text book or some other sturdy object to tilt you track slightly (something between 5^o and 15^o). Use the angle indicator on the track to determine just what angle your track is tilted at. It is not important that you achieve any particular angle, only that you know what angle you have achieved.

(a) Calculate what you expect the acceleration down the "inclined" ramp should be.

(b) Release the cart from rest and record its motion down the ramp.

(c) Plot the cart's position as a function of time moving down the ramp. Be sure to only use data from when the cart was freely moving down the ramp.

(d) Use your plot to determine the acceleration of the cart down the ramp. Does the measured acceleration agree with the expected acceleration?

5.3 Rolling Object

Use one of the older cart ramps for this measurement. Set it on your table with a slight incline like you did for the previous measurement. You will be rolling a basketball down this ramp. With the ramp turned upside down, the ball should roll straight down the ramp.

(a) Calculate what you would expect the acceleration down the "inclined" ramp to be.

(b) Release the basketball from rest and record its motion down the ramp.

(c) Plot the ball's position as a function of time moving down the ramp. Be sure to only use data from when the ball was freely rolling down the ramp.

(d) Use your plot to determine the acceleration of the ball down the ramp. Does the measured acceleration agree with the expected acceleration?