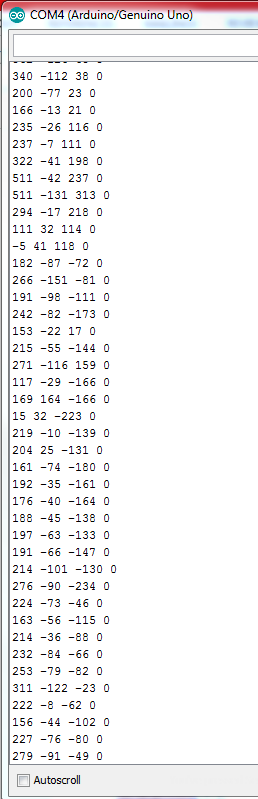
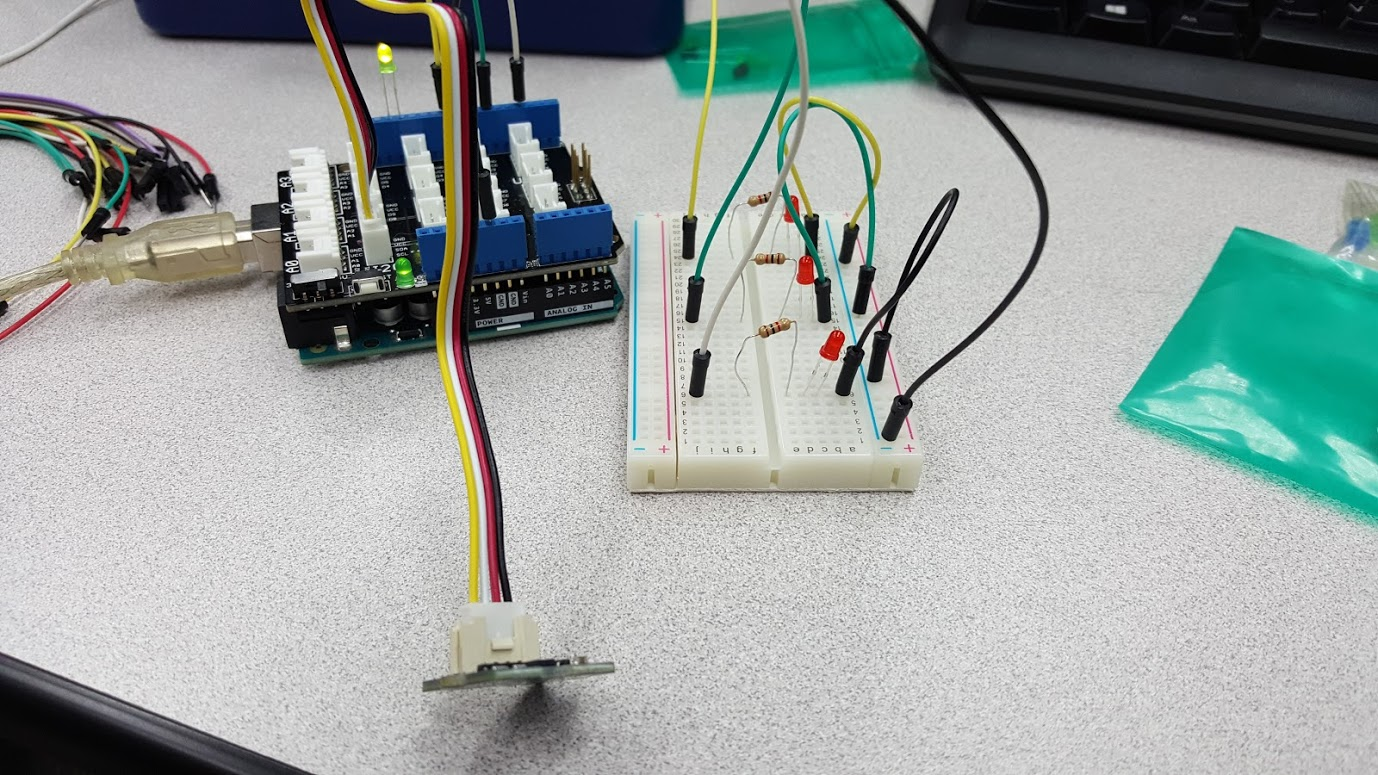
Sarah Wood

Embedded Programming

Ch6 Assg 2





Source Code

(I will spare you the agony of reading the class and member stuff again.)

Breakout2.cpp

#include <Wire.h>

#include "ADXL345.h"

ADXL345 adxl; //variable adxl is an instance of the ADXL345 library

int x, y, z;//added with v 2.0

const unsigned int X\_AXIS\_PIN = 3;

const unsigned int Y\_AXIS\_PIN = 6;

const unsigned int Z\_AXIS\_PIN = 9;

const unsigned int TEST\_PIN = 13;

const unsigned int NUM\_AXES = 3;

const unsigned int PINS[NUM\_AXES] = {

X\_AXIS\_PIN, Y\_AXIS\_PIN, Z\_AXIS\_PIN

};

const unsigned int BUFFER\_SIZE = 16;

const unsigned int BAUD\_RATE = 9600;

int buffer[NUM\_AXES][BUFFER\_SIZE];

int buffer\_pos[NUM\_AXES] = { 0 };

void setup() {

Serial.begin(BAUD\_RATE);

pinMode(X\_AXIS\_PIN, OUTPUT);

pinMode(Y\_AXIS\_PIN, OUTPUT);

pinMode(Z\_AXIS\_PIN, OUTPUT);

pinMode(TEST\_PIN, OUTPUT);

//added with v2.0

adxl.powerOn();

//set activity/ inactivity thresholds (0-255)

adxl.setActivityThreshold(75); //62.5mg per increment

adxl.setInactivityThreshold(75); //62.5mg per increment

adxl.setTimeInactivity(60); // how many seconds of no activity is inactive?

//look of activity movement on this axes - 1 == on; 0 == off

adxl.setActivityX(1);

adxl.setActivityY(1);

adxl.setActivityZ(1);

//look of inactivity movement on this axes - 1 == on; 0 == off

adxl.setInactivityX(1);

adxl.setInactivityY(1);

adxl.setInactivityZ(1);

//look of tap movement on this axes - 1 == on; 0 == off

adxl.setTapDetectionOnX(0);

adxl.setTapDetectionOnY(0);

adxl.setTapDetectionOnZ(1);

//set values for what is a tap, and what is a double tap (0-255)

adxl.setTapThreshold(50); //62.5mg per increment

adxl.setTapDuration(15); //625us per increment

adxl.setDoubleTapLatency(80); //1.25ms per increment

adxl.setDoubleTapWindow(200); //1.25ms per increment

//set values for what is considered freefall (0-255)

adxl.setFreeFallThreshold(7); //(5 - 9) recommended - 62.5mg per increment

adxl.setFreeFallDuration(45); //(20 - 70) recommended - 5ms per increment

//setting all interrupts to take place on int pin 1

//I had issues with int pin 2, was unable to reset it

adxl.setInterruptMapping( ADXL345\_INT\_SINGLE\_TAP\_BIT, ADXL345\_INT1\_PIN );

adxl.setInterruptMapping( ADXL345\_INT\_DOUBLE\_TAP\_BIT, ADXL345\_INT1\_PIN );

adxl.setInterruptMapping( ADXL345\_INT\_FREE\_FALL\_BIT, ADXL345\_INT1\_PIN );

adxl.setInterruptMapping( ADXL345\_INT\_ACTIVITY\_BIT, ADXL345\_INT1\_PIN );

adxl.setInterruptMapping( ADXL345\_INT\_INACTIVITY\_BIT, ADXL345\_INT1\_PIN );

//register interrupt actions - 1 == on; 0 == off

adxl.setInterrupt( ADXL345\_INT\_SINGLE\_TAP\_BIT, 1);

adxl.setInterrupt( ADXL345\_INT\_DOUBLE\_TAP\_BIT, 1);

adxl.setInterrupt( ADXL345\_INT\_FREE\_FALL\_BIT, 1);

adxl.setInterrupt( ADXL345\_INT\_ACTIVITY\_BIT, 1);

adxl.setInterrupt( ADXL345\_INT\_INACTIVITY\_BIT, 1);

//end v 2.0 additions

}

/\*

int get\_axis(const int axis) {

delay(1);

//buffer[axis][buffer\_pos[axis]] = analogRead(PINS[axis]); //superceded v2.0

buffer\_pos[axis] = (buffer\_pos[axis] + 1) % BUFFER\_SIZE;

long sum = 0;

for (int i = 0; i < BUFFER\_SIZE; i++)

sum += buffer[axis][i];

return round(sum / BUFFER\_SIZE);

}

\*/

void loop()

{

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//New code for v 2.0

digitalWrite(TEST\_PIN, HIGH);

adxl.readXYZ(&x, &y, &z);

//v2.0: we read them all at once, and assign them to x, y, z

Serial.print(x);

Serial.print(" ");

Serial.print(y);

Serial.print(" ");

Serial.print(z);

Serial.print(" ");

if(x<0)

{digitalWrite(X\_AXIS\_PIN, HIGH);}

else

{digitalWrite(X\_AXIS\_PIN, LOW);}

if(y<0)

{digitalWrite(Y\_AXIS\_PIN, HIGH);}

else

{digitalWrite(Y\_AXIS\_PIN, LOW);}

if(z<0)

{digitalWrite(Z\_AXIS\_PIN, HIGH);}

else

{digitalWrite(Z\_AXIS\_PIN, LOW);}

Serial.println("0");

}