

Департамент образования и науки города Москвы
Государственное бюджетное общеобразовательное учреждение города
Москвы "Школа № 1575"

Система помощи судейству на дрифт соревнованиях

Выполнил: учащийся 11 класса
Клитный Сергей Викторович
Научный руководитель: учитель робототехники
Юхарин Павел Иванович

Москва, 2023

Содержание

| | |
|-----------------------------------|----|
| • Введение | 3 |
| • Теоретическая часть..... | 5 |
| • Методика выполнения работы..... | 6 |
| • Программная часть..... | 12 |
| • Расчёт стоимости..... | 24 |
| • Анализ рынка..... | 27 |
| • Вывод..... | 28 |
| • Список литературы..... | 29 |
| • Приложение 1..... | 30 |
| • Приложение 2..... | 37 |
| • Приложение 3..... | 40 |
| • Приложение 4..... | 42 |
| • Приложение 5..... | 44 |

Введение

Мы привыкли воспринимать дрифт, как забаву для богатой молодёжи. Ведь они не редко попадают в аварии, а для восстановления машин необходимы большие деньги. Но на самом деле дрифт – это вид автоспорта, характеризующийся использованием управляемого заноса на максимально возможных для удержания на трассе скорости и угла к траектории. Соревнования проводятся на сухом асфальте, трассах с большим количеством поворотов. Этот вид автоспорта, основанный на зрелищности прохождения поворотов в заносе.

Существует два типа заездов: одиночные и парные. Победитель обычно определяется в нескольких заездах. В одиночных заездах судьи начисляют гонщику определённое количество очков в зависимости от скорости, траектории, угла заноса и зрелищности заезда в целом. В парных заездах первый участник должен проехать оцениваемый участок в соответствии с заданием (чаще всего по максимально правильной траектории), задачей второго участника является как можно сильнее приблизиться к своему сопернику во время движения в заносе, делать синхронные перекладки. Для определения победителя совершается два заезда, во втором заезде правила те же, но противники меняются местами. Победителем является тот пилот, который проехал ближе и лучше, будучи «догоняющим». Также, если оба заезда были безупречными или количество ошибок обоих пилотов суммарно одинаковое, судьи могут назначить повторный заезд.

Во время состязания оцениваются следующие характеристики:

- угол заноса;
- скорость движения автомобиля;
- траектория движения автомобиля.

Когда судьями являются люди, то угол заноса определяется визуально, это может привести к не правильному нахождению победителя. А электронная система поможет найти победителя основываясь на точных показаниях датчиков.

Цель проекта

Целью моего проекта является создание системы электронного судейства для состязаний по дрифту.

Задачи проекта

1. Выбор микроконтроллера и компонентов;
2. Создание схемы взаимодействия микроконтроллера и компонентов;
3. Разработка аппаратной и софт части сервера;
4. Сборка и монтаж элементов;
5. Написание и отладка кода;
6. Анализ рынка готовых изделий.

Теоретическая часть

Акселерометр – прибор, измеряющий проекцию кажущегося ускорения (разности между истинным ускорением объекта и гравитационным ускорением). Согласно технической документации к MPU6050 значение выхода акселерометра составляет 4096, когда к датчику прикладывается 1g. Среднее ускорение свободного падения на Земле приблизительно составляет 9,8 м/с², что соответствует 1g, поэтому значение выхода акселерометра в состоянии покоя составляет 4096.

Угол наклона системы можно определить если определить ускорения по всем осям. Так как оси взаимно перпендикулярно, тогда можно посчитать результирующий вектор. Его длина равна:

$$g = \sqrt{g_x^2 + g_y^2 + g_z^2}$$

А значит углы между осями будут:

$$\cos(\alpha) = \frac{g_x}{g}$$

$$\cos(\beta) = \frac{g_y}{g}$$

$$\cos(\gamma) = \frac{g_z}{g}$$

При проверки вычисленные углы были близки к истинным.

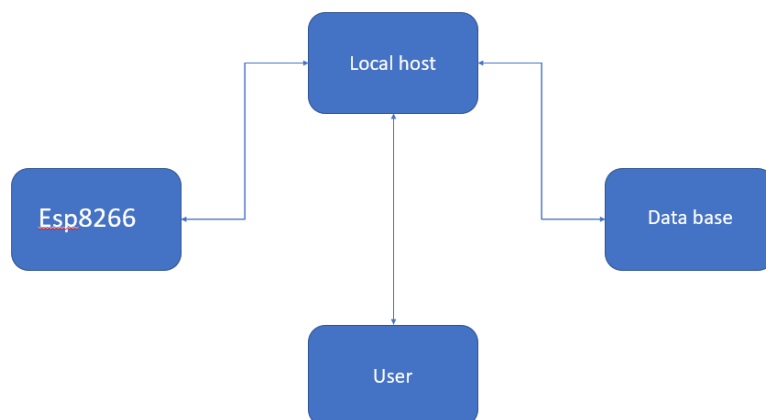
Методика выполнения работы

Архитектура

Данный проект реализован внутри одной локальной сети. Что позволяет беспрепятственно обращаться частям между собой при помощи ip адреса. Общение происходит с помощью socket. Socket - название программного интерфейса для обеспечения обмена данными между процессами. Процессы при таком обмене могут исполняться как на одной ЭВМ, так и на различных ЭВМ, связанных между собой только сетью. Сокет — абстрактный объект, представляющий конечную точку соединения. Сервер (localhost) запускают на машине в судебском пункте. А к ней подсоединяются клиенты (esp8266) и отправляют данные в режиме реального времени. На данной стадии отдача команд проходит через строку в Python. В будущем я планирую создать интерфейс для судьи (user). Сервер обрабатывает информацию и заполняет базу данных. (database). Данное решение имеет минусы. Из-за больших размеров трассы возможны потери сети, а следовательно, и потеря данных. Это проблема решается двумя способами:

1. Ставим усилитель сигнала на трассе.
2. Запускаем сервер на локальной машине и с помощью сервиса ngrok делаем его доступным из вне, а на каждый клиент ставим точку раздачи сети Wi-Fi, к которой подключаем клиент.

Ниже представлена схема архитектуры.



Выбор микроконтроллера и компонентов

Проанализировав имеющиеся микроконтроллеры мой выбор пал на Esp8266, рисунок 1, как на бюджетный и стабильный вариант. Так же данный контроллер имеет встроенный Wi-Fi модуль.

Ниже представлен список основных компонентов и их характеристики.

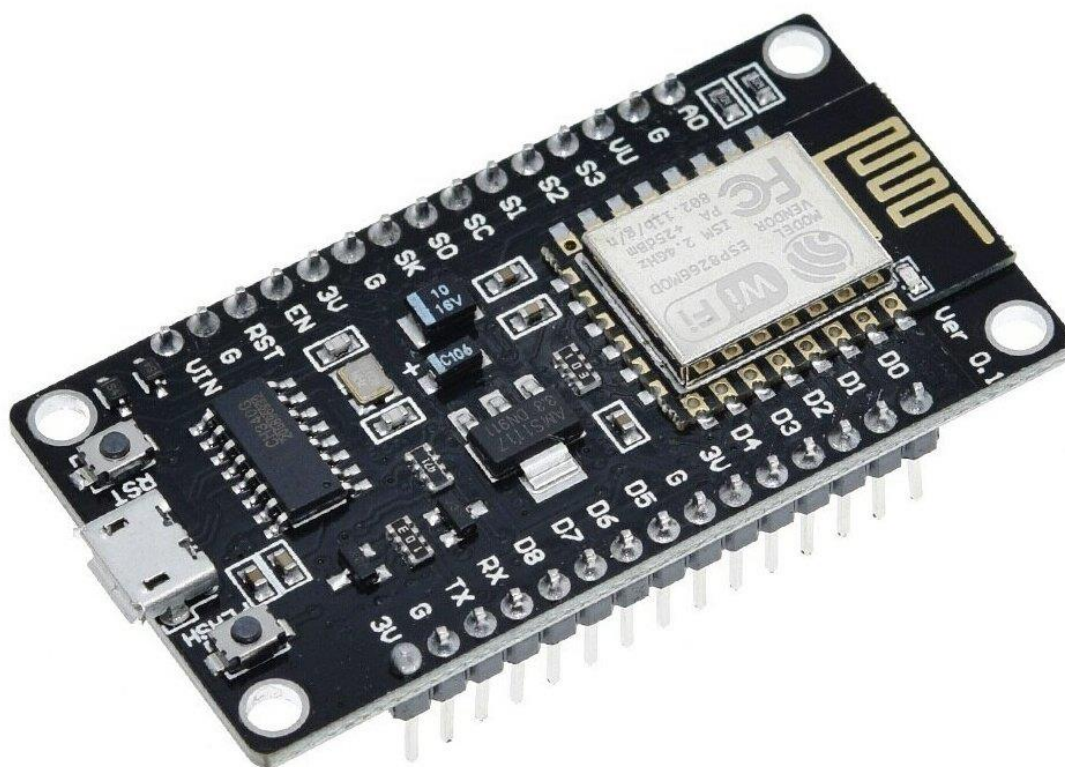
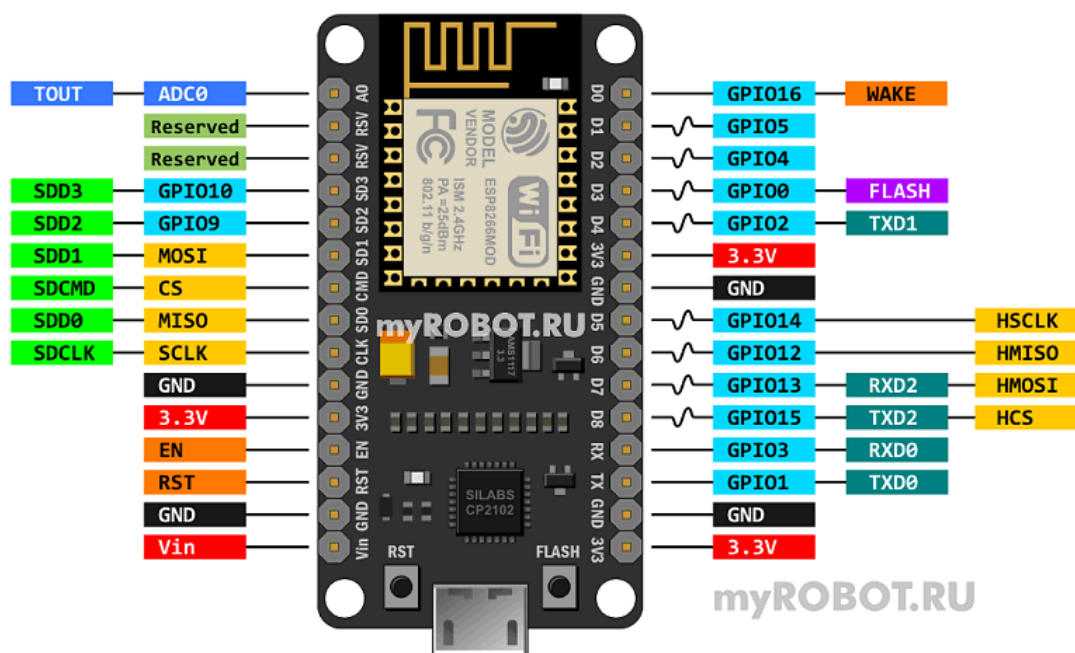


Рисунок 1. Esp8266

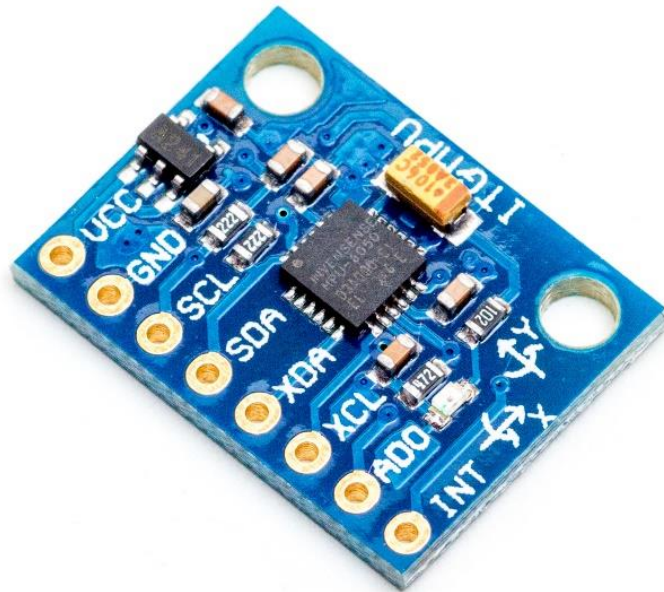
Характеристики:

| Specifications | ESP8266 | ESP32 |
|-------------------------|---------------------------------|--|
| MCU | Xtensa® Single-Core 32-bit L106 | Xtensa® Dual-Core 32-bit LX6 600 DMIPS |
| 802.11 b/g/n Wi-Fi | Yes, HT20 | Yes, HT40 |
| Bluetooth | None | Bluetooth 4.2 and below |
| Typical Frequency | 80 MHz | 160 MHz |
| SRAM | 160 kBytes | 512 kBytes |
| Flash | SPI Flash , up to 16 MBytes | SPI Flash , up to 16 MBytes |
| GPIO | 17 | 36 |
| Hardware / Software PWM | None / 8 Channels | 1 / 16 Channels |
| SPI / I2C / I2S / UART | 2/1/2/2 | 4/2/2/2 |
| ADC | 10-bit | 12-bit |
| CAN | None | 1 |
| Ethernet MAC Interface | None | 1 |
| Touch Sensor | None | Yes |
| Temperature Sensor | None | Yes |
| Working Temperature | - 40°C ~ 125°C | - 40°C ~ 125°C |

Приведём распиновку контактов Esp8266:



MPU 6050



Характеристики:

16-битный АЦП

Напряжение питания 3-5В

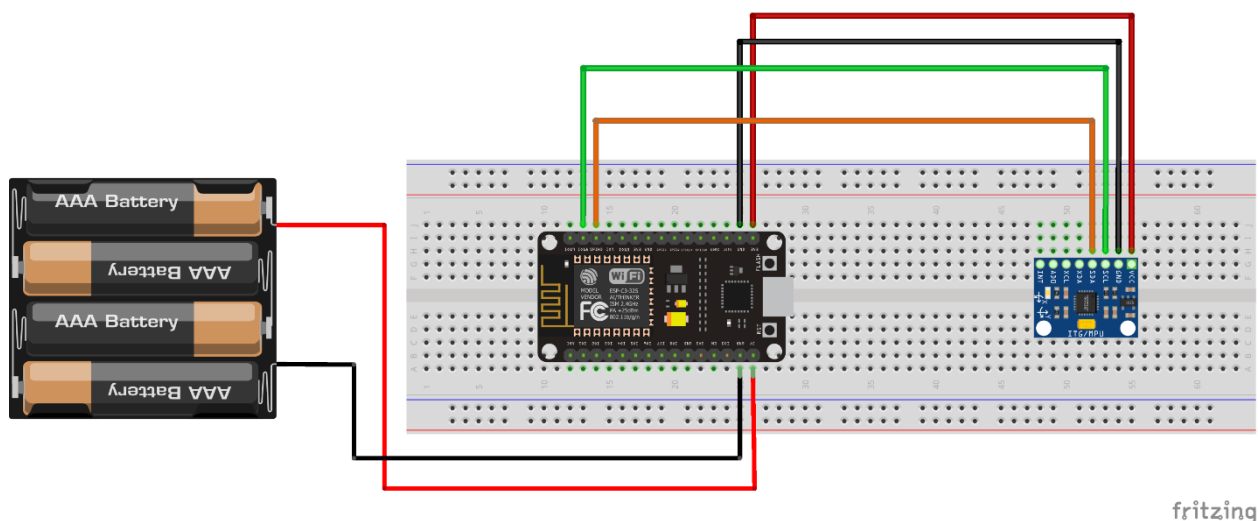
Поддержка протокола I2C

Диапазон ускорений 2, 4, 8, 16g

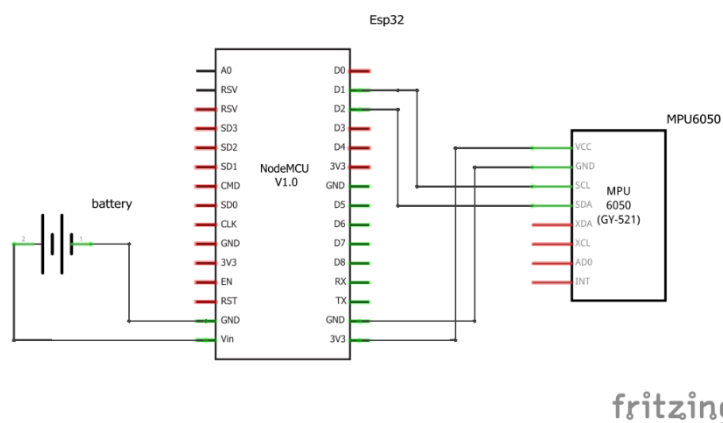
Создание принципиальных схем.

Для создания электрических схем будем использовать ПО Fritizing. Это бесплатное программное обеспечение для инженеров и конструкторов. Она помогает создать и документировать прототипы электрических цепей и схем, в том числе на базе

Arduino, ESP-8266, а также различного электронного оборудования для последующего производства печатных плат. В базе Fritzing есть все необходимые устройства, используемые в проекте. В этой программе я разработал схему устройства, которое помещается в машину. Схема приведена ниже.



Электрическая схема проекта



Принципиальная схема проекта

Сборка

Так как устройство будет устанавливаться в машину, то необходимо разработать систему, которая будет убирать вибрацию, так как гироскоп очень чувствителен к этому. В ходе анализа рынка я выбрал нейлоновые стойки, так как они не дорогие и поглощают почти всю вибрацию.



Нейлоновые стойки

В моем проекте вместо полноценного системы представлена модель только 1 устройства, так как вся система является модульной. На данный момент я разработал корпус. Для его моделирования была использована программа Solidworks 2020. Фото устройства и скриншоты модели находятся в приложении 4 и приложении 5.

Программная часть

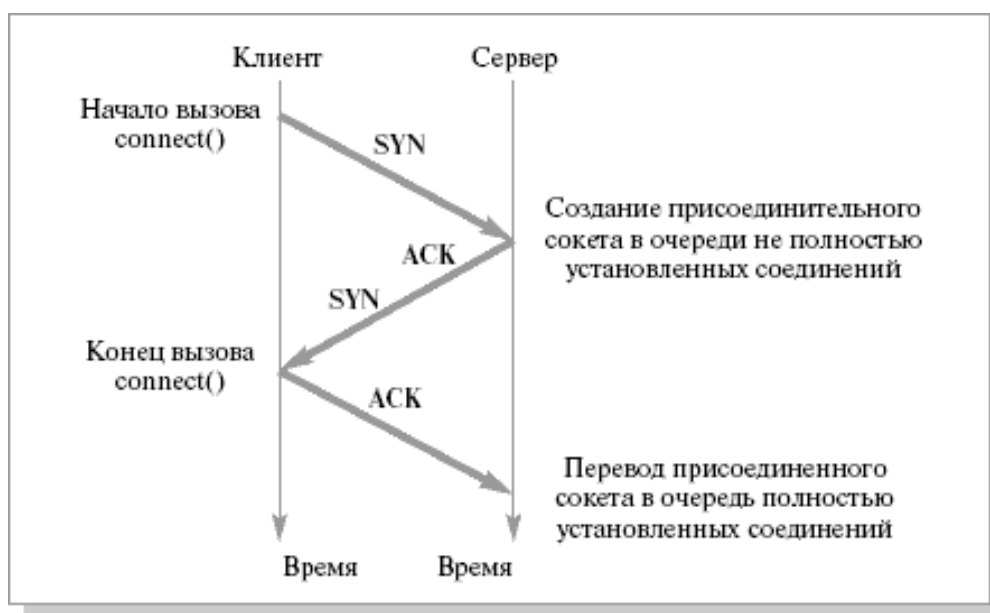
Код для esp-8266 (в качестве клиента) был написан в интегрированной среде разработки Arduino IDE 2.0.3. Для передачи данных с контроллера на сервер я использовал протокол TCP.

Перечислим некоторые достоинства этого протокола:

- Основное достоинство стека протоколов TCP/IP в том, что он обеспечивает надежную связь между сетевым оборудованием от различных производителей.
- Независимость от сетевой технологии — стек только определяет элемент передачи, дейтаграмму, и описывает способ ее движения по сети.
- Всеобщая связанность — стек позволяет любой паре компьютеров, которые его поддерживают, взаимодействовать друг с другом. Каждому компьютеру назначается логический адрес, а каждая передаваемая дейтаграмма содержит логические адреса отправителя и получателя. Промежуточные маршрутизаторы используют адрес получателя для принятия решения о маршрутизации.
- Подтверждения. Протоколы стека обеспечивают подтверждения правильности прохождения информации при обмене между отправителем и получателем.
- Стандартные прикладные протоколы. Протоколы стека TCP/IP включают в свой состав средства поддержки основных приложений, таких как электронная почта, передача файлов, удаленный доступ и т.д.

Данный протокол подразумевает конструкцию вопрос-ответ. Из-за этого будет возникать небольшая задержка. Данная задержка не является критичной, так как одновременное соединение с сервером будут иметь только 2 контроллера.

Ниже приведена схема данного протокола.



Для общения между клиентом и сервером была выбрана технология WebSocket. Это технология, которая позволяет клиенту установить двухстороннюю связь с сервером.

Написание клиентской части для esp8266

Для написания клиентской части я использовал библиотеки:

1. ESP8266WiFi.h – это встроенная библиотека. Будем её использовать для подключения контроллера к сети интернет и отправки сокета на сервер.
2. I2Cdev.h – это библиотека создана для общения по I2C.
3. MPU6050.h – с помощью этой библиотеке я работал с датчиком MPU6050.

Код программы представлен в Приложении 2.

Серверная часть

Сервер я написал на Python. В процессе работы я использовал некоторые библиотеки. Так как у нас будет несколько устройств, то необходимо сделать сервер многопоточным. После каждого подключения мы будем заносить данные в базу данных SQL. Для того чтобы сделать сервер многопоточным я использовал библиотеку threading.

Первым делом запустим сервер на своем устройстве. Для этого понадобится библиотека `socket`. Запустим сервер написав следующие строки кода:

```
localhost = '192.168.118.119'
port = 2023

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 2)

server.bind((localhost, port))
```

В 1 строке присвоим ip-адрес нашего сервера. Это ip-адрес, который присвоен нашему компьютеру в сети (сети, к которой подключен ноутбук). Чтобы его посмотреть откроем командную строку и введём `ipconfig`.

```
C:\Users\ksv>ipconfig

Настройка протокола IP для Windows

Неизвестный адаптер Подключение по локальной сети:

    Состояние среды. . . . . : Среда передачи недоступна.
    DNS-суффикс подключения . . . . . :

Адаптер беспроводной локальной сети Подключение по локальной сети* 1:

    Состояние среды. . . . . : Среда передачи недоступна.
    DNS-суффикс подключения . . . . . :

Адаптер беспроводной локальной сети Подключение по локальной сети* 2:

    Состояние среды. . . . . : Среда передачи недоступна.
    DNS-суффикс подключения . . . . . :

Адаптер беспроводной локальной сети Беспроводная сеть:

    DNS-суффикс подключения . . . . . :
    Локальный IPv6-адрес канала . . . : fe80::65ef:b942:e3d3:12aa%16
    IPv4-адрес. . . . . : 192.168.118.119
    Маска подсети . . . . . : 255.255.255.0
    Основной шлюз. . . . . : 192.168.118.253
```

Нам необходима строка IPv4-адрес.

Далее укажем порт, на котором будет запущен сервер. В моём случае это 2023, хотя можно взять любой, который больше 1024, чтобы не было конфликтов с внутренними приложениями.

В следующих строках кода я инициализирую сервер и запускаю его.

```
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 2)

server.bind((localhost, port))
```

Далее я считываю параметры заезда, разделённые /.

```
while True:
    data = [(elem) for elem in input("введите данные заезда: ").split('/')]
    data.append(input('ip: '))
    dat = []

    time_limit = int(input("Введите максимальную
длительность соревнований в секундах: "))

    if data[0] == 'par':
        dat = [elem for elem in input("введите данные
заезда: ").split('/')]
        dat.append(input('ip: '))

    time_start = time.time()
```

Если заезд парный, то создаю и этот массив, а также и максимальная длительность заезда.

На стороне сервера стартуем процесс реализующий tcp-socket в режиме listener.

```
while True:
    server.listen(1)
    clientsock, clientAddress = server.accept()

    newthread = ClientThread(clientAddress, clientsock)
    newthread.start()
    if time.time()-time_start > time_limit:
        break
```

Далее идет определения ip клиента.

```
class ClientThread(threading.Thread):
    def __init__(self, clientAddress, clientsocket):
        threading.Thread.__init__(self)
```

```
self.csocket = clientsocket
print("Новое подключение: ", clientAddress)
```

Далее я подключаюсь к базе данных, а если её нет, то создаю её.

```
def run(self):
    #print("Подключение с клиента : ", clientAddress)

    conn = sqlite3.connect('drift.db') #подключаемся к базе
данных
    cur = conn.cursor() #устанавливаем курсор

    cur.execute("""CREATE TABLE IF NOT EXISTS дрифт(
        'id' INT PRIMARY KEY,
        'Название соревнования' TEXT,
        'дата' TEXT,
        'организатор' TEXT,
        'место' TEXT,
        'номер машины' TEXT,
        'тип заезда' TEXT,
        'время' TEXT,
        'ip устройства' TEXT,
        'время от начала соревнования' TEXT,
        'угол по оси ox' INT,
        'угол по оси oy' INT,
        'угол по оси oz' INT
    );
    """)
```

далее я считываю сообщение и обрабатываю его. Генерирую массив для передачи его в базу данных.

```
msg = ''while True:
    data2 = self.csocket.recv(4096) # получаем сообщение от
клиента

    msg = data2.decode() # декодируем его
```



```

#print(msg.split('/'))

if msg != ""'\r\n"":      # если оно не пустое
    data2 = msg.split("/") # разбиваем сообщение
    db = [0]*13
    if msg == '':
        #print("Отключение")
        break

    else:

        #если ip совпадает с необходимым, то заносим его
        в базу данных
        if clientAddress[0] == data[-1]:
            db[0], db[1], db[2], db[3], db[4], db[5],
db[6], db[7], db[8], db[9] = int(time.time()*100), data[0],
data[1], data[2], data[3], data[4], data[5], data[6],
data[7], time.time() - time_start
            db[-3] = (data2[0])
            db[-2] = (data2[1])
            db[-1] = (data2[2])

            elif dat != []:
                db[0], db[1], db[2], db[3], db[4], db[5],
db[6], db[7], db[8], db[9] = int(time.time()*100), dat[0],
dat[1], dat[2], dat[3], dat[4], dat[5], dat[6], dat[7],
time.time() - time_start
                db[-3] = data2[-1]
                db[-2] = data2[-2]
                db[-1] = data2[-3]

            #print(len(db))
            if len(set(db)) == 13:
                cur.execute("INSERT INTO дрефт VALUES(?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?);", db)
                conn.commit()

            cur.close()

        else:
            cur.close()
            return

```

Первое сравнение в цикле происходит, чтобы отбросить мусорные сокет, которые посылает esp8266. Сокет, который посылает esp8266, выглядит следующим образом:

```
b' A1/A2/A3'
```

Где вместо **A1, A2, A3** будут показания углов по осям.

После этого я получаю все данные и генерирую массив, который передаю в базу данных. Также был написан класс, который строит график.

```
class Schedule():
    def __init__(self, data, time, db_ag, data2, time2,
db_ag2, angle):
        self.data = data
        self.db_ag = db_ag
        self.angle = angle
        self.time = time
        self.data2 = data2
        self.db_ag2 = db_ag2
        self.time2 = time2

    def reade(self):
        con = sqlite3.connect('drift.db') # подключаемся к
базе данных

        with con:
            cur = con.cursor()
            cur.execute("SELECT * FROM дрифт")
            rows = cur.fetchall()

            for row in rows:

                if self.data[1] == row[4] and self.data[2]
== row[2] and self.data[0] == row[3] and self.data[-2] ==
row[5] and row[6] == self.data[-1]:

                    if self.angle.lower() == 'x':
                        self.db_ag.append(row[-3])

                    elif self.angle.lower() == 'y':
```

```

        self.db_ag.append(row[-2])

        elif self.angle.lower() == 'z':
            self.db_ag.append(row[-1])

        self.time.append(row[0]/100)

        if self.data2 != []:

            if self.data2[1] == row[4] and
self.data2[2] == row[2] and self.data2[0] == row[3] and
self.data2[-2] == row[5] and row[6] == self.data2[-1]:

                if self.angle.lower() == 'x':
                    self.db_ag2.append(row[-3])

                elif self.angle.lower() == 'y':
                    self.db_ag2.append(row[-2])

                elif self.angle.lower() == 'z':
                    self.db_ag2.append(row[-1])

                self.time2.append(row[0]/100)

    plot.pl()

    def pl(self):

        fig, ax = plt.subplots()

        for l in range(len(self.time)):
            plt.scatter(self.time[l]-min(self.time),
self.db_ag[l], color='red', s=4)

        if self.time2 != 0:

            for l in range(len(self.time2)):
                plt.scatter(self.time2[l] - min(self.time2),
self.db_ag2[l], color='blue', s=4)
                plt.text(x=0, y=100, s = "Синий - второй гонщик,
Красный - первый")

        plt.xlabel('Время')
        plt.ylabel(f""""угол по оси {ag1}""")
        plt.title(f""""График угла по оси {ag1} от
времени""")

```

```
plt.grid(True)
plt.show()
pass
```

Функция для скачивания файла с данными заезда:

```
def archiving(dat, fil):

    try:
        fil += '.xlsx'

        workbook =xlsxwriter.Workbook(fil)
        worksheet = workbook.add_worksheet()

        j = -1

        con = sqlite3.connect('drift.db') # подключаемся к
базе данных

        with con:
            cur = con.cursor()
            cur.execute("SELECT * FROM дрифт")
            rows = cur.fetchall()

            for row in rows:
                if dat[1] == row[4] and dat[2] == row[2] and
dat[0] == row[3] and dat[-2] == row[5] and dat[-1] ==
row[6]:
                    j+=1
                    for i in range(len(row)):
                        worksheet.write(j, i, row[i])
                        #print()

            workbook.close()
            print(f"""Запись завершена в файл {fil}""")

    except:
        print("Такой файл уже существует")
        return
```

Ниже представлен ход выполнения кода

```
13 server = socket.socket(socket.AF_INET, socket.SOCK_STREAM) #создаём сокет
14 server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 2) #устанавливаем значение опций сокета
15
16 server.bind((localhost, port)) #привязываем сокет к адресу
17
18 def archiving(dat, fil):
19
20     try:
21         fil += '.xlsx'
22
23         workbook = xlswriter.Workbook(fil)
24         worksheet = workbook.add_worksheet()
25
26         j = -1
```

Run: server

C:\Users\kvs\PycharmProjects\Project_mos_robototecnik\venv\Scripts\python.exe C:\Users\kvs\PycharmProjects\Project_mos_robototecnik\server.py

1) Добавить соревнование
2) Построить график
3) Скачать файл заезда
Действие № 1
Введите для первой машины: организатора/место/дата/номер машины/тип заезда
организатор: жюккы/куркина/14.02.2023/2/фини
Введите название файла: 8
Запись завершена в файл 8.xlsx

После выполнения в базе данных появится следующий набор данных.

| id | номер ма... | тип заезда | ip устрой... | дата | время | угол по ос... | угол по ос... | угол по ос... |
|------------|-------------|------------|---------------|------------|--------------|---------------|---------------|---------------|
| 1676352929 | 1 | final | 127.168.37... | 14.02.2023 | 167635292... | -3 | 3 | -0.794 |
| 1676352930 | 1 | final | 127.168.37... | 14.02.2023 | 167635293... | 28 | -8 | -0.795 |
| 1676352931 | 1 | final | 127.168.37... | 14.02.2023 | 167635293... | -16 | 7 | -0.796 |
| 1676352932 | 1 | final | 127.168.37... | 14.02.2023 | 167635293... | -6 | -6 | -0.797 |
| 1676352933 | 1 | final | 127.168.37... | 14.02.2023 | 167635293... | -14 | 0 | -0.798 |
| 1676352934 | 1 | final | 127.168.37... | 14.02.2023 | 167635293... | 25 | 7 | -0.799 |
| 1676352935 | 1 | final | 127.168.37... | 14.02.2023 | 167635293... | 25 | 2 | -0.8 |
| 1676352936 | 1 | final | 127.168.37... | 14.02.2023 | 167635293... | 18 | -8 | -0.801 |
| 1676352937 | 1 | final | 127.168.37... | 14.02.2023 | 167635293... | -7 | 6 | -0.802 |
| 1676352938 | 1 | final | 127.168.37... | 14.02.2023 | 167635293... | 7 | 9 | -0.803 |
| 1676352939 | 1 | final | 127.168.37... | 14.02.2023 | 167635293... | 5 | -6 | -0.804 |
| 1676352940 | 1 | final | 127.168.37... | 14.02.2023 | 167635294... | 23 | -9 | -0.805 |
| 1676352941 | 1 | final | 127.168.37... | 14.02.2023 | 167635294... | 27 | 6 | -0.806 |

Для легкого определения ip устройства я также написал код, который определяет все ip в сети. Код представлен ниже.

```
import threading, socket, os, platform

def getMyIp():
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) #Создаем сокет (UDP)
    s.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1) # Настраиваем сокет на
    BROADCAST вещание.
    s.connect(('<broadcast>', 0))
    return s.getsockname()[0]

def scan_ip(ip):
    addr = net + str(ip)
    comm = ping_com + addr
    response = os.popen(comm)
    data = response.readlines()
    for line in data:
        if 'TTL' in line:
            print(addr, "Присутствует в сети")
            break

net = getMyIp()
print('You IP :', net)
net_split = net.split('.')
a = '.'
net = net_split[0] + a + net_split[1] + a + net_split[2] + a
start_point = int(input("Введите начальный номер: "))
end_point = int(input("Введите конечный номер: "))

oc = platform.system()
if (oc == "Windows"):
    ping_com = "ping -n 1 "
else:
    ping_com = "ping -c 1 "

print("Прогресс: ")

for ip in range(start_point, end_point):
    if ip == int(net_split[3]):
```

```
        continue
    potoc = threading.Thread(target=scan_ip, args=[ip])
    potoc.start()

potoc.join()

print("Все ip найдены")
```

Полный код, а также архитектура представлена в приложении.

Расчёт стоимости

За базу расчёта стоимости принимаем стоимость оборудования на торговой площадке Aliexpress

1. Esp8266:

The screenshot shows the AliExpress product page for a 'Беспроводной модуль V3 NodeMcu 4 м байты Lua WIFI Интернет вещей плата на основе ESP8266 ESP-12E для Arduino совместимый CH340/CP2102'. The page features a large image of the module, a search bar, and navigation links. The price is listed as 143,55 руб. with a 22% discount from 183,90 руб. The page also shows a 'В корзину' (Add to cart) button and a 'Купить сейчас' (Buy now) button.

AliExpress

Искать на Aliexpress

Беспроводной модуль V3 NodeMcu 4 м байты Lua WIFI Интернет вещей плата на основе ESP8266 ESP-12E для Arduino совместимый CH340/CP2102

★★★★★ 4.9 (1126 отзывов) 5982 купил WAVGAT Official Store (Рейтинг 96.06%)

Цвет CH340 serial port 8 вариантов

183,90-руб. -22%
143,55 руб.

есть бесплатная доставка, а еще есть [купоны от продавца](#)

Доставка Почтой 6 апр – Бесплатно

В корзину Купить сейчас

2. MPU6050

The screenshot shows the AliExpress product page for a 'GY-521 MPU-6050 MPU6050 Module 3 Axis analog gyro sensors+ 3 Axis Accelerometer Module'. The page features a large image of the module, a search bar, and navigation links. The price is listed as 84,58 руб. The page also shows a 'В корзину' (Add to cart) button and a 'Купить сейчас' (Buy now) button.

AliExpress

Искать на Aliexpress

Формула выгоды: скидки до 70% с 13 по 17 февраля >

Электронные компоненты и принадлежности > Электронные компоненты > Интегральные схемы

GY-521 MPU-6050 MPU6050 Module 3 Axis analog gyro sensors+ 3 Axis Accelerometer Module

★★★★★ 5 (63 отзыва) 1582 купил SAMIORE Store (Рейтинг 96.95%)

FEIYANG

84,58 руб.

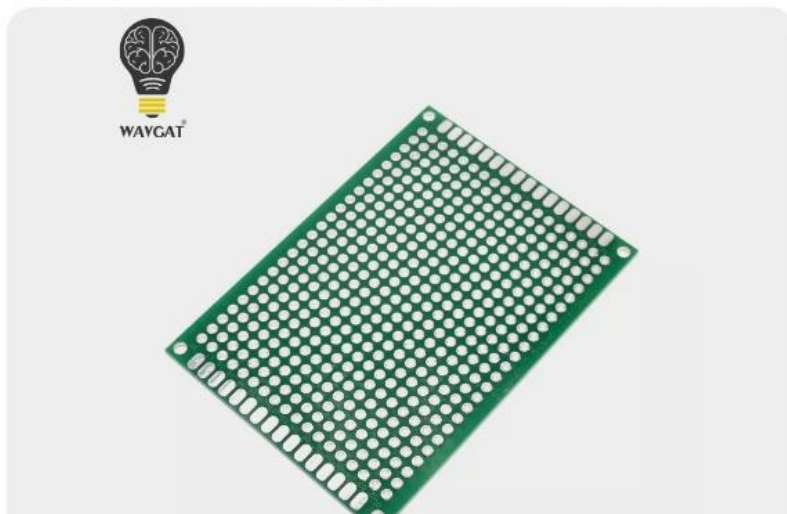
Доставка Почтой 6 апр – 24,05 руб.

В корзину Купить сейчас

3. Макетная плата:

WAVGAT 5*7 PCB 5x7 PCB 5cm 7cm Double Side Prototype PCB diy Universal Printed Circuit Board

★★★★★ 4.8 (263 отзыва) 3765 купили WAVGAT Official Store (Рейтинг 96.06%)



35,91 руб. -9%

32,79 руб.

есть бесплатная доставка, а еще есть купоны от продавца

Доставка Почтой 6 апр – Бесплатно

В корзину

Купить сейчас

4. Нейлоновые стойки:

aliexpress.ru/item/33020434460.html?sku_id=67186942395&spm=a2g2w.productlist.search_results.4.21e84aa6325UIL

AliExpress

Искать на Aliexpress



Войти Заказы Корзина

Формула выгоды: скидки до 70% с 13 по 17 февраля >

Строительство и ремонт > Крепеж и фурнитура, клеи и герметики > Застежки и крючки > Винты

Нейлоновая стойка M2 M2.5 M3 M4 m5 * L + 6 белый, черный, pcb, 20-50 ШТ.

★★★★★ 4.9 (327 отзывов) 841 купил shenzhen zhongfa metal Store (Рейтинг 96.27%)



Цвет black

Размер m3 50pcs 5 вариантов

m2.5 50pcs m4 20pcs m2 50pcs

Длина 10 мм 17 вариантов

7 мм 9mm 11mm 5 мм 6 мм

93,11 руб. -24%

70,61 руб.

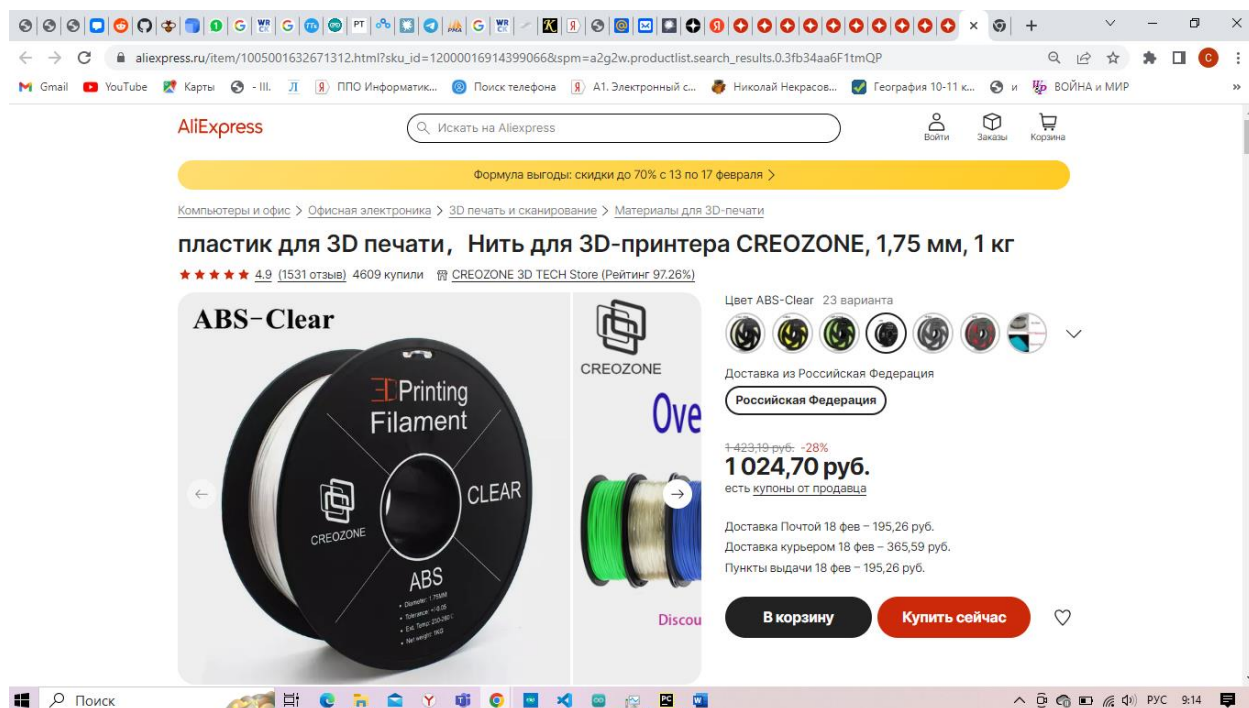
есть бесплатная доставка, а еще есть купоны от продавца

Доставка Почтой 6 апр – Бесплатно

Пункты выдачи 28 мар – 72,94 руб.

В корзину Купить сейчас

5. Также нам понадобится пластик для печати модели.



Но цена представлена за 1 кг, нам же хватит и 200 г.

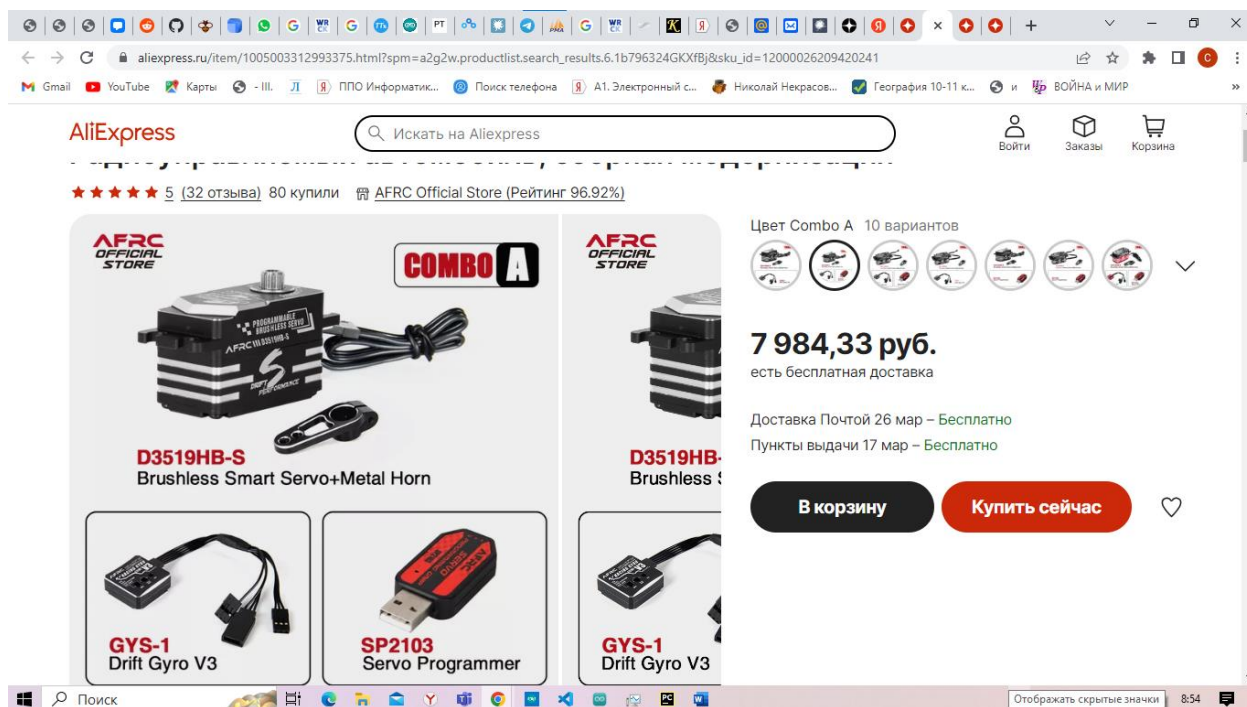
Сведём данные в таблицу, чтобы найти общую стоимость.

| Продукт | Цена (рубли) |
|-------------------|--------------|
| Плата Esp8266 | 143.55 |
| Датчик MPU6050 | 84.58 |
| Макетная плата | 32.79 |
| Нейлоновые стойки | 70.61 |
| Пластик | 204.94 |
| Прочие расходники | <100 |
| Итого: | 636.47 |

Значит стоимость проекта не превышает 1 тысячу рублей.

Анализ рынка

В ходе работы я провёл анализ рынка. Данные устройства почти не представлены на рынке. Но есть несколько датчиков, которые продаются на aliexpress. Но они продаются по заоблачным ценам. Скрин с ценой представлен ниже.



Данная цена очень высока. Поэтому мой проект является хорошей альтернативой, но с ценой в 8 раз меньше.

Вывод

В ходе работы мне удалось создать систему, с помощью которой можно определять угол заноса автомобиля. Данный продукт будет полезным для судейства на соревнованиях по дрифту. Он превосходит аналоги, и что несомненно важно, в 8 раз дешевле. Далее я планирую разработать клиентскую часть для пользователя, чтобы можно было по отдельности сравнивать соревнования и строить графики по этим точкам.

Список литературы:

https://intuit.ru/studies/professional_retraining/966/courses/216/lecture/5559?page=3#:~:text=%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D0%BD%D0%BE%D0%B5%20%D0%B4%D0%BE%D1%81%D1%82%D0%BE%D0%B8%D0%BD%D1%81%D1%82%D0%B2%D0%BE%20%D1%81%D1%82%D0%B5%D0%BA%D0%B0%20%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB%D0%BE%D0%B2%20TCP,%D1%81%D0%BF%D0%BE%D1%81%D0%BE%D0%B1%20%D0%B5%D0%B5%20%D0%B4%D0%B2%D0%B8%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F%20%D0%BF%D0%BE%20%D1%81%D0%B5%D1%82%D0%B8.

<https://ru.hexlet.io/blog/posts/что-такое-websocket-и-как-они-вообще-работают>

<https://aliexpress.ru/>

<https://qna.habr.com/>

<https://www.cyberforum.ru/>

<https://techtutorialsx.com/2017/11/01/esp32-arduino-websocket-client/>

<https://github.com/>

https://www.kgasu.ru/upload/iblock/cd2/up_zashita_ot_vibracii.pdf

Приложение 1 - Код для серверной части

```
import socket, threading
import sqlite3
import time
import matplotlib.pyplot as plt
import xlswriter

#инициализируем порт и адрес сервера
localhost = '192.168.11.119'
port = 2023

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
#создаём сокет
server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 2)
#устанавливаем значение опций сокета

server.bind((localhost, port)) #привязываем сокет к адресу

def archiving(dat, fil):

    try:
        fil += '.xlsx'

        workbook = xlswriter.Workbook(fil)
        worksheet = workbook.add_worksheet()

        j = -1

        con = sqlite3.connect('drift.db') # подключаемся к
базе данных

        with con:
            cur = con.cursor()
            cur.execute("SELECT * FROM дрефт")
            rows = cur.fetchall()

            for row in rows:
                if dat[1] == row[4] and dat[2] == row[2] and
dat[0] == row[3] and dat[-2] == row[5] and dat[-1] ==
row[6]:
                    j+=1
                    for i in range(len(row)):
                        worksheet.write(j, i, row[i])
```

```

        #print()

        workbook.close()
        print(f"""Запись завершена в файл {fil}""")

    except:
        print("Такой файл уже существует")
        return

class Schedule():
    def __init__(self, data, time, db_ag, data2, time2,
db_ag2, angle):
        self.data = data
        self.db_ag = db_ag
        self.angle = angle
        self.time = time
        self.data2 = data2
        self.db_ag2 = db_ag2
        self.time2 = time2

    def reade(self):
        con = sqlite3.connect('drift.db') # подключаемся к
базе данных

        with con:
            cur = con.cursor()
            cur.execute("SELECT * FROM дрифт")
            rows = cur.fetchall()

            for row in rows:

                if self.data[1] == row[4] and self.data[2]
== row[2] and self.data[0] == row[3] and self.data[-2] ==
row[5] and row[6] == self.data[-1]:

                    if self.angle.lower() == 'x':
                        self.db_ag.append(row[-3])

                    elif self.angle.lower() == 'y':
                        self.db_ag.append(row[-2])

                    elif self.angle.lower() == 'z':
                        self.db_ag.append(row[-1])

```

```

        self.time.append(row[0]/100)

        if self.data2 != []:

            if self.data2[1] == row[4] and
self.data2[2] == row[2] and self.data2[0] == row[3] and
self.data2[-2] == row[5] and row[6] == self.data2[-1]:

                if self.angle.lower() == 'x':
                    self.db_ag2.append(row[-3])

                elif self.angle.lower() == 'y':
                    self.db_ag2.append(row[-2])

                elif self.angle.lower() == 'z':
                    self.db_ag2.append(row[-1])

            self.time2.append(row[0]/100)

    plot.pl()

    def pl(self):

        fig, ax = plt.subplots()

        for l in range(len(self.time)):
            plt.scatter(self.time[l]-min(self.time),
self.db_ag[l], color='red', s=4)

        if self.time2 != 0:

            for l in range(len(self.time2)):
                plt.scatter(self.time2[l] - min(self.time2),
self.db_ag2[l], color='blue', s=4)
                plt.text(x=0, y=100, s = "Синий - второй гонщик,
Красный - первый")

        plt.xlabel('Время')
        plt.ylabel(f""""угол по оси {agl}""")
        plt.title(f""""График угла по оси {agl} от
времени""")
        plt.grid(True)
        plt.show()
        pass

```



```

class ClientThread(threading.Thread):
    def __init__(self, clientAddress, clientsocket):
        threading.Thread.__init__(self)
        self.csocket = clientsocket
        #print("Новое подключение: ", clientAddress)

    def run(self):
        #print("Подключение с клиента : ", clientAddress)

        conn = sqlite3.connect('drift.db') #подключаемся к
базе данных
        cur = conn.cursor() #устанавливаем курсор

        cur.execute("""CREATE TABLE IF NOT EXISTS дрифт(
            'id' INT PRIMARY KEY,
            'Название соревнования' TEXT,
            'дата' TEXT,
            'организатор' TEXT,
            'место' TEXT,
            'номер машины' TEXT,
            'тип заезда' TEXT,
            'время' TEXT,
            'ip устройства' TEXT,
            'время от начала соревнования' TEXT,
            'угол по оси ox' INT,
            'угол по оси oy' INT,
            'угол по оси oz' INT
        );
        """)

        msg = ''

        while True:
            data2 = self.csocket.recv(4096) # получаем
сообщение от клиента

            msg = data2.decode() # декодируем его
            #print(msg.split('/'))

            if msg != ""'\r\n"": # если оно не пустое
                data2 = msg.split("/") # разбиваем сообщение
                db = [0]*13
                if msg == '':
                    #print("Отключение")

```

```

        break

    else:

        #если ip совпадает с необходимым, то
        заносим его в базу данных
        if clientAddress[0] == data[-1]:
            db[0], db[1], db[2], db[3], db[4],
            db[5], db[6], db[7], db[8], db[9] = int(time.time()*100),
            data[0], data[1], data[2], data[3], data[4], data[5],
            data[6], data[7], time.time() - time_start
            db[-3] = (data2[0])
            db[-2] = (data2[1])
            db[-1] = (data2[2])

            elif dat != []:
                db[0], db[1], db[2], db[3], db[4],
                db[5], db[6], db[7], db[8], db[9] = int(time.time()*100),
                dat[0], dat[1], dat[2], dat[3], dat[4], dat[5], dat[6],
                dat[7], time.time() - time_start
                db[-3] = data2[-1]
                db[-2] = data2[-2]
                db[-1] = data2[-3]

            #print(len(db))
            if len(set(db)) == 13:
                cur.execute("INSERT INTO дрейф
VALUES(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);", db)
                conn.commit()

        cur.close()

    else:
        cur.close()
        return

#запускаем бесконечный цикл, чтобы не перезапускать
программу
while True:
    time.sleep(1)
    print("\n1) Добавить соревнование \n2) Построить
график\n3) Скачать файл заезда")

    action = input("Действие № ")

```

```

# если добавление заезда, то выполняется код, который
написан ниже
if action == "1":
    print("Данные вводить через '/',
Название/Дата/организатор/место/Номер машины/Тип
заезда/время")

    data = [(elem) for elem in input("введите данные
заезда: ").split('/')]
    data.append(input('ip: '))
    dat = []

    time_limit = int(input("Введите максимальную
длительность соревнований в секундах: "))

    ok = input("Парный ли заезд? ")

    # если заезд парный, то добавляется второе
устройство, создав индентичный массив
    if ok.lower() == "да":
        dat = [elem for elem in input("введите данные
заезда: ").split('/')]
        dat.append(input('ip: '))

    time_start = time.time()      # записываем время
начала процесса

    # проверяем правильно ли заполнен массив (ы)
    if (len(dat) == 0 and len(data) == 8) or (len(dat)
== 8 and len(data) == 8):

        while time.time()-time_start < time_limit:

            server.listen(1)      # переводим сервер в
режим постоянной ожидания подключения
            clientsock, clientAddress = server.accept()
# принимаем соединение

            if clientAddress[0] == data[-1] or (dat!= []
and clientAddress[0] == dat[-1]):

                newthread = ClientThread(clientAddress,
clientsock) # создаём объект данного класса (начинаем новый
поток)

```

```

        newthread.start() # вызываем его

    else:
        print("Пожалуйста, повторите ввод")

    elif action == "2":

        data = input("Введите для первой машины:
организатора/место/дата/номер машины/тип
заезда\n").split('/')
        agl = input("Введите ось, по которой построить
график: ")
        ok = input("Парный ли заезд?: ")
        data2 = []

        if ok.lower == "да":
            data2 = input("Введите для второй машины:
организатора/место/дата/номер машины/тип
заезда\n").split('/')

        plot = Schedule(data, [], [], data2, [], [], agl)
        plot.reade()

    elif action == "3":
        data = [elem for elem in input("Введите для первой
машины: организатора/место/дата/номер машины/тип
заезда\n").split('/')]

        if len(data) == 5:
            file = input("Введите название файла: ")
            archiving(data, file)

        else:
            print("Введена не корректная информация")

    else:
        print("Данные не верны. Повторите попытку ввода.")

# b' '

```

Приложение 2 - Код для клиентской части esp8266

```
#include "I2Cdev.h"
#include "MPU6050.h"
#include <ESP8266WiFi.h>

#define TO_DEG 57.2957f //константа перевода из радиан в градусы
#define TIME_OUT 10

MPU6050 accgyro;

long int t1;
const int MPU_addr = 0x68;
int agl[4];
bool flag = true;
const unsigned long *number;

// Определяем параметры сети
const char *ssid = "AndroidAPBF9F";
const char *password = "ser123456";

// Определяем адрес сервера
const char *ADDR = "192.168.11.119";

// Определяем url подключения
const char *URL = "/";
// Определяем порт
const uint16_t PORT = 2023;

void setup() {

    Serial.begin(115200);
    accgyro.initialize();
    Wire.begin();
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
    }
}
```

```

    }
}

void loop() {

    angl_xyz(&agl[0], &agl[1], &agl[2], &agl[3]);
    delay(10);
}

void angl_xyz(int *ax, int *ay, int *az, int *t) {
    *t = millis();
    if (t1 < *t) {
        int16_t ax, ay, az, gx, gy, gz;

        accgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
        // Serial.println(gx);
        // Serial.println(gy);
        // Serial.println(gz);
        float g = sqrt(gx * gx + gy * gy);
        float g2 = sqrt(gx * gx + gy * gy + gz * gz);

        float cos_x = gx / g;
        float cos_y = gy / g;
        float cos_z = gz / g2;

        //получить значение в градусах (возможно убрать - 90)
        int aglx = abs(TO_DEG * acos(cos_x) - 90);
        int agly = abs(TO_DEG * acos(cos_y) - 90);
        int aglz = abs(TO_DEG * acos(cos_z) - 90);

        seeding(aglx, agly, aglz);
    }
}

void seeding(float aglx, float agly, float aglz){
    WiFiClient client;

    if (!client.connect(ADDR, PORT)) {
        delay(50);
    }
    String msg;

```

```
msg = String(aglx) + '/' + String(agly) + '/' + String(aglz);  
Serial.println(msg);  
client.println(msg);  
client.stop();
```

```
}
```

Приложение 3 - Код для определения всех ip в сети

```
import threading, socket, os, platform

def getMyIp():
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) #Создаем сокет (UDP)
    s.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1) # Настраиваем сокет на
BROADCAST вещание.
    s.connect(('<broadcast>', 0))
    return s.getsockname()[0]

def scan_Ip(ip):
    addr = net + str(ip)
    comm = ping_com + addr
    response = os.popen(comm)
    data = response.readlines()
    for line in data:
        if 'TTL' in line:
            print(addr, "Присутствует в сети")
            break

net = getMyIp()
print('You IP :', net)
net_split = net.split('.')
a = '.'
net = net_split[0] + a + net_split[1] + a + net_split[2] + a
start_point = int(input("Введите начальный номер: "))
end_point = int(input("Введите конечный номер: "))

oc = platform.system()
if (oc == "Windows"):
    ping_com = "ping -n 1 "
else:
    ping_com = "ping -c 1 "

print("Прогресс: ")

for ip in range(start_point, end_point):
    if ip == int(net_split[3]):
        continue
    potoc = threading.Thread(target=scan_Ip, args=[ip])
```

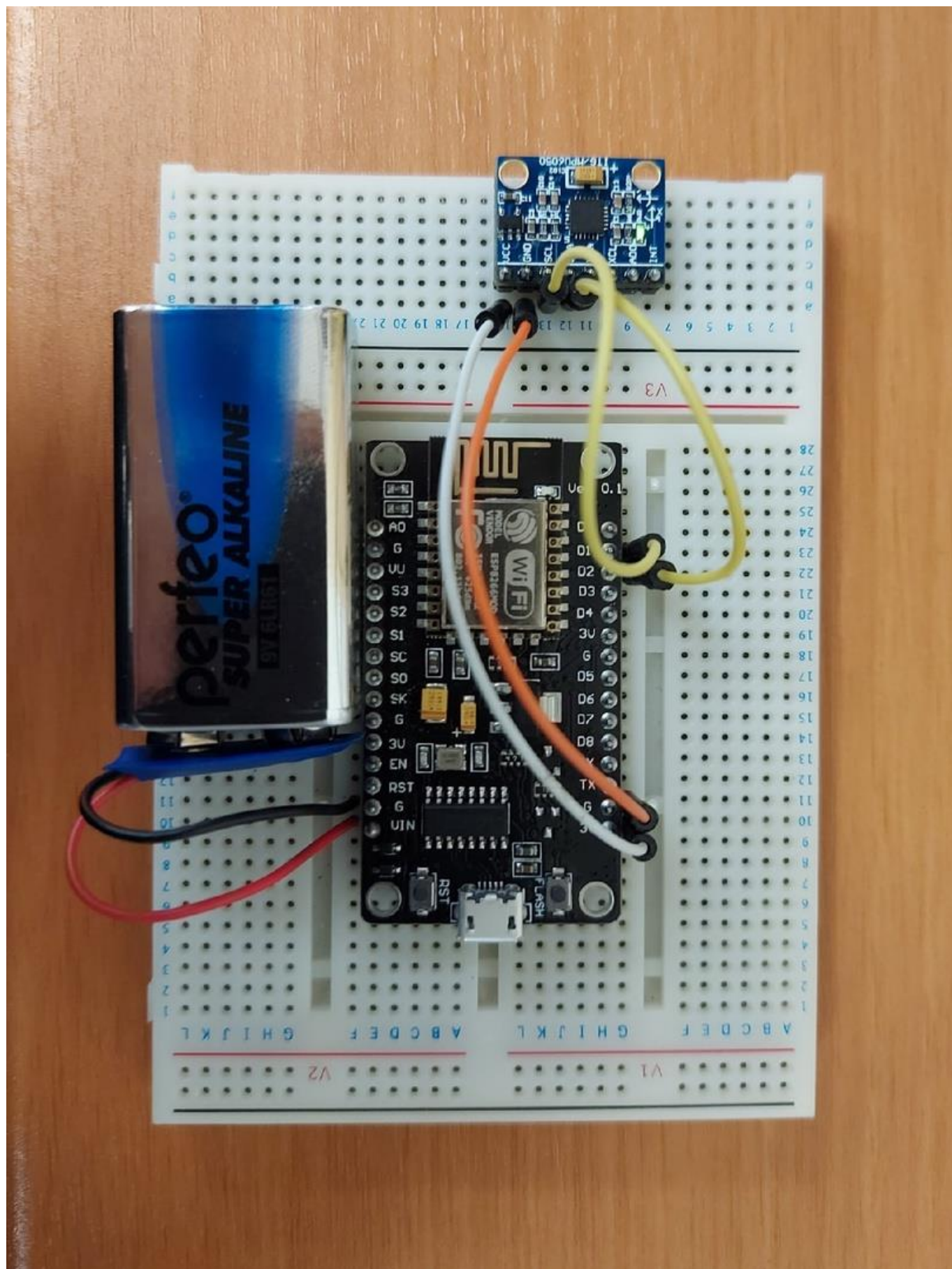


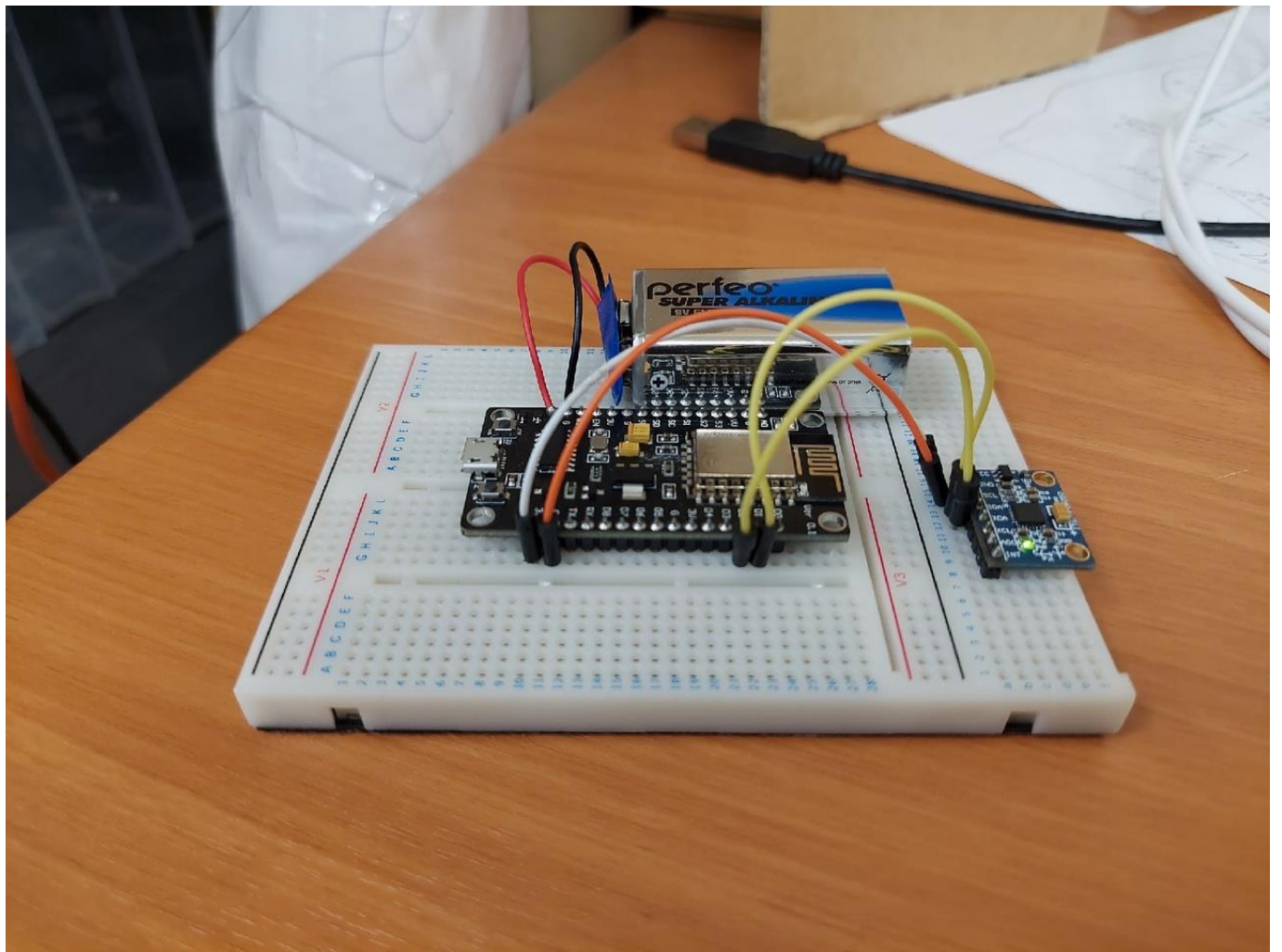
```
potoc.start()

potoc.join()

print("Все ip найдены")
```

Приложение 4 – Фото собранного устройства





Приложение 5 – снимок экрана из программы

