```
// Car Accelerometer

#include <LiquidCrystal.h>
#include <Adafruit_ADXL345_U.h>
#include <SparkFun_External_EEPROM.h>

LiquidCrystal lcd( 12, 11, 10, 9, 8, 7 );  // create an lcd object and assign the pins

Adafruit_ADXL345_Unified accel = Adafruit_ADXL345_Unified(12345); // create an accelerometer object (accel) and assign it a
unique ID
sensors_event_t accelEvent;                                       // declare a sensor event variable

ExternalEEPROM extDisk;                  // instantiate an External EEPROM object and call it extDisk ( External Disk )
                                         // External EEPROM to write to or read from

const int noOfRows     =  40;
const int noOfCols     =  3;
const int noOf2dArrays =  2;

int saveGforcesDelay = 50;

int arrayNo = 0;

float gForcesArray[noOfRows][noOfCols][noOf2dArrays];

float x = 0.0;        // declare a variable to hold the x component of acceleration
float y = 0.0;        // declare a variable to hold the y component of acceleration
float z = 0.0;        // declare a variable to hold the z component of acceleration

const int pushButtonPin = 6;
const int buzzer        = 5;
const int LEDs          = 4;
const int xTriggerPin   = A3;
const int yTriggerPin   = A2;

float xMax = 0.0;
float yMax = 0.0;
float zMax = 0.0;
```

```
void setup()
  {
  int startPause = 2000;

  Serial.begin(9600);

  pinMode(pushButtonPin, INPUT_PULLUP);
  pinMode(buzzer, OUTPUT);
  pinMode(LEDs, OUTPUT);

  lcd.begin(16, 2);                    // set the display to 16 columns, 2 rows
  lcd.clear();                         // clear the display
  lcd.setCursor(1, 0);                 // set the cursor in the top row, 3 spaces to the right
  lcd.print("Accelerometer");
  lcd.setCursor(3, 1);                 // set the cursor in the bottom row, 6 spaces to the right
  lcd.print("By: Roman");
  delay(startPause);
  lcd.clear();


  lcd.setCursor(1, 0);                 // position the cursor at column 1 row 1
  lcd.print("Accelerometer");          // display "Accelerometor"

  if (accel.begin())                   // Initialize the accelerometer
    {
    lcd.setCursor(3, 1);               // Accelerometer was detected
    lcd.print("Detected");             // desplay "Detected"
    }
  else
    {
    lcd.setCursor(1, 1);               // Accelerometer was NOT detected
    lcd.print("NOT Detected");         // desplay "NOT Detected"

    while(true)                        // go into an infinite loop
      {
      // do nothing
      }
    }

  delay(startPause);
```

```
 // Set Accelerometer Range
 // set the operating range for the accelerometer to +/- 2 g.
 // higher values will have a wider measurement range.  Lower values will have more sensitivity.
 // 1 g = 32.1740 ft per sec per sec or 9.80665 meters per sec per sec.
 accel.setRange(ADXL345_RANGE_2_G);

 // Set Data Rate
 // this sets the rate at which th accelerometer output is updated.
 // rates above 100 Hz will exhibit increaed noice
 // rates below 6.25 Hz will be more sensitive to temperature variations.
 accel.setDataRate(ADXL345_DATARATE_25_HZ);


if (!extDisk.begin())
   {
   lcd.clear();                              // clear th Liquid Crystal Display (LCD)
   lcd.setCursor(1, 0);                      // set the LCD cursor to row 0, column 1
   lcd.print("Unable to Find");              // display "Unable to find"
   lcd.setCursor(1, 1);                      // set the LCD cursor to row 1, column 1
   lcd.print("EEPROM Memory");               // display "Freezing"
   while (true){;}                           // Freeze - go into "do nothing" infinite loop
   }
 else
   {
   lcd.clear();                              // clear th Liquid Crystal Display (LCD)
   lcd.setCursor(1, 0);                      // set the LCD cursor to row 0, column 1
   lcd.print("EEPROM Memory");               // display "Eeprom Memory"
   lcd.setCursor(3, 1);                      // set the LCD cursor to row 1, column 3
   lcd.print("Detected");                    // display "Detected"
   delay(startPause);                         // pause for "pauseTime" seconds

   lcd.clear();                              // clear th Liquid Crystal Display (LCD)
   lcd.setCursor(2, 0);                      // set the LCD cursor to row 0, column 2
   lcd.print("Memory Size");                 // display "Memory Size"
   lcd.setCursor(2, 1);                      // set the LCD cursor to row 1, column 5
   lcd.print(extDisk.length());              // display the memory size of the EEPROM
   lcd.print(" Bytes");                      // display "Memory Size"
   delay(startPause);                         // pause for "pauseTime" seconds
   }
```

```
 int epromArrayNo;
 int epromRowNo;
 extDisk.get(0, epromArrayNo);
 extDisk.get(2, epromRowNo);

if((epromArrayNo == noOf2dArrays) && (epromRowNo == noOfRows))
  {
  lcd.clear();
  lcd.print("  Reading Data");
  lcd.setCursor(0, 1);
  lcd.print("  From EEPROM");
  delay(startPause);

  int loc = 32;

  for (int i = 0; i < noOf2dArrays; i++)
    {
    for(int j = 0; j < noOfRows; j++)
      {
      for(int k = 0; k < noOfCols; k++)
        {
        extDisk.get(loc, gForcesArray[j][k][i]);
        loc = loc + 4;
        }
      }
    }
  }
 else
  {
   lcd.clear();
   lcd.print("Rows and Arrays");
   lcd.setCursor(0, 1);
   lcd.print("  Don't Match");
   delay(startPause);
   lcd.clear();
   lcd.print(" Erasing EEPROM");
   extDisk.put(0, noOf2dArrays);
```

```
    extDisk.put(2, noOfRows);

    int loc = 32;

    for (int i = 0; i < noOf2dArrays; i++)
      {
      for(int j = 0; j < noOfRows; j++)
        {
        for(int k = 0; k < noOfCols; k++)
          {
          extDisk.put(loc, 0.00);
          loc = loc + 4;
          }
        }
      }
    }
  int noOfSpaces = 7;
  int noOfStarts = 0;
  extDisk.get(4, noOfStarts);
  lcd.clear();
  lcd.print("Number of Starts:");
  if(noOfStarts > 99){noOfSpaces = noOfSpaces -1;}
  if(noOfStarts > 9999){noOfSpaces = noOfSpaces -1;}
  lcd.setCursor(noOfSpaces, 1);
  lcd.print(noOfStarts);
  delay(startPause);

  noOfStarts = noOfStarts + 1;
  extDisk.put(4, noOfStarts);
  }


void loop()
  {
  if (digitalRead(pushButtonPin) == LOW)
    {
    displayGforces();
    lcd.clear();
    lcd.setCursor(1,0);
    lcd.print(" - Plotting - ");
```

```
    plotGforces();
    delay(3000);
    }

  getGforces();

  float xTrigger = getTrigger(xTriggerPin);
  float yTrigger = getTrigger(yTriggerPin);

  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("X:");
  if(x>=0){lcd.print(" ");}
  lcd.print(x);
  lcd.print(" Y:");
  if(y>=0){lcd.print(" ");}
  lcd.print(y);
  lcd.setCursor(4,1);
  lcd.print("Z:");
  if(z>=0){lcd.print(" ");}
  lcd.print(z);
  lcd.setCursor(0,1);
  lcd.print(xTrigger,1);
  lcd.setCursor(12,1);
  lcd.print(yTrigger,1);

  if (x > xTrigger || abs(y) > yTrigger)
    {
    if (arrayNo < noOf2dArrays)
      {
      saveGforces(arrayNo);
      epromSave(arrayNo);
      arrayNo = arrayNo + 1;
      delay(500);
      }
    }

  delay(500);
  }
```

```
void getGforces()
  {
  accel.getEvent(&accelEvent);  // get an accelerometer event
                                // the event variable will contain the accelerometer ID
                                // as well as the x, y and z components of the acceleration

  x = accelEvent.acceleration.x;
  y = accelEvent.acceleration.y;
  z = accelEvent.acceleration.z;

  int xCal = 0;
  int yCal = 0;
  int zCal = 0;

  if (x >= 0)
    {
    xCal = map(x * 100, 0, 1080, 0, 981);
    }
  else
    {
    xCal = map(x * 100, 0, -990, 0, -981);
    }

  if (y >= 0)
    {
    yCal = map(y * 100, 0, 1000, 0, 981);
    }
  else
    {
    yCal = map(y * 100, 0, -1080, 0, -981);
    }

  if (z >= 0)
    {
    zCal = map(z * 100, 0, 890, 0, 981);
    }
  else
    {
    zCal = map(z * 100, 0, -1120, 0, -981);
```

```
    }

  y = xCal;
  z = yCal;
  x = -zCal;

  x = x/981;
  y = y/981;
  z = z/981;
  }


void saveGforces(int arrayNo)
  {
  for (int i = 0; i < noOfRows; i++)
    {
    getGforces();

    gForcesArray[i][0][arrayNo] = x;
    gForcesArray[i][1][arrayNo] = y;
    gForcesArray[i][2][arrayNo] = z;

    digitalWrite(buzzer, HIGH);
    digitalWrite(LEDs, HIGH);
    delay(saveGforcesDelay);
    digitalWrite(buzzer, LOW);
    digitalWrite(LEDs, LOW);
    delay(saveGforcesDelay);
    }
  }


void epromSave(int arrayNo)
  {
  int loc = 32 + (arrayNo *((noOfRows*noOfCols)*4));


  for(int i = 0; i <noOfRows; i++)
    {
    for(int j= 0; j < noOfCols; j++)
```

```
      {
      extDisk.put(loc, gForcesArray[i][j][arrayNo]);
      loc = loc + 4;
      }
    }
  }




void displayGforces()
  {
  for (int i = 0; i < noOf2dArrays; i++)
    {
    for (int j = 0; j < noOfRows; j++)
      {
      lcd.clear();
      lcd.print("X:");
      if(gForcesArray[j][0][i] >= 0){lcd.print(" ");}
      lcd.print(gForcesArray[j][0][i]);

      lcd.print(" Y:");
      if(gForcesArray[j][1][i] >= 0){lcd.print(" ");}
      lcd.print(gForcesArray[j][1][i]);
      lcd.setCursor(4,1);

      lcd.print("Z:");
      if(gForcesArray[j][2][i] >= 0){lcd.print(" ");}
      lcd.print(gForcesArray[j][2][i]);
      lcd.setCursor(0,1);
      lcd.print(i+1);
      lcd.setCursor(14,1);
      lcd.print(j+1);

      delay(250);
      }
     displayMaxGforces(i);
    }
  }
```

```
void plotGforces()
  {
  Serial.println("X-Axis Y-Axis Z-Axis");

  for (int i = 0; i < noOf2dArrays; i++)
    {
    for (int j = 0; j < noOfRows; j++)
      {
      for (int k = 0; k < noOfCols; k++)
        {
        Serial.print(10 * gForcesArray[j][k][i]);
        Serial.print(",");
        }

      Serial.println();
      }

    for (int j = 0; j < 10; j++)
      {
      Serial.print(0);
      Serial.print(",");
      Serial.print(0);
      Serial.print(",");
      Serial.println(0);
      }

    for (int j = 0; j < 10; j++)
      {
      getMaxGforces(i);
      Serial.print(xMax * 10);
      Serial.print(",");
      Serial.print(yMax * 10);
      Serial.print(",");
      Serial.println(zMax * 10);
      }

    for (int j = 0; j < 30; j++)
      {
      Serial.print(0);
```

```
      Serial.print(",");
      Serial.print(0);
      Serial.print(",");
      Serial.println(0);
      }
    }

    for (int j = 0; j < 20; j++)
      {
      Serial.print(0);
      Serial.print(",");
      Serial.print(0);
      Serial.print(",");
      Serial.println(0);
      }
    }


float getTrigger(int triggerPin)
  {
  int potValue = map(analogRead(triggerPin), 0, 1023, 0, 100);
  float triggerValue = (float)potValue/100;
  return(triggerValue);
  }

void displayMaxGforces(int arrayNo)
    {
    getMaxGforces(arrayNo);

    lcd.clear();
    lcd.print("X:");
    if(xMax >= 0){lcd.print(" ");}
    lcd.print(xMax);

    lcd.print(" Y:");
    if(yMax >= 0){lcd.print(" ");}
    lcd.print(yMax);

    lcd.setCursor(4,1);
    lcd.print("Z:");
```

```
    if(zMax >= 0){lcd.print(" ");}
    lcd.print(zMax);
    lcd.setCursor(0,1);
    lcd.print(arrayNo+1);
    lcd.setCursor(13,1);
    lcd.print("MAX");

    delay(10000);
    }


void getMaxGforces(int arrayNo)
  {
  xMax = 0.0;
  yMax = 0.0;
  zMax = 0.0;

  for (int j = 0; j < noOfRows; j++)
    {
    if(abs(gForcesArray[j][0][arrayNo]) > abs(xMax))
      {
      xMax = gForcesArray[j][0][arrayNo];
      }

    if(abs(gForcesArray[j][1][arrayNo]) > abs(yMax))
      {
      yMax = gForcesArray[j][1][arrayNo];
      }

    if(abs(gForcesArray[j][2][arrayNo]) > abs(zMax))
      {
      zMax = gForcesArray[j][2][arrayNo];
      }
    }
  }
```