

**ДВАДЕСЕТА УЧЕНИЧЕСКА КОНФЕРЕНЦИЯ**

**УК’20**

**ТЕМА НА ПРОЕКТА**

**DriveStyle**

**Автор:**

**Владимир Валентинов Вълков**  
ППМГ „Добри Чинтулов”, гр.Сливен, 9в клас  
  
Email: [**vladimirvylkov07@gmail.com**](mailto:vladimirvylkov07@gmail.com)Телефон: **0877 478497**

**Научен ръководител (консултант):**

Петър Веселинов Стоянов

Управител, Астра Пейджинг ЕООД

Email: **peter@astrapaging.com** Телефон: **0878 624434**

**Резюме на български език  
  
DriveStyle**

Всички фирми, които разполагат с голям брой служебни автомобили инсталират GPS устройства с които следят както маршрута на шофьорите, така и скоростта с която се движат по пътя. За съжаление, поддържането на ниска скорост не е единствения фактор за безопасно шофиране и опазване на автомобила. „Спортния“ стил на каране, невниманието за дупки по пътя, рязко тръгване и спиране на светофари са масово явление особено когато се шофира „чужда“ кола.

DriveStyle е проект, базиран на Arduino Uno, чиято цел е да следи както маршрут и скорост, така и ускоренията на автомобила по трите оси в пространството. Данните се записват на SD карта и в последствие се анализират със софтуер написан на Python.

Проектът може да помогне на много собственици на автомобили, като ги информира за начина по който те са управлявани.

**Резюме на английски език  
  
DriveStyle**

All companies that have a large number of company cars install GPS devices. With these GPS devices they follow the route and the speed of the driver. Unfortunately keeping low speed is not the only indicator of safe driving. “Sports” style of driving, inattention to holes along the way, sudden starting and stopping at traffic lights are massive phenomena especially when driving a foreign car.

DriveStyle is a project based on Arduino Uno. Its aim is to track the route, speed and accelerations in all three axes. All data is saved onto a SD card and analyzed with a Python software.

The project can help many car owners by informing them how their cars are driven.

**УВОД**Целта на проекта е да се създаде **устройство, което анализира стила на каране на шофьора.**

Опитът ми със C/C++ ми помогна да навляза бързо в света на микроконтролерите, както и да създавам различни проекти с помощта на Arduino.   
Завършеното устройство има за цел да помогне на много хора в България като ги информира за техният стил на каране на автомобил.  
  
Поставените цели са:

1.Увеличаване на пътната безопасност

2.Следене на маршрут и скорост

3.Следене на ускоренията на автомобила по трите оси в пространството.

4.Анализиране на резултатите.

Етапи на разработка:

1. Определяне на цялостната идея на проекта.   
2. Разделяне на проекта на отделни етапи и планиране на срокове за завършване.   
3. Запознаване с начина на работа на Arduino, както и останалите компоненти на устройството.   
4. Изработване на цялостна схема на проекта и закупуване на нужните хардуерни части.   
5. Създаване на цялостен сорс код, управляващ устройството.   
6. Разработка на тестови модели на устройството.   
7.Анализиране на положителните страни и недостатъците на продукта.   
8. Създаване на краен продукт, работещ надеждно и безотказно.   
10. Оформяне на документация и цялостен завършек на проекта.

Документацията е разделена на следните глави:   
**Глава 1 - Проектиране и реализация на хардуера**   
В тази глава съм описал основните компоненти на устройството и неговата функционална схема.   
**Глава 2 - Софтуер**Тук е представен и разгледан подробно целия сорс код, който управлява устройството.   
**Глава 3 - Основни моменти от разработката на проекта**   
В тази глава са описани основните проблеми и моменти, през които преминах в процеса на създаване на устройството.   
**Глава 4 - Снимки и начин на работа**   
Представени са снимки на крайния продукт и начина му на използване в ежедневието.   
**Глава 5 - Заключение**   
Описание на постигнатите резултати и бъдещи идеи за развитие на проекта.

**Глава 1- Проектиране и реализация на хардуера   
   
1.1 Компоненти**

**1.1.1 Микроконтролер**

В основата на проекта стои контролер Arduino Uno, който намира все по-голямо приложение в различните проекти свързани с автоматизация. Всеки човек с минимални познания по електроника може да експериментира с него без големи парични вложения, тъй като са както евтини, така и лесни за употреба.



**Arduino Uno Спецификации:**

Микроконтролер: ATmega328P

Работно напрежение: 5 V

Захранващо напрежение (препоръчително): 7-12 V

Цифрови I/O порта: 14 (от които 6 могат да са PWM изходи)

Аналогови входове: 6

Максимален ток на I/O порт: 40 mA

Прогрaмируема памет: 32 KB, от които 0,5 KB заети от буутлоудъра

SRAM: 2 KB

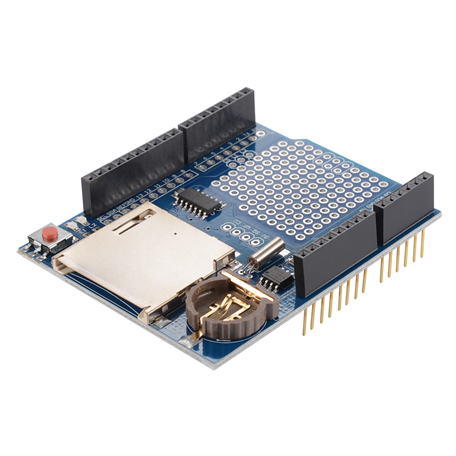
EEPROM: 1 KB

Тактова честота: 16 MHz

**1.1.2 Data Logger Shield**

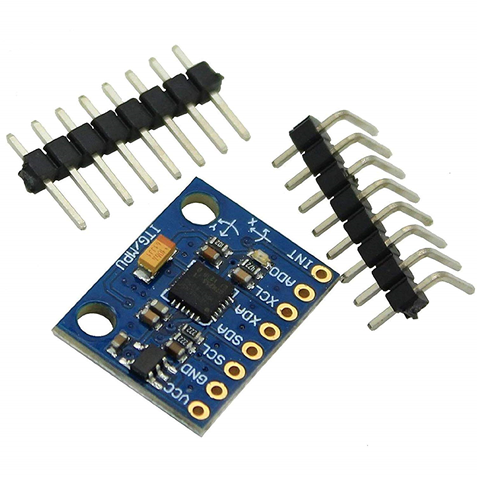
Другият основен компонент на устройството е Data Logger Shield.Той се разполага на Arduino Uno контролера и има две основни части: SD card logger и Real Time Cloak.

**Снимки на и компонента:**



# 1.1.3 Accelerometer

В устройството се използва сензора за намиране на ускоренията по трите оси в пространството – акселерометър. Разполага се на Data Logger Shield и е много чуствителен.



**1.1.4** **GPS модул**

Друг сензор, който се използва в устройството е GPS модула. От него се взимат географска дължина, географска широчина и скороста в km/h.Разполага се на Data Logger Shield.



**1.2 Функционална схема на устройството**



**1.2.1 Описание на всички връзки между компонентите:**

* Data logger Shield се свързва към Arduino контролера
* Accelerometer се свързва към Arduino микроконтролера чрез пинове A4 и A5
* GPS модул се свързва към Arduino контролера чрез пинове D3 и D4

**Глава 2 - Софтуер**

Системата има два софтуерни компонента:

* Софтуер на C за работа на хардуера
* Софтуер на Python за графично изобразяване на данните

**2.1 Софтуер на Arduino**

Устройството се управлява от функционален сорс код, написан на средата за разработка (**Integrated Development Environment - IDE**) на Arduino, която може да се изтегли безплатно от официалния сайт. Arduino използват адаптирана версия на езика C/C++, а в мрежата са налични много безплатни упътвания, уроци и примери, както за начинаещи, така и за напреднали.

**2.1.1 Сорс код**

#include "RTClib.h"

#include <SPI.h>

#include <SD.h>

#include<Wire.h>

#include <TinyGPS++.h>

#include <SoftwareSerial.h>

unsigned long m = 0, m1 = 0;

int counter = 0;

int buff\_id = 0;

const int MPU\_addr=0x69;

int16\_t AcX,AcY,AcZ, AcXold, AcYold, AcZold;

float dX, dY , dZ,dXa,dYa,dZa,dXaold,dYaold,dZaold;

char filename[20],csv[100];

const int chipSelect = 10;

static const int RXPin = 4, TXPin = 3;

static const uint32\_t GPSBaud = 9600;

double latitude=0.0, longitude=0.0, velocity=0.0;

File myFile;

Sd2Card card;

SdVolume volume;

SdFile root;

RTC\_DS1307 rtc;

TinyGPSPlus gps;

SoftwareSerial ss(RXPin, TXPin);

void fLog()

{

char tstamp[30];

myFile = SD.open(filename, FILE\_WRITE);

if (myFile)

{

DateTime now3 = rtc.now();

sprintf(tstamp,"%04d-%02d-%02d %02d:%02d:%02d;",now3.year(),now3.month(),now3.day(),now3.hour(),now3.minute(),now3.second());

myFile.print(tstamp);

myFile.print((unsigned int)dXa, DEC);

myFile.print(';');

myFile.print((unsigned int)dYa, DEC);

myFile.print(';');

myFile.print((unsigned int)dZa, DEC);

myFile.print(';');

myFile.print(latitude,5);

myFile.print(';');

myFile.print(longitude,5);

myFile.print(';');

myFile.println(velocity,0);

}

myFile.close();

}

void readAcceleration()

{

Wire.beginTransmission(MPU\_addr);

Wire.write(0x3B);

Wire.endTransmission(false);

Wire.requestFrom(MPU\_addr,14,true);

AcX=Wire.read()<<8|Wire.read();

AcY=Wire.read()<<8|Wire.read();

AcZ=Wire.read()<<8|Wire.read();

}

void setup() {

Serial.begin(9600);

ss.begin(GPSBaud);// setup GPS

//setup Accelerometer

Wire.begin();

Wire.beginTransmission(MPU\_addr);

Wire.write(0x6B); // PWR\_MGMT\_1 register

Wire.write(0); // set to zero (wakes up the MPU-6050)

Wire.endTransmission(true);

//setup RTC

if (! rtc.begin()) {

Serial.println("Couldn't find RTC");

}

if (! rtc.isrunning()) {

Serial.println("RTC is NOT running!");

rtc.adjust(DateTime(F(\_\_DATE\_\_), F(\_\_TIME\_\_)));

}

//generate filename

DateTime now = rtc.now();

sprintf(filename, "%02d%02d%02d%02d.csv", now.month(), now.day(), now.hour(), now.minute()); //set filename

//setup SD card

SD.begin(chipSelect);

//initialize accelerations

readAcceleration();

}

void loop() {

//log the data into the SD card every 1s

if(millis() - m >= 1000)

{

m = millis();

dXa = round(sqrt(dXa/ counter));

dYa = round(sqrt(dYa/counter));

dZa = round(sqrt(dZa/counter));

counter = 0;

latitude = gps.location.lat();

longitude = gps.location.lng();

velocity = gps.speed.kmph();

fLog();

Serial.print("Latitude = ");

Serial.print(latitude, 5);

Serial.print(" Longitude = ");

Serial.print(longitude, 5);

Serial.print(" Speed = ");

Serial.print(velocity, 0);

Serial.print(" km/h X = ");

Serial.print(dXa, 0);

Serial.print(" Y = ");

Serial.print(dYa, 0);

Serial.print(" Z = ");

Serial.println(dZa, 0);

}

//get acceleration every 100ms

if(millis() - m1 >= 100)

{

m1 = millis();

DateTime now2 = rtc.now();

AcXold = AcX;

AcYold = AcY;

AcZold = AcZ;

readAcceleration();

dX = AcX - AcXold;

dY = AcY - AcYold;

dZ = AcZ - AcZold;

dXa += dX \* dX;

dYa += dY \* dY;

dZa += dZ \* dZ;

counter++;

}

//loop to get GPS cordinates

while(ss.available() > 0)

{

gps.encode(ss.read());

}

}

**2.2 Софтуер на Python**

Сорс кодът за изобразяване на графиките е изключително прост, но функционален. Написан е на средата за разработка Jupyter, която може да се изтегли от официалния сайт .

**2.2.1 Сорс код**

import matplotlib.pyplot as plt

import numpy as np

import csv

time = []

x = []

y = []

z = []

velocity = []

with open('test.csv','r') as csvfile:

    plots = csv.reader(csvfile, delimiter=';')

    for row in plots:

        time.append(np.datetime64(row[0]))

        x.append(float(row[1]))

        y.append(float(row[2]))

        z.append(float(row[3]))

        velocity.append(float(row[6]))

fig = plt.figure(figsize=(30,15))

plt.subplot(4,1,1)

plt.plot(time, x,color = 'red',marker='o' ,label='X axis',linewidth=2, markersize=5)

plt.ylabel('Values', size = 25)

plt.legend(fontsize = 'xx-large')

plt.title('Acceleration', size = 25)

plt.subplot(4,1,2)

plt.plot(time, y,color = 'green',marker='o' ,label='Y axis',linewidth=2, markersize=5)

plt.ylabel('Values', size = 25)

plt.legend(fontsize = 'xx-large')

plt.subplot(4,1,3)

plt.plot(time, z,color = 'blue',marker='o' ,label='Z axis',linewidth=2, markersize=5)

plt.ylabel('Values', size = 25)

plt.legend(fontsize = 'xx-large')

plt.subplot(4,1,4)

plt.plot(time, velocity, label='velocity',marker= 'o',linewidth=2, markersize=5)

plt.xlabel('Time', size = 25)

plt.ylabel('Values', size = 25)

plt.legend(fontsize = 'xx-large')

plt.title('Speed in km/h', size = 25)

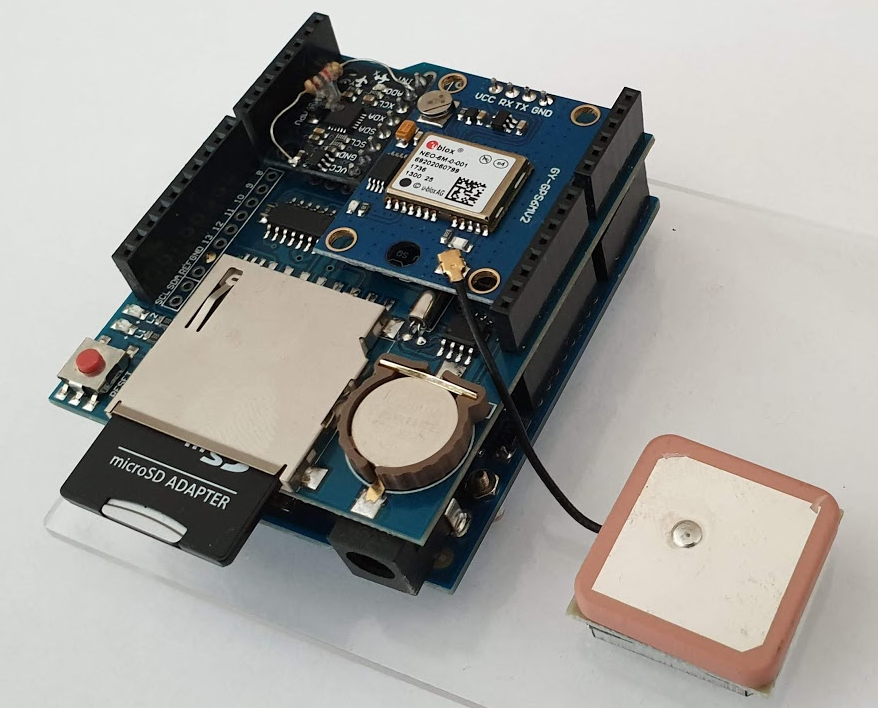
plt.show()

**Глава 3 - Основни моменти от разработката на проекта**

За да се получи завършен и безотказно работещ продукт трябваше да се преодолеe един проблем.   
  
Акселерометъра и часовника се намират на една и съща шина и споделят едно и също ID. Това се оказа проблем, когато се иска информация от часовника, защото се получаваше информация от акселерометъра. Реших този проблем като промених ID на акселерометъра чрез PULL\_UP резистор на PIN AD0.

**Глава 4 - Снимки и начин на работа**

**4.1 Завършеното устройство**



**4.2 Начин на работа:**

Начинът на работа е много прост. Първо Accelerometer измерва ускоренията и вибрациите по трите оси в пространството. Тези стойности ги пресмята с формулата средно квадратично с цел превръщане на стойностите в положителни. След това се измерва GPS координати и скорост в km/h. В SD карта се запаметяват следните стойности: 1) Време 2) Вибрации по X 3) Вибрации по Y 4) Вибрации по Z 5) Географска ширина 6) Географска дължина 7) Скорост на автомобила (км/ч).Ето няколко примерни реда от csv файла:

**2019-11-23 07:54:36;933;2235;2004;42.67390;26.32141;42**

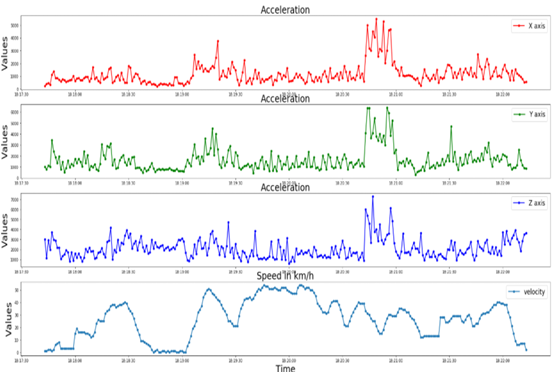
**2019-11-23 07:54:37;811;1727;2741;42.67383;26.32153;43**

**2019-11-23 07:54:38;1436;1530;1683;42.67376;26.32164;45**

**2019-11-23 07:54:39;964;1269;1828;42.67370;26.32176;44**

**2019-11-23 07:54:40;1292;1736;1682;42.67363;26.32187;44**

Тези резултати се изобразяват на графика чрез Python. Ето една примерна графика:

****

**Глава 5 – Заключение**

**5.1 Цел и постигане на целта**

Целта на проекта е да се направи **устройство, което анализира стила на каране на шофьора.**Целта е постигната посредством използване на Arduino микроконтролер със специализиран софтуер написан на езика „С“.   
Основните предимства на устройството са:

•Устройството с много голяма чувствителност прави запис на всички ускорения и вибрации на автомобила по трите оси в пространството.

•Паралелно с всички отчети се записват координатите и скоростта от GPS, което позволява не само анализ на начина на шофиране, а анализ на състоянието на пътната настилка

•Високата чувствителност позоволява сравнителен анализ на вибрациите причинени от еднакъв начин на шофиране и еднаква настилка при различни марки автомобили.

**5.2 Използвани софтуерни продукти**

За реализацията на проекта е използвана средата за разработка на Arduino микроконтролери - **Arduino IDE**. За изобразяването на резултатите на графика е използванo IDE - **Jupyter** .

* 1. **Бъдещо развитие**

•Създаване на мобилно приложение, което да постига същите резултати с помощта на сензорите вградени във всички съвременни мобилни телефони

•Създаване на web базирана система и база данни, които да обединят и анализират данните получени от много на брой шофьори по различните пътища на България. По този начин е възможно да се създаде електронна карта на състоянието на пътната настилка и опасните участъци.

**ИЗТОЧНИЦИ НА ИНФОРМАЦИЯ**

* Accelerometer – <https://create.arduino.cc/projecthub/Nicholas_N/how-to-use-the-accelerometer-gyroscope-gy-521-6dfc19>
* GPS - <https://create.arduino.cc/projecthub/ruchir1674/how-to-interface-gps-module-neo-6m-with-arduino-8f90ad>
* Matplotlib - <https://matplotlib.org/>