Interactive Jewellery Making

Computer style

Run Before You Can Walk

Ever been told that you have to walk before you can run?

Annoying.

Not true.

Definitely not true.

But please try the Jewellery Making course first – then this could be for you if you are the sort of person who likes running before they can walk.

Interactive Jewellery

Suppose you program your wearable and go out wearing it. You might get bored with it and wish you could change the colour scheme.

Of course, if you have your PC, MAC or Linux notebook with you, no problem.

But what about making your wearable interactive so that it changes as you move?

That's where the accelerometer comes in.

Introducing the Accelerometer

It's that little black chip on the apex of the centre triangle.

It can sense the g forces along three axes – x, y and z – as you move or shake it.

Remember x, y and z from Alice? You learned to move characters along the x, y and z axes.

Our Sketch in Outline

We could make the Engduino light up RED if the g force along the x axis is above a certain value, BLUE for the y axis and YELLOW for the z axis. You can choose whichever colours you want.

Our sketch will have the usual three sections.

First we will include the libraries of code for the LEDs and the accelerometer and also one called Wire.

In the setup section, we will start the code in these libraries.

Then the loop.

Basically, we will have these steps:

1. Make an array, which is like a table with three cells to store the g forces along each axis
2. Then we will have some lines of code to print out the g forces so that we know how much g force is being applied to the Engduino as we move it or shake it around.
3. Lastly, we need to some to switch on some of the LEDs to RED if the force along the x axis reaches a certain level.
   When we know how to do this, we can repeat it for the other two axes with different LEDs and different colours.

Re-use Some Code

Most programmers use code someone else has written or code they have already written themselves. Recycling code saves time and helps programmers avoid errors. If it works, why reinvent the wheel?

```
#include <EngduinoLEDs.h>
#include <EngduinoAccelerometer.h>
#include <Wire.h>

void setup()
{
  EngduinoLEDs.begin();
  EngduinoAccelerometer.begin();
}

void loop()
{
  float accelerations[3];

  EngduinoAccelerometer.xyz(accelerations);

  float x = accelerations[0];
  float y = accelerations[1];
  float z = accelerations[2];

  Serial.print("Acceleration: x = ");
  Serial.print(x);
  Serial.print("g y = ");
  Serial.print(y);
  Serial.print("g z = ");
  Serial.print(z);
  Serial.println("g");
```

This code came from the engduino website.

You will be able to understand the first two sections.

The line

```
  float accelerations[3];
```

creates an array called accelerations to hold three values – the g forces along the x, y and z axes. float means that the data held will have decimal places. We would use int if we just wanted whole numbers, but they would not be accurate enough.

The next line collects the g forces.

```
EngduinoAccelerometer.xyz(accelerations);
```

The three lines

```
  float x = accelerations[0];
  float y = accelerations[1];
  float z = accelerations[2];
```

store the values for the g forces in the array. Remember that computers start counting at zero, so x goes in the first place in the array which has an index number of zero. The force along the y axis goes in array index 1, which is the second place.

Then we have some lines which print out text and the values of the g forces to the Serial Monitor which is a window that you can display on your screen.

`Serial.print("` displays text on the screen, like **`print(`** in Python.

In the last line, `Serial.println` creates a new line (like pressing Enter on the keyboard).

Enter all of this code carefully and finish with a line to make the program wait for 5 seconds before displaying a new message. Here is the complete code with a curly brace to complete the code block.

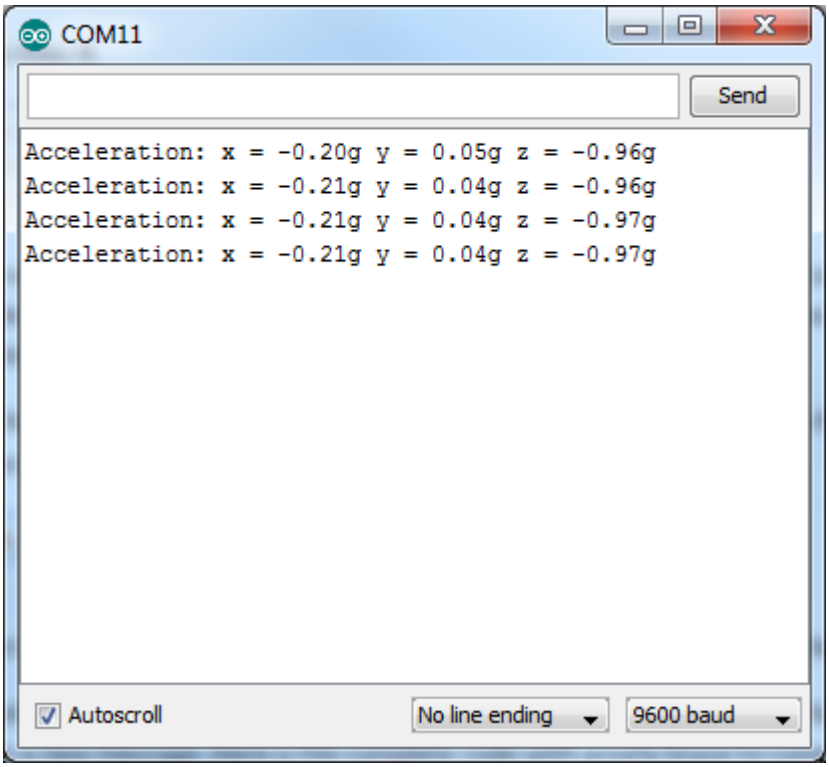
```
#include <EngduinoLEDs.h>
#include <EngduinoAccelerometer.h>
#include <Wire.h>

void setup()
{
  EngduinoLEDs.begin();
  EngduinoAccelerometer.begin();
}

void loop()
{
  float accelerations[3];

  EngduinoAccelerometer.xyz(accelerations);

  float x = accelerations[0];
  float y = accelerations[1];
  float z = accelerations[2];

  Serial.print("Acceleration: x = ");
  Serial.print(x);
  Serial.print("g y = ");
  Serial.print(y);
  Serial.print("g z = ");
  Serial.print(z);
  Serial.println("g");

  delay(5000);
}
```

Connect the Engduino to your PC, MAC or Linux box using a USB extension lead – unless you want to shake your laptop or MacBook Air – not recommended!.

Save, compile, sort out any typos and upload.

When you see “Done uploading”, click on the icon for the Serial Monitor. It looks like a magnifying glass.

You will see something like this:



Shake the Engduino. It is robust but don’t bang into anything or anybody. Notice how the x, y and z values change.

Move it up and down. Which axis shows a change in values?

Experiment to see what sort of motion changes the values on each axis. Make a note of what you find out.

Develop the Code

Now for the challenge.

Suppose you find that moving the Engduino up and down produces a value of 0.2 on the x axis. We can program the Engduino to flash RED when the g force on the x axis reaches 0.2.

We need an **if** block.

In Arduino (and Javascript), what we are testing (the condition) goes in brackets. The code that will run if the condition is true goes in curly braces.

```
Serial.print("Acceleration: x = ");
Serial.print(x);
Serial.print("g y = ");
Serial.print(y);
Serial.print("g z = ");
Serial.print(z);
Serial.println("g");

if (x > 0.2)
{


}
```

To make LED1 RED for two seconds, we need

```
if (x > 0.2)
{
  EngduinoLEDs.setLED(1, RED);
  delay(2000);
  EngduinoLEDs.setLED(1, OFF);
}
```

Compile, upload and test by moving the Engduino up and down for a few seconds, then put it down carefully. Wait at least five seconds? (Why five seconds?) Does it do what you expected?

Of course, you could have several LEDs come on when the force on the x axis reaches a certain level. Do they have to be the same colour?

You can repeat the code for the y and z axes.

You might find that you want an LED or a block of LEDs to come on when the g force is outside a range, for example, if it is less than -0.3 (negative 0.3) or more than 0.3. The syntax using or is:

if (x < -0.3 or x > 0.3)

Note that you have to repeat x. if (x < -0.3 or > 0.3) will produce an error.

Have fun.

What's the Point?

OK, it's fun, well maybe.

Interactive wearables are becoming big business, particularly in medical sciences.

As I write this, some people are dying because it is so hot. In the winter people will die because it can get so cold.

How could a wearable help?

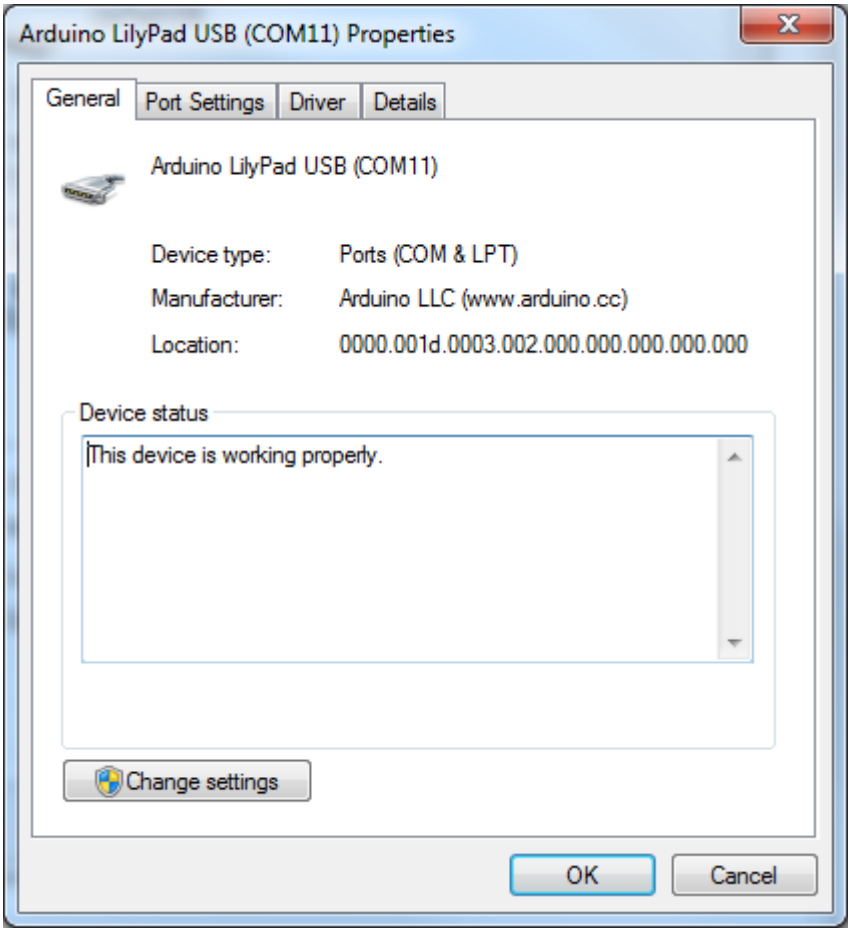
How could UCL develop the Engduino to help people with common but serious medical conditions?

As you develop through the school and into uni, how can **you** use this technology to invent the future?

Troubleshooting - Hardware

The first time you use the Engduino, an adult will probably have set it up for you to make sure it works.

If you are working on your own, first make sure that you can see the Engduino when you choose Devices and Printers (in Windows 7). It will appear as LilyPadUSB if it is installed and switched on. Right click and choose Properties, then Hardware, then Properties.



The Engduino (also known as Arduino LilyPad USB) is installed on COM Port 11.

Back in the Arduino IDE, choose

- Tools → Board to check that the software is using the LilyPad Arduino USB
- Tools → Serial Port to check that it is using the correct COM (serial) port.

Troubleshooting – Software

If your program won't compile

Look for typos, upper case instead of lower case, lower case instead of upper case, missing colons….

What do you think will happen with this code?

```
void loop()
{
  EngduinoLEDs.SetAll(RED)
}
```

It won't compile

- because SetAll should be setAll
- and beause the colon is missing from the end of the line.

If your program compiles but doesn't do what you expect

You probably have a logic error, which means that the computer is doing what you asked but not what you meant. You did not think it through carefully! Don't worry, everyone has made logic errors at some time, it is normal. You learn by making mistakes.

Suppose you want the Engduino to flash red at one second intervals.

Why does this code not flash the LEDs?

```
void loop()
{
  EngduinoLEDs.setAll(RED);
  delay(1000);
  EngduinoLEDs.setAll(OFF);
}
```

If you can't see why, discuss it with a friend. Try it out. Get it to work!