Arduino Programming for Accelerometer

Rohan Panda
ECE 480 Design Team 2
11/12/2015

Abstract: This application notes describes the Arduino programming process for accelerometer. The procedure describes the coding and connection process to show how to make the accelerometer sense vibration and tilt shifting.

Key Words: vibration, accelerometer, Arduino, monitoring, imbalance, pins

Introduction: The Arduino UNO Rev 3 is a microcontroller based on the ATmega32P with 14 digital input and output pins. Out of these 14 pins, 6 are capable of PWM outputs, 6 analog inputs, a USB connection, a power jack, an ICSP header, 16 MHz quartz crystal, and a reset button. The convenient part of Arduino is that it is already pre built with every microcontroller support hence simply plugging in with an USB cable will suffice. Also an advantage of using the UNO is that the chip used in it can be replaced for relatively cheap cost. UNO board is the very first of the Arduino boards and even though there are more advanced boards available, for this project, the UNO board will be enough. The figure 1 on the next page shows the technical specifications of this Arduino board.

The accelerometer used in this project would be ADXL335 small, thin, low power, 3-axis accelerometer that contains signal conditioned voltage outputs. It is designed to measure the acceleration with a full scale range of ±3 g. It measures two kinds of motion. First kind of motion that it measures is the static acceleration of gravity when the accelerometer is tilted. So during the diamond polishing phase if the arm tilts it will send a signal. The second kind of acceleration it measures is the dynamic acceleration which will help to detect the change is vibration that may occur when the diamond polishing machine is at work.
ADXL335 is powered by a 3.3v power source that causes it to generate 3.3v peak outputs. It has three analog outputs for X, Y, and Z axis which require an ADC microcontroller that is provided by the analog functions of Arduino board.

**Figure 1: Technical Specifications for Arduino Board**

### Technical specs

<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microcontroller</td>
<td>ATmega328P</td>
</tr>
<tr>
<td>Operating Voltage</td>
<td>5V</td>
</tr>
<tr>
<td>Input Voltage (recommended)</td>
<td>7-12V</td>
</tr>
<tr>
<td>Input Voltage (limit)</td>
<td>6-20V</td>
</tr>
<tr>
<td>Digital I/O Pins</td>
<td>14 (of which 6 provide PWM output)</td>
</tr>
<tr>
<td>PWM Digital I/O Pins</td>
<td>6</td>
</tr>
<tr>
<td>Analog Input Pins</td>
<td>6</td>
</tr>
<tr>
<td>DC Current per I/O Pin</td>
<td>20 mA</td>
</tr>
<tr>
<td>DC Current for 3.3V Pin</td>
<td>50 mA</td>
</tr>
<tr>
<td>Flash Memory</td>
<td>32 KB (ATmega328P)</td>
</tr>
<tr>
<td></td>
<td>of which 0.5 KB used by bootloader</td>
</tr>
<tr>
<td>SRAM</td>
<td>2 KB (ATmega328P)</td>
</tr>
<tr>
<td>EEPROM</td>
<td>1 KB (ATmega328P)</td>
</tr>
<tr>
<td>Clock Speed</td>
<td>16 MHz</td>
</tr>
<tr>
<td>Length</td>
<td>68.6 mm</td>
</tr>
<tr>
<td>Width</td>
<td>53.4 mm</td>
</tr>
<tr>
<td>Weight</td>
<td>25 g</td>
</tr>
</tbody>
</table>

Fig 1. Technical Specifications of Arduino shows that the board is relatively light and small weighing only 25g with length of 68.6 mm and width of 53.4 mm. It gives the Flash memory and clock speed are respectively 32 KB and 16 MHz which indicates it is enough for small accelerometer programming.
**Process:** The process will describe how to connect the Arduino and the accelerometer and then download the Arduino software to program and run the codes. Figure 2 and 3 will be referenced in order to describe the connection.

**Figure 2: Arduino UNO Rev 3 pins and ports**

![Arduino UNO Rev 3 pins and ports](image)

*Figure 2 shows the Arduino UNO Rev 3 ports and pins and where the accelerometer will be connected.*

**Figure 3: ADXL335 Accelerometer**

![ADXL335 Accelerometer](image)

*Figure 3 shows the X, Y, Z and GND pins for the accelerometer and it will be used to reference the power pin connection.*
The parts required for this process are the Arduino board, ADXL335 accelerometer, connecting wires, and USB cable to connect Arduino board to computer.

**Process for Circuit building:**
Accelerometer has 5 pins and all of these are connected to Arduino. First connect the GND to Arduino’s GND. Then connect the VCC to Arduino’s 5V, X to Arduino’s Analog Pin A5, Y to Arduino’s Analog Pin A4, and Z to Arduino’s Analog Pin A3. Finally the AREF is connected to 3.3v on Arduino to set the reference voltage of ADXL355 to 3.3v.

The final product should look something like the following figure 4.

**Figure 4: Final connection of Arduino to Accelerometer**

![Image of Arduino and Accelerometer connection]

**Process of coding the Accelerometer:**
Before beginning the coding, the Arduino 1.6.6 software needs to be downloaded for compatible operating system. Opening it up would show something like the following figure.
Figure 5: Window for Arduino Software

Figure 5 shows the Arduino window it provides a space for writing codes. It has a → button that is used to upload the codes to machine. The black bar under the window shows the error messages.

Next the following code is put in the Arduino software.

```c
const int ap1 = A5;
const int ap2 = A4;
const int ap3 = A3;
int sv1 = 0;
int ov1 = 0;
int sv2 = 0;
int ov2 = 0;
int sv3 = 0;
int ov3 = 0;

void setup() {
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);
  
  void loop() {
    analogReference(EXTERNAL);  // connect R.V to REF
    // read the analog in value:
    sv1 = analogRead(ap1);
    // map it to the range of the analog out:
    sv1 = map(sv1, 0, 1023, 0, 255);
    // change the analog out value:
    delay(2);
    //
    ov1 = analogWrite(ap2);
    //
    delay(2);
    //
    sv2 = analogRead(ap3);
    sv2 = map(sv2, 0, 1023, 0, 255);
    // print the results to the serial monitor:
    Serial.print("Error = ");
    Serial.println(sv1);
    Serial.println("Input = ");
    Serial.println(ov1);
```

```c
}
```
// initialize serial communications at 9600 bps:
Serial.begin(9600);

void loop() {
    analogReference(EXTERNAL);  // connect 3.3v to AREF
    // read the analog input value:
    sv1 = analogRead(ap1);
    // map it to the range of the analog output:
    ov1 = map(sv1, 0, 1023, 0, 255);
    // change the analog output value:
    delay(2);
    //
    sv2 = analogRead(ap2);
    ov2 = map(sv2, 0, 1023, 0, 255);
    //
    delay(2);
    //
    sv3 = analogRead(ap3);
    ov3 = map(sv3, 0, 1023, 0, 255);
    // print the results to the serial monitor:
    Serial.print("Xsensor1 = ");
    Serial.print(sv1);
    Serial.print("\toutput1 = ");
    Serial.println(ov1);
    Serial.print("Ysensor2 = ");
    Serial.print(sv2);
    Serial.print("\toutput2 = ");
    Serial.println(ov2);
    Serial.print("Zsensor3 = ");
    Serial.print(sv3);
    Serial.print("\toutput3 = ");
    Serial.println(ov3);
    delay(3000);

}
**Results:** After the serial monitoring is pressured the result should give something like the following in figure 6.

![Figure 6: Results of Accelerometer Vibration](image)

**Figure 6: Results of Accelerometer Vibration**

\[
\begin{align*}
X_{sensor1} &= 771 & output1 &= 192 \\
Y_{sensor2} &= 798 & output2 &= 198 \\
Z_{sensor3} &= 956 & output3 &= 238 \\
X_{sensor1} &= 758 & output1 &= 188 \\
Y_{sensor2} &= 790 & output2 &= 196 \\
Z_{sensor3} &= 936 & output3 &= 233 \\
X_{sensor1} &= 765 & output1 &= 190 \\
Y_{sensor2} &= 795 & output2 &= 198 \\
Z_{sensor3} &= 945 & output3 &= 235 \\
X_{sensor1} &= 761 & output1 &= 189 \\
Y_{sensor2} &= 784 & output2 &= 195
\end{align*}
\]

Figure 6 shows that there are two different analog values that are output. The first one is the ADC value in 10 bit resolution ranging from 0 to 1023 and the second is the mapped PWM value that is in 8 bit resolution ranging from 0 to 255. Then three values of X, Y, and Z are displayed at the same time and repeated after a specified interval in the code. The outputs are varied because during this measuring process, vibration was produced. It is not constant which shows that the accelerometer is detecting the vibration.

**Conclusion:** All in all the Arduino board is a quite inexpensive and reliable way to program the accelerometer to show imbalances in the diamond polishing arm. The hardware parts are relatively easy to build. Figure 6 shows that the accelerometer is sensitive enough to differentiate between output of 195, 190, and 189 which means the user can set up an external alarm so it can go off when an undesired vibration is produced.

**Reference:**
