Instructions for Device

Setup and Use

Provided by

Andrew Spiewak

- The following pages will give a head start to anyone perusing the measurement of Mechanomyography, by multiple means, and detailed instruction.
- While countless numbers of researchers have tackled this, the major members of this iteration of study include the following. Their contact info has been provided as known at the time of this document's creation. Contact is allowed but by no means guaranteed.

- Joseph English (Researcher, Piezoelectric, Raspberry PI) jcenglish@me.com
- Alexander Mackey (Researcher, Accelerometers, Arduino, Raspberry PI) mackeyasm@gmail.com
- Kirsten Miller (Researcher, Biomedical/Biomechanical, Wiring and Shielding) miller.kc.1@knights.ucf.edu
- Kayla Parker (Researcher, Team Leader/ Organizer, Vision Realization) p.kayla918@knights.ucf.edu
- Sam Rodriguez (Researcher, Data acquisition, Programming, Machine Learning) samurodro@gmail.com
- Andrew Spiewak (Researcher, Photonics, Data Acquisition, Procurement) andrewspiewak7@gmail.com
- Dr. Hansen Mansy (Associate Professor, University of Central Florida) Hansen.Mansy@ucf.edu

Section 1: Initial Purchases and Justification.

A computer with USB 3.0 or later, LabView, MATLAB, Arduino IDE, and CAD software.

Many headaches can be avoided by starting with good hardware. Suggested retailers include: Mouser, Sparkfun, Digi-key, and Amazon.

Accelerometer Parts list

Arduino DUE	1 @ $41.95
ADXL355BEZ Unmounted Accelerometer (digital)	1 @ $49.50
EVAL-ADXL355-PMDZ (digital)	1 @ $30.00
Alex Tech PET Braided Sleeve ¼ inch	25 ft. @ $7.99
Alex Tech PET Braided Sleeve ⅛ inch	25 ft. @ $7.99

Electriduct Metal Braided Shielding ⅛ inch	25 ft. @ $13.99
Electriduct Metal Braided Shielding ¼ inch	25 ft. @ $19.99
Hxtape Copper Foil Tape 1”	66 ft. @ $8.98
Striveday 5 Color 30 AWG wire	1 box @ $12.99
LAMPVPATH 2 AA Battery Holder with Switch	set of 3 @ $6.49
VNDEFUL Black Waterproof Project Case	set of 5 @ $7.49
2 AA Batteries	set of 6 AA @ $4.90
Super Soldier 60 % 0.8mm	1 lb. @ $23.55
Assorted Heat Shrink Tubing	I box @ $7.77
Loctite 2 Part Epoxy	1 set @ $12.44
3m Red Dot Electrodes	Pack of 50 @ $12.04
AD8232 Heart Rate Monitor	1 @ $19.95

Additional Parts (Measuring using other than Accelerometer)

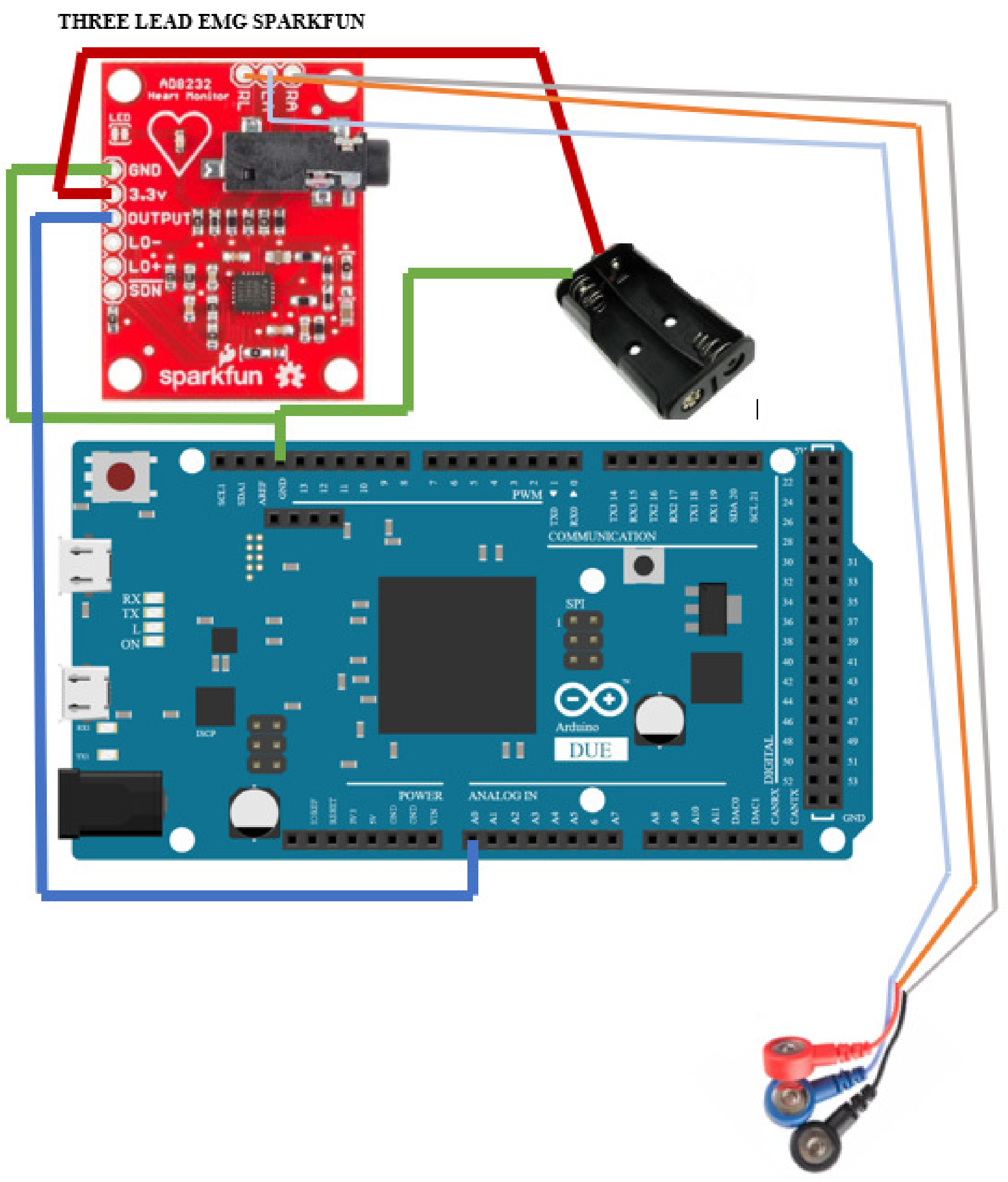
Piezoelectric 102-1126-ND	1 @ $1.31
Green laser module	1 @ $17.95
Green photodiodes VEMD5510CF-GS15	1 @ $2.04
584-EVAL-ADXL354BZ (analog)	1 @ $43.75

An attempt was made to use the ADXL355 chip without the evaluation board with the result of a lower signal to noise ratio than the eval board mounted version, the exact cause of this is not known to us. It is highly recommended to start with the EVAL-ADXL355-PMDZ (digital) and work it out from there.

Because of the elevated data rate of the digital accelerometer the Arduino Due with a 32-bit processor will be required, a simpler Arduino UNO will not run ADXL355 digital.

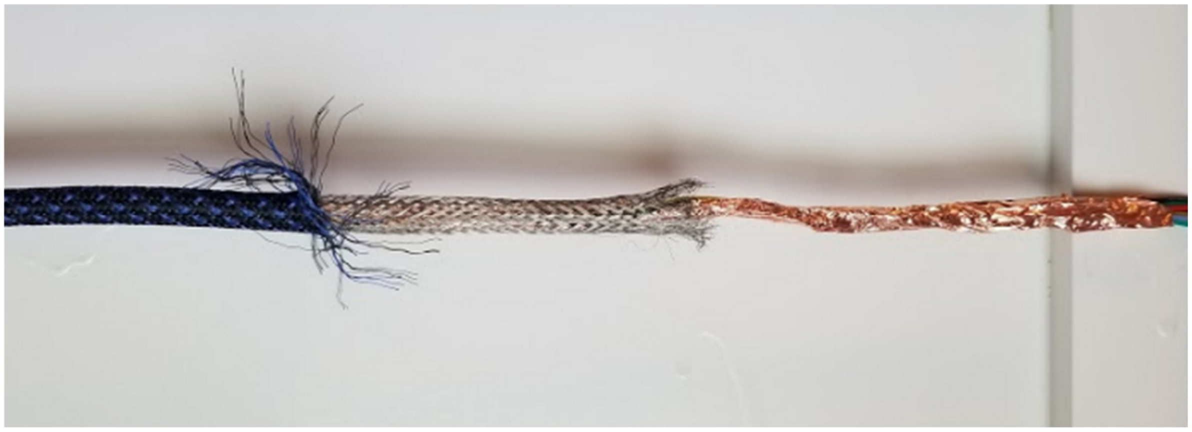
The Raspberry PI is a far more complicated system and not recommended for beginners. See the latter chapter on Raspberry PI if more info is desired .

Section 2: Wiring and Shielding

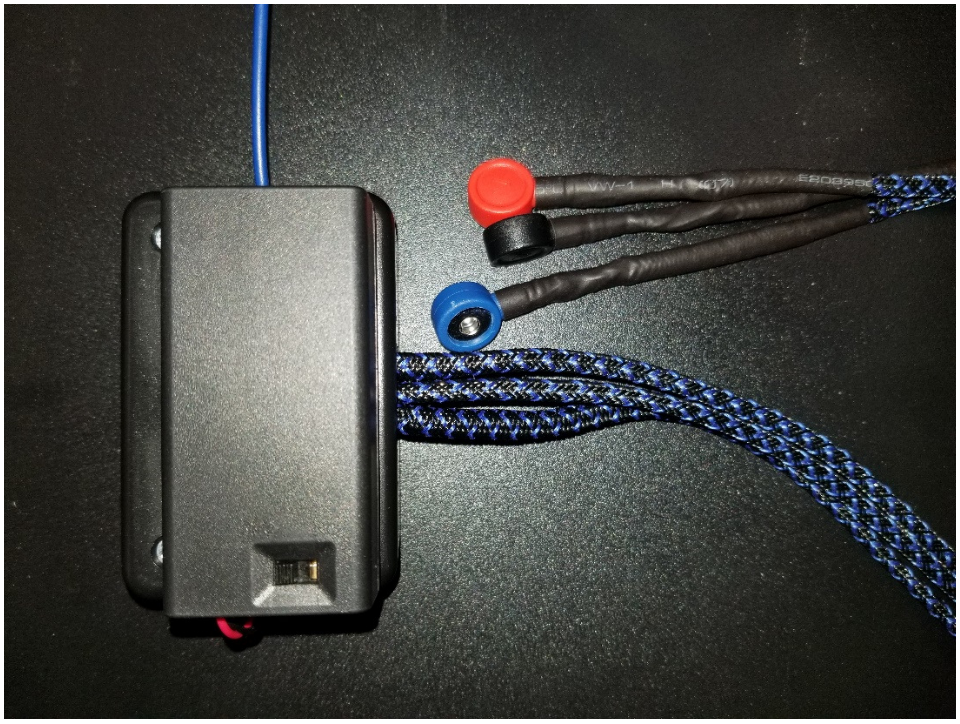


Wiring the EMG device should be completed as follows. The OUTPUT on AD8232 (Blue) should be led to the Analog in port A0/A1 whichever is specified in the code. The 3.3V supply will connect to the positive (red) lead of a 2 AA battery harness. The ground (green) wires of the batteries, Arduino DUE, and the EMG should all be connected to each other as shown. There are three electrode leads RED, BLUE, and BLACK. The RED lead (after shielding) should be soldered to the RL connection. This electrode will always be placed on the belly or thickest part of the muscle. The BLUE lead (after shielding) should be soldered to the LA connection. This electrode will always be placed next to the RED electrode, on the muscle off center. The BLACK lead (after shielding) should be soldered to the RA connection. The BLACK electrode should be attached to a close by bone, or boney area like the elbow. Always use fresh batteries and new electrodes.

Apply shielding to wires as seen below in Figure 1

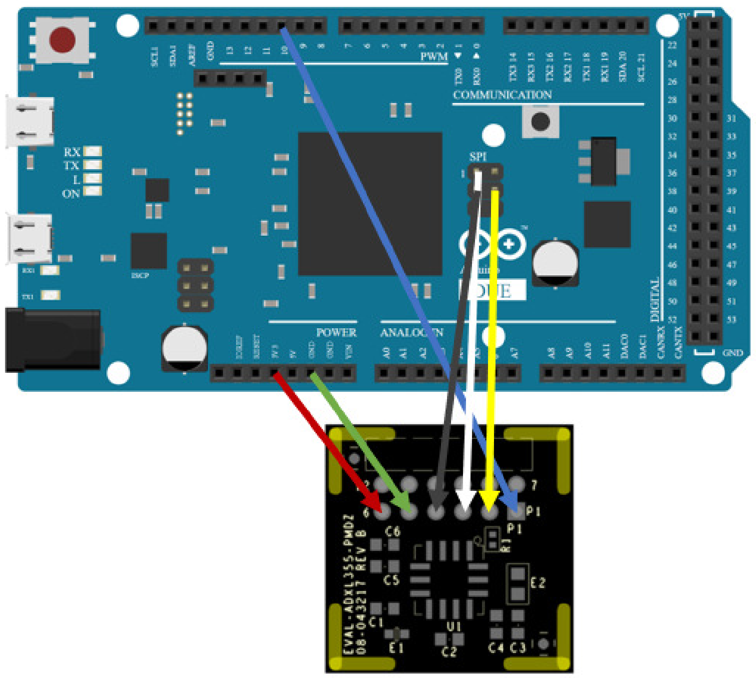


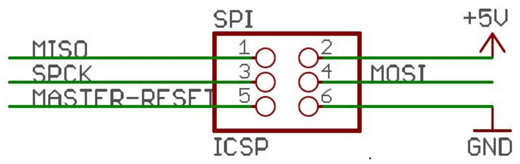
After determining the number of wires and how long of wire leads will be needed for the application begin the shielding with the copper tape. This provides shield as well as protection to the small gauge wire from the abrasive braided stainless sleeve. A long-pointed stick or skewer worked well for applying the braided stainless, a campfire marshmallow stick was used here. Finally, it is recommended to cover the shielded cables in a braided PET sleeve seen in black and blue above. This configuration is extremely important for the electrode leads on the EMG because they are analog and collect noise from the environment like an antenna. This process will also add a professional touch of beauty to your work.



The finished EMG device with battery pack and fully shielded wiring and shielded project case can be seen above. The Final wiring is Ultra low noise as needed to collect usable data. Again, the shielding of this device is paramount.

The accelerometer will also need to have leads soldered to the board, and all the wiring should be shielded and protected. Below is that process.





P1 Pin Number	Pin Function	Mnemonic
Pin 1	Chip Select	/CS
Pin 2	Master Out Slave In	MOSI
Pin 3	Master In Slave Out	MISO
Pin 4	Serial Clock	SCLK
Pin 5	Digital Ground	DGND
Pin 6	Digital Power	VDD
Pin 7	Interrupt 1	INT1
Pin 8	Not Connected	NC
Pin 9	Interrupt 2	INT2
Pin 10	Data Ready	DRDY
Pin 11	Digital Ground	DGND
Pin 12	Digital Power	VDD

Above on the left is the Arduino pin out and the right is the Eval board pins. P1 (Blue) should be connected to digital pin #10 on the Arduino DUE (specified in the code). P2 (Yellow) is the MOSI connection and should be linked with pin #4 on the SPI port of the DUE. P3 (White) MISO should be linked to pin #1 on the DUE. P4 (Black) is the serial clock SCLK and should be linked to the #3 pin on the SPI port of the Arduino. P5 (green) is the ground, and P6 is the power which can only be connected to 3.3V, 5V will damage the accelerometer. Both the accelerometer and the EMG can be run from one Arduino DUE.

Section 3: Arduino code

Below is an example of working code for the ADXL355 it can be copied and pasted directly into the Arduino IDE program and uploaded into the board. Improvements can be made but this current code will run the accelerometer and EMG on the same plot.

```
#include <SPI.h>  //------------------------------------ library for SPI in arduino must be added to
run SPI protocal.
// telling the arduino where to read/write, after assignment there is no options needed.
const int WRITE_TO = 0x00; //-----------------------------This is the address we will give data to.
const int READ_FROM = 0x01; //----------------------------This is the address we will get data from.
// There are 3 addresses per axis that data is stored in 8+8+4 bits. We will need to tell it to not
read 4-7 on all of the _1 locations
const int Z_1 = 0x10; //----------------------------------Bits 3,2,1,0 only will store Z data.
const int Z_2 = 0x0F; //----------------------------------We use all 8 bits at this address.
const int Z_3 = 0x0E; //----------------------------------We use all 8 bits at this address.
const int Y_1 = 0x0D; //----------------------------------Bits 3,2,1,0 only will store Y data.
const int Y_2 = 0x0C; //----------------------------------We use all 8 bits at this address.
const int Y_3 = 0x0B; //----------------------------------We use all 8 bits at this address.
const int X_1 = 0x0A; //----------------------------------Bits 3,2,1,0 only will store x data.
const int X_2 = 0x09; //----------------------------------We use all 8 bits at this location.
const int X_3 = 0x08; //----------------------------------We use all 8 bits at this location.
// We want to have the filter active
const int FILTER = 0x3; //----------------------------This is the filter address.
const int SET_FILTER = 0x6A; //-----------------------------This creates a bandpass (Table 43, pg.
37)
// We want the Range to measure 2G and below.
const int RANGE = 0x2C; //--------------------------------This is the address to set the range
setting.
const int MEAS_2G = 0x01; //------------------------------This is the code to set the range to 2G.
// We need to turn the device off/on, and set it to measure only acceleration.
const int POWER = 0x2D; //---------------------------------This is the address for the power setting.
const int PW_SET = 0x06; //--------------------------------First digit DRDY (1=off, 0=on), second
digit TEMP (1=off, 0=on), third digit STANDBY (1=standby, 0=measure)
// We need to select a digital pin on the arduino to run the slave.
const int ARD_PIN = 10; //---------------------------------Specifies pin 10 as chip select pin.
const int EMGPIN = A1;
//Naming the reset functions
```

```
const int RESET = 0x2F; //---------------------------------This is the location to reset the device
const int NEW = 0x00; //-----------------------------------This is the code to sent to reset device
const int EXPANDSIGNBASE = 0xFFF;
int EXPANDSIGNSHIFTED = EXPANDSIGNBASE << 20;
int CalibrationCounter = 0;
int CalibrationCheck =0;
int microa;
int microb;
int rate;
int CalibrationSamples = 501;
int CalibrationCalculationCount = 0;
float CalculationStoragea = 0;
float CalculationStorageb = 0;
float CalculationStoragec = 0;
float XCalibrationValue=0;
float YCalibrationValue=0;
float ZCalibrationValue=0;




// Now that our constants are set we can write the loop
void setup() {
 Serial.begin(250000); //------------------------------------This sets the number of bits per second.
(sampling frequency x 20 bits). BAUD
 SPI.begin(); //--------------------------------------------This enables the Arduino SPI pin locations
as follows... clock CLK is pin 13...MISO is pin 12...MOSI is pin 11.
 pinMode(ARD_PIN,OUTPUT); //--------------------------------Tells us pin 10 should be an output
// Reset the device to make sure it starts clean
writeRegister(RESET,NEW); //--------------------------------This resets all setting to default each
time its plugged in.
delay(250);//-----------------------------------------------The reset takes time to finish
// Now we will configure the settings we named earlier
writeRegister(RANGE,MEAS_2G); //----------------------------This activates the range of 2G
measurements
writeRegister(POWER,PW_SET); //-----------------------------This enables measurement and dissables
TEMP and DRDY
//writeRegister(FILTER,SET_FILTER); //------------------------This enables filters

delay(100); //----------------------------------------------more initial setup time
```

MMG

```
}
void loop() {
// Begin our loop by organizing the data collected
int DataAddresses[] = {X_3,X_2,X_1,Y_3,Y_2,Y_1,Z_3,Z_2,Z_1}; //-------------------Creates a vector
containing all the X-axis data addresses.
int DataMeasures[] = {0,0,0,0,0,0,0,0,0};
int dataSize = 9;
int XaxisAddresses[] = {X_3,X_2,X_1};
int XaxisMeasures[] = {0, 0, 0};//--------------------------------Creates a vector containing all the
X-axis data points.
int YaxisAddresses[] = {Y_3,Y_2,Y_1}; //-------------------Creates a vector containing all the Y-axis
data addresses.
int YaxisMeasures[] = {0, 0, 0};//--------------------------------Creates a vector containing all the
Y-axis data points.
int ZaxisAddresses[] = {Z_3,Z_2,Z_1}; //-------------------Creates a vector containing all the Z-axis
data addresses.
int ZaxisMeasures[] = {0, 0, 0};//--------------------------------Creates a vector containing all the
Z-axis data points.
int axisdataSize = 3; //---------------------------------------------------------------State the size
of the vectors
int XCalibrationStorage[CalibrationSamples];
int YCalibrationStorage[CalibrationSamples];
int ZCalibrationStorage[CalibrationSamples];

/*readMultipleData(XaxisAddresses, axisdataSize, XaxisMeasures); //------------------------How to read
the data
// Merging split Data into one
  int XDATA = (XaxisMeasures[2] >> 4) + (XaxisMeasures[1] << 4) + (XaxisMeasures[0] << 12); // shifts
first entry right 4 bits, second entry left 4 bits, third entry left 12 bits
int XTestValue = XDATA << 12;
if (XTestValue < 0){
  XDATA = XDATA+EXPANDSIGNSHIFTED;
}
//Serial.print("X=");//-------------------------------------COMMENTED OUT SO THAT THE PLOTTER WILL
WORK, ALSO REMOVES LABELS IN SERIAL MONITOR
Serial.print(XDATA);// -------------------------------------COMMENT OUT TO NOT GRAPH X
Serial.print("\t");
//delay(.25);
readMultipleData(YaxisAddresses, axisdataSize, YaxisMeasures); //------------------------How to read
the data
// Merging split Data into one
```

```
  int YDATA = (YaxisMeasures[2] >> 4) + (YaxisMeasures[1] << 4) + (YaxisMeasures[0] << 12); // shifts
first entry right 4 bits, second entry left 4 bits, third entry left 12 bits
int YTestValue = YDATA << 12;
if (YTestValue < 0){
  YDATA = YDATA+EXPANDSIGNSHIFTED;
}
//Serial.print("Y=");//-------------------------------------COMMENTED OUT SO THAT THE PLOTTER WILL
WORK, ALSO REMOVES LABELS IN SERIAL MONITOR
Serial.print(YDATA);// -------------------------------------COMMENT OUT TO NOT GRAPH X
Serial.print("\t");
//delay(.25);
readMultipleData(ZaxisAddresses, axisdataSize, ZaxisMeasures); //------------------------How to read
the data
// Merging split Data into one
  int ZDATA = (ZaxisMeasures[2] >> 4) + (ZaxisMeasures[1] << 4) + (ZaxisMeasures[0] << 12); // shifts
first entry right 4 bits, second entry left 4 bits, third entry left 12 bits
int ZTestValue = ZDATA << 12;
if (ZTestValue < 0){
  ZDATA = ZDATA+EXPANDSIGNSHIFTED;
}
//Serial.print("Z=");//-------------------------------------COMMENTED OUT SO THAT THE PLOTTER WILL
WORK, ALSO REMOVES LABELS IN SERIAL MONITOR
Serial.print(ZDATA);// -------------------------------------COMMENT OUT TO NOT GRAPH X
Serial.print("\n");
//delay(.25);
*/




// Read the data from the accelerometer and EMG
readMultipleData(DataAddresses, dataSize,DataMeasures);
int XDATA = (DataMeasures[2] >> 4) + (DataMeasures[1] << 4) + (DataMeasures[0] << 12);
int YDATA = (DataMeasures[5] >> 4) + (DataMeasures[4] << 4) + (DataMeasures[3] << 12);
int ZDATA = (DataMeasures[8] >> 4) + (DataMeasures[7] << 4) + (DataMeasures[6] << 12);
int XTestValue = XDATA << 12;
int YTestValue = YDATA << 12;
int ZTestValue = ZDATA << 12;
//Correct the sign of the data
if (XTestValue < 0){
  XDATA = XDATA+EXPANDSIGNSHIFTED;
}
if (YTestValue < 0){
  YDATA = YDATA+EXPANDSIGNSHIFTED;
}
if (ZTestValue < 0){
  ZDATA = ZDATA+EXPANDSIGNSHIFTED;
}
//get EMG Data
analogReadResolution(12);
int EMGDATA = ((analogRead(EMGPIN))*1000-1600000);



//get data capture rate
/*microa = micros();
rate = 1000000/(microa-microb);
microb = microa;
*/
    //print data to be read
    //Serial.print("XDATA \t");
    Serial.print(XDATA + 75000);
    Serial.print("\t");
    //Serial.print("YDATA \t");
    Serial.print(YDATA);
    Serial.print("\t");
    //Serial.print("ZDATA \t");
    Serial.print(ZDATA - 75000);
    Serial.print("\t");
    //Serial.print("Rate \t");

    //Serial.print(rate);
    //Serial.print("\t EMGDATA \t");
    Serial.print(EMGDATA);
    Serial.print("\n");
```

```
// Set this delay for the loop rate
//delay (1);// ----------------------------------------------milisecond delay to set data sampling
rate
}
// This Changes the Data stored on the Accelerometer by a specific value
void writeRegister(byte thisRegister, byte thisValue) {
  byte dataToSend = (thisRegister << 1) | WRITE_TO;
  digitalWrite(ARD_PIN, LOW);
  SPI.transfer(dataToSend);
  SPI.transfer(thisValue);
  digitalWrite(ARD_PIN, HIGH);
}
// Read registry in specific device address
unsigned int readRegistry(byte thisRegister) {
  unsigned int result = 0;
  byte dataToSend = (thisRegister << 1) | READ_FROM;
  digitalWrite(ARD_PIN, LOW);
  SPI.transfer(dataToSend);
  result = SPI.transfer(0x00);
  digitalWrite(ARD_PIN, HIGH);
  return result;
}
// Read multiple registries
void readMultipleData(int *addresses, int dataSize, int *readedData) {
  digitalWrite(ARD_PIN, LOW);
  for(int i = 0; i < dataSize; i = i + 1) {
    byte dataToSend = (addresses[i] << 1) | READ_FROM;
    SPI.transfer(dataToSend);
    readedData[i] = SPI.transfer(0x00);
    //delay(2.5);
  }
  digitalWrite(ARD_PIN, HIGH);
}
```

// END OF CODE