```
#include <Arduino.h>

 //define analog-digital pin for ADXL335
#define ADXL335_X 0
#define ADXL335_Y 1
#define ADXL335_Z 2
#define BATTERY_PIN 3

#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <SoftwareSerial.h>
#include <Adafruit_MMA8451.h>
#include <Adafruit_Sensor.h>
#include <avr/io.h>
#include <avr/eeprom.h>
#include <avr/interrupt.h>

void getDigitalAcc();
void getAnalogAcc();
void getVelocityAndDisplacement();
void initOffsets();
void initDigitalAcc();
void sendValuesToSlave();
void dispTestVals();
void displayLCD();
void initTimer();
void getBatteryLevel();
void manageTimer();
void initButtons();
void initLEDs();
void checkButtons();
void initBestVals();
void checkBestVals();
void clearEepromBest();
void get5sAvg();
void getRoadType();
void initADC();
int adc(int adc_channel);
int ADC_read(int adc_channel);
ISR(TIMER0_COMPA_vect );
void Receive_GPS_Data();



const int ARDUINO_SLAVE_ADDRESS = 9;

LiquidCrystal_I2C lcd(0x27,20,4);  // set the LCD address to 0x27 for a 16 chars
  and 2 line display

float z_acc=0, y_acc=0, x_acc=0; //accelerations
float z_5s=0, y_5s=0, x_5s=0; //5s avgs
float y_vel=0, x_vel=0; //velocities
float x_disp=0, y_disp=0; //displacements
float mag_acc=0, mag_vel=0; float mag_disp=0; //magnitudes (vector sums)
```

```
float x_sum=0, y_sum=0, z_sum=0;
int passed_5s=0;
float x_sum_last=0, y_sum_last=0, z_sum_last=0;
int iter_5savg=1;

float x_acc_sum=0, y_acc_sum=0; //sums for calculating vel and disp
int iter_acc = 0; //iteration for above

float best_vel=0, best_acc=0, best_disp=0;
//addresses for EEPROM
unsigned int best_vel_addr=0;
unsigned int best_acc_addr=10;
unsigned int best_disp_addr=30;

String road_type = "Smooth";
int mode=0;
int battery_percentage=0;

float zoffset=0, yoffset=0, xoffset=0; //offsets for initial acceleration values
  (denying gravitation)
int measurement_timegap=0, measurement_timegap_start=0; //variables to measure time
   between two acceleration measurements for caluclationg vel and disp

char updateDisplay = 1;
volatile int timerCnt = 0;

boolean recently_clicked = false;

//digital accelerometer
Adafruit_MMA8451 mma = Adafruit_MMA8451();

  int Gpsdata;             // for incoming serial data
  unsigned int finish =0;  // indicate end of message
  unsigned int pos_cnt=0;  // position counter
  unsigned int lat_cnt=0;  // latitude data counter
  unsigned int log_cnt=0;  // longitude data counter
  unsigned int flg    =0;  // GPS flag
  unsigned int com_cnt=0;  // comma counter
  char la[20];            // latitude array
  char lg[20];             // longitude array

SoftwareSerial gpsSerial(11,12);

void getDigitalAcc() {
  measurement_timegap_start = millis();
  sensors_event_t event;
  mma.getEvent(&event);
  float x_sum_test = 0, z_sum_test=0, y_sum_test=0;
  for(int i = 0; i< 10; i++) {
    z_sum_test+=event.acceleration.z - zoffset;
  }

  z_acc = z_sum_test/10;
  z_sum_last = z_acc;

}
```

```
void getAnalogAcc() {

  int xadc, yadc;
  xadc = ADC_read(ADXL335_X);
  yadc = ADC_read(ADXL335_Y);

  x_acc = (5.09*(float)(xadc)/1023.0-1.65)/0.33*9.806-xoffset;
  y_acc = (5.09*(float)(yadc)/1023.0-1.65)/0.33*9.806-yoffset;

  x_acc_sum += x_acc;
  y_acc_sum += y_acc;

  x_sum_last = x_acc;
  y_sum_last = y_acc;

  iter_acc++;
  mag_acc = sqrt(x_acc*x_acc+y_acc*y_acc);
}

void getVelocityAndDisplacement() {
  float x_acc_test=0,y_acc_test=0;
  x_acc_test = x_acc_sum/iter_acc;
  y_acc_test = y_acc_sum/iter_acc;
  if(abs(x_acc_test)>=0.15){
    x_vel += x_acc_test*20/1000*3.6; //*3.6 = m/s -> km/h
  }
  if(abs(y_acc_test)>=0.15){
    y_vel += y_acc_test*20/1000*3.6; //*3.6 = m/s -> km/h
  }
  if(abs(x_vel)>=5.0) {
    x_disp += x_vel/3.6*2.0/10;
  }
  if(abs(y_vel)>=5.0) {
    y_disp += y_vel/3.6*2.0/10;
  }
  mag_vel = sqrt(x_vel*x_vel+y_vel*y_vel);
  mag_disp = ((sqrt(x_disp*x_disp+y_disp*y_disp)));
}


void initOffsets() {
  getDigitalAcc();
  getAnalogAcc();
  zoffset = z_acc;
  xoffset = x_acc;
  yoffset = y_acc;
}

void initDigitalAcc() {
  mma.begin();
  mma.setRange(MMA8451_RANGE_2_G);
  delay(100);
}

void sendValuesToSlave() {
```

```
  Receive_GPS_Data();

  Wire.beginTransmission(ARDUINO_SLAVE_ADDRESS);
  Wire.write("lon");
  delay(100);
  Wire.write(lg);
  delay(100);
  Wire.write("lat");
  Wire.write(la);
  Wire.endTransmission();
  delay(1);

  finish = 0;pos_cnt = 0;
}

void dispTestVals() {
  lcd.clear();
  lcd.setCursor(1,0);
  lcd.print(String(x_vel,6));

}

void displayLCD() {
  if(updateDisplay==0) {
    return;
  } else {
    updateDisplay = 0;
  }
  lcd.clear();
  if(mode==0) {
    lcd.setCursor(0,0);
    lcd.print("Acceleration");
    lcd.setCursor(0,1);
    lcd.print("x: "+String(x_acc,1));
    lcd.setCursor(0,2);
    lcd.print("y: "+String(y_acc,1));
    lcd.setCursor(0,3);
    lcd.print("z: "+String(z_acc,1));
  }
  else if(mode==1) {
    lcd.setCursor(0,0);
    lcd.print("5s average acc");
    lcd.setCursor(0,1);
    lcd.print("x: "+String(x_5s,1));
    lcd.setCursor(0,2);
    lcd.print("y: "+String(y_5s,1));
    lcd.setCursor(0,3);
    lcd.print("z: "+String(z_5s,1));
  }
  else if(mode==2) {
    lcd.setCursor(0,0);
    lcd.print("Velocity");
    lcd.setCursor(0,2);
    lcd.print("x: "+String(x_vel,1));
    lcd.setCursor(0,3);
    lcd.print("y: "+String(y_vel,1));
```

```
  }
  else if(mode==3) {
    lcd.setCursor(0,0);
    lcd.print("Displacement");
    lcd.setCursor(0,2);
    lcd.print("x: "+String(x_disp,4));
    lcd.setCursor(0,3);
    lcd.print("y: "+String(y_disp,4));
  }
  else if(mode==4) {
    lcd.setCursor(0,0);
    lcd.print("Magnitude");
    lcd.setCursor(0,1);
    lcd.print("Acc: " +String(mag_acc,1));
    lcd.setCursor(0,2);
    lcd.print("Vel: " +String(mag_vel,1));
    lcd.setCursor(0,3);
    lcd.print("Disp: " +String(mag_disp));
  }
  else if(mode==5) {
    lcd.setCursor(0,0);
    lcd.print("Best scores");
    lcd.setCursor(0,1);
    lcd.print("Acc: "+String(best_acc,1));
    lcd.setCursor(0,2);
    lcd.print("Vel: "+String(best_vel,1));
    lcd.setCursor(0,3);
    lcd.print("Disp: "+String(best_disp));
  }
  else if(mode==6) {
    lcd.setCursor(0,0);
    lcd.print("Battery: "+String(battery_percentage)+"%");
    lcd.setCursor(0,2);
    lcd.print(road_type + " road");
  }
}

void initTimer() {
  cli();
  TCCR0A = 0;// set entire TCCR0A register to 0
  TCCR0B = 0;// same for TCCR0B
  TCNT0  = 0;//initialize counter value to 0
  TCCR0A |= (1<<WGM01);
  OCR0A = 0xF9;
  TIMSK0 |= (1 << OCIE0A);
  TCCR0B |= (1 << CS02);
  sei();
}

void getBatteryLevel() {
  battery_percentage = analogRead(BATTERY_PIN);
  battery_percentage = map(battery_percentage, 684, 845, 0, 100);
  if(battery_percentage>=70) {
    PORTD |= (1<<PORTD4);
    PORTD &= ~(1<<PORTD2);
    PORTD &= ~(1<<PORTD3);
```

```
  } else if(battery_percentage>=40 && battery_percentage<70) {
    PORTD |= (1<<PORTD3);
    PORTD &= ~(1<<PORTD2);
    PORTD &= ~(1<<PORTD4);
  } else if(battery_percentage<40) {
    PORTD |= (1<<PORTD2);
    PORTD &= ~(1<<PORTD3);
    PORTD &= ~(1<<PORTD4);
  }
  }
}

void manageTimer() {
  if(timerCnt%100==0) {
    getBatteryLevel();
    updateDisplay = 1;
    checkBestVals();
  }
  if(timerCnt%6==0) {
    getVelocityAndDisplacement();
    iter_acc=0;
    x_acc_sum=0;
    y_acc_sum=0;
  }
  if(timerCnt%200==0) {
    recently_clicked = false;
    timerCnt=0;
  }
  if(timerCnt>=1250) {
    passed_5s++;
    timerCnt = 0;
  }
}

void initButtons() {
  DDRD &= ~(1<<DDD5);
  DDRD &= ~(1<<DDD6);
  DDRD &= ~(1<<DDD7);
}

void initLEDs() {
  DDRD |= (1<<PORTD2);
  DDRD |= (1<<PORTD3);
  DDRD |= (1<<PORTD4);
}

void checkButtons() {
  if(recently_clicked) {
    return;
  }
  if(digitalRead(5)==HIGH) {
    if (mode>=6) {
      mode=0;
    } else {
      mode++;
    }
```

```
    displayLCD();
    recently_clicked = true;

  }
  else if(digitalRead(6)==HIGH) {
    zoffset=z_acc+zoffset;
    xoffset=x_acc+xoffset;
    yoffset=y_acc+yoffset;
    z_acc=0; y_acc=0; x_acc=0;
    z_5s=0; y_5s=0; x_5s=0;
    y_vel=0; x_vel=0;
    x_disp=0; y_disp=0;
    mag_acc=0; mag_vel=0; mag_disp=0;
    recently_clicked = true;
  }
  else if(digitalRead(7)==HIGH) {
    clearEepromBest();
    recently_clicked = true;
  }
}

void initBestVals() {
  best_vel = eeprom_read_float((float*)best_vel_addr);
  best_acc = eeprom_read_float((float*)best_acc_addr);
  best_disp = eeprom_read_float((float*)best_disp_addr);
}

void checkBestVals() {
  if(best_vel<mag_vel) {
    best_vel=mag_vel;
    eeprom_write_float((float*)best_vel_addr,best_vel);
  }
  if(best_acc<mag_acc) {
    best_acc=mag_acc;
    eeprom_write_float((float*)best_acc_addr,best_acc);
  }
  if(best_disp<mag_disp) {
    best_disp=mag_disp;
    eeprom_write_float((float*)best_disp_addr,best_disp);
  }
}

void clearEepromBest() {
  eeprom_write_float((float*)best_vel_addr,0.0);
  eeprom_write_float((float*)best_acc_addr,0.0);
  eeprom_write_float((float*)best_disp_addr,0.0);
}

void get5sAvg() {
  if(passed_5s>=1) {
    x_sum = x_sum + x_acc - x_sum_last;
    y_sum = y_sum + y_acc - y_sum_last;
    z_sum = z_sum + z_acc - z_sum_last;
    x_5s = x_sum/iter_5savg;
    y_5s = y_sum/iter_5savg;
    z_5s = z_sum/iter_5savg;
```

```
  } else {
    x_sum += x_acc;
    y_sum += y_acc;
    z_sum += z_acc;
    x_5s = x_sum/(iter_5savg);
    y_5s = y_sum/(iter_5savg);
    z_5s = z_sum/(iter_5savg);
    iter_5savg++;
  }
}

void getRoadType() {
  if(abs(z_5s)>=0.54) {
    road_type="Bumpy";
  } else {
    road_type="Smooth";
  }
}

void initADC() {
  ADMUX = (1<<REFS0);
  ADCSRA = (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0) | (1 << ADEN);
}

int adc(int adc_channel)
{
  ADMUX &= 0xF0; //clear any previous used channel, but keep internal reference
  ADMUX |= adc_channel; //set the desired channel
  //start conversion
  ADCSRA |= (1 << ADSC);
  //wait till conversion is finished
  while ( (ADCSRA & (1 << ADSC)) );
  //now return result to the calling function as a 16 bit unsigned int
  return ADC;
}

int ADC_read(int adc_channel)
{
  float sum = 0;
  int samples = 10;
  float tmpVal;
  for (int i = 0; i < samples; i++)
  {
    sum += adc(adc_channel);
  }
  return sum/samples;
}

void setup() {
  Wire.begin();
  lcd.init();
  lcd.backlight();
  gpsSerial.begin(9600);
  initADC();
  initDigitalAcc();
```

```
  initOffsets();
  initTimer();
  initButtons();
  initLEDs();
  initBestVals();
}

void loop() {
  getDigitalAcc();
  getAnalogAcc();
  get5sAvg();
  getRoadType();
  displayLCD();
  manageTimer();
  checkButtons();

  sendValuesToSlave();

}

ISR(TIMER0_COMPA_vect) {
  timerCnt+=1;
}

  void Receive_GPS_Data()
  {
    while(finish==0){
      while(gpsSerial.available()>0){         // Check GPS data
       Gpsdata = gpsSerial.read();
       flg = 1;
      if( Gpsdata=='$' && pos_cnt == 0)   // finding GPRMC header
        pos_cnt=1;
      if( Gpsdata=='G' && pos_cnt == 1)
        pos_cnt=2;
      if( Gpsdata=='P' && pos_cnt == 2)
        pos_cnt=3;
      if( Gpsdata=='R' && pos_cnt == 3)
        pos_cnt=4;
      if( Gpsdata=='M' && pos_cnt == 4)
        pos_cnt=5;
      if( Gpsdata=='C' && pos_cnt==5 )
        pos_cnt=6;
      if(pos_cnt==6 &&  Gpsdata ==','){   // count commas in message
        com_cnt++;
        flg=0;
      }

      if(com_cnt==3 && flg==1){
       la[lat_cnt++] =  Gpsdata;         // latitude
       flg=0;
      }

      if(com_cnt==5 && flg==1){
        lg[log_cnt++] =  Gpsdata;         // Longitude
        flg=0;
      }
```

```
        if( Gpsdata == '*' && com_cnt >= 5){
          com_cnt = 0;                        // end of GPRMC message
          lat_cnt = 0;
          log_cnt = 0;
          flg     = 0;
          finish  = 1;

       }
     }
}
}
```