**Earthquake Monitor**

The aim of this exercise is to become familiar with the Arduino development platform, accelerometer sensors and low-power radio communications. This will be achieved through the development of a basic, networked earthquake sensor device. The components you will be using will be as follows:

|  |  |  |
| --- | --- | --- |
| Macintosh HD:Users:slock:Desktop:nano.jpg | Macintosh HD:Users:slock:Desktop:GY-61.jpg | Macintosh HD:Users:slock:Desktop:nordic.jpg |
| Arduino Nano | GY-61 Accelerometer | Nordic Radio |

**Task 1: Blink !**

The first program that anyone writes on an Arduino is usually “Blink” – it is the “Hello World” of Arduino development ! First open up the Arduino application on your laptop and then from the top-bar menu select: File> Examples> 01.Basics> Blink

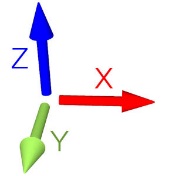
You should now see the Blink code in your editor window.

Plug your Arduino into your laptop with a suitable USB cable, then make sure that “Tools> Board> Arduino Nano” is selected and that “Tools> Port” is the serial port that your Arduino is connected to. If you are unsure what this means, just ask !

Next we need to upload the code onto the Arduino. This is done by clicking the “upload” button 

Near the bottom of the Arduino window you will see a variety of progress messages including: “Compiling sketch”, “Uploading”, “Uploading Done” or if you are really unlucky, a red error message!

If everything worked OK, you should see one of the lights on the Arduino flashing on and off every second. Congratulations, you are running your first Arduino program !!

**Task 2: Accelerometer hookup**

Now that you are familiar with some of the features of the Arduino development environment, we’ll do something a little bit more interesting. We are going to hook up the Arduino to an accelerometer board that can be used to sense movement and vibration in three different dimensions: X Y Z (see diagram)

The first thing to do is to UNPLUG YOUR ARDUINO FROM THE USB. This is just in case you slip with a wire and accidentally short the chip (unlikely, but possible).

Next connect the accelerometer board with the ribbon cable provided. Make the following Accelerometer->Arduino connections: VCC->5V GND->GND X\_OUT->A3 Y\_OUT->A4 Z\_OUT->A5

Once the Accelerometer board is connected, you can plug the USB cable back in.

**Task 3: Accelerometer Code**

We now need to read in values from the Accelerometer sensor. Update your Blink program with the following features (might be wise to save it to your desktop first, so that you don’t lose any code):

In the “setup” function, switch the data pins (A3, A4 and A5) to “input” mode, This is done using the “pinMode” function, for example: *pinMode(A3, INPUT);* Make sure you do it for all 3 pins !

Also in the setup function, add the following line of code: *Serial.begin(9600);* This initialises serial communication so that we can send data down the USB cable, back to the computer.

Next, in the “loop” function read in the current value of each of the 3 accelerometer dimensions. This is done using the “analogRead” function, for example: *int x = analogRead(A3);* Make sure you do it for all 3 input pins (using different variable names to store the readings !)

After each value you read in, use the “Serial.println” function to send X, Y and Z acceleration data down the USB cable to your computer. For example: *Serial.println(x);* If you want to put a blank line between any values, just use *Serial.println(“ “);*

Upload your updated program onto your Arduino in the same way as before.

Once uploaded, you should be able to see the data streaming in by open up the “Serial Monitor”:

**Task 4: Gravity Baseline**

In addition to sensing vibration, the accelerometer will also detect the pull of gravity. Depending on the orientation of your sensor board, this might be in X, Y, Z, or a combination of different directions! In your setup function, read the initial values from X, Y and Z and remember them as a “baseline” (you can create three integer variables at the top of your program to store this information in). You will need to subtract these initial values from your later readings (made in the loop function) in order to factor out the effects of gravity.

**Task 5: Magnitude calculation**

Sending 3 separate values down the serial cable is a bit unnecessary – we probably only need to send a single value. Perform a calculation on the Arduino to resolve the three separate readings into a single absolute “magnitude” value (hint: you might need to use Pythagoras to do this). Finally, change your code so that you only send the single value down the serial line. Check in serial monitor to see if everything is working as you might expect.

**Task 6: Nordic radio**

Instead of sending your magnitude value down the serial cable to the Serial Monitor, we are now going to send it via radio to the grapher application (plugged into the projection computer). A template Arduino program has been provided to help you get started on this activity, use it as the basis for this stage of your work.

You will also need to install a radio comms library on your computer (if it is not already installed !), this can be found at:

<http://people.cs.bris.ac.uk/slock/mirf.zip>

Unzip it and copy the whole folder into: My Documents/Arduino/Libraries

You will need a Nordic radio and a specially constructed 8-wire connection cable.

First, wire up the Nordic radio to the Arduino following the below wiring colour scheme:

|  |  |
| --- | --- |
|  |  |

MAKE SURE THAT YOU USE THE 3.3V POWER PIN – AND NOT THE 5V PIN !!!!!!!!!!!!!!!!!!!!!!!!!!!!

Add code to the template file to read data from your accelerometer, factor out the effects of gravity, calculate the magnitude and send it via the Nordic radio. Much of the radio initialisation code has been provided for you in the template. You just need to replace the lines in the loop function containing “//…” with code of your own.

In order for the grapher to successfully receive your message, you need to obey the following rules:

* Your message must always contain a total of 9 bytes
* The first byte must be a special “message begins” byte 0x00
* The next 5 bytes must be the characters of a unique name for your device (e.g. “STEVE”)
* The next 2 bytes must be your magnitude (integers are stored in 2 bytes on these Arduinos)
* The final byte must be a special “message ends” byte 0xFF

Note: In order to split the magnitude value into two separate bytes, you will need to use the Arduino highByte() and lowByte() functions. The grapher expects to see the high byte first, followed by the low byte in the incoming message.

**Task 7: Repeater Node (optional)**

The range of the Nordic radios is fairly limited (even with an external antenna). As an optional extension to this exercise, create a “repeater” node. For this task, you will need an additional Arduino and another Nordic radio board. The code running on your “repeater” Arduino should receive incoming messages addressed to “REPTR” (or something similar or your own choice) and will resend them on to the “GRAPH” node. To help you complete this task, you may need to refer to the examples on the MIRF radio library github page:

<https://github.com/aaronds/arduino-nrf24l01/tree/master/Mirf/examples>

Think about how you might chain these repeater nodes together in order to further extend the range of the earthquake monitoring network.