|  |  |  |
| --- | --- | --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ Федеральное государственное автономное  образовательное учреждение высшего образования  «Южно-Уральский государственный университет  (национальный исследовательский университет)»  Высшая школа электроники и компьютерных наук  Кафедра «Электронные вычислительные машины» | | |
| РАБОТА ПРОВЕРЕНА  Рецензент  \_\_\_\_\_\_\_\_\_\_\_\_Е.А. Зверева  « » июня 2021 г. | | ДОПУСТИТЬ К ЗАЩИТЕ  Заведующий кафедрой ЭВМ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Г.И. Радченко  «\_\_\_»\_\_\_\_\_\_\_\_\_\_ 2021 г. |
| Система поддержки indoor соревнований по велосипедному спорту | | |
| ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ | | |
|  | Руководитель работы,  к.т.н., доцент каф. ЭВМ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_И.Л. Кафтанников «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_2021 г. | |
|  | Автор работы,  студент группы КЭ-222 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_О.И. Морозов «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_2021 г. | |
|  | Нормоконтролёр,  ст. преп. каф. ЭВМ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С.В. Сяськов «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2021 г. | |
| Челябинск-2021   |  |  | | --- | --- | | МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ Федеральное государственное автономное  образовательное учреждение высшего образования  «Южно-Уральский государственный университет  (национальный исследовательский университет)»  Высшая школа электроники и компьютерных наук  Кафедра «Электронные вычислительные машины» | | |  | УТВЕРЖДАЮ  Заведующий кафедрой ЭВМ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Г.И. Радченко «\_\_\_»\_\_\_\_\_\_\_\_\_\_2020 г. | | **ЗАДАНИЕ**  **на выпускную квалификационную работу магистра**  студенту группы КЭ-222  Морозову Олег Ивановичу  обучающемуся по направлению  09.03.01 «Информатика и вычислительная техника» | | | 1. Тема работы: Разработка системы цифрового соревнования с помощью велотренажера утверждена приказом по университету от 24 апреля 2021 г. № 627 | | | 2. Срок сдачи студентом законченной работы: 1 июня 2021 г. | | | 3. Исходные данные к работе:  – Языки программирования: C#, Python.  – Платформы разработки: ARM, Windows, Linux.  – Библиотеки ARM: I2C 128x64 OLED Display, Wire, Mouse, HID.  – Библиотеки Linux: requests,serial,time,VR,. | | | | |

– Сервер на операционной системе Ubuntu/Debian, с установленными Apache2/NGINX,MySQL/PosgreSQL, node.js/ Django.

– Для отладки базы данных используется PGAdmin.

– Микроконтроллер передаёт состояние велотренажера на установленный клиент персонального компьютера.

– Когда вы крутите педали на станке, датчики отправляют данные на компьютер через проводной(USB) или беспроводной (Bluetooth 5.0, WI-FI5, ANT+). Клиент Компьютера обрабатывает эти данные и позволяет кататься и соревноваться вместе с другими пользователями по всему миру.

– VR Гарнитура позволит вам поворотом головы изменить своё направление взгляда, как в реальной жизни.

4. Перечень подлежащих разработке вопросов:

– анализ литературы по теме умные тренажеры и соревнования по велоспорту;

– рассмотрение существующих аналогов эмуляторов соревнования по велоспорту, оценка их сильных и слабых сторон;

– разработка собственного сервиса соревнования по велоспорту;

– оценка работоспособности севиса с разными режимами и в разных условиях.

5. Дата выдачи задания: 1 декабря 2020 г.

Руководитель работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/И.Л. Кафтанников /

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/О.И. Морозов /

КАЛЕНДАРНЫЙ ПЛАН

|  |  |  |
| --- | --- | --- |
| Этап | Срок сдачи | Подпись  руководителя |
| Введение и обзор литературы | 01.03.2020 |  |
| Разработка модели, проектирование | 01.04.2020 |  |
| Реализация аппаратного прототипа велосипедного тренажера | 15.05.2020 |  |
| Реализация программного прототипа сервиса велосипедных соревнований | 01.05.2020 |  |
| Тестирование, отладка, эксперименты | 15.05.2020 |  |
| Компоновка текста работы и сдача на  нормоконтроль | 24.05.2020 |  |
| Подготовка презентации и доклада | 30.05.2020 |  |

|  |  |
| --- | --- |
| Руководитель работы | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/И.Л. Кафтанников / |
| Студент | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/О.И. Морозов/ |

АННОТАЦИЯ

|  |  |
| --- | --- |
|  | О.И. Морозов. Организация соревнований по велоспорту в удалённом режиме. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШЭКН; 2021, XX с., XX ил., библиогр. список – XX наим. |

В рамках выпускной квалификационной работы производится детальный анализ литературы по темам: домашний симулятор соревнований по велоспорту. Организуется разработка программного обеспечения серверной, интерфейсной, микроконтроллерной частей, а также базы данных. Производится выборка и анализ результатов работы разработанного велотренажера в предложенных режимах. Рассматриваются преимущества и недостатки разработанного программно-аппаратного комплекса. Доказывается способность системы разработанного велосипедного тренажера и сервиса велосипедных соревнований.

Пояснительная записка включает в себя введение, оглавление, основную часть, заключение и библиографический список.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ 7

# ВВЕДЕНИЕ

**Актуальность данной темы**

Спорт играет большую роль в жизни людей. Он укрепляет здоровье, воспитывает характер, делает человека сильным и выносливым, закаляет организм. Век назад физические качества – выносливость, сила – ценились людьми. Но роль физической силы падает из-за развития техники и снижения уровня агрессии в обществе. По мере развития технологий работа, требовавшая физической силы, переходит к машинам, а оператору машины особая физическая сила уже не нужна. В настоящее время людей тянет заниматься спортом больше для поддержания здоровья или хобби.

Международный олимпийский комитет (IOC) провел опрос среди спортсменов и других представителей отрасли, по итогам которого выяснилось, что простой из-за пандемии коронавируса вылился для атлетов в потерю мотивации, а для функционеров и организаторов соревнований — в целый комплекс проблем материального характера.[1]

**Описание проблемы**

За последние годы отрасль тренировок на велотренажерах активно развивалась, а появление разумных тренажеров и программ стало началом трансформации велоспорта изнутри. Это не только жизнеспособная и довольно реальная альтернатива велосипедной езде, которой пользуются любители и профессионалы. Такие тренировки также доставляют удовольствие. Это идеальный вариант для катания в плохую погоду, при ограниченном количестве времени или при необходимости сфокусироваться на тренировке без мыслей о плохих дорогах. Или же когда «за окном» карантин и приходится находиться в самоизоляции.

Появилась цель с подвинуть людей заниматься спортом, превратив спорт в доступную из дома соревновательную платформу со множеством пользователей.

Для решения описанной проблемы следует разработать сервис имитации занятия велоспортом с погружением в виртуальную реальность с физическими нагрузками.

Он поможет малозамотивированным людям выбрать любую погоду и ландшафт для занятия спортом, собирать статистику занятия велоспортом и увеличить характеристики выносливости.

Решение проводить виртуальные соревнования по велоспорту найдут применение в разных категориях велоспорта:

* Начальный уровень – позволит скоротать время с пользой для здоровья в красивых местах реального или вымышленного мира.
* Средний уровень – позволит организовать любительское соревнование в кругу своих знакомых или небольших районов с соблюдением ряда нормативов, если некие штрафные санкции ограничивают живую встречу, а система принятия решений поможет явно определить победителей.
* Высокий уровень – соревновательный. Организация более строгих соревновательных и регулярных соревнований с составлением рейтинга лучших спортсменов. Данное соревнование может увидеть каждый, если кото из тренирующего или поддерживающего состава не сможет поехать с на реальную встречу, а так он может следить за соревнованием.

Вопросы, которые нужно решить с точки зрения спортивной оценки:

* воссоздать реальные условия соревнования в разных категориях;
* составить Нормы, требования и условия их выполнения по виду спорта.

Вопросы, которые нужно решить с точки зрения IT-специалиста:

* выбрать мобильную платформу для приложения;
* выбрать язык программирования;
* выбрать среду разработки;
* исследовать протоколы связи тренажеров с сервером.

1.АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. ЦЕЛЬ ДИПЛОМНОЙ РАБОТЫ

Целью дипломной работы является разработка программно-аппаратного комплекса для интерактивного велотренажера. Он позволит подключить большинство старых моделей велотренажеров к ПК, для эмуляции тренировки в открытом мире или трассе.

Для достижения данной цели необходимо сделать устройство, аппаратная часть которого будет на основе микроконтроллера, устанавливающейся на велотренажере и собирающее данные о количестве оборотов и поворотах, сделанных пользователем и передающее их на ПК через интерфейс USB.

С помощью программной части, данные будут преобразовываться и воспроизводиться в игре, разработанной на Unity.

**1.2. ОБЗОР ЛИТЕРАТУРЫ**

Был проведен анализ наиболее популярных бесплатных или условно бесплатных средств разработки компьютерных игр. Для сравнения движков был выбран электронный ресурс [x], в котором подробно описаны самые популярные средства разработки игр, предоставляющих свою бесплатную версию пользователю. Эти платформы доступны для свободного использования.

Из рассматриваемых платформ особенно выделились Unity и Unreal Engine 4 [x], так как они понятны для использования, в них схожие возможности и они бесплатны, что очень важно для начинающих программистов. Остальные платформы не подошли, так как они уступали Unity и Unreal Engine 4 по сравниваемым параметрам.

Для разработки игрового приложения была выбрана платформа Unity[x]. Данная платформа имеет низкий порог вхождения, большое количество обучающих материалов [xx] и сообщество разработчиков, вследствие чего, с ней можно быстрее начать работать.

Raspberry Pi часто используется как мозг робота, домашний сервер или просто компьютер. Во книге Саймона Монк «Raspberry Pi. Сборник рецептов. Решение программных и аппаратных задач» [xx]содержится свыше 240 полезных рекомендаций и советов по практическому применению Raspberry Pi. Рассматриваются такие вопросы, как настройка компьютера с Linux, написание программ на Python, управление двигателями и датчиками, а также взаимодействие Raspberry Pi с другими электронными устройствами, включая Arduino и проекты IoT (интернет вещей). Опытный разработчик и автор популярных учебных пособий Саймон Монк знакомит читателей с базовыми принципами построения любительского электронного оборудования, которое основано на популярной микроконтроллерной платформе Raspberry Pi, обладающей невероятно большим потенциалом для применения в серьезных коммерческих проектах.

1.3. ЭТАПЫ РЕШЕНИЯ ЗАДАЧИ

Для достижения цели выпускной работы необходимо выполнить несколько этапов:

1. Анализ существующих решений.

2. Выявление достоинств и недостатков существующих устройств.

3. Разработка программно-аппаратной части комплекса:

– разработка структуры устройства;

– выбор компонентов схемы;

– интеграция с компьютерным приложением;

– тестирование работы комплекса.

1.4. ОБЗОР АНАЛОГОВ

Учитывая нынешний глобальный кризис, все больше и больше велосипедистов обращают свое внимание на дома, поскольку они надеются сохранить форму в условиях самоизоляции и социального дистанцирования. Один из лучших способов сохранить мотивацию в эти нестабильные времена - добавить в свою жизнь виртуальную программу тренировок. Рассмотрим наиболее популярные.

* + 1. Zwift

Zwift — это игра с турбо-тренером, которая позволяет вам подключить турбо-тренажер к компьютеру, iPad, iPhone или Apple TV, позволяя вам кататься с другими велосипедистами в виртуальной среде, тем самым помогая облегчить скуку, связанную с катанием в помещении. вестный из них мобильных приложений для виртуальных тренировок.

Помимо соревнований с другими гонщиками в гонщиках Zwift, те, кто ищет конкретные тренировки, могут получить доступ к тренировкам, разработанным профессиональными тренерами, и они могут быть выполнены в группах с гонщиками, выполняющими усилия с одинаковой интенсивностью на основе процента от их FTP[2]. Из недостатков, данная платформа не бесплатная. Zwift стоит около 15 долларов в месяц.

* + 1. Onelap

Onelap – Китайский аналог Zwift. В Onelap есть возможность заниматься и развлечься. Onelap создает реалистичную среду с удивительно четкой графикой деталей, великолепными цветами и физической моделью, которая имитирует градиент, ветер и сопротивление качению, вы можете адаптировать каждый аспект своего аватара, велосипеда и маршрутов - независимо от холмистости маршрут, ровный маршрут или горный маршрут, вы не будете ездить на велосипеде в повторяющейся сцене в пределах 100 км пути. Большинство игрков из Азии, но увеличивается и число европейцев. На данный момент игра бесплатна, но что будет в дальнейшем неизвестно.

* + 1. RGT Cycling

RGT Cycling — это приложение для проведения тренировок в помещении, которые симулируют реальные велотрассы по всему миру с помощью умного велотренажера. Вместо того чтобы крутить педали в виртуальном мире, вы будете ездить по известным маршрутам, испытывая реальные ощущения.

Также существуют различные структурированные тренировки, разработанные тренерами, и вы можете создать свою собственную гонку, загрузив файл маршрута.

Особенность RGT Cycling заключается в том, что во время гонки вы лучше ощущаете реальность по сравнению с некоторыми конкурентами, если вы движетесь впритык за другим гонщиком, а также создается эффект торможения на поворотах. Если вы хотите обогнать своего соперника, вам придется крутить педали быстрее или прикладывать больше усилий.

RGT Cycling утверждает, что данные о стандартной силе в их приложении являются более реалистичными, чем в других симуляторах велогонок, а скорость исчисляется исходя из реальных данных, и ваше изображение будет автоматически замедляться, чтобы вы не врезались в гонщика, который едет перед вами.

Также есть звание «горный король», таблицы классификации участков трассы, таким образом вы можете бросить вызов самому себе.

Таблица XX – Сравнение приложений велосоревнования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Название** | **Zwift** | **Onelap** | **RGT Cycling** | **Российские аналоги** |
| Язык | Английский | Китайский | Английский | - |
| Интерфейсы связи | BLE, ANT+ | ANT+ | ANT+ , BLE | - |
| Режим тренировки | + | + | + | - |
| Режим любительский | + | + | + | - |
| Режим профессиональный | + | - | + | - |
| Стоимость  оботудования | 300 долларов | 300 долларов | 300 долларов | - |
| Стоимость  услуги | 10 долларов США в месяц | - | 15 долларов США в месяц | - |

Особенность RGT Cycling заключается в том, что во время гонки вы лучше ощущаете реальность по сравнению с некоторыми конкурентами, если вы движетесь впритык за другим гонщиком, а также создается эффект торможения на поворотах. Если вы хотите обогнать своего соперника, вам придется крутить педали быстрее или прикладывать больше усилий.

RGT Cycling утверждает, что данные о стандартной силе в их приложении являются более реалистичными, чем в других симуляторах велогонок, а скорость исчисляется исходя из реальных данных, и ваше изображение будет автоматически замедляться, чтобы вы не врезались в гонщика, который едет перед вами.

Также есть звание «горный король», таблицы классификации участков трассы, таким образом вы можете бросить вызов самому себе.

**1.5. АНАЛИЗ И ПОДБОР КОМПОНЕНТОВ АППАРАТНОЙ**

**ЧАСТИ КОМПЛЕКСА**

В данном разделе сравним готовое аппаратное решение для велосипедного соревнования в помещении с компонентами для самостоятельной сборки аппаратного комплекса.

**Велосипедный тренажер Deuter MT-04**

Тренажер с блоком переднего колеса и быстросъемным шпагатом. Готовое решение для тренировки на собственном велосипеде в комнате. Подходит для велосипедов с диаметром колеса 26 "~ 28". Имеется проводной контроллер на 6 скоростей магнитного сопротивления. Внешний вид умного тренажёра показан на рисунке 1.



Рисунок 1- Велосипедный тренажер Deuter MT-04

Преимущества:

* максимальное снижающий шум магнитный ролик колеса, разработан с теплоотводящим вентилятором;
* легко собрать без дополнительных инструментов, все включено в посылка;
* 6 уровней скорости для переключения передач, регулировка не нужно снимать с велосипеда;
* самая низкая цена на ту же модель, мы производители;
* прочный материал, максимальная грузоподъемность более 135 кг.

Недостатки:

* Минимальная стоимость тренажера — 23 000 рублей.

Составим список компонентов для сборки собственного умного тренажера. В ходе выполнения выпускной квалификационной работы магистра был проведен анализ наиболее подходящих к данной работе микроконтроллеров, главными требованиями к выбору стали небольшая стоимость устройства, небольшой размер, достаточная мощность для считывания всех сигналов без задержки и возможность подключения по Bluetooth Low Energy.

**Raspberry Pi Zero W**

Одноплатный компьютер с возможностью подключения к беспроводной локальной сети и Bluetooth. Внешний вид показан на рисунке 2.

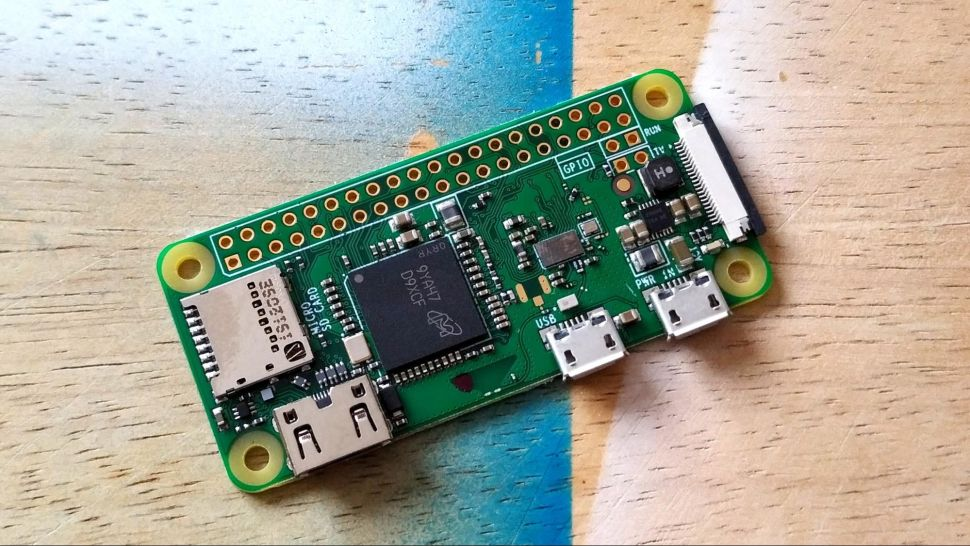


Рисунок 2 – Raspberry Pi Zero W

Raspberry Pi Zero W обладает всеми необходимыми функциональными возможностями:

* 802.11 b/g/n wireless LAN;
* Bluetooth 4.1+LE;
* одноядерный процессор с тактовой частотой 1 ГГц;
* 512 МБ ОЗУ;
* 28 портов GPIO;
* Питание от 5 В 2А.

**ESP32-WROVER-E Espressif**

ESP32-WROVER-E и ESP32-WROVER-IE - это два мощных универсальных модуля MCU WiFi-BT-BLE, предназначенных для широкого спектра приложений, от сетей датчиков с низким энергопотреблением до самых сложных задач, таких как кодирование голоса, музыка, потоковая передача и декодирование MP3. Внешний вид показан на рисунке 3



Рисунок 3 – ESP32-WROVER-E Espressif

ESP32-WROVER-E поставляется с антенной на печатной плате, а ESP32-WROVER-IE - с антенной IPEX. Оба они оснащены внешней флэш-памятью SPI объемом 4 МБ и дополнительной псевдостатической оперативной памятью SPI объемом 8 МБ (PSRAM). Информация в этом техническом описании применима к обоим модулям.

Сравнение двух контроллеров указаны в таблице 1.

Таблица 1 – Сравнение микроконтроллеров для комплекса

|  |  |  |
| --- | --- | --- |
|  | Raspberry Pi Zero W | ESP32-WROVER-E Espressif |
| Макс. частота, МГц | 1000 | 240 |
| RAM, Мб | 1024 | 8 |
| ROM, Мб | \* | 4 |
| I/O выводов | 40 | 38 |
| Цена, руб. | 1300 | 400 |
| Размер, см | 3x4 | 2x3 |

Вывод: в таблице представлено сравнение микроконтроллеров по основным характеристикам. Из представленных, дорогим, но наиболее подходящим является Raspberry Pi Zero W, так как в соответствии с требованиями размеров и стоимости подходит больше всего.

Для считывания количества оборотов педалей велотренажера, был выбран датчик Холла TLE4905L представленный на рисунке 4.



Рисунок 4 – Датчик Холла TLE4905L

За последние годы появилось множество гаджетов, которые снабжают велосипедистов информацией, мониторы сердечного ритма, разнообразные приложения с GPS-трекингом и т.д., но измеритель мощности велосипеда находится в особом статусе, безусловно ввиду своей высокой стоимости, но также за счет сути данных, которые он собирает.

В велоспорте единица измерения «Ватты» — это энергия, необходимая для перемещения массы на определенное расстояние за известный промежуток времени.

Мощность (Вт) = Сила х Расстояние / Время

1Вт = 1Нм/с.

Иначе говоря, чтобы массу, весом в 1 Ньютон переместить в пространстве на 1 метр за 1 секунду, необходимо затратить 1 Ватт энергии.

В умных тренажёрах присутствует динамическое сопротивление, регулируемое специальными программами для тренировок разной интенсивности.

В современных тренажерах нагрузка создается за счет магнитного, воздушного либо гидравлического тормоза.

Магнитный порошок в рабочей камере будет соединяться в состоянии связи под действием магнитного потока, генерируемого от хомута, когда ток проходит катушку возбуждения, магнитная муфта порошка может передавать крутящий момент, полагаясь на силу сдвига, генерируемую магнитной цепью и трением, генерируемым магнитным порошком и рабочей поверхностью. Внешний вид магнитного тормоза показан на рисунке 5.



Рисунок 5- Магнитный тормоз

Интегрированая тормозная система в велосипеде может использоваться как имитатор сопротивления. Преимущество в том что не нужно его покупать, но подключение к умному тренажеру требует электропривод для регулировки натяжения тормоза.

Универсальным решением будет щёточный двигатель. Его можно использовать в качестве тахометра, снимающего через приложеный к колесу вал скорость вращения. А при условии подачи напряжения в обратном направлениии движения колеса можно имитировать нагрузку в тренировке.

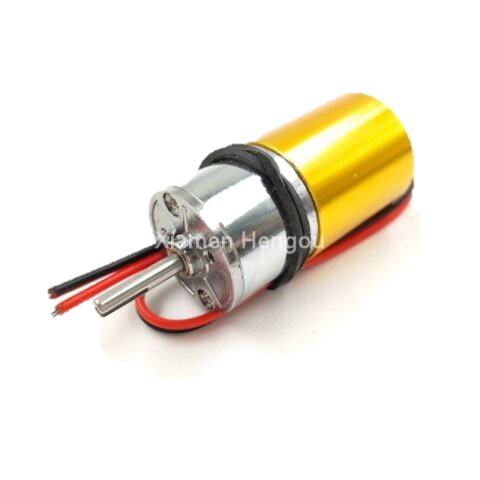
****

Рисунок 5- Тахогенератор



Готовые решения умных тренажеров имеют безпроводной протокол сопряжения и передачи данных.

ANT+ - технология беспроводной связи, использует нелицензируемый диапазон 2,4 ГГц. Дальность связи приблизительно сопоставима с Bluetooth: спецификация протокола ограничивает ее 30 метрами.

Еще одной полезной особенностью является существенно более низкое энергопотребление. Для смартфона это не столь критично, а вот то, что другие устройства, использующие интерфейс ANT+, могут питаться от плоской батарейки – это куда более важно для разработчиков.

Но ключевым свойством стандарта является его многоканальность. Связь организована по принципу master-slave, и ведущий аппарат способен получать информацию сразу от нескольких ведомых, при этом не мешающих друг другу.

Именно это сделало данный протокол столь привлекательным для создания всевозможного спортивного снаряжения.

В моде есть библиотека, которая прослушивает сигнал беспроводного протокола ANT + от тренера, совместимого с ANT + FE-C (Tacx, Wahoo, Elite, Bkool, Kinetic, Saris и т. Д.), Или измерителя мощности, или беговой дорожки Smart, или стопы. pod, считывает скорость и применяет ее к транспортному средству, которым управляет ваш персонаж во время игры. Он считывает наклон местности, неровность (материал) и ветер в игре и отправляет всю эту информацию в умный тренажер, чтобы он мог воспроизвести твердость местности, по которой вы на самом деле катаетесь. В таблице 2 приведено сравнение протокола ANT+ с Bluetooth Low Energy.

|  |  |  |
| --- | --- | --- |
| **Технология** | **ANT/ANT+** | **Bluetooth Low Energy** |
| Частотный диапазон | 2.4 – 2.483 ГГц | 2.4 – 2.483 ГГц |
| Поддерживаемые сетевые топологии | точка-точка, звезда, кластерное дерево, mesh | точка-точка, звезда |
| Модуляция | GFSK | GFSK |
| Ширина канала | 1 МГц | 2 МГц |
| Протокол | простой | более сложный |
| Скорость передачи данных | 1 Мбит/с | 1 Мбит/с |
| Радиус действия | 50 метров | 50 метров |
| Безопасность | 64-битный ключ | 128-битный алгоритм   шифрования AE |
| Стоимость | 1000 руб | 0 руб (встроен в кннтроллер) |

Вывод: ANT+ популярный протокол в области умных тренажеров, но довольно дорогой относительно BLE. Но не все сервисы поддерживают умные тренажеры с протоколом BLE.

В условиях хорошо проветриваемых помещений, комфортный теплоотвод за счет конвекции и испарения составляет порядка 585 Вт (при температуре воздуха должна быть 12.8 С). Эта цифра используется при дизайне индустриальных цехов.

Учитывая историю о метаболизме, велосипедист выдавал бы в таком цехе максимум 150 Вт на станке. Пределы разумного - 375 Вт [3]. В этом случае ожидается, что весь избыток тепловой энергии, а такой будет порядка 600 Вт поверх охлаждаемых 585 Вт, должен пойти на нагревание. Для велосипедиста массой 70 кг это означает повышение температура тела на 2 градуса в течении 10 минут.

Ни у кого нет желания потренироваться при температуре тела 39 С. Даже при низких скоростях, поток создаваемого воздуха гарантирует охлаждение, которое превышает тепловое рассеивание велосипедиста. Поэтому, рекомендуется использовать вместе с тренажером вентилятор.

Невзирая на неадекватный, с точки зрения физики, воздушный поток, 90 Вт вентилятор создает впечатляющий эффект обдувания. Это вентилятор, если и не остужает, то хотя бы добавляет ощущение комфорта на велотренажере - лужа пота под станком никуда не девается.

Таблица 3 – Сравнение электровентиляторов разных производителей

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Наименование | Уровень шума, дБ | Обдуваемая мощность, Вт | Диаметр, см | Стоимость, руб | Производительность, м3/ч |
| Electrolux EFF-1004i | 48 | 55 | 40 | 3 990 | 3000 |
| AEG VL 5606 WM | 25 | 100 | 40 | 3 290 |  |
| Rix NPSF-8000 |  | 90 | 45 | 3 190 |  |
| Midea FS4543 |  | 100 | 45 | 2 990 |  |

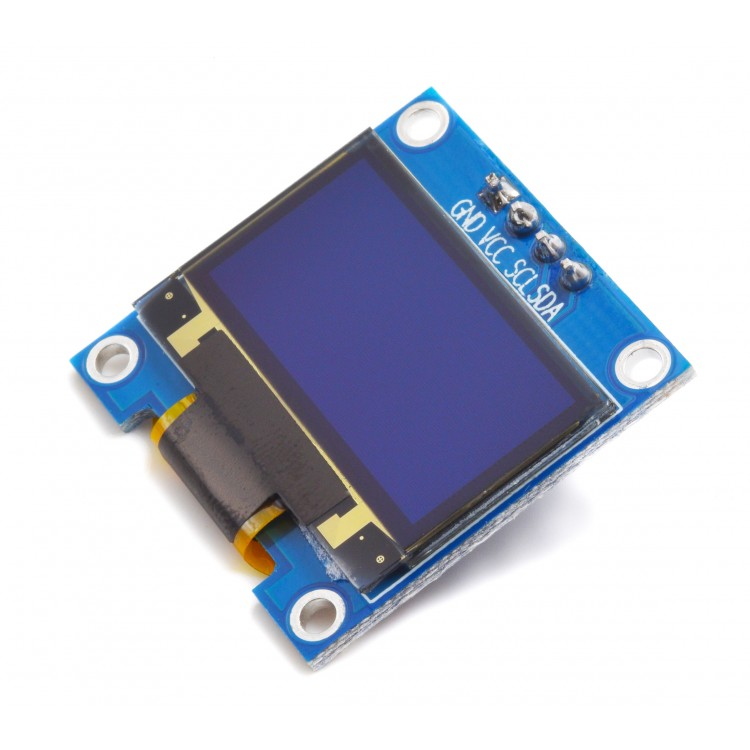


Рисунок xx – Дисплей SSD1315

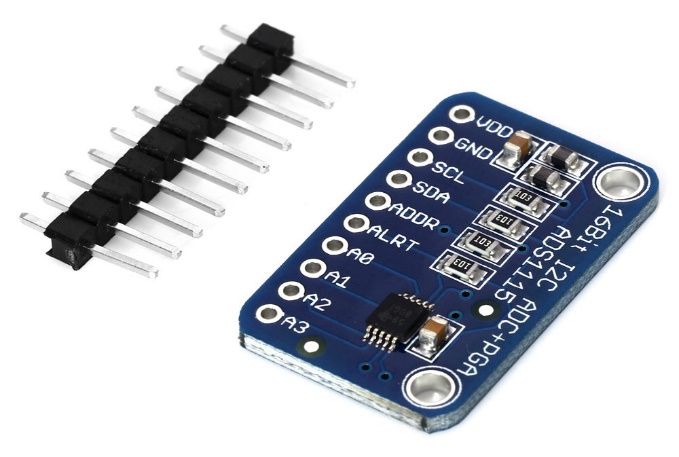


Рисунок xx – Аналогово-цифровой преобразователь ADS1115

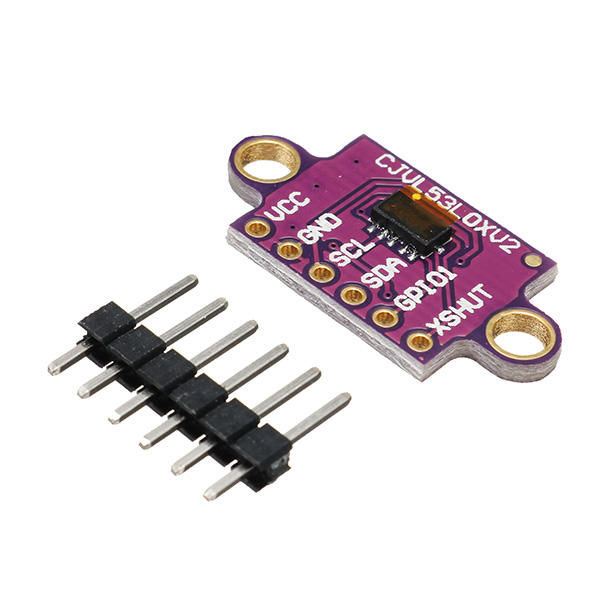


Рисунок xx – Модуль расстояний VL53L0X

**1.6. ВЫБОР ВИДЕОМОНИРОРА**

Виртуальная реальность обещает самые разные вещи, но одно она, безусловно, может сделать эти скучные тренировки в помещении на велосипеде немного более увлекательными.

Погружение в виртуальную реальность (VR) - это ощущение физического присутствия в нефизическом мире. Восприятие создается путем окружения пользователя системы VR изображениями, звуками или другими стимулами, которые создают захватывающую общую среду.

**Oculus rift,**

Основное отличие — есть провод. За счёт подключения к ПК можно использовать шлем в любых VR-играх с отличным качеством картинки. Контроллеры в комплекте те же, что и у Quest — маленькие и удобные, могут долго работать без замены батареек.

Этот шлем — один из наиболее лёгких и удобных в эксплуатации, однако у него физически нельзя отрегулировать расстояние между глазами — только через программу.

Плюсы: приемлемая цена, удобство и простота использования, хороший внутренний inside-out-трекинг, для которого не нужны базовые станции.

Минусы: нельзя отрегулировать расстояние между глазами. Нет официального российского представительства и сервиса в РФ.

**Sony HMZ-T1**

Это не шлем виртуальной реальности, как может показаться с первого взгляда, но и не простые 3D-очки – возможно правильнее было бы назвать его «стерео-шлем». Стереоэффект достигается за счет использования двух OLED-дисплеев для каждого глаза в отдельности. При такой схеме нет никаких перекрестных наложений, всякого рода помех и искажений картинки. Картинка должна быть идеально четкой. Идея такой компоновки далеко не нова. Ей десяток-другой лет. Но вот в таком качестве, серийном производстве и началом мировых продаж всех опередила Sony.

**Silico MicroDisplay ST1080**

Компания Silicon Micro Display разработала портативный дисплей в виде очков. Модель ST1080 поддерживает контент в 2D- и 3D-формате с разрешением Full HD 1080p. Пользователь видит изображение эквивалентное экрану с диагональю 100 дюймов, удаленном на расстояние 3 м. Кроме того, очки обладают прозрачностью 10%, что позволяет использовать их в системах дополненной реальности. Девайс получает видеосигнал через HDMI-кабель, а питание – через USB. Вес модели – 180 г. Цена Silicon Micro Display ST1080 составляет $800.

Таблица 4 – Сравнение шлемов виртуальной реальности

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Oculus Rift** | **Sony HMZ-T1** | **Silico MicroDisplay   ST1080** |
| **Разрешение** | 1920×1080 | 1280×720 | 1920×1080 |
| **Тип экрана** | LCD | OLED | LCoS |
| **Разъем для подключения** | DVI/HDMI | HDMI 1.4 | HDMI 1.4 |
| **Способ подачи 3D сигнала** | Отдельные экраны для каждого глаза | Frame Packaging | Frame Packaging |
| **Уровень обзора** | 110 Градусов   по диагонали | 45 градусов по диагонали | 45 градусов по диагонали |
| **Отслеживание движения головы** | Да | Нет | Нет |

**Xiaomi Mi VR + Trinus VR**

Многие люди после использования очков виртуальной реальности могут испытывать головокружение. Чтобы избежать этого, Xiaomi использует сам телефон в качестве аппаратного ускорителя, что позволяет повысить чувствительность в 17 раз. Из-за того, что даже малейшее движение головы улавливается датчиками, достигается полная синхронизация движений головы с видимой виртуальной реальностью. Таким образом можно забыть о головной боли или о каких-либо других неудобствах при использовании очков в течении долгого времени.

TrinusVR - программа для стриминга. Тринус позволяет запускать большинство 3D приложений и игр. На рисунке 6 показаны спосоды подключения смартфона в качестве шлема виртуальной реальности.

Для использования стриминга вам необходимы следующие условия:

• VR гарнитура Xiaomi Mi VR или аналог;

• Первональный компьютер;

• смартфон с наличием гироскопа, акселерометра и магнитометра;

• быстрое и стабильное Wi-Fi соединение;

• Microsoft .NET;

• сервер, установленный на ПК;

• клиент, установленный на смартфон.

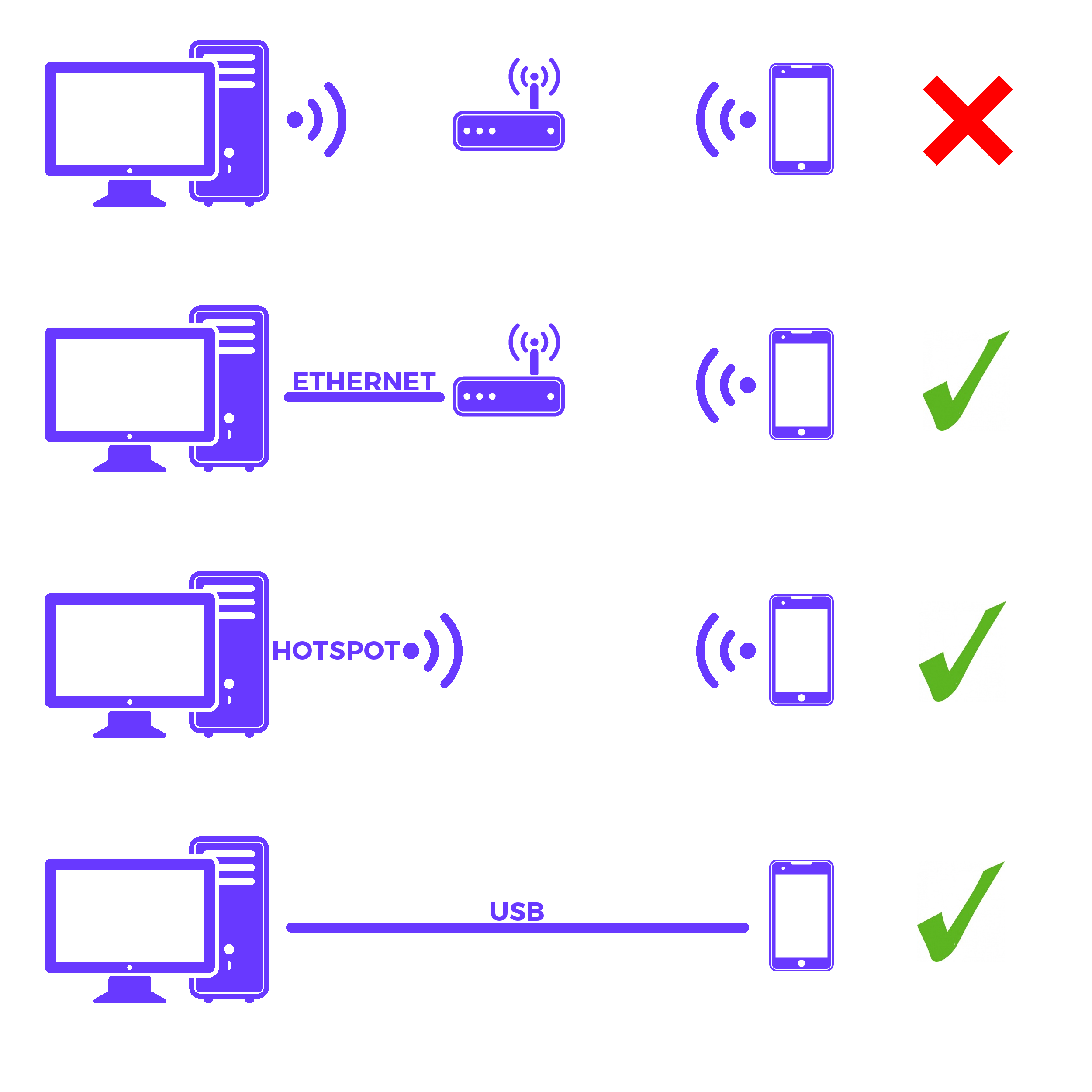


Рисунок 6- Способы сопряжения смартфона с ПК через TrinusVR

**1.6. ВЫБОР ИГРОВОГО ДВИЖКА**

Для реализации программной части комплекса, необходимо выбрать игровой движок, на котором будет сделана игра. Игровой движок – это модуль игры, который включает в себя игровую логику. Процесс разработки приложения сильно облегчается за счёт экономии времени и сил, посредством встроенных инструментов. В настоящее время существует огромное количество таких средств. Для сравнения, были выбраны наиболее популярные.

**Unity3D**

Unity 3D — это мощная среда для разработки 3D игр и приложений. Данная платформа создана в 2005 году. Главный плюс Unity 3D это простота разработки приложений. В данной среде разрабатывается огромное количество игр под различные платформы.

Одним из главных преимуществ использования платформы Unity является ее подробная документация, с описанием всех функциональных возможностей, а также как их правильно применить.



Рисунок 7 - Интерфейс платформы Unity

Основные возможности и плюсы Unity 3D:

– доступный и понятный интерфейс;

– поддержка двух языков программирования: C# и JavaScript, на которых пишутся скрипты;

– большое сообщество;

– поддержка перетягивания объектов в редакторе;

– возможность дополнения функционала;

– возможность использования систем контроля версий.

Минусы:

– ограниченный набор инструментов;

– не самая лучшая графика.

**Unreal Engine**

Unreal Engine 4 – среда разработки, созданная Epic Games. Unreal Engine 4 – самая популярная среда разработки для создания фильмов и ААА-проектов. Данная платформа обладает высокими графическими возможностями. С Unreal Engine 4 есть возможность разрабатывать игры под PC, Mac, консоли, IOS, Android. В отличие от Unity, UE4 имеет мощный инструмент для дизайна игровых уровней прямо в сцене, достаточно удобную систему Blueprint, не имеющую аналогов, красивый дизайн самой платформы и интуитивность в использовании. Из всех сред разработки, Unreal Engine 4 является самым инновационным. Он сочетает в себе высокую производительность, лучшую графику, простой язык программирования и удобность в использовании.

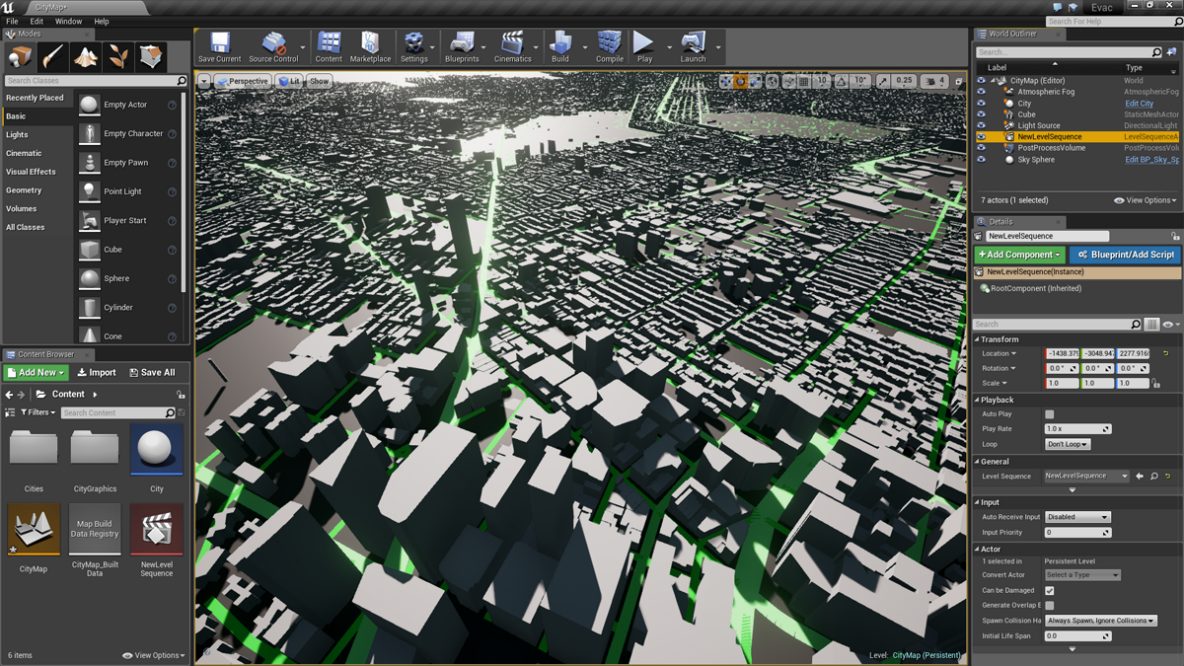


Рисунок 8 - Интерфейс платформы Unreal Engine

Плюсы:

– большое сообщество;

– возможно, напрямую использовать в проекте файлы с исходным кодом на C++;

– широкий ассортимент инструментов для различных целей;

– совместим с различными платформами.

Минусы:

– сложно привыкнуть к определенным инструментам;

– небольшой выбор готовых инструментов в официальном магазине;

При выборе средств разработки наиболее важными критериями были:

– порог вхождения;

– поддерживаемые платформы и используемые языки программирования;

– цена;

– исходный код.

Данные игровые движки схожи по функционалу, они бесплатны, имеют

хорошую документацию и поддержку, но среда Unity имеет менее сложный язык проектирования. Таким образом, был выбран игровой движок Unity3D.

**1.7. ВЫВОДЫ ПО ГЛАВЕ 1**

В ходе анализа предметной области был проведен обзор литературы, которая поможет в разработке работы.

Были рассмотрены проекты, схожие по назначению к разрабатываемую комплексу. Выявлены их достоинства и недостатки.

Были рассмотрены устройства сбора и передачи данных. Сравнивая

Raspberry Pi Zero, ESP32-WROVER-E Espressif, выбор остановился на Raspberry Pi Zero, так как по необходимым критериям, а это, небольшая стоимость устройства, небольшой размер, достаточная мощность для считывания всех сигналов без задержки и возможность подключения по Bluetooth BLE. Поэтому он подходит больше.

Для считывания данных о движениях пользователя, был выбран датчик Холла KY-003, чтобы определять количество оборотов, сделанных пользователем, а также кнопки, чтобы передавать данные о сделанном повороте вправо или влево.

Для теплового рассеивания велосипедиста в аппаратный комплекс рекомендуется использовать вместе с тренажером вентилятор. Из предложенных вариантов выбран Bionaire BAC 14 из-за оптимальных характеристик мощности и шума за свою стоимость.

Были рассмотрены средства разработки игры, а именно Unity и Unreal Engine 4, они понятны для использования, в них схожие возможности и они бесплатны, что очень важно для программистов. В качестве платформы для разработки игры будет использоваться Unity, так как был небольшой опыт использования данного игрового движка, и он подходит для достижения поставленной цели

**2. ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ К КОМПЛЕКСУ**

Для реализации данной системы необходим следующий набор подсистем:

– приложение на Microsoft Windows. Приложение обеспечивает пользователю симуляцию передвижения на велосипеде;

– графический интерфейс приложения;

– аппаратная реализация системы. Устройство, позволяющее передавать данные о движениях пользователя непосредственно в приложение;

– велотренажер, крепления которого, позволят установить устройство.

2.1. ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

– считывание устройством данных о движениях пользователя;

– передача данных устройством о количестве оборотов, сделанных пользователем;

– передача данных устройством о нажатых кнопок направления движения;

– обработка данных, переданных устройством на ПК;

– воспроизведение действий пользователя на виртуальной модели в приложении.

2.2. НЕФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

2.2.1 ТРЕБОВАНИЯ АППАРАТНОЙ ЧАСТИ КОМПЛЕКСА

– задержка передачи данных от устройства должна быть минимальной (не более 200мс);

– размеры устройства должны превышать следующих параметров: 20\*35\*35мм;

– вес устройства не должен превышать 500 грамм.

2.2.2 ТРЕБОВАНИЯ ПРОГРАММНОЙ ЧАСТИ КОМПЛЕКСА

– обработка данных для воспроизведения в приложении не должна быть заметна пользователю;

– изменение пользователем настроек разрешении экрана в приложении;

– возможность начать игру;

– изменение уровня громкости в настройках игры;

– возможность зайти в настройки из начатой игры.

2.2.3 ТРЕБОВАНИЯ К ЛИНГВИСТИЧЕСКОМУ ОБЕСПЕЧЕНИЮ

Пользовательский интерфейс должен быть на русском языке.

2.2.4 ТРЕБОВАНИЯ К ДОКУМЕНТАЦИИ

В документации на устройство должны содержаться технические характеристики устройства, которые включают следующие требования: вес, размеры, а также инструкцию для пользователя, как правильно установить систему.

**3. ПРОЕКТИРОВАНИЕ**

Проектирование – процесс определения архитектуры, компонентов, интерфейсов и других характеристик системы или её части. Результатом проектирования является проект – целостная совокупность моделей, свойств или характеристик, описанных в форме, пригодной для последующей реализации.

**3.1. ФУНКЦИОНАЛЬНЫЙ СОСТАВ**

Для описания функционального состава системы можно представить функциональную схему. Данная схема поясняет отдельные виды процессов, протекающих в целостных функциональных блоках. На рисунке XX представлена функциональная схема системы.



Рисунок XX — Функциональная схема системы

Из функциональной схемы видно, что установленные на велотренажер датчики, взаимодействуют с микроконтроллером, который будет передавать сигналы от них, непосредственно на ПК.

Приложение получает эти данные от персонального компьютера и симулирует на виртуальной модели сигналы, полученные от пользователя.

<…>

**3.2. ОБЩАЯ СТРУКТУРА АППАРТАНОЙ ЧАСТИ**

Аппаратное обеспечение состоит из различных электронных элементов. В его состав входят следующие модули:

– питание информационной части от ПК;

– датчики Холла, поворотов направо и налево;

– обмен информацией с системой.

Каждый из которых обладает определенным функционалом, благодаря которому программно-аппаратный комплекс сможет действовать как одно целое. На рисунке XX представлена структурная схема.



Рисунок XX – Схема аппаратного комплекса

Рисунок xx, где

ПК – персональный компьютер;

МК – микроконтроллер;

АКБ – аккумуляторная батарея;

И – инвертор (DC-AC);

Н – нагрузка;

С – сенсоры;

БП – блок преобразования (регулятор, ШИМ, БУ);

ИО – исполнительный орган (двигатель + редуктор);

V – показатель скорости;

M – показатель момента;

Напр – показатель направления;

Uзад – напряжение задания;

Uупр – напряжение управления;

ДТ – датчик тока;

ДС – датчик скорости;

ω – скорость.



Рисунок XX – Структурная схема велотренажера

Ризунок xx, где

ЗИ – задатчик интенсивности;

РС – регулятор скорости (пропорциональный);

РТ – регулятор тока (пропорциональный);

Косс – коэффициент обратной связи по скорости;

Кост – коэффициент обратной связи по току;

ДТ – датчик тока;

ТГ – тахогенератор;

М – двигатель постоянного тока, параллельного возбуждения;

БУ – блок управления на IBGT транзисторах.



Рисунок XX – Функциональная схема велотренажера



Рисунок XX – Концепция велотренажера

<…>

**3.3. СТРУКТУРНЫЙ СОСТАВ ПРОГРАММНОЙ ЧАСТИ**



Рисунок XX – Структурная схема приложения

Структурная схема приложения включает в себя внутренние и внешние компоненты. Состав внешних компонентов: графический интерфейс приложения. Во внутренние компоненты входят: модуль настроек пользователя, прогресс пользователя и модуль обработки сигналов от контроллера.

**3.4. ДИАГРАММА ПРЕЦЕДЕНТОВ ПРИЛОЖЕНИЯ**



Рисунок XX – Диаграмма прецедентов главное меню

На основе данных потребностей выделены следующие варианты

использования:

– играть;

– игровой прогресс;

– настройки;

– выход;

– вернуться (в предыдущее меню);

– игра;

– управление;

– графика;

– звук;

Рассмотрим более подробно каждый из этих прецедентов.

Таблица XX -

|  |  |
| --- | --- |
| Прецедент | Краткое описание |
| играть | Позволяет пользователю начать управлять велосипедом посредством  подключенного к нему велотренажера. |
| игровой прогресс | Позволяет пользователю перейти в подменю прогресса игры |
| настройки | Позволяет пользователю перейти в меню настроек. Этот вариант  использования начинается, когда игрок нажимает на кнопку паузы. |
| выход | Позволяет пользователю выйти из приложения. |
| звук | Позволяет пользователю настроить параметры  «общая громкость»,  «громкость музыки»,  «громкость эффектов», |
| графика | Позволяет пользователю перейти в подраздел настройки «графика». |
| игра | Позволяет пользователю перейти в подраздел настройки «игра». |
| управление | Позволяет пользователю перейти в подраздел настройки «управление» |
| общий прогресс | Позволяет пользователю просмотреть данные о своем общем прогрессе. |
| вернуться | Позволяет пользователю выйти из меню настроек и вернуться в главное  меню. Этот вариант использования начинается, когда игрок нажимает кнопку  «Назад». |

3.4 КАЛИБРОВКА ТРЕНАЖЕРА

Для передачи корректных показаний тренажера с установленным велосипедом необходимо производить калибровку аппаратного комплекса под действующий велосипед.

В зависимости от диаметра колеса человек может иметь разные скоростные характеристики. Перед каждым запуском тренажера датчик расстояния будет измерять расстояние от себя до камеры колеса.

Корпус тренажера и положение датчика расстояния неизменимое и расстояние от центра установки колеса до датчика равно 381 мм (15”).

При условии расстояния от датчика до покрышки колеса, равным 50,8 мм (2”) диаметр колеса вычисляется таким образом:

Диаметр колеса=(381 мм – 50,8)\*2=660,4 (26”)

В последствии из тахогенератора и диаметра колеса тренажер передаёт достоверную скорость велосипеда с используемыми размерами колёс. Вычисление скорости велосипеда вычисляется следующим образом:

Скорость [км/ч] = ((скорость вращения колеса[об.мин]/60)\*π\*(диаметр колеса[мм]/1000000))

3.5 ВЫВОДЫ ПО ГЛАВЕ 3

В ходе проектирования комплекса был описан его функциональный состав, показаны структуры аппаратной и программных частей, а также разработана UML-диаграмма вариантов использования из главного меню.

4 РЕАЛИЗАЦИЯ

Для разработки игрового приложения была выбрана платформа Unity. Данная платформа имеет низкий порог вхождения, большое количество обучающих материалов и сообщество разработчиков, в следствии чего с ней можно быстрее начать работать. Для создания и редактирования графической составляющей игры использовался графический редактор Adobe Photoshop.

Для разработки 3D-моделей и анимации использовался Blender 3D.

Для программирования микроконтроллера использовался язык Python в Thonny IDE.

4.1 РЕАЛИЗАЦИЯ ПРОГРАММНОЙ ЧАСТИ

4.1.1 ФАЙЛОВАЯ СТРУКТУРА ПРИЛОЖЕНИЯ

Разработанный проект содержит каталоги, в которых хранятся:

– 3D-модели;

– игровые сцены;

– скрипты;

– музыка;

– шрифты;

– шаблоны объектов;

– анимации объектов.

Файловая структура представлена на рисунке XX.



Рисунок XX – Файловая структура

В папке «Animations» находятся анимации всех объектов игры. В папке «Audio» находятся вся музыка и звуки для игры. В директории «Prefabs» находятся готовые шаблоны игровых объектов. Директория «Scene» содержит сцену игры, в которой происходят все действия. В папке «Scripts» находятся скрипты с описанием всех классов и взаимодействий.

В каталоге «Scripts» содержатся три подкаталога:

– «Animations» содержит в себе скрипты, управляющие анимацией в игре;

– «Assistants» содержит в себе скрипты-помощники, которые управляют логикой всей игры;

– «UI» Содержит в себе скрипты, управляющие графическим интерфейсом;

В каталоге «3D models» содержатся три подкаталога:

– «Bicycle» содержит в себе 3D-модели различных велосипедов;

– «Buildings» содержит в себе 3D-модели всех зданий присутствующих в игре;

– «Sceney» содержит в себе 3D-модели всех второстепенных объектов, таких как столбы, скамейки, деревья и т.д.

4.1.2 РЕАЛИЗАЦИЯ ИНТЕРФЕЙСОВ

В ходе создания игры для взаимодействия пользователя с программой было создано главное меню, показное на рисунке xx, которое включает в себя следующие пункты:

– игра;

– прогресс игры;

– настройки;

– выход.

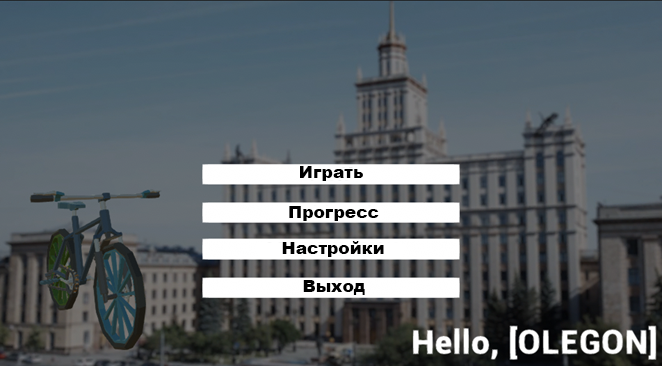


Рисунок XX - Главное меню

После нажатия на пункт настройки, пользователь попадает в подменю, в которым можно выбрать настройки игры, управления и графики. Они представлены на рисунках XX.

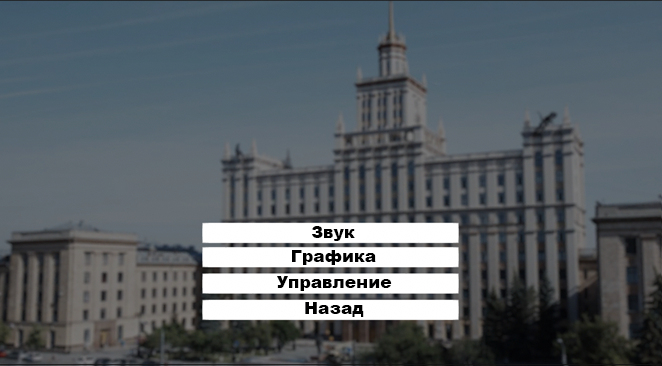


Рисунок XX – Настройки игры

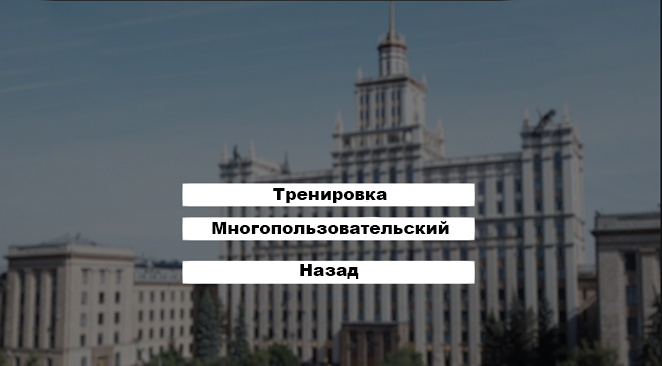


Рисунок XX – Режимы игры



Рисунок XX – Настройки управления

При выборе пункта прогресса игры, пользователь может увидеть свои общие результаты, а также результаты за последнюю игровую сессию. Они представлены на рисунках XX и XX .



Рисунок XX – Общий прогресс игры

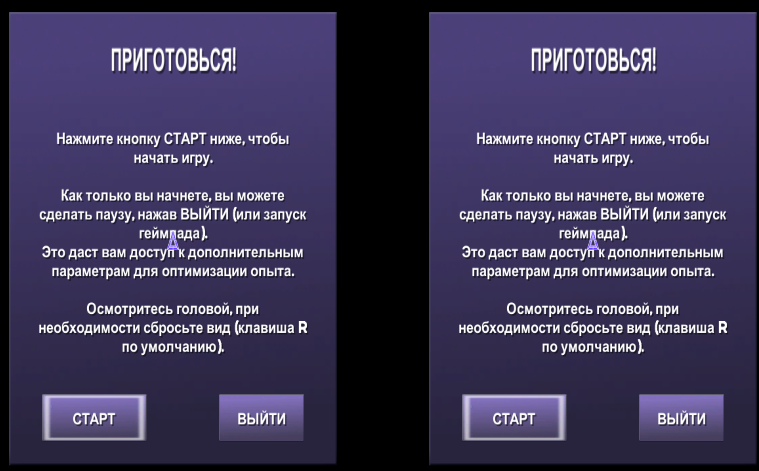


Рисунок XX – Включение VR

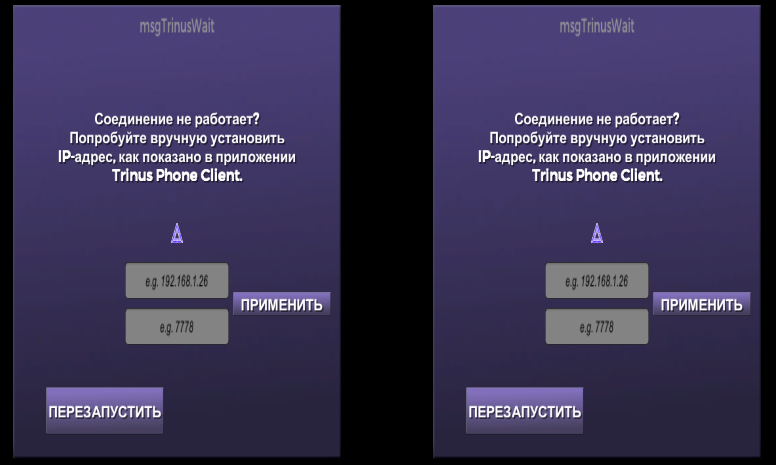


Рисунок XX – Подключение смартфона

На рисунке xx, показано, как после нажатия на пункт игра, перед пользователем появляется экран загрузки, после которого, при нажатии любой из кнопок, он перейдет непосредственно к игре.

При запуске игры, пользователь видит перед собой город, модель велосипеда, которым может управлять через подключенный велотренажер посредством USB интерфейса. Также, в левом нижнем углу изображен велокомпьютер, который показывает системное время, скорость в км/ч и расстояние в километрах. Процессы изменения этих показателей отображены на рисунках xx и xx.



Рисунок XX – Тренировка в VR режиме

4.2 РЕАЛИЗАЦИЯ АППАРАТНОЙ ЧАСТИ

Для проверки работоспособности устройства, был собран опытный образец. Его реализация демонстрируется на рисунках xx и xx.

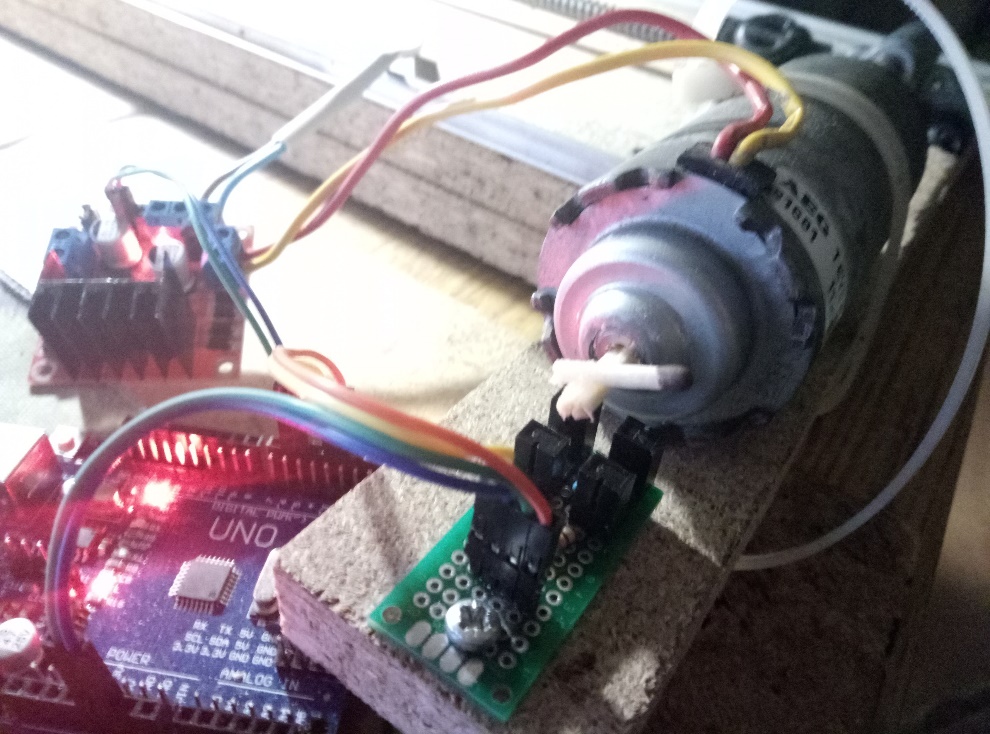


Рисунок xx – Тахогенератор

После прокрутки педали, на котором закреплен магнит, датчик Холла передает сигнал в игру для управления виртуальной моделью велосипеда. Чем больше скорость прокрутки педалей, тем быстрее начинает двигаться модель.

Для управления поворотами, были подключены кнопки, они прикреплены к рулю велотренажера, это показано на рисунке xx.

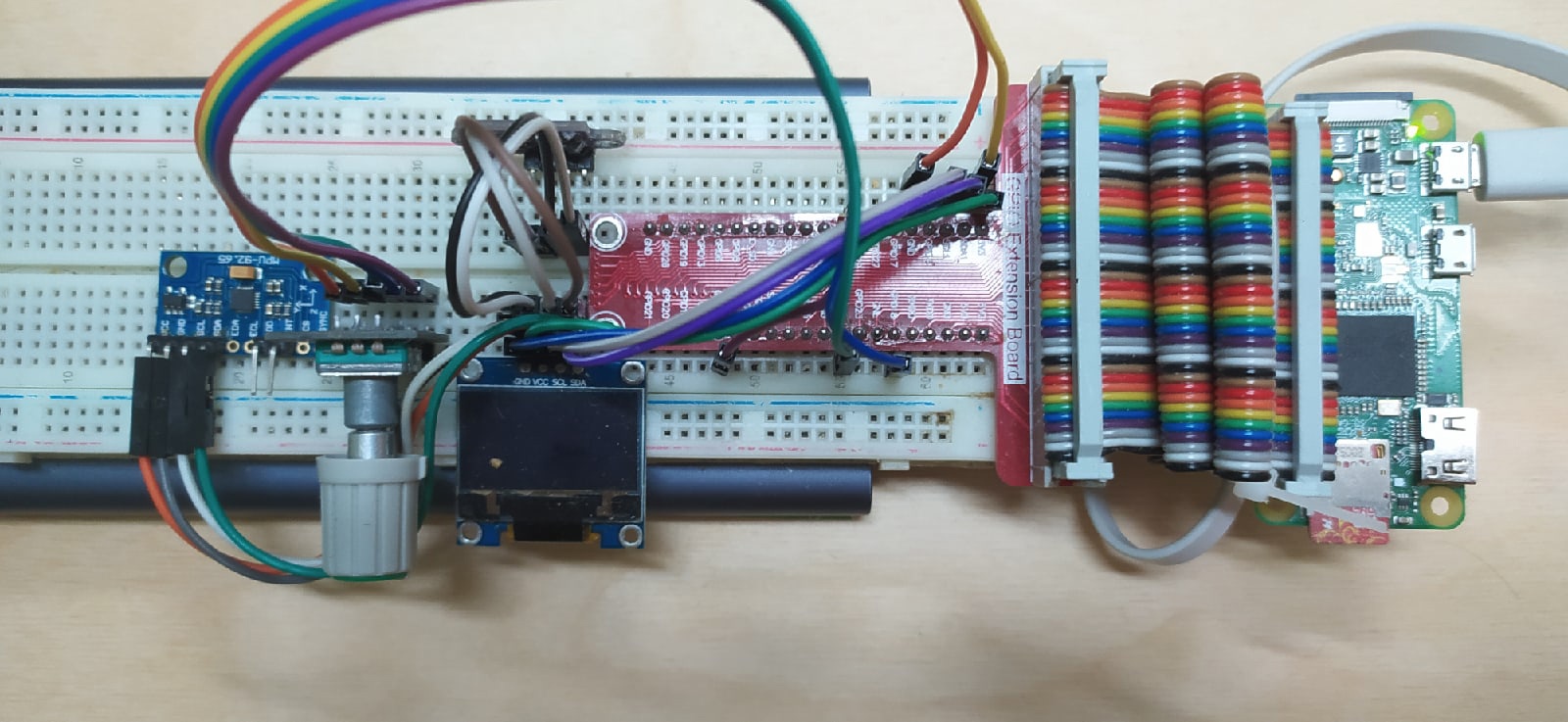


Рисунок xx – Гироскоп

4.3 ВЫВОДЫ ПО ГЛАВЕ 4

Спроектированные ранее устройство и игра, были успешно реализованы. Выполнена возможность управления велотренажером в разработанной компьютерной игре. Смоделированы модели города, внутриигровых объектов, а также велосипеда. Для проверки корректной работы игры, необходимо тестирование.

5 ТЕСТИРОВАНИЕ

5.1 ФУНКЦИОНАЛЬНОЕ ТЕСТИРОВАНИЕ

При тестировании приложения использовался метод функционального тестирования.

Функциональное тестирование – это один из видов тестирования, направленного на проверку соответствий функциональных требований ПО к его реальным характеристикам.

Основной его задачей является подтверждение того, что разрабатываемый программный продукт обладает всем функционалом, требуемым заказчиком.

5.2 ПРОВЕДЕНИЕ ПРОЦЕДУРЫ ТЕСТИРОВАНИЕ

Результаты тестирования:

Тест №1. Передвижение пользователя в игре.

Входные данные: пользователь крутит педали на велотренажере.

Ожидаемый результат: модель велосипеда в игре начнет движение вперед.

Полученный результат: совпадает с ожидаемым.

Тест пройден успешно.

Тест №2. Пользователь нажимает кнопки поворота на велотренажере.

Входные данные: пользователь нажимает кнопки поворота.

Ожидаемый результат: модель велосипеда в игре поворачивает в нужное направление.

Полученный результат: совпадает с ожидаемым.

Тест пройден успешно.

Тест №3. Выход из игры.

Входные данные: пользователь находится в главном меню.

Ожидаемый результат: пользователь нажимает кнопку выхода из игры, и она закрывается.

Полученный результат: совпадает с ожидаемым.

Тест пройден успешно.

Тест №4. Изменение громкости.

Входные данные: пользователь находится в настройках игры.

Ожидаемый результат: пользователь перемещает ползунок громкости и громкость изменяется.

Полученный результат: совпадает с ожидаемым.

Тест пройден успешно.

Тест №5. Переход игры в оконный режим.

Входные данные: пользователь находится в настройках графики.

Ожидаемый результат: пользователь нажимает кнопку «в окне» и игра переходит в такой режим.

Полученный результат: совпадает с ожидаемым.

Тест пройден успешно.

Тест №6. Нажатие одновременно двух кнопок поворота.

Входные данные: пользователь нажимает две кнопки управления одновременно.

Ожидаемый результат: модель велосипеда не поворачивает.

Полученный результат: совпадает с ожидаемым.

Тест пройден успешно.

Тест №7. Навигация в игре.

Входные данные: пользователь находится в режиме игры.

Ожидаемый результат: пользователь нажимает кнопку паузы и появляется окно меню.

Полученный результат: совпадает с ожидаемым.

Тест пройден успешно.

Тест №8. Обновление данных о прогрессе игры.

Входные данные: пользователь находится в игре.

Ожидаемый результат: после преодоления некоторого расстояния, а затем последующего перехода в статистику «прогресса игры», данные, отображенные

там, станут другими.

Полученный результат: совпадает с ожидаемым.

Тест пройден успешно.

5.3 ВЫВОДЫ ПО ГЛАВЕ 5

Было проведено функциональное тестирование, чтобы убедиться в правильной работе компонентов приложения, а также правильной работы аппаратной части. Для каждого теста был описан ожидаемый

результат и шаги тестирования. Все проводимые тесты были успешно пройдены.

ЗАКЛЮЧЕНИЕ

В рамках дипломной работы был спроектирован и разработан программно-аппаратный комплекс для интерактивного велотренажера.

Программная часть была реализована на платформе Unity, аппаратная часть была основана на микроконтроллере digispark ATtiny85.

Для достижения этих целей, нужно было выполнить следующие задачи:

– провести обзор литературы необходимой для решаемых задач;

– провести анализ и выбор средств реализации устройства управления;

– провести анализ и выбор средств реализации игры;

– спроектировать аппаратную систему;

– спроектировать программную систему;

– реализовать программно-аппаратный комплекс;

– провести тестирование программно-аппаратного комплекса.

В результате выпускной квалификационной работы был разработан комплекс, с помощью которого, пользователь может модернизировать свой велотренажер и заниматься на нем играя в приложение.

В дальнейшем, приложение может быть расширено, добавив туда онлайн составляющую, а также искусственный интеллект. Для распространения игры и продажи готового устройства будет создан сайт.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Фотиев М.М.Электропривод и электрооборудование металлургических цехов: учеб. пособие для вузов – М.: Металлургия, 1990. – 352 с.

2. Грудев А.П. Технология прокатного производства – М.: Изд-во «Металлургия», 1994. – 326 с.

ПРИЛОЖЕНИЕ А

Листинг кода на Python в PyCharm

from mpu6050 import mpu6050

import time

import pickle

import VL53L0X

import Adafruit\_SSD1306

import RPi.GPIO as GPIO

import socket

import sys, subprocess, struct

import argparse

import asyncio

import sys

import os

import subprocess

from PIL import Image, ImageFont, ImageDraw

print("libs imported")

PATH\_CFG = "./ports.conf"

ROOT\_LOG = '/var/log'

class app():

def \_\_init\_\_(self):

self.UNIT = 0x1

self.bufferSize = 1024

self.serverAddressPort = ('192.168.0.175', 20001)

self.disp = display()

print("display inited")

self.disp.chek\_IP()

self.sock = socket.socket()

self.sock.connect(self.serverAddressPort)

self.restartMotionDetector = True

self.loop = asyncio.get\_event\_loop()

self.dic\_tasks = {}

self.tasks = ()

self.serv\_tasks = ()

self.init\_service\_task()

self.tasks = self.serv\_tasks

print(str("CHislo zadach: ").encode('utf-8'), len(self.tasks))

print("-----------------------zapuskaem planirovshik------------------------------")

self.loop.run\_until\_complete(self.task\_manager())

asyncio.run(self.task\_manager())

def init\_service\_task(self):

"""

funkcia inicializacii zadach

"""

# spisok zadack

tasks = []

#print("dobavlyaem svoyu korutinu")

# dobavlyaem svoyu korutinu

#my\_task = self.loop.create\_task(self.task\_recive\_udp())

#tasks.append(my\_task)

print("dobavlyaem svoyu korutinu")

# dobavlyaem svoyu korutinu

my\_task2 = self.loop.create\_task(self.task\_send\_udp())

tasks.append(my\_task2)

tmp = (t for t in tasks)

self.serv\_tasks = tuple(tmp)

print(self.serv\_tasks)

async def task\_manager(self):

try:

self.gather = asyncio.gather(\*self.tasks)

await self.gather

print("-----------------------zapuskayem vse zadachi------------------------------")

except asyncio.CancelledError:

print('Osnovnaya zadacha otmenena!')

# raise

# raise

async def task\_recive\_udp(self):

while True:

print ("waiting...")

await asyncio.sleep(0.5)

msgFromServer = self.UDPClientSocket.recvfrom(self.bufferSize)

array = pickle.loads(msgFromServer[0])

# address = msgFromServer[1]

msg = "Message from Server:{}".format(array)

print (msg)

if array[0] == "angle":

print(array[1])

await asyncio.sleep(1)

async def task\_send\_udp(self):

print("task\_send\_udp started")

self.laser=\_laser()

cycle\_mm=self.laser.get\_range\_cycles()

print (cycle\_mm/2.54/10,' inch')

self.cycle\_diam=cycle\_mm

self.encoder = \_encoder(self.sock, self.cycle\_diam)

self.gyro = \_gyroscope()

while True:

#print("task\_send\_udp worked")

#print(self.gyro.gyve\_angle())

await asyncio.sleep(1)

class \_encoder():

def \_\_init\_\_(self,sock,diam):

self.speed\_rpm = 0

self.Enc\_A = 18

self.Enc\_B = 12

self.Enc\_SW = 24

GPIO.setwarnings(True)

GPIO.setmode(GPIO.BCM)

GPIO.setup(self.Enc\_A, GPIO.IN)

GPIO.setup(self.Enc\_B, GPIO.IN)

GPIO.setup(self.Enc\_SW, GPIO.IN)

GPIO.add\_event\_detect(self.Enc\_A, GPIO.RISING, callback=self.rotation\_decode, bouncetime=10)

GPIO.add\_event\_detect(self.Enc\_SW, GPIO.FALLING, callback=self.callback\_switch, bouncetime=10)

self.sock=sock

self.diam=diam

def rotation\_decode(self,Enc\_A):

time.sleep(0.002)

Switch\_A = GPIO.input(self.Enc\_A)

Switch\_B = GPIO.input(self.Enc\_B)

if (Switch\_A == 1) and (Switch\_B == 0):

self.speed\_rpm += 1

print("direction -> ", self.speed\_rpm)

self.send\_speed()

while Switch\_B == 0:

Switch\_B = GPIO.input(self.Enc\_B)

while Switch\_B == 1:

Switch\_B = GPIO.input(self.Enc\_B)

elif (Switch\_A == 1) and (Switch\_B == 1):

self.speed\_rpm -= 1

print ("direction <- ", self.speed\_rpm)

self.send\_speed()

while Switch\_A == 1:

Switch\_A = GPIO.input(self.Enc\_A)

def callback\_switch(self,Enc\_SW):

self.speed\_rpm=0

print("direction <- ", self.speed\_rpm)

self.send\_speed()

def send\_speed(self):

speed\_kmh=self.convert\_to\_kmh(self.speed\_rpm)

msg={}

msg['speed']=speed\_kmh

self.sock.send(pickle.dumps(msg))

print(msg,' kmh')

def convert\_to\_kmh(self,rot):

Perimeter\_m=self.diam\*2.54\*3.14

return (Perimeter\_m/1000)\*(self.speed\_rpm/60)

class display():

def \_\_init\_\_(self):

RST = None

self.disp = Adafruit\_SSD1306.SSD1306\_128\_64(rst=RST)

self.disp.begin()

self.disp.clear()

self.disp.display()

self.width = self.disp.width

self.height = self.disp.height

self.image = Image.new('1', (self.width, self.height))

self.draw = ImageDraw.Draw(self.image)

self.draw.rectangle((0, 0, self.width, self.height), outline=0, fill=0)

self.padding = -2

self.top = self.padding

self.bottom = self.height - self.padding

self.x = 0

self.font = ImageFont.load\_default()

print("display inited")

def chek\_IP(self):

self.draw.rectangle((0, 0, self.width, self.height), outline=0, fill=0)

self.cmd = "hostname -I | cut -d\' \' -f1"

IP = subprocess.check\_output(self.cmd, shell=True)

# self.cmd = "top -bn1 | grep load | awk '{printf \"CPU Load: %.2f\", $(NF-2)}'"

self.draw.text((self.x, self.top), "IP: " + str(IP), font=self.font, fill=255)

print("IP: " + str(IP))

self.disp.image(self.image)

self.disp.display()

time.sleep(.1)

class \_gyroscope():

def \_\_init\_\_(self):

self.sensor = mpu6050(0x68)

def gyve\_angle(self):

accelerometer\_data = self.sensor.get\_accel\_data()

#{'x': -6.2201359130859375, 'y': 6.904877587890625, 'z': 3.308786694335937}

return accelerometer\_data['x']

class \_laser():

def \_\_init\_\_(self):

self.height\_trainers\_mm = 35\*2.54\*10/2

#self.com\_height\_mm=0

self.tof = VL53L0X.VL53L0X(i2c\_bus=1, i2c\_address=0x29)

# I2C Address can change before tof.open()

# tof.change\_address(0x32)

self.tof.open()

def get\_range\_cycles(self):

self.com\_height\_mm=0

# Start ranging

self.tof.start\_ranging(VL53L0X.Vl53l0xAccuracyMode.BETTER)

self.timing = self.tof.get\_timing()

print("Timing %d ms" % (self.timing / 1000))

if self.timing < 20000:

self.timing = 20000

for count in range(1, 10):

distance = self.tof.get\_distance()

if distance > 0:

print("%d mm, %d cm, %d" % (distance, (distance / 10), count))

self.com\_height\_mm=self.com\_height\_mm+distance

time.sleep(self.timing / 1000000.00)

self.avg\_distance\_mm = self.com\_height\_mm / (10-1)

print("avd distance :",self.avg\_distance\_mm)

diameter\_mm=(self.height\_trainers\_mm-self.avg\_distance\_mm)\*2

print("diameter :", diameter\_mm,"mm")

self.tof.stop\_ranging()

return diameter\_mm

#self.tof.close()

if \_\_name\_\_ == '\_\_main\_\_':

app()

ПРИЛОЖЕНИЕ Б

Листинг кода на С# в Unity

TrinusCamera.cs

using UnityEngine;

using System.Collections;

using System.Collections.Generic;

using System;

using System.Reflection;

namespace trinus{

public class TrinusCamera : MonoBehaviour

{

public enum CAMERA\_MODE{

DISABLED,

SINGLE,

DUAL,

UNITY\_VR

}

Camera leftCamera;

Camera rightCamera;

Camera splitCamera;

Camera uiCameraMain;

Camera uiCameraSub;

Vector3 offset;

Vector3 defaultRotation;

public delegate void cameraEnabled(TrinusCamera camera, bool toggle);

cameraEnabled cameraEnabledDelegate;

CAMERA\_MODE activeMode = CAMERA\_MODE.DISABLED;

[Tooltip("The Camera mode determines if the view should be monoscopic or stereosopic (using Unity VR mode or a two camera rig). Refer to the manual for details")]

public CAMERA\_MODE defaultMode = CAMERA\_MODE.UNITY\_VR;

void Awake(){

defaultRotation = new Vector3(transform.localEulerAngles.x , transform.localEulerAngles.y, transform.localEulerAngles.z);

Transform t = transform.Find ("LeftCamera");

if (t != null)

leftCamera = (Camera)t.GetComponent<Camera> ();

t = transform.Find ("RightCamera");

if (t != null)

rightCamera = (Camera)t.GetComponent<Camera> ();

t = transform.Find ("SplitCamera");

if (t != null)

splitCamera = (Camera)t.GetComponent<Camera> ();

if (splitCamera == null && transform.GetComponent<Camera>() != null) {

splitCamera = transform.GetComponent<Camera>();

if (leftCamera == null)

leftCamera = splitCamera;

if (rightCamera == null)

rightCamera = splitCamera;

defaultMode = CAMERA\_MODE.SINGLE;

Debug.Log("TrinusCamera using external camera (" + splitCamera + "). Switching to single camera mode");

}

uiCameraMain = GameObject.Find ("TrinusUICamera").GetComponent<Camera>();

uiCameraSub = uiCameraMain.transform.Find ("Camera").GetComponent<Camera> ();

setMode(defaultMode);

}

public void setFov(int f){

if (activeMode == CAMERA\_MODE.DUAL) {

leftCamera.fieldOfView = f;

rightCamera.fieldOfView = f;

} else

splitCamera.fieldOfView = f;

PlayerPrefs.SetInt ("fov", f);

}

public Vector3 getDefaultRotation(){

return defaultRotation;

}

public Vector3 getOffset(){

return offset;

}

public void setOffset(Vector3 offset){

this.offset = offset;

}

public int getFov(){

return PlayerPrefs.GetInt ("fov", 100);

}

public void setConvergence(float c){

}

public float getConvergence(){

return 0;

}

public void setSeparation(float s){

}

public float getSeparation(){

return 0;

}

public Camera getMainCamera(){

if (activeMode == CAMERA\_MODE.DUAL)

return leftCamera;

return splitCamera;

}

public Camera[] getMainDualCamera(){

return new Camera[]{ leftCamera, rightCamera };

}

public Camera getUICamera(){

return uiCameraMain;

}

public CAMERA\_MODE getMode(){

return activeMode;

}

public void setDefaultMode(){

setMode (defaultMode);

}

public void setMode(CAMERA\_MODE mode){

if (mode == activeMode)

return;

activeMode = mode;

switch (mode) {

case CAMERA\_MODE.DISABLED:

leftCamera.gameObject.SetActive(false);

rightCamera.gameObject.SetActive(false);

splitCamera.gameObject.SetActive(false);

break;

case CAMERA\_MODE.SINGLE:

leftCamera.gameObject.SetActive(false);

rightCamera.gameObject.SetActive(false);

splitCamera.gameObject.SetActive(true);

UnityEngine.XR.XRSettings.enabled = false;

uiCameraSub.gameObject.SetActive (false);

uiCameraMain.rect = new Rect (0, 0, 1, 1);

break;

case CAMERA\_MODE.UNITY\_VR:

leftCamera.gameObject.SetActive (false);

rightCamera.gameObject.SetActive (false);

splitCamera.gameObject.SetActive (true);

UnityEngine.XR.XRSettings.LoadDeviceByName("GoogleVR");

UnityEngine.XR.XRSettings.enabled = true;

uiCameraSub.gameObject.SetActive (false);

uiCameraMain.rect = new Rect (0, 0, 1, 1);

break;

case CAMERA\_MODE.DUAL:

leftCamera.gameObject.SetActive (true);

rightCamera.gameObject.SetActive (true);

if (splitCamera != leftCamera)

splitCamera.gameObject.SetActive (false);

uiCameraSub.gameObject.SetActive (true);

uiCameraSub.gameObject.SetActive (true);

uiCameraSub.rect = new Rect (0.5f, 0, 0.5f, 1);

uiCameraMain.rect = new Rect (0, 0, 0.5f, 1);

break;

}

setFov (getFov ());

}

public void setCameraEnabledDelegate(cameraEnabled ce){

cameraEnabledDelegate = ce;

}

public void OnEnable(){

enabledCamera ();

}

public void enabledCamera(){

if (cameraEnabledDelegate != null)

cameraEnabledDelegate (this, false);

}

public void syncCameras(){

if (leftCamera != null & rightCamera != null) {

syncComponents (leftCamera.gameObject, rightCamera.gameObject);

syncComponents (rightCamera.gameObject, leftCamera.gameObject);

}

}

private void syncComponents(GameObject source, GameObject dest){

List<Component> list = new List<Component>();

source.GetComponents (list);

foreach(Component component in list){

copyComponent(dest, component);

}

}

private static void copyComponent<T>(GameObject gameObject, T other) where T : Component

{

Type type = other.GetType();

if (gameObject.GetComponent(type) != null)

return;//already exists

Component newComponent = gameObject.AddComponent (type);

if (other is Behaviour)

((Behaviour)newComponent).enabled = (other as Behaviour).enabled;

BindingFlags flags = BindingFlags.Public | BindingFlags.NonPublic | BindingFlags.Instance | BindingFlags.Default | BindingFlags.DeclaredOnly;

PropertyInfo[] pinfos = type.GetProperties(flags);

foreach (PropertyInfo pinfo in pinfos) {

if (pinfo.CanWrite) {

try {

pinfo.SetValue(newComponent, pinfo.GetValue(other, null), null);

}

catch { }

}

}

FieldInfo[] finfos = type.GetFields(flags);

foreach (FieldInfo finfo in finfos) {

finfo.SetValue(newComponent, finfo.GetValue(other));

}

}

}

}