|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |

Институт Информационных технологий

IT Академия Samsung, трек «Интернет вещей»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к проекту по дисциплине «Информационные технологии интернета вещей»

**Тема проекта «Персональный помощник теннисиста»**

**Студент группы** ИНБО-06-19, Сахневич Кирилл Русланович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись студента)

**Руководитель проекта** Миронов Антон Николаевич, Дергунов Александр Александрович, Левандровский Александр Максимович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись руководителя)

Работа представлена к защите «02» июля 2021 г.

Допущен к защите «28» июня 2021 г.

Москва 2021

**СОДЕРЖАНИЕ**

[1 ЦЕЛИ И ЗАДАЧИ ПРОЕКТА 3](#_Toc76072011)

[2. АРХИТЕКТУРА ПРОЕКТА 5](#_Toc76072012)

[2.1 Общая схема архитектуры проекта 5](#_Toc76072013)

[2.2 Описание интерфейсов между компонентами решения и форматов данных 5](#_Toc76072014)

[2.3 Описание и обоснование аппаратных решений 5](#_Toc76072015)

[2.4 Описание и обоснование используемых технологий связи 7](#_Toc76072016)

[2.5 Описание и обоснование используемых программных компонентов 7](#_Toc76072017)

[3 ОПИСАНИЕ ПРОТОТИПА ПРОЕКТА 15](#_Toc76072018)

[1.1 Расчет энергопотребления прототипа 16](#_Toc76072019)

[1.2 Оценка стоимости прототипа 16](#_Toc76072020)

[4 ПЕРСПЕКТИВЫ РАЗВИТИЯ ПРОЕКТА 17](#_Toc76072021)

[Список литературы 18](#_Toc76072022)

# 1 ЦЕЛИ И ЗАДАЧИ ПРОЕКТА

В любом спорте здоровье, время и деньги используемые для изучения техники и последующей ее постановки являются основными ресурсами, и оптимизация их использования является очень важной проблемой как для профессионалов, так и для любителей, поэтому мой проект может оказаться очень востребованным.

Двенадцати лет я посвятил профессиональному спорту, что дало мне бесценный жизненный опыт в организации своего тренировочного процесса.

И так данную проблему я решил рассмотреть на примере тенниса.

На данный момент на рынке существует две компании представителя представившие свои продукты – HEAD [1] и ZEPT [2].

Их датчики собирают статистику о: типе удара, времени игры, скорости удара, кол-ве потраченных калорий и пятне попадания.

Обе компании ориентированы на сбор статистических данных в целях создания конкурентной среды между игроками.

Также они предоставляют возможность отточить некоторые навыки с помощью записанных обучающих видео и возможностью воспроизвести движение подачи, что уже предоставляет неплохие возможности для развития игроков.

Но эти возможности можно расширить следующими способами:

* Для игрока

1. Возможность поиска ошибок в движении в сравнении с идеальным. Идеальное движение может быть задано тренером или выбрано из базы заготовленных разработчиком, другими пользователями.
2. Отчет после тренировки или матча включает полную статистику о перемещении игрока по корту и стратегических решениях игрока.
3. Возможность увидеть копию своих движений на большом экране или смартфоне в любой реальном времени или в записи
4. Возможность планировать тренировочный процесс и свою профессиональную карьеру, не выходя из приложения

* Для тренера:

1. Возможность обучить больше игроков улучшив качество тренировок
2. Возможность планировать и следить за карьерой своих спортсменов
3. Возможность получать только максимум КПД, без ущерба здоровью как минимум за счет взвешенного расписания.

Важно в конченом итоге не принуждать конечного пользователя обвешиваться во время тренировки более чем одним носимым модуля и при этом носимый модуль должен быть максимально лёгким, насколько это возможно. Данное требование является необходимым, т.к. конченый продукт должен помогать пользователю без вреда или помех тренировочному процессу.

# 2. АРХИТЕКТУРА ПРОЕКТА

## 2.1 Общая схема архитектуры проекта

Общая архитектура прототипа состоит из трех блоков:

1. Блок смартфона – смартфон, приложение.
2. Блок ракетки – 6-ти осевой (акселерометр, гироскоп) IMU сенсор MPU-6050[3], Arduino UNO, Bluetooth модуль HC-05[4]
3. Блок дополнительного поднятия точности, состоит из инфракрасной камеры и метки (не реализован в силу ограниченности времени и ресурсов)

На рисунке 1 представлена визуализация взаимодействия трех блоков.

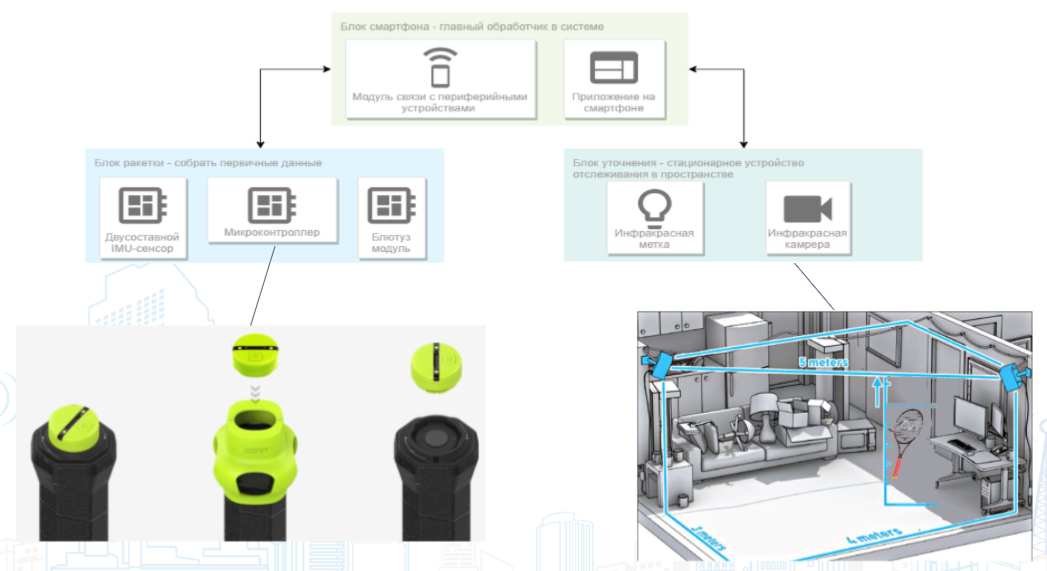


Рис. 1 Архитектура прототипа

## 2.2 Описание интерфейсов между компонентами решения и форматов данных

Общение между микроконтроллером, IMU сенсором проходит по протоколу I2C и Bluetooth модулем по Serial интерфейсу.

## 2.3 Описание и обоснование аппаратных решений

Академия выдала микроконтроллер Arduino (дешево, популярно, в наличии) и дала возможность использовать Bluetooth модулем HC-05(Troyka BT).

Также был выдан IMU сенсор на 10 степеней свободы, его настройка и подбор фильтров для стабилизации датчика не дали ожидаемого результата. Это проявлялось в аномалиях, получаемых от акселерометра и дрейфе гироскопа, последним при тестировании использовался расширенный фильтр Калмана который позволил стабилизировать кватернион, описывающий вращение датчика на первые 3-5 секунд работы. К сожалению такой результат являлся неудовлетворительным для дальнейшего использования этого датчика, т.к. в этом проекте важно видеть стабильные показания датчика на протяжении активной, многочасовой сессии.

Далее было принято решение купить новый датчик более точный и современный. По результатам поиска было принято решение, что новый датчик обязательно должен иметь блок DMP (Digital Motion Processor) и регистры смещения [5]. Был выбран шестиосевой сенсор MPU-6050, т.к. он оснащен вышеописанными функциями и был в наличии в магазинах Москвы.

Новый сенсор позволил откалиброваться достаточно точно для проведения тестов на протяжении 15 минут без потерь в ориентации. Но погрешность измерений сенсора стабильна только на трёх знаках после запятой, что несет потери при переходе к скорости. Также в данный момент сенсор работает в режиме чувствительности 1g, что приведёт к потерям данных при резких движениях и сильных ударах. Данную проблему было решено исправить с помощью второго, более точного сенсора, настроенного на другую чувствительность.

Поэтому был дозаказан сенсор CJMCU-20948[6], который будет работать параллельно в режиме перегрузок 2g или 4g и даст возможность компенсировать потери данных при соответствующих перегрузках.

## 2.4 Описание и обоснование используемых технологий связи

Обратимся к рисунку 1 и рассмотрим взаимодействие блоков смартфон-ракетка и смартфон-камера (блок поднятия точности).

Смартфон-ракетка.

Общаются по Serial-over bluetooth c баудрейтом 1382400, данные записываются в структуру, кодируются в байтовую строку и отправляются на смартфон, где распарсив строку по размеру переменных происходит дальнейшая обработка, визуализация и запись данных в файл.

Структура состоит из данных о:

1. Кватернионе: qw, qx, qy, qz;
2. Ускорении: ax, ay, az;
3. Времени между итерациями: dt.

Данный баудрейт и самописный протокол общения были использованы для максимизации скорости отправки данных, в данный момент dt между измерениями составляет около 700 микросекунд. Такая задержка связана в первую очередь со временем отработки методов приведения абсолютного ускорения к линейному и к мировому, подробнее расскажу в программной части.

Смартфон-камера (пока на уровне идеи).

Данный блок состоит из инфракрасной метки которую нужно будет внедрить в датчик на ракетке и инфракрасной камеры, прикрепленной к смартфону. Инфракрасная камера жёстко соединена со смартфоном по проводу. Смартфон и камера должны быть закреплены на штативе и стоять на территории корта так, чтобы инфракрасная камера видела полностью территорию маневра игрока, в конечном итоге это позволит получить как более точное положение ракетки, так и всю историю перемещений игрока.

## 2.5 Описание и обоснование используемых программных компонентов

Arduino IDE – для настройки блока ракетки. В скетче последней прошивки происходит обработка ускорения, мы получаем из абсолютного ускорения – мировое с помощью поворота вектора абсолютного ускорения на соответствующий кватернион и сложения результата с вектором силы тяжести. Эти вычисления приводят нас к увеличению задержки с 60 до 700 микросекунд. В дальнейшем методы будут перенесены в Unity.

Unity – для визуализации движения датчика в реальном времени и записи в файл обработанных данных.

Среда и движок Unity были выбраны, т.к. было необходимо иметь возможность быстро визуализировать вращение кватерниона и перемещение, полученное с помощью интегрирования мирового ускорения. Также использование Unity дает возможность быстро собрать проект как под смартфон, так и под ПК.

На рисунке 2 и 3 представлена 3D сцена в приложении Unity

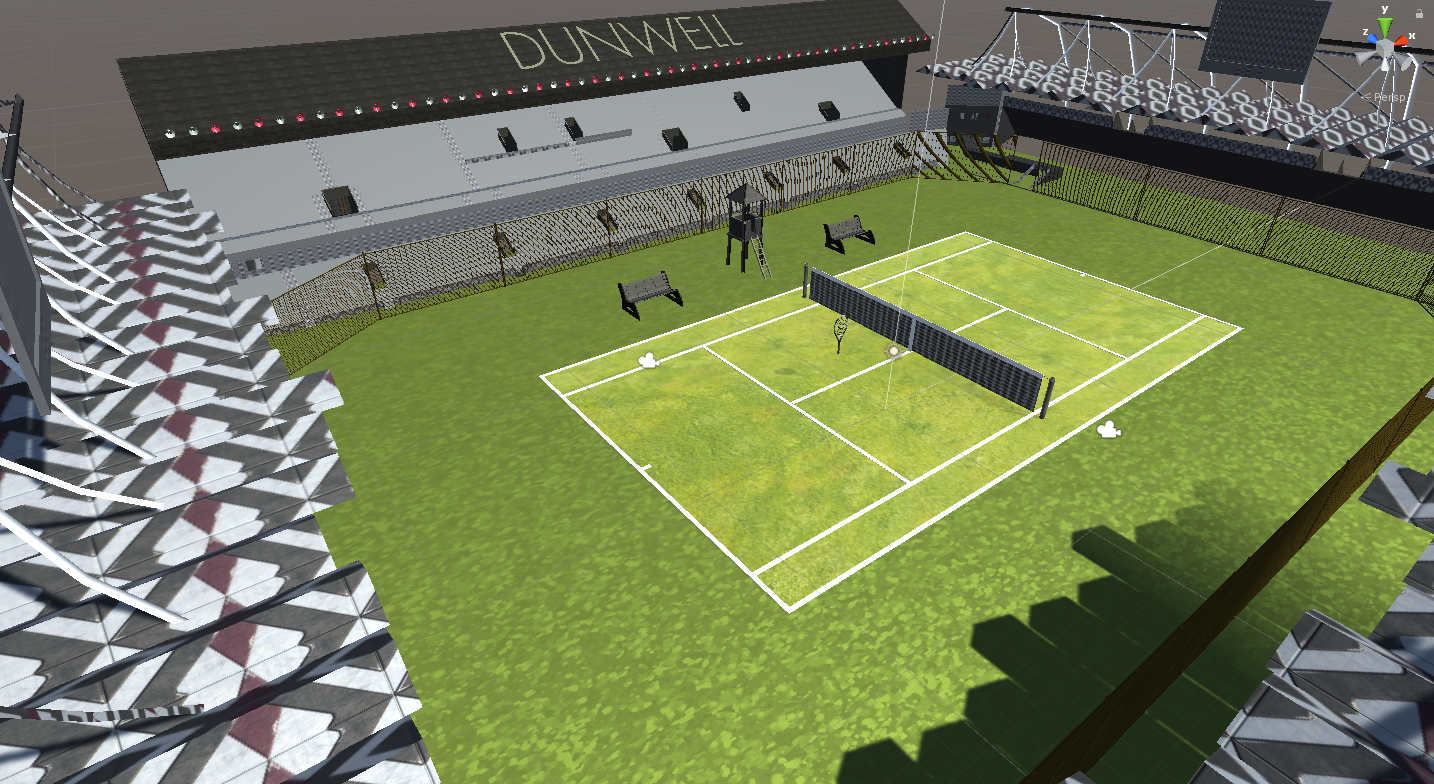


Рис. 2 3D Сцена в приложении Unity

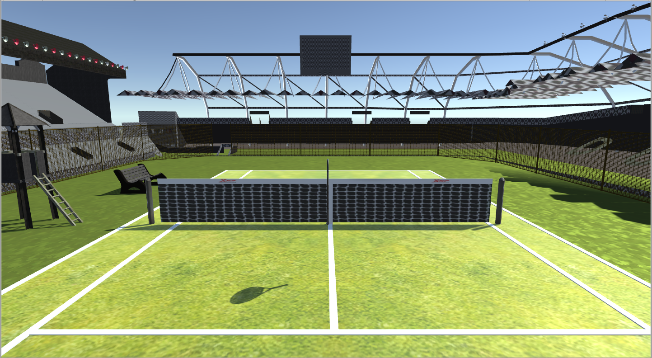
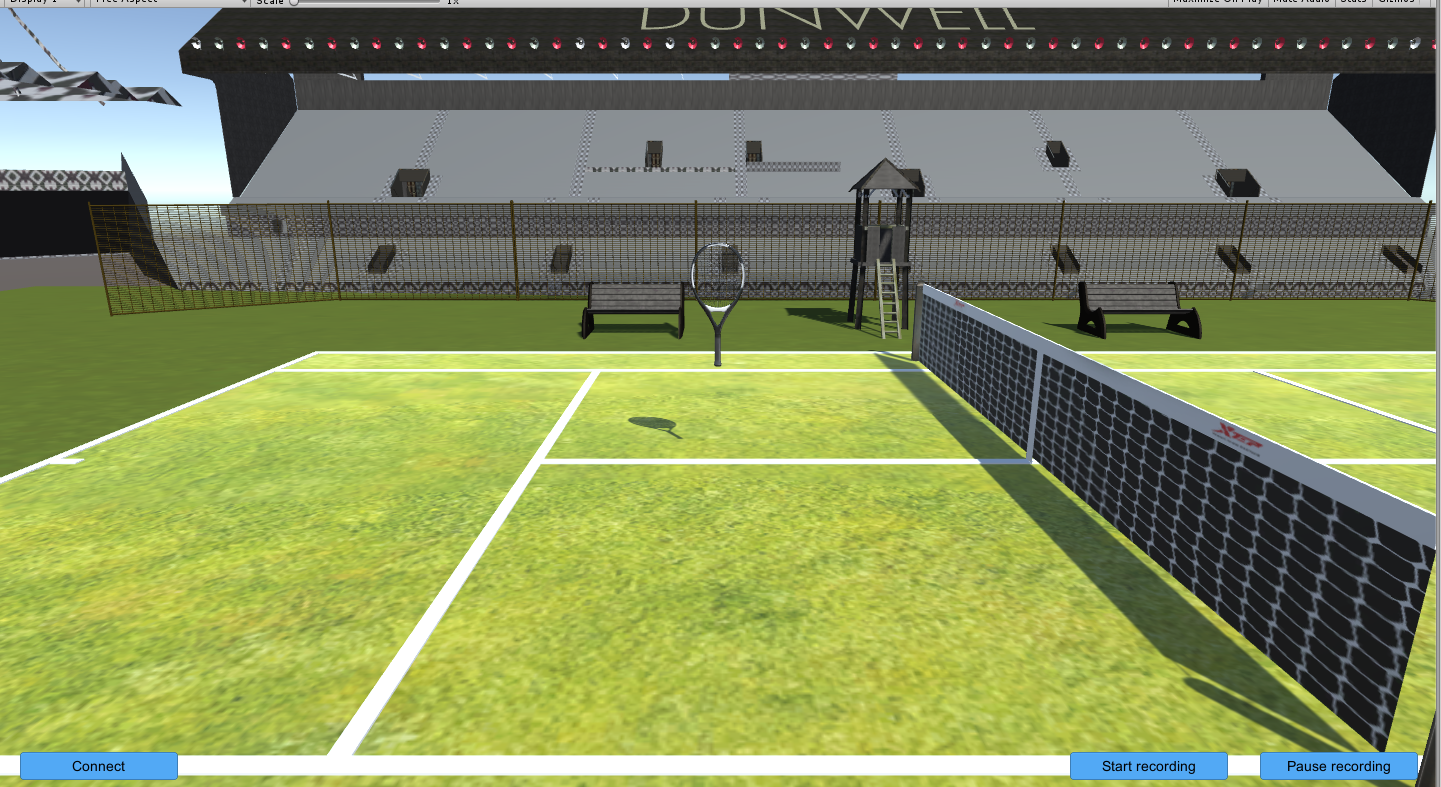


Рис. 3 3D Сцена в приложении Unity от лица пользователя

Также в Unity были предприняты попытки по переходу к траектории движения ракетки по следующим формулам:

|  |  |
| --- | --- |
|  | (1) |

Где Acc –ускорение, Vel – скорость, на первой итерации равная нулю.

|  |  |
| --- | --- |
|  | (2) |

Где S – перемещение, на первой итерации равное нулю.

По результатам тестирования в Unity на первом этапе моделька, ракетки которой мы присваивали получившиеся значения смещалась с огромной погрешностью, что можно считать провалом. Таким образом было решено выводить полученные данные в файл для дальнейшего анализа в Python, что в бедующем должно дать возможность обучить нейросеть или дерево решений на записанных данных.

Таким образом разработка перешла в Python, где было создано графическое приложения для разметки данных, и несколько программ позволяющих построить графики визуализирующие полученные данные.

На рисунке 4 представлена зависимость изменения скорости от времени по осям X, Y, Z, данные используемые для построения этого графика были взяты при подключении по проводу.

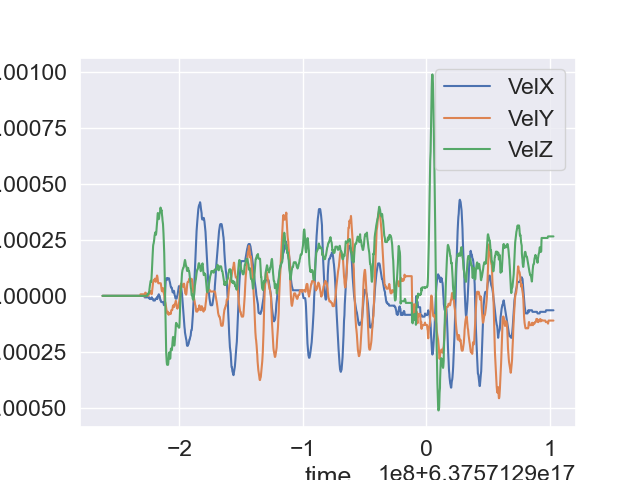


Рис. 4 График зависимости скорости от времени

Из рисунка 4 следует, что скорость не сходится к нулю после резкого движения, это объясняется тем, что мне не хватило частоты считывания данных с акселерометра. На данный момент интервал между итерациями dt составляет 700 микросекунд, его можно уменьшить до 60 микросекунд, что должно дать хороший прирост в точности, но дальнейшую оптимизацию было решено отложить и попробовать типизировать движения с помощью кватернионов, для этого необходимо было bluetooth подключение и дерево решений или нейронная сеть, которая смогла бы автоматически определять типы ударов.

На рисунке 5 представлены графики, показывающие зависимость ускорения от времени полученные по bluetooth соединению.

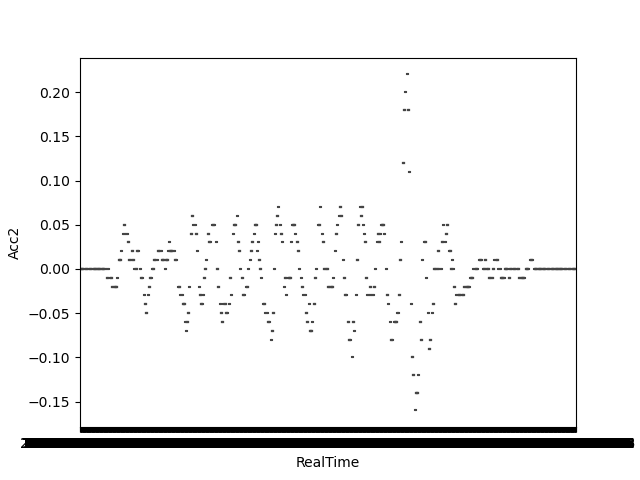


Рис. 5 График зависимости ускорения от времени

Из рисунка 5 следует, что данные получаемые по bluetooth не содержат аномалий, давайте посмотрим на рисунок 6.

На рисунке 6 представлен общий вид приложения написанного с помощью библиотеки PyQT [7] после разметки данных.

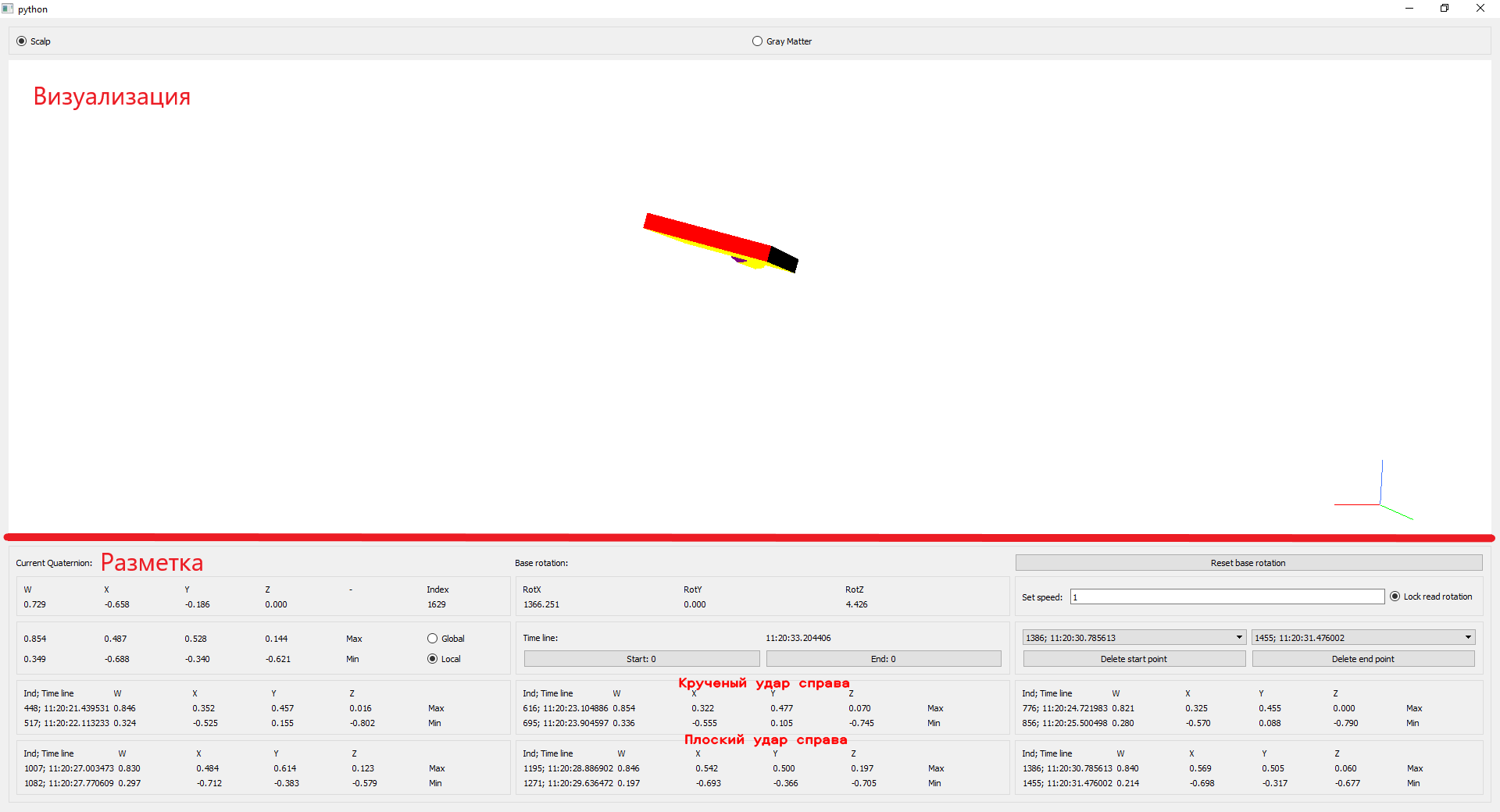


Рис. 6 Общий вид приложения после разметки движений «Крученый удар справа» и «плоский удар справа»

Разобьем приложение на два блока – «Визуализация» и «Разметка».

Рассмотрим блок «Визуализация», по центру приложения расположена упрощённая моделька ракетки, состоящая из двух примитивов – прямоугольника, изображающего головку ракетки и цилиндра с восемью гранями представляющего ручку ракетки. Опорная точка находится на нижнем конце ракетки.

На рисунке 7 представлено сравнительный пример нарисованной модельки и фотография в реальности.

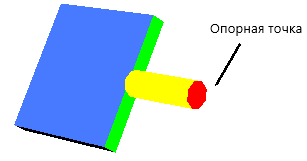


Рис. 7 Моделька ракетки, её представление в реальности

Для изменения опорной точки модельки в приложении есть возможность изменять положение или поворот с помощью удержания левой или правой кнопки мыши соответственно, совместно со смещением мыши в необходимую сторону. Также можно приблизить или отдалить камеру от объекта с помощью колеса прокрутки мыши. Для ориентации в пространстве были добавлены оси X –красная, Y – зеленая и Z –голубая, которые вращаются вместе с изображённой моделькой.

Теперь рассмотрим блок «Разметка», его можно поделить также на два блока статический и динамический с точки зрения появления новых графических блоков. На рисунке 8 представлена визуализация подблоков в блоке «Разметка».

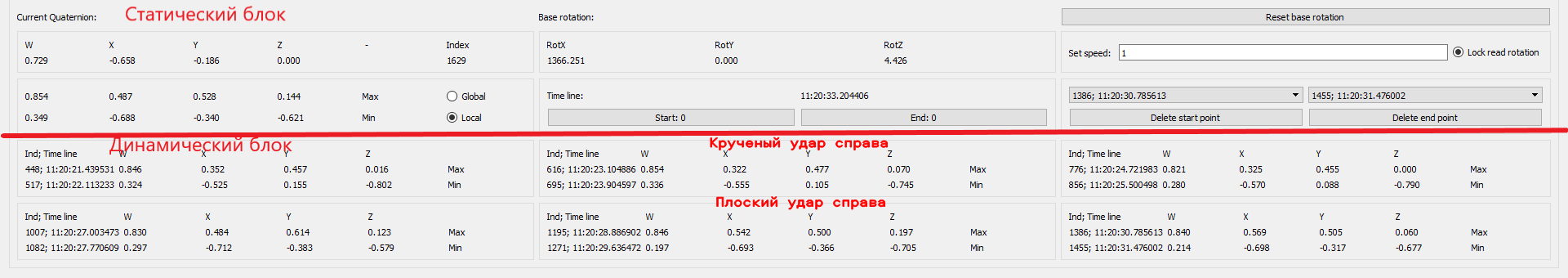


Рис. 8 Статический блок и динамический.

Рассмотрим статический блок и разделим его как представлено на рисунке 9.

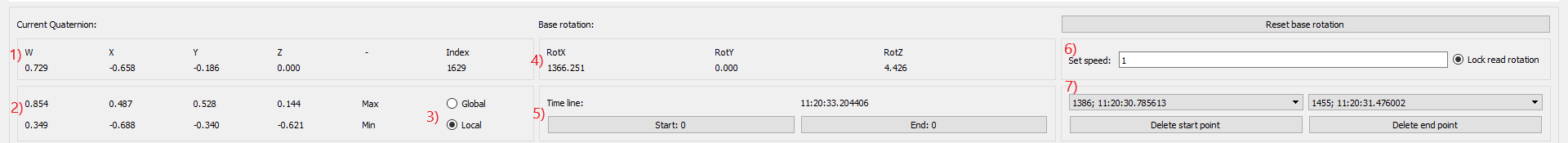


Рис. 9 Разделение статистического блока

Статический блок состоит из следующих частей:

1. Показания текущего кватерниона и индекс в считываемых данных таблицы;
2. Максимум и минимум по каждой из осей кватерниона;
3. Выбранный вид визуализации части 2 – глобальный, локальный;
4. Поворот опорной точки вокруг осей X, Y, Z;
5. Временная линия, считываемая из таблицы. Кнопки отвечающий за разметку начала и окончания движения;
6. Кнопка Reset base rotation – сбрасывает примененные в части 4. Линия для редактирования – предназначена для установки скорости одной итерации (целый значения);
7. История поставленных начальных и конечных точек, а также кнопки для удаления выбранных точек.

Теперь рассмотрим динамический блок, представленный на рисунке 10.

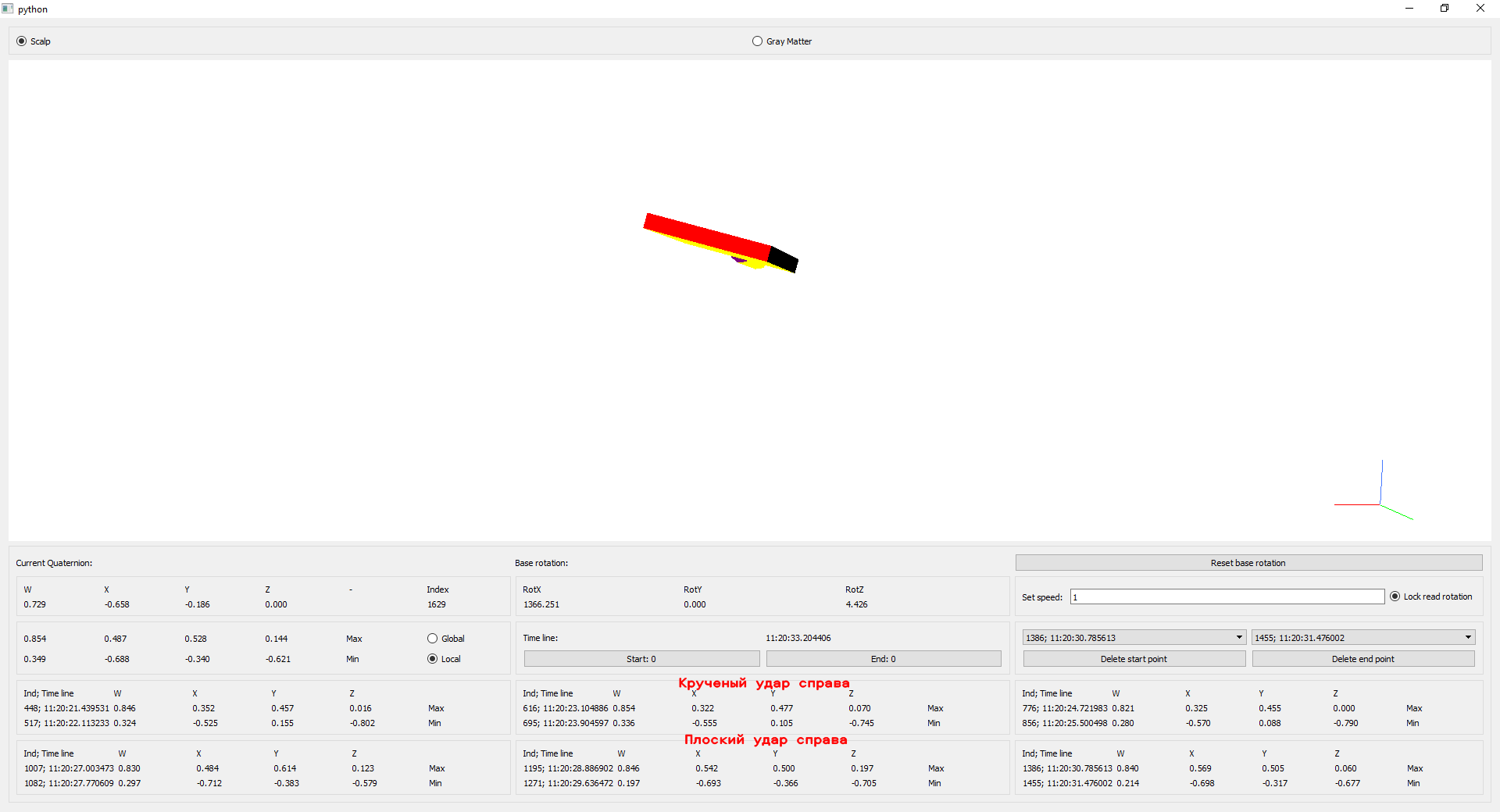


Рис. 10 Динамический блок

Динамический блок представляет собой историю найденных локальных минимумов и максимумов, числящихся в списках пункта 7 статического блока.

Динамический блок генерируется автоматически при переключении в режим отображения локальных минимумов, максимумов и добавлении – удалении конченой точки.

На рисунке 10 уже представлены некоторые результаты разметки, в первой строке размечен «Крученый удар справа», во второй строке «Плоский удар справа».

Из результатов разметки следует, что для минимума и максимума каждых из фаз или типов движения можно вычислить среднее значение с некоторым радиусом. Что позволит на раннем этапе подобрать условный блок, позволяющий специфицирующий каждое из движений. А далее на большем кол-ве размеченных данных нужно обучить дерево решений или нейросеть, где входные параметры будут расширены мировым ускорением.

# 3 ОПИСАНИЕ ПРОТОТИПА ПРОЕКТА

На рисунке 11 представлен внешний вид прототипа.

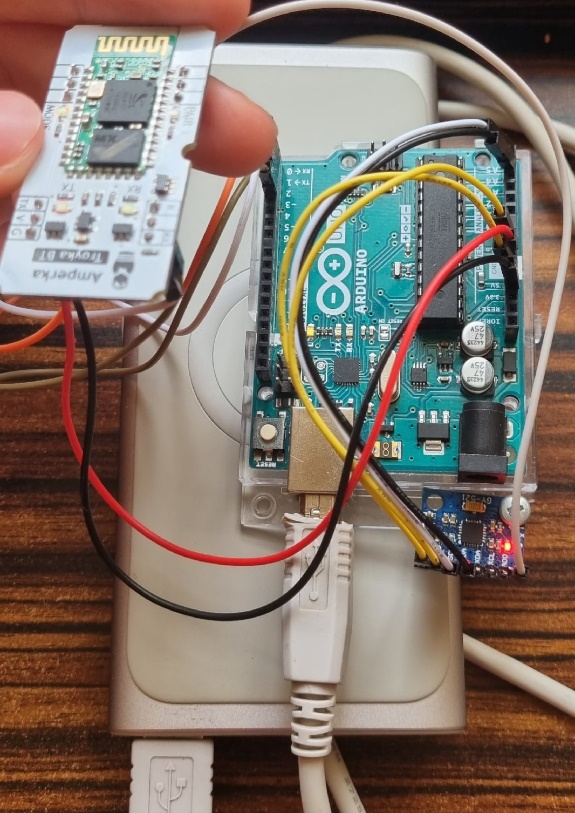


Рис. 11 Внешний вид прототипа

На рисунке 12 представлена схема подключения датчиков.

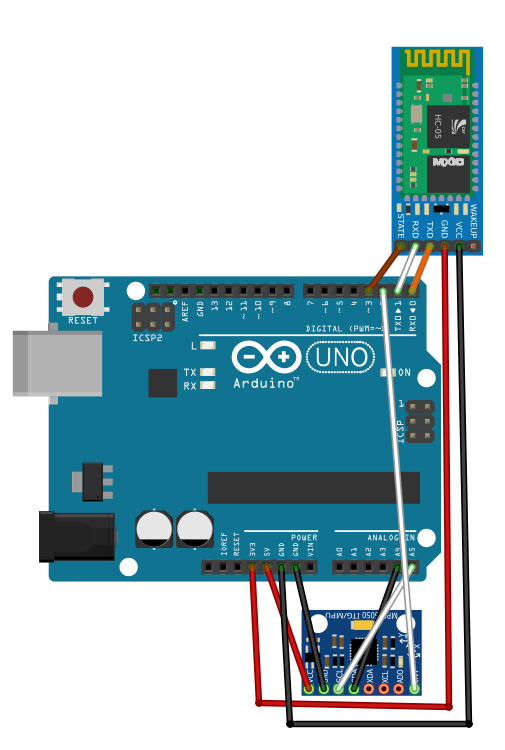


Рис. 12 Схема подключения

## Расчет энергопотребления прототипа

* HC-05 - 50 мА
* MPU-6050 - 10 мА
* Arduino UNO - 50 мА

**Итого:**

Расчетное среднее энергопотребление – 45 мА

Время работы от внешнего аккумулятора – 3-5ч

## Оценка стоимости прототипа

* HC-05 – 380 р
* Внешний аккумулятор – 1000 р
* Arduino UNO– 500 р
* IMU датчик MPU-6050 – 100 р

**Итого:** 2480 р

# 4 ПЕРСПЕКТИВЫ РАЗВИТИЯ ПРОЕКТА

**Планы на будущее:**

1. Расширить чувствительность системы за счет добавления второго инерциального датчика, настроенного на большие перегрузки.
2. Получить траекторию движения, научиться находить ошибки в траектории (есть идеальная – тренера или игрока, к которому стремится спортсмен и есть фактическая, ошибка – оценка разницы в движении между идеальным и фактическим по некоторым пороговым значениям на каждой из фаз движения). Для этого необходимо поднять точность измерений за счет добавления третьего блока, увеличения частоты измерений во втором блоке и расширения чувствительности второго блока.
3. Создать графический интерфейс, позволяющий с удобством вместе работать игрокам и тренерам.
4. Добавить связь с сервером для индексации пользователей и расширения возможностей коммуникации между ними.

# Список литературы

1. Продукт компании HEAD: официальный сайт. — международный — URL: <https://www.head.com/en/sensor> (дата обращения 29.06.2021).
2. Продукт компании ZEPP: официальный сайт. — международный — URL: <http://www.zepplabs.com/en-us/tennis/> (дата обращения 29.06.2021).
3. Сенсор компании MPU-6050: официальный сайт. — Международный — URL: <https://invensense.tdk.com/products/motion-tracking/6-axis/mpu-6050/> (дата обращения 29.06.2021).
4. Продукт Bluetooth модулю hc-05 (Troyka BT) и документация — URL: <http://wiki.amperka.ru/%D0%BF%D1%80%D0%BE%D0%B4%D1%83%D0%BA%D1%82%D1%8B:troyka-bluetooth> (дата обращения 01.07.2021).

1. Подробности о возможностях датчика MPU-6050: неофициальный сайт. URL: <https://russianblogs.com/article/2224834212/> (дата обращения 29.06.2021).
2. Документация к сенсору ICM-20948: официальный сайт. — международный — URL: <https://invensense.tdk.com/wp-content/uploads/2016/06/DS-000189-ICM-20948-v1.3.pdf> (дата обращения 29.06.2021).
3. Библиотека PyQt5: официальный сайт. — URL: https://pypi.org/project/PyQt5/ (дата обращения 25.06.2021).