

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Московский институт электроники и математики

Козьмин Андрей Викторович, группа БИВ247

Корсаев Артемий Батаевич, группа БИВ247

**БУДИЛЬНИК
С ТЕХНОЛОГИЕЙ РАСПОЗНАВАНИЯ ПОЗЫ ЧЕЛОВЕКА**

Междисциплинарная курсовая работа
по направлению 09.03.01 Информатика и вычислительная техника
студентов образовательной программы бакалавриата
«Информатика и вычислительная техника»

Студент _____
подпись И.О. Фамилия

Студент _____
подпись И.О. Фамилия

Руководитель
Старший преподаватель

И.О. Фамилия

Москва 202_ г.

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

МОСКОВСКИЙ ИНСТИТУТ ЭЛЕКТРОНИКИ И МАТЕМАТИКИ

ЗАДАНИЕ

на междисциплинарную курсовую работу бакалавра
студенту группы БИВ247 Козьмину Андрею Викторовичу

1. Тема работы

Будильник с технологией распознавания позы человека.

2. Требования к работе.

2.1 Устройство может воспроизводить звуковые сигналы.

2.2 Устройство может отправлять/получать данные по Wi-Fi.

2.3 Устройство может снимать видео.

3. Содержание работы

3.1 Написание программы для отправки данных микроконтроллером по http.

3.2 Написание программы для получения данных с камеры.

3.3 Написание программы для воспроизведения звуковых сигналов.

3.4 Проектирование и разработка устройства (электрическая схема и корпус).

4. Сроки выполнения этапов работы

Первый вариант МКР предоставляется студентом в срок до «___» _____ 202_г.

Итоговый вариант МКР предоставляется студентом в срок до «___» _____ 202_г.

Задание выдано «___» _____ 202_г. _____ А.М. Елисеенко

подпись руководителя

Задание было принято

к исполнению «___» _____ 202_г. _____ А.В. Козьмин

подпись студента

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

МОСКОВСКИЙ ИНСТИТУТ ЭЛЕКТРОНИКИ И МАТЕМАТИКИ

ЗАДАНИЕ

**на междисциплинарную курсовую работу бакалавра
студенту группы БИВ247 Корсаеву Артемию Батаевичу**

1. Тема работы

Будильник с технологией распознавания позы человека.

2. Требования к работе.

2.1 Программа может обрабатывать фотографии.

2.2 Программа может отправлять/получать данные по Wi-Fi.

2.3 Программа может сравнивать фотографии.

3. Содержание работы

3.1 Написание программы загрузки данных о позе.

3.2 Написание программы для передачи данных на устройство.

3.3 Написание программы для сравнения с загруженной позой.

4. Сроки выполнения этапов работы

Первый вариант МКР предоставляется студентом в срок до «___» _____ 202_г.

Итоговый вариант МКР предоставляется студентом в срок до «___» _____ 202_г.

Задание выдано «___» _____ 202_г. _____ А.М. Елисеенко

подпись руководителя

Задание было принято

к исполнению «___» _____ 202_г. _____ А.Б. Корсаев

подпись студента

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

МОСКОВСКИЙ ИНСТИТУТ ЭЛЕКТРОНИКИ И МАТЕМАТИКИ

График выполнения междисциплинарной курсовой работы бакалавра
студента группы Козьмина Андрея Викторовича

Тема работы

Будильник с технологией распознавания позы человека.

Дата согласования первого

варианта МКР

«__» _____ 202_г.

А.М. Елисеенко

подпись руководителя

Дата согласования

итогового варианта МКР

«__» _____ 202_г.

А.В. Козьмин

подпись студента

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

МОСКОВСКИЙ ИНСТИТУТ ЭЛЕКТРОНИКИ И МАТЕМАТИКИ

График выполнения междисциплинарной курсовой работы бакалавра
студента группы Корсаева Артемия Батаевича

Тема работы

Будильник с технологией распознавания позы человека.

Дата согласования первого

варианта МКР

«__» _____ 202_г.

А.М. Елисеенко

подпись руководителя

Дата согласования

итогового варианта МКР

«__» _____ 202_г.

А.Б. Корсаев

подпись студента

Содержание

Введение	8
Актуальность	8
Цель работы	8
Задачи	8
Анализ существующих решений	8
Используемые компоненты	9
Глава I: Проектирование и реализация программного продукта	12
Постановка задачи	12
Проектирование интерфейса и программной архитектуры	12
Выбор технологий	13
Реализуемая архитектура согласно паттернам проектирования	13
Реализация основных функций приложения	13
Реализация функции преобразования изображения человека в вектора-кости.	13
Реализация HTTP сервера	15
Реализация Android приложения	16
Реализация функции подключения устройства к WIFI	16
Реализация BLE на устройстве	17
Реализация синхронизации времени на устройстве с мировым временем	17
Реализация функционала отправки и получения данных по HTTP	17
Реализация функционала работы с камерой	18
Реализация функционала работы с энергонезависимым хранилищем	18
Реализация функционала передачи сообщений по каналам	18
Реализация функционала проигрывания музыки	19
Реализация конвертора midi в формат для устройства	19
Тестирование и отладка	20
Тестирование модели распознавания позы	20
Тестирование Android приложения	20
Тестирование устройства	20
Глава II: Анализ результатов и возможные доработки	21

Оценка полученного результата	21
Дальнейшие шаги по улучшению	21
Возможности масштабирования	21
Проблемы, с которыми мы столкнулись	21
Заключение.....	22
Выводы по результатам работы.....	22
Достижение цели и выполнение задач.....	22
Личный опыт и приобретённые навыки	22
Перспективы продолжения работы.....	22
Список использованных источников	23
Приложения	24
Приложение - Листинг кода.....	24

Введение

Актуальность

Сон является неотъемлемой частью жизни каждого человека. Многие пользуются будильниками и изо дня в день просыпаются под однообразную музыку, что может раздражать или надоедать (рис. 1). В связи с этим возникает необходимость создания устройства, позволяющего разнообразить ежедневную рутину пробуждения. В нашей работе мы разработали будильник, который отключается в момент, когда человек принимает необходимую позу. Это способствует более осознанному и активному пробуждению, а также уменьшает вероятность повторного засыпания.

Цель работы

Создать будильник и необходимое ПО для его функционирования, которые предоставляют следующий функционал: отключение музыки после принятия человеком необходимой позы.

Задачи

1. Разработать приложение для удобного и интуитивного взаимодействия с будильником.
2. Разработать ПО для обработки пользовательских поз.
3. Разработать устройство и написать ПО для его функционирования.

Анализ существующих решений

Проведя анализ открытых источников аналогичных решений найдено, не было. Но были выявлены решения, со схожими идеями:

1. Alarmy – android приложение, которое при срабатывании требует от пользователя совершение какой либо активности: решить математическую задачу, сделать фотографию заданного объекта или же небольшая физическая активность – потрясти телефон / дойти до определённого места.
2. Barcode Alarm Clock и QRAlarm – IOS и Android приложения соответственно, идея которых заключается в том, что для выключения звукового сигнала требуется

просканировать определённый QR код, который пользователь заранее распечатал и поместил в помещении в определённое место.

3. Также существует множество будильников, основанных на инфракрасном датчике. Используя различные устройства – зачастую это пистолет – необходимо попасть инфракрасным лазером в приёмник, после чего будильник выключится.

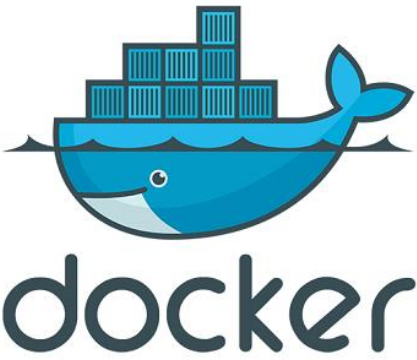



Исходя из полученных данных, можно сделать вывод, что функционал нашего будильника не имеет прямых аналогов, но в то же время уже предпринимались попытки создания нестандартных решений в данной области.

Используемые компоненты

Было принято решение разрабатывать проект, используя компоненты, указанные в таблице 1. Выбор данных компонентов основан в большинстве на личном опыте, а также по другим причинам, которые указаны в таблице.

Таблица 1 - Используемые компоненты.

Название и описание	Причины	Изображение
Микроконтроллер ESP32S3 – необходим для создания самого устройства будильника.	<ol style="list-style-type: none"> 1. Большое количество обучающего материала. 2. Достаточное количество GPIO. 3. Переходник на камеру. 4. Достаточная производительность для многозадачной работы. 	
Android studio – официальная среда разработки приложений на Android.	<ol style="list-style-type: none"> 1. Большое количество встроенных инструментов для разработки. 2. Наличие опыта работы в данной среде. 	
ESP IDF – официальный фреймворк для разработки ПО для микроконтроллеров ESP32.	<ol style="list-style-type: none"> 1. Обширная документация. 2. Большое количество примеров. 3. Наличие всех необходимых инструментов для разработки. 	
FastAPI – фреймворк для разработки RESP API сервисов на python.	<ol style="list-style-type: none"> 1. Высокая производительность. 2. Понятная документация. 3. Большое количество примеров и статей. 4. Асинхронная обработка запросов. 	

Название и описание	Причины	Изображение
Docker – инструмент для контейнеризации приложений.	<ol style="list-style-type: none"> 1. Наличие опыта использования данного инструмента. 2. Обширная документация. 3. Множество обучающих материалов и примеров. 4. Простота развёртывания приложения. 	
OpenCV – мощная библиотека для обработки изображений.	<ol style="list-style-type: none"> 1. Наличие опыта использования этого инструмента. 2. Популярное решение для множества задач. 3. Хорошая производительность. 	
TensorFlow – фреймворк для обучения и использования моделей искусственного интеллекта.	<ol style="list-style-type: none"> 1. Имеется опыт использования данного фреймворка. 2. Предоставляет большой функционал для построения моделей, обрабатывающих изображения. 	
Git – самая популярная система контроля версий.	<ol style="list-style-type: none"> 1. Большая распространённость. 2. Имеется опыт использования этой системы контроля версий 	

Глава I: Проектирование и реализация программного продукта

Постановка задачи

Требуется разработать прототип устройства и сервер. Устройство будет включаться в заданное пользователем время, делать снимки человека и отправлять их на сервер. Сервер будет анализировать изображение на то, в правильной ли позе находится человек. После чего отправлять вердикт на устройство, которое в свою очередь будет отключать звуковой сигнал, если человек принял правильную позу.

Проектирование интерфейса и программной архитектуры

Было принято разделить задачу на две части: сервер и устройство. Как было описано в постановке задачи на сервере изображения будут обрабатываться, а устройство будет реализовывать вердикт сервера. Для решения такой задачи на сервере был реализован HTTP сервер, который получает POST запрос с изображением, после чего прогоняет его через модель MoveNet. На выходе из модели получается набор векторов-костей, которые сравниваются с эталонным и в качестве ответа сервер отправляет пустой ответ с кодом 200, если человек в правильной позе или пустой ответ с кодом 400 если человек в неправильной позе.

Программная архитектура на устройстве выглядит следующим образом. Основная задача была разбита на самостоятельные подзадачи, которые выполняются параллельно и общаются между собой посредством каналов, через которые передаются команды и данные. Для этого было принято решение использовать операционную систему FreeRTOS. Были созданы следующие подзадачи:

1. BLE – задача, отвечающая за взаимодействие с bluetooth low energy.
2. HTTP – задача, отвечающая за взаимодействие с сервером по HTTP протоколу.
3. LOGIC – задача, отвечающая за логику работы устройства. Она отправляет команды другим задачам.

4. CAM – задача, отвечающая за взаимодействие с камерой.

5. MUSIC – задача, отвечающая за проигрывание музыки.

Были так же реализованы отдельные модули, не являющиеся самостоятельными задачами, но необходимые для работы устройства:

1. WIFI – модуль отвечает за подключение устройства к wifi.

2. CHANNELS – модуль предоставляет каналы для общения между задачами.

3. STORAGE – модуль предоставляет удобный функционал для работы с энергонезависимой памятью.

4. SNTP – модуль отвечает за получение реального мирового времени.

Выбор технологий

Используемые технологии и причины их выбора были детально описаны во введении. Так же с ними можно ознакомиться в таблице 1.

Реализуемая архитектура согласно паттернам проектирования

На устройстве была реализована архитектура Orchestration + Event Channel. За оркестрацию отвечает задача LOGIC, и все задачи обмениваются между собой данными и событиями по каналам.

На сервере реализован паттерн Chain of Responsibility. После получения POST запроса сервер прогоняет его через цепочку обработчиков, после чего возвращает итоговый результат.

Реализация основных функций приложения

Реализация функции преобразования изображения человека в векторности.

Одной из важнейших функций обработки изображения является movenet(), которая отвечает за нахождение ключевых точек на изображении.

```

12 model_name = "movenet_lightning"
13 input_size = 256
14 interpreter = tf.lite.Interpreter(model_path="./CV/model.tflite")
15 interpreter.allocate_tensors()
16
17 def movenet(input_image):
18     input_image = tf.cast(input_image, dtype=tf.uint8)
19     input_details = interpreter.get_input_details()
20     output_details = interpreter.get_output_details()
21     interpreter.set_tensor(input_details[0]['index'], input_image.numpy())
22     interpreter.invoke()
23     keypoints_with_scores = interpreter.get_tensor(output_details[0]['index'])
24     return keypoints_with_scores

```

Рисунок 1 листинг функции movenet()

Эта функция используется уже в `get_keypoints()`, которая преобразует изображения в вид, необходимый для функции `movenet()`. Так, в список преобразований входит возможность получения на вход картинки как строки, символизирующей путь к изображению, так и с `numpy`-массивами. Вход преобразуется тип данных `Tensor()`, используемых библиотекой `tensorflow`, а затем проходит масштабирование `Tensor()`, который посылается уже в функцию `movenet()`, результат которой возвращается из `get_keypoints()`.

```

65 def get_keypoints(image, from_npparray=False):
66     """get keypoints on skeleton.
67     Args:
68         image: a detected picture by web camera. Can be a nparray and path to picture/
69         from_npparray: how parse image
70     Returns:
71         keypoints"""
72     if from_npparray:
73         image = tf.convert_to_tensor(image, dtype=np.uint8)
74     else:
75         image_path = image
76         image = tf.io.read_file(image_path)
77         image = tf.convert_to_tensor(image)
78         image = tf.image.decode_jpeg(image)
79
80     input_image = tf.expand_dims(image, axis=0)
81     input_image = tf.image.resize_with_pad(input_image, input_size, input_size)
82     keypoints_with_scores = movenet(input_image)
83     return keypoints_with_scores

```

Рисунок 2 - листинг функции get_keypoints()

Одной из ключевых функций является `keypoints_and_edges_for_display()` (Приложение), которая на вход получает ключевые точки, возвращая информацию для представления позы на изображении в виде линий(костей).

Реализация HTTP сервера

HTTP сервер написан на языке программирования Python с использованием фреймворка FastAPI. Он реализует один эндпоинт, который ожидает POST запрос с изображением. Изображение передаётся в модель, после чего результат обработки изображения сравнивается с эталонным значением, после чего возвращается ответ клиенту – 200, если поза совпала и 400, если поза не совпадает. Эталонные значения получаются с помощью инъекции зависимостей (рис. 3).

```
12 #load pattern image
13 async def get_etalon_edges_with_names():
14     _, _, _, res = keypoints_and_edges_for_display(get_keypoints('template.jpg'), 1280, 720, names=True)
15     return res
16
17
18 #endpoint for conversation with esp32
19 @app.post("/")
20 async def root(request: Request, etalon_edges_with_names: Annotated[dict, Depends(get_etalon_edges_with_names)]):
21     data: bytes = await request.body()
22     print(len(data))
23     image = cv2.imdecode(np.asarray(bytearray(data), dtype=np.uint8), cv2.IMREAD_COLOR)
24
25     _, _, _, edges_with_names = keypoints_and_edges_for_display(get_keypoints(image, from_narray=True), 1280, 720, names=True)
26
27     is_correct_pose = estimate(etalon_edges_with_names, edges_with_names)
28     response = HTTPStatus.OK if is_correct_pose else HTTPStatus.BAD_REQUEST
29     return Response(status_code=response)
```

Рисунок 3 – Реализация эндпоинта и инъекции зависимостей.

Так же для удобства и воспроизводимости запуска сервер был контейнеризирован с использованием Docker (рис. 4).

```
1 FROM ubuntu:24.04
2
3 WORKDIR /project/
4
5 RUN apt update
6 RUN apt upgrade -y
7 RUN apt install -y curl
8 RUN apt install -y ffmpeg
9 RUN apt install -y libsm6
10 RUN apt install -y libgl1
11 RUN apt install -y libxext6
12 RUN curl -LsSf https://astral.sh/uv/install.sh | sh
13
14 COPY ./python-version ./
15 RUN $HOME/.local/bin/uv python install
16
17 COPY ./pyproject.toml ./
18 RUN $HOME/.local/bin/uv sync
19
20 COPY ./template.jpg ./
21 COPY ./server.py ./
22 COPY ./CV/ ./CV/
23
24 CMD $HOME/.local/bin/uv run gunicorn server:app --workers $(nproc) --worker-class uvicorn.workers.UvicornWorker --bind 0.0.0.0:80
```

Рисунок 4 – Dockerfile для HTTP сервера.

Так же для менеджмента зависимостей используется специальный инструмент `uv` от компании `astral-sh`. Он позволяет сохранять конкретные версии используемых библиотек, автоматически управляет виртуальным окружением, а также предоставляет некоторые другие возможности. Все эти возможности в совокупности позволяют удобно работать с виртуальным окружением, зависимостями, версией `python` и т.п. Это в значительной степени ускоряет разработку программного обеспечения.

Реализация Android приложения

Для реализации Android приложения был использован язык программирования `Kotlin`, т.к. `Google` рекомендуют переходить на него вместо `Java`, а также `Kotlin` проще в освоении. Цель Android приложения – конфигурация устройства. Для этого в приложении был реализован функционал для работы с `Bluetooth low energy` (Приложение), т.к. передача данных между устройством и телефоном пользователя происходит именно по этому протоколу.

Алгоритм работы приложения следующий – сначала человек должен используя `Bluetooth` подключиться к устройству. После этого, в приложении появится список всех устройств, которые находятся в памяти смартфона. В этом списке пользователь должен выбрать будильник и произойдёт подключение. После чего можно будет задать название `WIFI` сети и пароль для неё, а также указать время срабатывания устройства.

При установке времени срабатывания устройства или данных для подключения к `WIFI` происходит запись в специальные характеристики `BLE` на устройстве.

Реализация функции подключения устройства к WIFI

Для того, чтобы реализовать функцию подключения `esp32` к `WIFI` была использована официальная документация, в которой прямым текстом говорилось о том, что рекомендуется использовать пример из репозитория. Поэтому в качестве шаблона был использован именно пример из официального репозитория. Далее туда были внесены небольшие правки, а именно при неудачном подключении `esp32` пробует подключаться к `WIFI` до тех пор, пока не получится. А также для авторизации

используются данные, хранящиеся непосредственно в энергонезависимом хранилище, в то время как в примере из документации используются переменные передаваемые на этапе компиляции.

Реализация BLE на устройстве

BLE сервис был реализован на основе примера из официального репозитория esp-idf. В стандартный пример были внесены следующие изменения: параметры BLE характеристик и их количество были сконфигурированы в соответствии с нашими задачами. Во время записи значения в определённую характеристику вызывается определённое событие, исходя из которого значение, записанное в характеристику, записывается под специальный ключ в энергонезависимую память. Например, когда человек вводит название и пароль для WIFI сети на смартфоне, эти значения сначала записываются в соответствующие им характеристики, после чего происходит вызов события записи этих характеристик и данные записываются в энергонезависимую память esp32 под ключами ssid и password соответственно. Аналогичный процесс происходит с установкой времени срабатывания будильника.

Реализация синхронизации времени на устройстве с мировым временем

После того, как устройство успешно подключилось к WIFI, происходит запрос по протоколу sntp на сервер pool.ntp.org. При получении успешного ответа внутреннее время синхронизируется с мировым. При получении ответа с ошибкой происходит повторная попытка получения мирового времени до тех пор, пока время не получится синхронизировать. Для реализации этой задачи был так же использован пример из официального репозитория, который был модифицирован под текущую задачу. А именно был добавлен часовой пояс Москвы. Так же было добавлено логирование, для последующей отладки.

Реализация функционала отправки и получения данных по HTTP

Для реализации данной задачи в качестве шаблона был использован пример из официального репозитория с доработкой под текущую задачу. Были добавлены две константы содержащие IP адрес и порт сервера, на который отправляются фотографии

для проверки позы. Работа с сокетами была оставлена по умолчанию как в примере. Была изменена структура HTTP запроса – тип отправляемых данных (Content-Type: image/jpeg), а в качестве тела добавлены байты, представляющие изображение. HTTP сообщения отправляются при получении данных из канала с изображениями. Результат ответа от сервера отправляется в канал для основной задачи LOGIC.

Реализация функционала работы с камерой

Для реализации функционала получения изображений с камеры используется дополнительный компонент из официального репозитория esp32-camera. Он позволяет работать со множеством камер, в том числе с доступной нам. Так же для этого компонента есть пример, который и был использован в качестве шаблона. Функция получения снимков срабатывает, когда основная задача LOGIC отправляет соответствующий сигнал в специальный канал. После получения снимка он отправляется по каналу в HTTP задачу. Получение снимков происходит до тех пор, пока задача LOGIC не отправит стоп-сигнал.

Реализация функционала работы с энергонезависимым хранилищем

Работа с энергонезависимым хранилищем в esp-idf доступна в двух вариациях, а именно в C стиле и C++ стиле. Для удобства и большей функциональности было принято решение использовать C++ стиль. Были написаны несколько функций обёрток над стандартными, доступными из официального SDK. Для большей функциональности написанные функции были реализованы с использованием шаблонов (Приложение), что позволило сохранять и загружать значения различных типов. Так же было реализовано перечисление (Приложение), содержащее результат вызова той или иной функции.

Реализация функционала передачи сообщений по каналам

Для реализации данного функционала были выбраны очереди из FreeRTOS (Приложение). Такой выбор обусловлен тем, что этот инструмент создан специально под подобные задачи. Такие очереди используют синхронизацию, что не допускает гонки данных и других многопоточных уязвимостей. Так же у таких очередей можно

установить максимальный размер, что предотвратит неконтролируемый расход памяти. В дополнение к вышеперечисленным преимуществам, очереди так же имеют удобное API, позволяя удалять, добавлять, забирать, копировать и т.д. сообщения из очереди. Это позволяет гибко использовать их в различных сценариях.

Реализация функционала проигрывания музыки

Для проигрывания музыки используется пьезоэлектрический излучатель. Данный компонент был выбран из-за простоты в использовании и достаточных возможностей для демонстрации работы прототипа. Воспроизводимая музыка записывается в память устройства ещё на этапе компиляции. Она представляется в виде двух массивов одинакового размера. Первый массив является массивом частот, обозначающий последовательные звуки. Второй массив содержит в себе длительность звучания каждой частоты. Когда задаче MUSIC приходит сигнал о необходимости включить музыку начинается цикл итерации по массиву частот, на каждой итерации которого конфигурируется частота широтно-импульсной модуляции на назначенном выходе микроконтроллера в соответствии с текущим элементом массива. Этот сигнал звучит столько времени, сколько указано во втором массиве (Приложение). Музыка прекращает играть, когда об этом приходит соответствующий сигнал из задачи LOGIC. В случае, когда музыка полностью проигрывается от начала и до конца, она начинает играть заново.

Реализация конвертора midi в формат для устройства

Для удобства была реализована дополнительная программа на языке программирования Python, которая конвертирует midi файл в код, который необходимо использовать в esp32, чтобы проигрывать различные мелодии. Для реализации этого скрипта была использована библиотека mido, которая позволяет загружать midi файлы, а также итерироваться по командам внутри них. Алгоритм работы заключается в том, что сначала в файле находится информация о скорости воспроизведения. Далее программа итерируется по командам midi. Если команда означает включение звука, то с помощью специальной формулы, которая была

выведена на основе проведённых исследований, номер ноты конвертируется в её частоту. Эта частота добавляется в массив, содержащий частоты нот. Когда встречается команда выключения звука или же уменьшение его громкости до 0, происходит запись во второй массив отметки о том, сколько времени длилась эта нота. После такой итерации получается два массива, которые потом конвертируются в код на C++. Так же для оптимизации времени выполнения конвертации используется кэширование функции расчёта частоты ноты из её midi номера. Код программы можно посмотреть в приложении.

Тестирование и отладка

Тестирование модели распознавания позы

Для тестирования модели распознавания позы на начальном этапе была написана программа для её отладки. В качестве камеры, откуда берется поза, используется веб-камера компьютера.

Следующим этапом тестирование модели распознавания позы было использования камеры устройства на базе ESP32S3 вместо веб-камеры компьютера.

Тестирование Android приложения

Для тестирования Android приложения на начальном этапе было использовано логирование. После чего, для тестирования приложения в совокупности с устройством было использовано так же логирование самого устройства, откуда было видно все записи характеристик и т.д.

Тестирование устройства

Для тестирования устройства было использовано логирование. Этот метод позволил показывать результаты выполнения задач в реальном времени, а также исключить неправильное поведение. Так же были опробованы всевозможные варианты использования устройства в совокупности с сервером и Android приложением.

Глава II: Анализ результатов и возможные доработки

Оценка полученного результата

В результате был получен прототип, демонстрирующий реализацию поставленных задач. Данное устройство в полном объёме выполняет заданные требования, что можно считать успешным решением.

Дальнейшие шаги по улучшению

Планируется сделать загрузку пользователем собственного шаблона позы, поддержку распознавания нескольких поз на фотографии, улучшение графического интерфейса, а также улучшить функцию сравнения позы: например, чтобы функция оценивала не просто картинку, а видео длиной в 5 секунд.

Так же в перспективе переход от прототипа к полноценному варианту. Для этого нужно спроектировать корпус, а также использовать динамик вместо пьезоэлектрического излучателя. Так же в корпус будет необходимо добавить источник света, чтобы устройство могло работать даже в тёмное время суток.

Возможности масштабирования

Модель сервера можно оптимизировать для GPU, что позволит обрабатывать больше изображений за тот же промежуток времени.

После разработки данной работы, можно сказать, что переход на другой микроконтроллер, с необходимыми характеристиками будет отличным вариантом масштабирования. Это позволит реализовать дополнительный функционал.

Проблемы, с которыми мы столкнулись

Весь процесс проходил в основном гладко без проблем, но во время реализации функционала проигрывания музыки были выявлены некоторые проблемы. А именно недостаточное количество GPIO микроконтроллера. Это связано с тем, что в устройстве используется камера, которая берёт на себя большое количество входов. Из-за этого возможность подключить micro-SD карту пропала. Это к тому, что музыку для устройства можно загрузить только во время компиляции. Так же были

предприняты несколько попыток о реализации передачи музыки через BLE, но из-за недостаточности знаний такое решение не удалось реализовать.

Так же возникли проблемы с определением функции оценки позы и подбором порогового значения для неё.

Заключение

Выводы по результатам работы

Разработка минимально жизнеспособного продукта проведена успешна. Проект имеет большой потенциал за счёт своих перспектив развития и заложенного фундамента.

Достижение цели и выполнение задач

Все поставленные цели были достигнуты успешным решением поставленных задач.

Личный опыт и приобретённые навыки

Козьмин Андрей дополнил навыки программирования микроконтроллера esp32.

Артемий Корсаев приобрел навыки работы в команде, работа с фреймворком FastAPI, а также разработкой с помощью инструментов tensorflow.

Перспективы продолжения работы

У проекта огромный потенциал. Пока что он находится на стадии прототипа и у него присутствуют недостатки. В перспективе можно избавиться от недочётов и перейти от прототипа к полноценному устройству.

Список использованных источников

- 1 ESP-IDF Examples Repository. URL: <https://github.com/espressif/esp-idf/tree/master/examples> (дата обращения: 06.06.2025).
- 2 ESP-IDF Programming Guide for ESP32-S3. URL: <https://docs.espressif.com/projects/esp-idf/en/stable/esp32s3/index.html> (дата обращения: 06.06.2025).
- 3 FastAPI Documentation. URL: <https://fastapi.tiangolo.com/> (дата обращения: 06.06.2025).
- 4 Matplotlib Documentation. URL: <https://matplotlib.org/stable/index.html> (дата обращения: 06.06.2025).
- 5 Mido: MIDI Objects for Python. URL: <https://mido.readthedocs.io/en/stable/> (дата обращения: 06.06.2025).
- 6 MoveNet: Ultra fast and accurate pose detection. URL: <https://www.tensorflow.org/hub/tutorials/movenet?hl=ru> (дата обращения: 06.06.2025).
- 7 NumPy Documentation. URL: <https://numpy.org/doc/stable/> (дата обращения: 06.06.2025).
- 8 OpenCV Documentation. URL: <https://docs.opencv.org/5.x/> (дата обращения: 06.06.2025).
- 9 TensorFlow Python API. URL: https://www.tensorflow.org/api_docs/python/tf/all_symbols (дата обращения: 06.06.2025).

Приложения

Приложение - Листинг кода

```
57 def keypoints_and_edges_for_display(  
58     keypoints_with_scores,  
59     height,  
60     width,  
61     keypoint_threshold=0.11,  
62     names=False  
63 ):  
64     """Returns high confidence keypoints and edges for visualization.  
65  
66     Args:  
67         keypoints_with_scores: A numpy array with shape [1, 1, 17, 3] representing  
68             the keypoint coordinates and scores returned from the MoveNet model.  
69         height: height of the image in pixels.  
70         width: width of the image in pixels.  
71         keypoint_threshold: minimum confidence score for a keypoint to be  
72             visualized.  
73  
74     Returns:  
75         A (keypoints_xy, edges_xy, edge_colors) containing:  
76             * the coordinates of all keypoints of all detected entities;  
77             * the coordinates of all skeleton edges of all detected entities;  
78             * the colors in which the edges should be plotted.  
79     """  
80     keypoints_all = []  
81     keypoint_edges_all = []  
82     edge_colors = []  
83     num_instances, _, _, _ = keypoints_with_scores.shape  
84     for idx in range(num_instances):  
85         kpts_x = keypoints_with_scores[0, idx, :, 1]  
86         kpts_y = keypoints_with_scores[0, idx, :, 0]  
87         kpts_scores = keypoints_with_scores[0, idx, :, 2]  
88         kpts_absolute_xy = np.stack(  
89             [width * np.array(kpts_x), height * np.array(kpts_y)], axis=-1)  
90         kpts_above_thresh_absolute = kpts_absolute_xy[  
91             kpts_scores > keypoint_threshold, :]  
92         keypoints_all.append(kpts_above_thresh_absolute)  
93  
94         edges_with_names = dict()  
95         for edge_pair, color in KEYPOINT_EDGE_INDS_TO_COLOR.items():  
96             if (kpts_scores[edge_pair[0]] > keypoint_threshold and  
97                 kpts_scores[edge_pair[1]] > keypoint_threshold):  
98                 x_start = kpts_absolute_xy[edge_pair[0], 0]  
99                 y_start = kpts_absolute_xy[edge_pair[0], 1]  
100                 x_end = kpts_absolute_xy[edge_pair[1], 0]  
101                 y_end = kpts_absolute_xy[edge_pair[1], 1]  
102                 line_seg = np.array([[x_start, y_start], [x_end, y_end]])  
103                 keypoint_edges_all.append(line_seg)  
104                 edge_colors.append(color)  
105                 name = f'{{KEYPOINT_DICT_INVERTED[edge_pair[0]]}}-{{KEYPOINT_DICT_INVERTED[edge_pair[1]]}}'  
106                 edges_with_names[name] = line_seg  
107  
108     if keypoints_all:  
109         keypoints_xy = np.concatenate(keypoints_all, axis=0)  
110     else:  
111         keypoints_xy = np.zeros((0, 17, 2))  
112  
113     if keypoint_edges_all:  
114         edges_xy = np.stack(keypoint_edges_all, axis=0)  
115     else:  
116         edges_xy = np.zeros((0, 2, 2))  
117     if names:  
118         return keypoints_xy, edges_xy, edge_colors, edges_with_names  
119     else:  
120         return keypoints_xy, edges_xy, edge_colors
```



```

48 bleDevicesList.clear()
49
50 val bluetoothManager: BluetoothManager = this.getSystemService(Context.BLUETOOTH_SERVICE) as BluetoothManager
51 val bluetoothAdapter: BluetoothAdapter = bluetoothManager.adapter
52
53 if (!bluetoothAdapter.isEnabled) {
54     val requestBluetoothEnableIntent: Intent = Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE)
55     val requestBluetoothEnable = registerForActivityResult(StartActivityForResult()) {
56         if (!bluetoothAdapter.isEnabled) {
57             val message: Toast = Toast.makeText(this, "Please turn on Bluetooth so that the app works correctly", Toast.LENGTH_LONG)
58             message.show()
59             finish()
60         } else {
61             val message: Toast = Toast.makeText(this, "Thanks for turning on Bluetooth", Toast.LENGTH_SHORT)
62             message.show()
63             bleDevicesList.addAll(bluetoothAdapter.bondedDevices)
64         }
65     }
66     requestBluetoothEnable.launch(requestBluetoothEnableIntent)
67 }
68
69 requestPermissionLauncher = registerForActivityResult(RequestPermission()) { isGranted: Boolean ->
70     if (isGranted) {
71         val message: Toast = Toast.makeText(this, "Permissions have been successfully submitted", Toast.LENGTH_SHORT)
72         message.show()
73     } else {
74         val message: Toast = Toast.makeText(this, "To make the app work correctly, please provide the necessary permissions", Toast.LENGTH_LONG)
75         message.show()
76         finish()
77     }
78 }
79
80 permissionExplainLauncher = registerForActivityResult(StartActivityForResult()) { result ->
81     if (result.resultCode == RESULT_OK) {
82         bleDevicesList.addAll(bluetoothAdapter.bondedDevices)
83     } else {
84         val message: Toast = Toast.makeText(this, "To make the app work correctly, please provide the necessary permissions", Toast.LENGTH_LONG)
85         message.show()
86         finish()
87     }
88 }
89
90 when {
91     ContextCompat.checkSelfPermission(this, Manifest.permission.BLUETOOTH_CONNECT) == PackageManager.PERMISSION_GRANTED -> {
92         bleDevicesList.addAll(bluetoothAdapter.bondedDevices)
93     }
94     ActivityCompat.shouldShowRequestPermissionRationale(this, Manifest.permission.BLUETOOTH_CONNECT) -> {
95         val intent: Intent = Intent(this, PermissionsActivity::class.java)
96         intent.putExtra("message", "To make the app work correctly, please provide the necessary permissions")
97         intent.putExtra("permission", Manifest.permission.BLUETOOTH_CONNECT)
98         permissionExplainLauncher.launch(intent)
99     }
100     else -> {
101         requestPermissionLauncher.launch(Manifest.permission.BLUETOOTH_CONNECT)
102     }
103 }
104
105 requestPermissionLauncher = registerForActivityResult(RequestPermission()) { isGranted: Boolean ->
106     if (isGranted) {
107         val message: Toast = Toast.makeText(this, "Permissions have been successfully submitted", Toast.LENGTH_SHORT)
108         message.show()
109     } else {
110         val message: Toast = Toast.makeText(this, "To make the app work correctly, please provide the necessary permissions", Toast.LENGTH_LONG)
111         message.show()
112         finish()
113     }
114 }
115
116 permissionExplainLauncher = registerForActivityResult(StartActivityForResult()) { result ->
117     if (result.resultCode != RESULT_OK) {
118         val message: Toast = Toast.makeText(this, "To make the app work correctly, please provide the necessary permissions", Toast.LENGTH_LONG)
119         message.show()
120         finish()
121     }
122 }
123
124 val bleDeviceMacAddress: String? = intent.getStringExtra("mac")
125 if (bleDeviceMacAddress == null) finish()
126
127 val bluetoothManager: BluetoothManager = this.getSystemService(Context.BLUETOOTH_SERVICE) as BluetoothManager
128 val bluetoothAdapter: BluetoothAdapter = bluetoothManager.adapter
129
130 val bleDevice: BluetoothDevice = bluetoothAdapter.getRemoteDevice(bleDeviceMacAddress)
131

```

```

94 val bleGattCallback: BluetoothGattCallback = object : BluetoothGattCallback() {
95     override fun onConnectionStateChange(gatt: BluetoothGatt?, status: Int, newState: Int) {
96         super.onConnectionStateChange(gatt, status, newState)
97         Log.d("DEBUG_TAG", "connection state changed to $newState")
98         if (newState == BluetoothProfile.STATE_CONNECTED) {
99             connectionStatus.value = "connected"
100             when {
101                 ContextCompat.checkSelfPermission(this@DeviceActivity, Manifest.permission.BLUETOOTH_CONNECT) == PackageManager.PERMISSION_GRANTED
102                 gatt?.discoverServices()
103             }
104             ActivityCompat.shouldShowRequestPermissionRationale(this@DeviceActivity, Manifest.permission.BLUETOOTH_CONNECT) -> {
105                 val intent: Intent = Intent(this@DeviceActivity, PermissionsActivity::class.java)
106                 intent.putExtra("message", "To make the app work correctly, please provide the necessary permissions")
107                 intent.putExtra("permission", Manifest.permission.BLUETOOTH_CONNECT)
108                 permissionExplainLauncher.launch(intent)
109             }
110             else -> {
111                 requestPermissionLauncher.launch(Manifest.permission.BLUETOOTH_CONNECT)
112             }
113         }
114     }
115     if (newState == BluetoothProfile.STATE_CONNECTING) {
116         connectionStatus.value = "disconnecting"
117     }
118     if (newState == BluetoothProfile.STATE_DISCONNECTED) {
119         connectionStatus.value = "disconnected"
120     }
121     if (newState == BluetoothProfile.STATE_DISCONNECTING) {
122         connectionStatus.value = "disconnecting"
123     }
124 }
125
126 override fun onServicesDiscovered(gatt: BluetoothGatt?, status: Int) {
127     super.onServicesDiscovered(gatt, status)
128
129     if (status == BluetoothGatt.GATT_SUCCESS) {
130         Log.d("DEBUG_TAG", "new services discovered")
131     } else {
132         Log.w("DEBUG_TAG", "onServicesDiscovered not success. status: $status")
133     }
134 }
135
136 override fun onCharacteristicWrite(
137     gatt: BluetoothGatt?,
138     characteristic: BluetoothGattCharacteristic?,
139     status: Int
140 ) {
141     super.onCharacteristicWrite(gatt, characteristic, status)
142     Log.d("DEBUG_TAG", "Characteristic has been written. Status: $status")
143 }

```

```

23 enum StorageError {
24     OK,
25     FAILED_TO_READ_VALUE,
26     FAILED_TO_WRITE_VALUE,
27 };

```

```

54 template<typename T> StorageError load(const std::string& key, T& value) {
55     esp_err_t res = handle->get_item(key.c_str(), value);
56     if (res != ESP_OK) return FAILED_TO_READ_VALUE;
57     return OK;
58 }
59
60
61 template<typename T> StorageError load(const std::string&& key, T& value) {
62     esp_err_t res = handle->get_item(key.c_str(), value);
63     if (res != ESP_OK) return FAILED_TO_READ_VALUE;
64     return OK;
65 }
66
67
68 template<typename T> StorageError save(const std::string& key, T& value) {
69     esp_err_t res = handle->set_item(key.c_str(), value);
70     if (res != ESP_OK) return FAILED_TO_WRITE_VALUE;
71     handle->commit();
72     return OK;
73 }
74
75
76 template<typename T> StorageError save(const std::string&& key, T& value) {
77     esp_err_t res = handle->set_item(key.c_str(), value);
78     if (res != ESP_OK) return FAILED_TO_WRITE_VALUE;
79     handle->commit();
80     return OK;
81 }

```

```

18 typedef struct {
19     std::uint8_t* buffer;
20     std::uint64_t size;
21 } image_t;
22 constexpr UBaseType_t image_channel_len = 10;
23 QueueHandle_t image_channel;
24
25 const UBaseType_t server_status_channel_len = 100;
26 QueueHandle_t server_status_channel;
27
28
29 void init() {
30     image_channel = xQueueCreate(image_channel_len, sizeof(image_t));
31     server_status_channel = xQueueCreate(server_status_channel_len, sizeof(int32_t));
32 }

```

```

23 constexpr uint32_t freqs[] = {147, 147, 147, 147, 14
24 constexpr int32_t durs[] = {429, 214, 429, 214, 429,
25 constexpr int32_t music_size = 383;

```

```

71 void main(void* args) {
72     while (true) {
73         for (int32_t i = 0; i < music_size; ++i) {
74             if (!flag) {
75                 ESP_ERROR_CHECK(ledc_set_duty(LED_MODE, LEDC_CHANNEL, 0));
76                 ESP_ERROR_CHECK(ledc_update_duty(LED_MODE, LEDC_CHANNEL));
77                 break;
78             }
79             ESP_LOGI(TAG, "Music playing");
80             play_note(freqs[i]);
81             vTaskDelay(pdMS_TO_TICKS(durs[i]));
82         }
83         vTaskDelay(pdMS_TO_TICKS(1000));
84     }
85 }

```

```

7 class Converter:
8     def __init__(self, file_name: str) -> None:
9         self.midi = MidiFile(file_name)
10
11     @cache
12     def midi_num_to_freq(self, midi_num: int) -> float:
13         return 440 * pow(2, (midi_num - 69) / 12)
14
15     def convert(self) -> tuple[list[float], list[float]]:
16         meta, track = self.midi.tracks
17
18         tempo = 0
19         for msg in meta:
20             if msg.type == "set_tempo":
21                 tempo = msg.tempo
22
23         notes_freq = []
24         notes_duration = []
25         for msg in track:
26             if msg.type == "note_on":
27                 notes_freq.append(self.midi_num_to_freq(msg.note))
28             if msg.type == "note_off":
29                 notes_duration.append(tick2second(msg.time, self.midi.ticks_per_beat, tempo) * 1000)
30
31         return notes_freq, notes_duration
32
33     def convert_to_cpp(self) -> str:
34         freqs, durs = self.convert()
35         freqs = ", ".join(f"{round(note)}" for note in freqs)
36         durs = ", ".join(f"{round(dur)}" for dur in durs)
37         return f"constexpr uint32_t freqs[] = {{{freqs}}};\nconstexpr int32_t durs[] = {{{durs}}};\nconstexpr int32_t music_size = {len(freqs)};\n"
38
39
40 def main():
41     converter = Converter("pirates.mid")
42     print(converter.convert_to_cpp())

```