Pet Feeder

Хранилка за домашни любимци

Проект по дисциплината: Практическа роботика
и умни „неща“

Съставил:

Рая Георгиева, ФН 62013

ФМИ, 15.07.2020г.

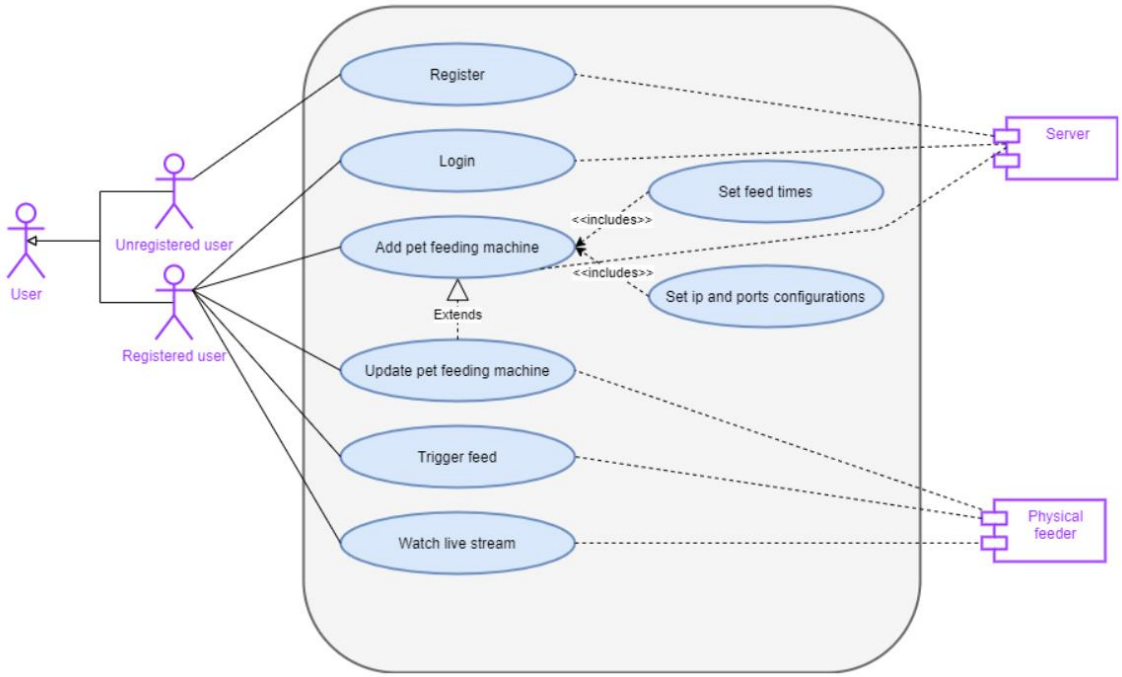
Съдържание

1. Описание на основните фунцкионални и нефункционални изисквания. Диаграма на поребителските случаи.

1.1. Функционални изисквания

Идентификатор	Текст
ФИ-1	Системата трябва да предоставя възможност за регистрация на *нерегистриран потребител*.
ФИ-2	*Регистриран потребител* трябва да може да влезе в системата.
ФИ-3	При несъвпадащи потребителско име и парола системата трябва да дава съобщение за грешка.
ФИ-4	Всеки *регистриран потребител* трябва да има уникален идентификатор (ид или имейл).
ФИ-5	*Регистриран потребител* може да добавя и премахва хранилки.
ФИ-6	*Регистрираният потребител* трябва да може да променя вече добавени хранилки.
ФИ-7	*Регистрираният потребител* трябва да има възможност да задава вид животно съответстващ на хранилка. Видовете животни са: котка, куче.
ФИ-8	*Регистрираният потребител* трябва да може да пусне и да гледа камерата на устройството на живо.
ФИ-9	*Регистрираният потребител* трябва да може да задава интервал на хране на животното на съответната хранилка.
ФИ-10	*Регистрираният потребител* трябва да може ръчно да предизвика хранене.
ФИ-11	При недостатъчно количество храна на *потребителя* трябва да се изпрати съобщение, че има нужда от презареждане на машината.
ФИ-12	При ръчно предизвикано хранене то се осъществява при засечено животно от съответния вид (видове животно виж ФИ-7)

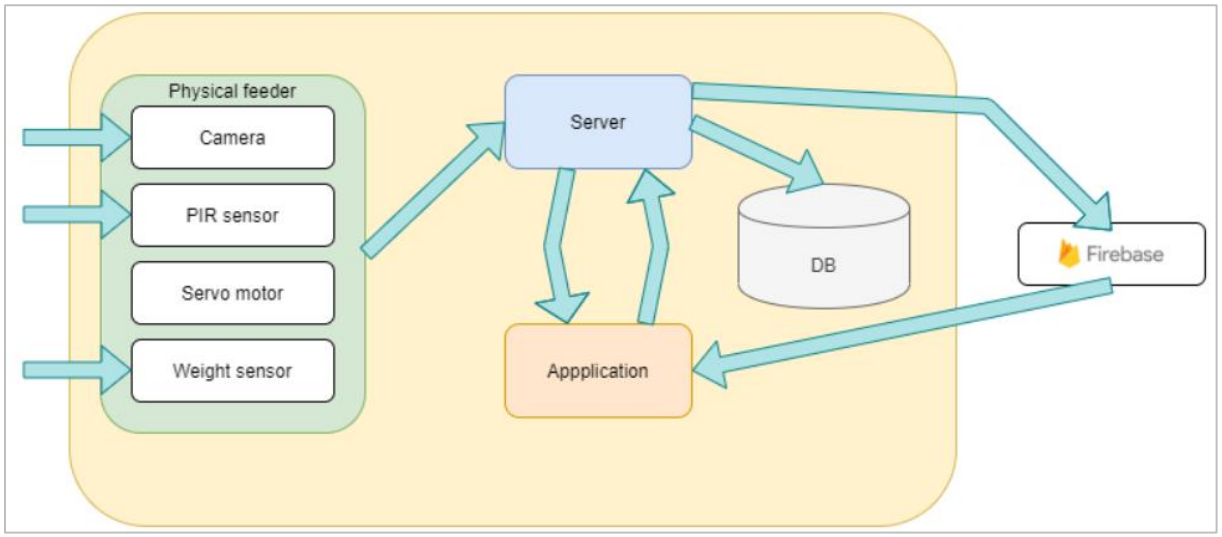
1.2. Диаграма на потребителските случаи



Фигура 1 Диаграм на потребителските случаи

2. Описание на използваните технологии и структурата на системата

Системата се състои структурно от следните елементи: Хардуерна част (устройството, което се намира в дома на човек, самата хранилка), Сървърна част (контролира бизнес логиката на системата и служи за диспечер на заявки между клиента и хранилката), База данни, Мобилно приложение (Фигура 2)



Фигура 2 Схема на системата

2.1. Хардурна част – Raspberry Pi

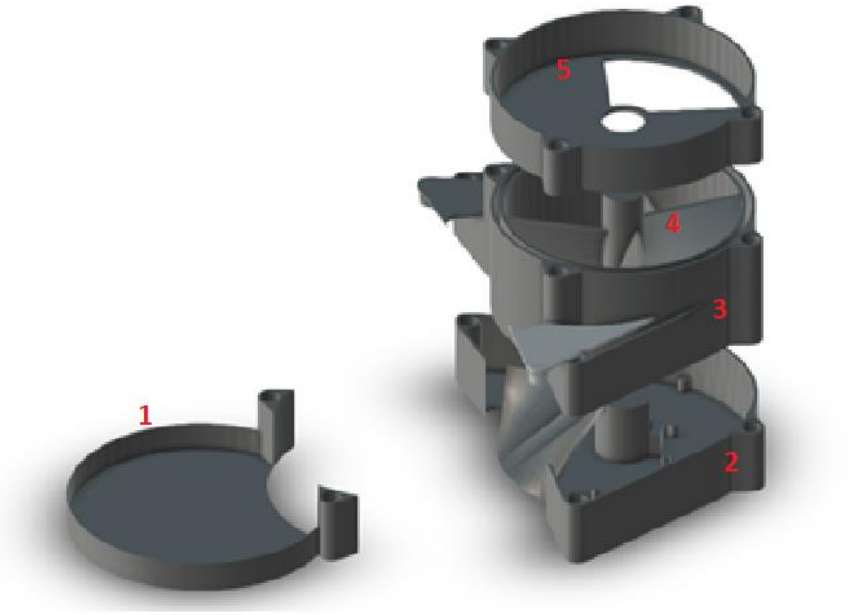
Хардуерната част включва следните елементи:

- 3D-принтиран копрус
- Raspberry Pi 3 (Model B)
- Серво-мотор – SG90 *[1]*
- Raspberry Pi Camera Module V2
- PIR Motion Sensor *[3]*
- Сензор за натиск *[2]*

2.1.1.3D-принтиран копрус

Моделът за корпуса е начертан чрез системата Fusion360 и включва следните пет елемента [8] (фиг. 3):

1. Купичка, в която се изсипва храната
2. Отделение, в което се помещават платката и мотора.
3. Отделение, в което се помещава перката.
4. Перка, закрепва се в долната си част за мотора.
5. Капак-поставка, над който се поставя кутия за храна



Фигура 3 3Д модел на кутията

2.1.2.Raspberry Pi

Всички останали компоненти са свързани за него, а то ги контролира. На него се изпълняват 2 процеса: сървърен процес, който следи за заявки, такава

заявка може да бъде например за ръчно хранене, и основен процес на работа на машинката, в него се изпълнява основната логика по работата на програмата, например изпълняване на храненията през определен период от време, отчитане на сензорите и изпращане на HTTP заявка, указваща недостатъчно наличие на храна.

Крайни точки (endpoints) на сървъра, който се намира на Raspberry Pi:

Име	Описание
GET move	Изпълнява ръчно движение. Връща JSON, съдържащ статуса на движението: дали се е изпълнило успешно или не.
POST setSleepTime	Променя времето между храненията на устройството.

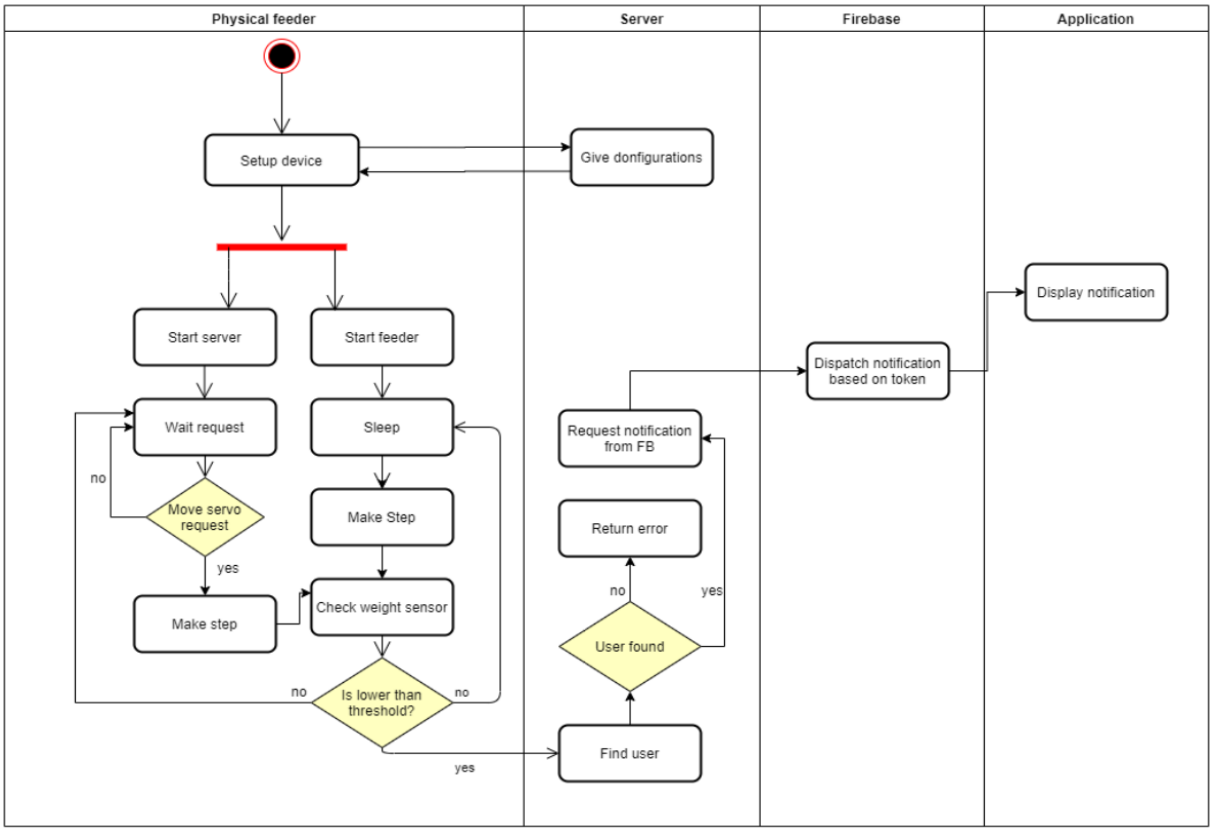
Таблица 1 Списък endpoints на сървъра, който се помещава на Raspberry Pi

Списък на заявките, които изпраща към Сървъра:

Име	Описание
GET /fm/config/{mac}	Изпълнява се при инициализация, взема запазени настройки от сървъра.
POST /fm/empty/{mac}	Сигнализира за празно устройство.

Таблица 2 Списък на заявките, които се изпращат от Physical feeder

Следната activity диаграма представя работата на тази част от системата:



Фигура 4 Activity диаграма

Описание на диаграмата:

1. Конфигурациите се изтеглят от сървъра (виж Таблица 2).
2. Двата процеса се разклоняват:
    a. Сървърът чака за заявки, ако получи заявка за завъртане на моторчето, го извършва и извършва съответните проверки (точка 3).
    b. Основният цикъл на работа се състои в поддържане на устройството в заспало състояние за определения период от време, след което се извършва завъртане и проверки (точка 3).
3. Проверки: след всяко завъртане се отчитат показанията на сензора за натиск. При показания по-малки от 100г се изпраща известие към сървъра за празна машина.

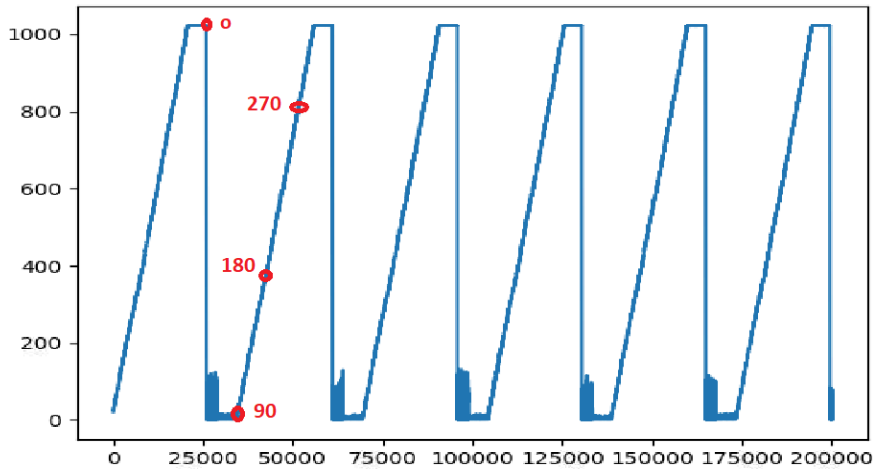
2.1.3.Серво-мотор

Моделът на мотора е SG90, като при закупуване този модел позволява въртене до 180 градуса [4]. Да се върти на 360 градуса му пречеха от една страна ограничител, който откачихме и потенциометър, който предоставя информация за точното положение на мотора, от друга страна. На мястото на потенциометъра закачихме делител на напрежение (два резистора), а потенциометъра свързах към аналогово-цифров преобразувател за да мога да контролирам позицията на мотора. За контрол над движението на мотора използвам хардуерна широчинно-импулсна модулация чрез библиотеката PiGPIO. Използваното запълване е ≈ 7,6 %.

Входът, получаван от потенциометъра, е между 0 и 1023, като в енда трета от оборот, се получава шум и е по-трудно да се определи точната позиция. На фиг. 5 се вижда графиката на показанията на потенциометъра. За позиция 0 градуса считам края на горното плато (стойност 1023), за позиция 90 градуса, края на долното плато (стойност 0), за 180 градуса позиция с показатели около 390 и последно за позиция 270 градуса показатели около 835. Тези стойности са определени емпирично.

2.1.4.Камера

Камерата е свързана за устройството по стандартен начин посредством лентовия си кабел. По-интересен е въпросът с видео стрийминга. Кодът за стрийминга е взет директно от документацията на камерата *[5]*. Това, което реално се случва е, че се създава сървър, който обслужва на порт 8001 и предоставя видео във формат MJPEG с честота около 30 фрейма в секунда, това е максималната честота, с която могат да се заснемат последователни кадри.

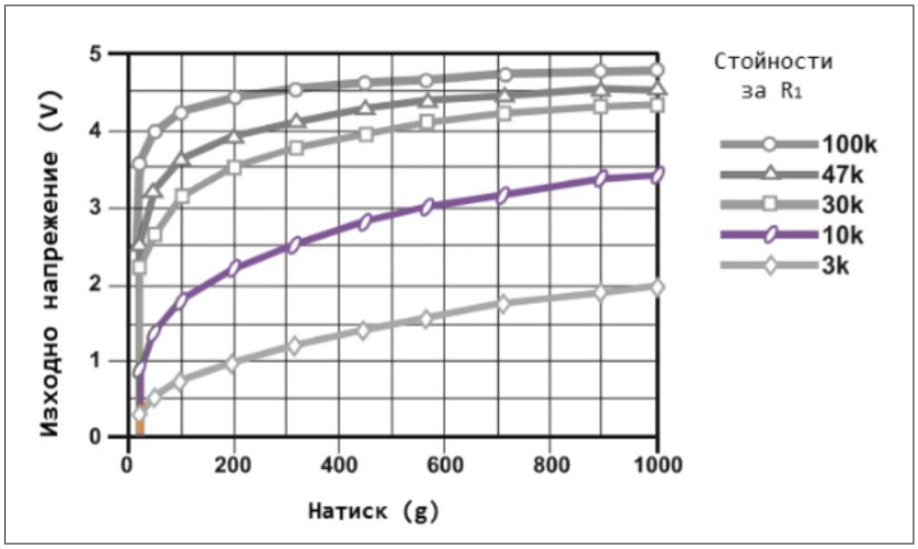


Фигура 5 Показания на потенциометъра, закачен за мотора

2.1.5.Сензор за натиск

Този сензор променя съпротивлението си в зависимост от натиска, който е приложен над него. Съответно входът, който предоставя е аналогов, но Raspberry Pi няма пинове за аналогов вход. Това наложи използването на аналогово-цифров преобразувател. Моделът, на който се спрях е MCP3008 [6], като го свързах за Raspberry Pi платката на пиновете за хардуерен SPI. Този АЦП предоставя възможност за вход на до 8 канала и преобразува аналоговите показания в цифрови със стойности от 0 до 1023.

Основното приложение на сензора за натиск е да отчете, когато теглото на храната е прекалено малко и съответна да изпрати известие за това. Под прекалено малко разбираме под 100г. За определяне на съответствието между показанията на сензора и реалното тегло използвах следната таблица:



Фигура 6 Съответствие между показанията на сензора за натиск и реалното тегло

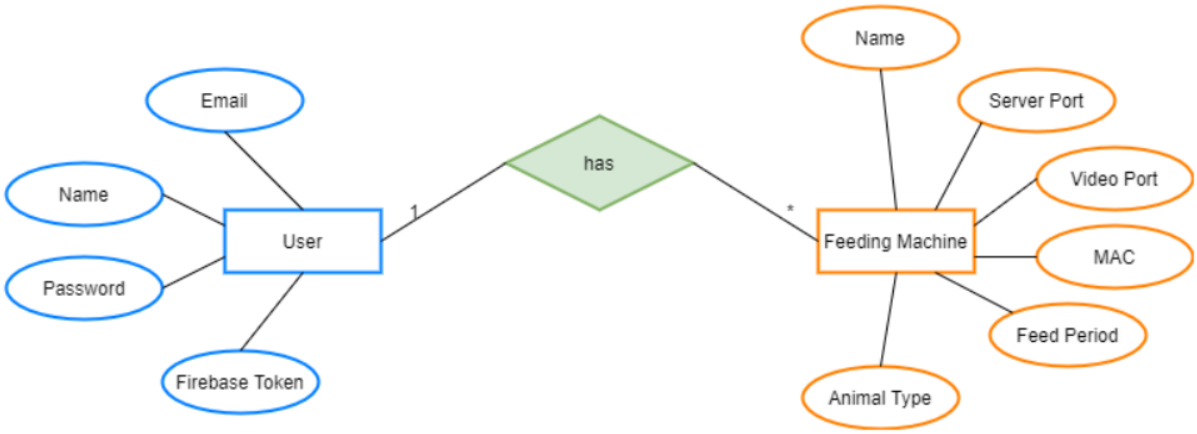
С лилаво е оцветена кривата, която ни интересува (тъй като сме ползвали 10K резистор), а изчисленията правя като съм сметнала стойността, която би трбявало да отговаря на 100г по следния начин:

$$\frac{x}{1024} \approx \frac{1.8}{3.4}$$

Това е приблизително 542, тоест заявка за известие се изпраща при стойност по-малка или равна на 542.

2.2. Сървърна част

Сървърната част е Spring Boot приложение с CRUD действия. Основните същности са User (потребител) и FeedingMachine (хранилка). Като има релация между тях, такава, че един потребител може да има повече от една хранилки, но на една хранилка съответства точно един потребител (фиг.5).



Фигура 7 Entity-Relationship диаграма на данните

Достъпът до данните се извършва с помощта на JPARepository, като има такива интерфейси и за двете същности. Данните се съхраняват в MySql база данни, която е хостната локално на моя компютър и се достъпва чрез MySql Connector Java. За паролите се използва за криптиране се използва класът BcryptPasswordEncoder, предоставен от SpringFramework.Security.

Крайни точки (endpoints) на сървъра:

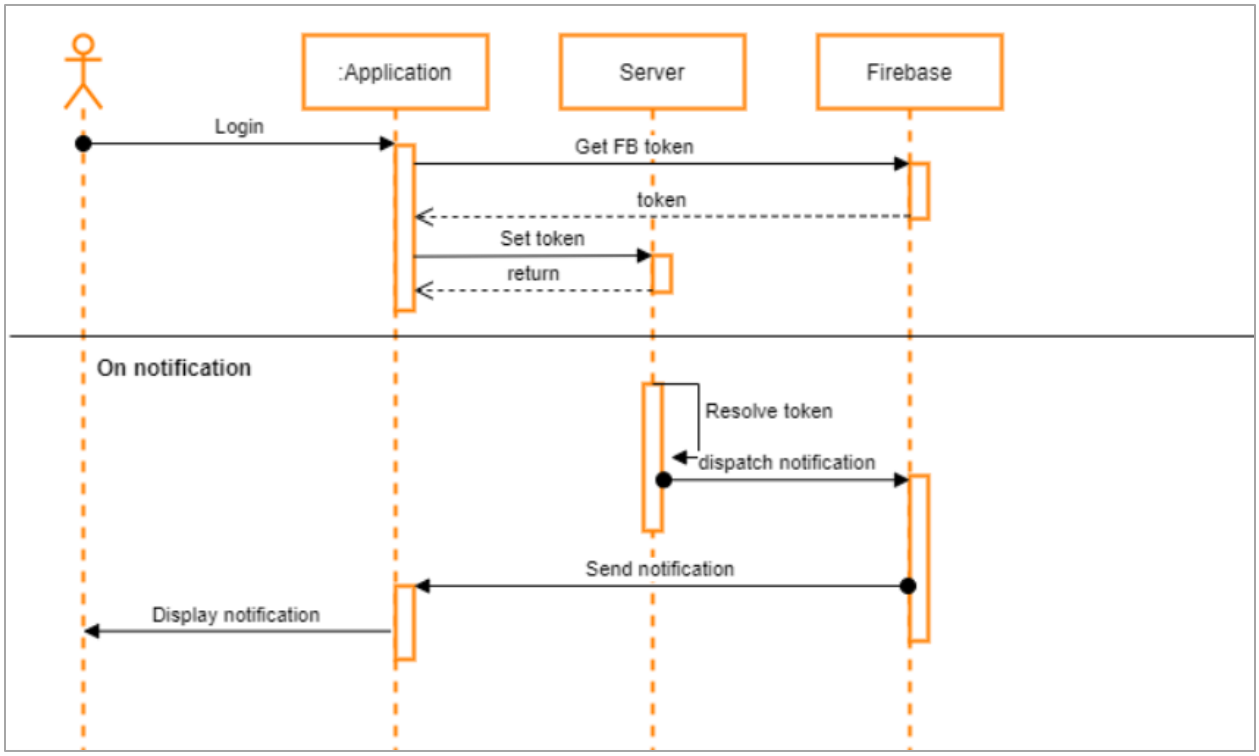
Име	Описание
GET /users	Връща списък с обекти, съответстващи на всички текущи потребители в базата данни.
GET /user/feedingmachines/{id}	Връща всички хранилки на потребителя с идентификатор, равен на {id}.
POST /user/authenticate	Тялото на заявката съдържа обект с имейл и чиста парола, които се проверяват в базата от данни за съответствие. Ако се открие

	такова, се връща съответния обект, в противен случай се връща NULL.
POST /register	Регистрира потребител с данни, съответстващи на данните от тялото на заявката.
POST /user/update	Обновява данните на потребител в съответствие с данните от тялото на заявката.
POST /user/add_feeding_machine/{id}	Добавя хранилка с данни данните от тялото на заявката към потребител с идентификатор, равен на {id}.
GET fm/move/{id}	Тази заявка отразява желание от страна на потребителя за хранене. Включва изпращане на HTTP заявка към физическата машина.
GET fm/config/{mac}	Връща конфигурациите на хранилка с MAC адрес, равен на {mac}.
POST fm/update	Обновява данните на хранилката, чийто идентификатор е този от тялото на заявката с данните от тялото на заявката.
POST /fm/empty/{mac}	Получава се от физическото устройство при засичане на празна хранилка
DELETE /fm/{id}	Изтрива хранилка с идентификатор, равен на {id}.

2.3. Клиентска част

Клиентската част е приложение, написано с помощта на Ionic Framework (Angular) *[10]*, като за native поведение се използва Capacitor. Приложение е съставено само за системата Android. Клиентската част основно се уповава на вградения шаблон tabs. Като основно се състои от отделни странници за профил, регистрация, вход, преглед на животно/и, изход. Моделите, които се използват съответстват на основните същности.

Основна част от изпълнението на клиентската е част е регистрирането на Firebase токен. Такъв токен се дава всяко мобилно устройство, като при вход този токен се изпраща към Сървъра, където се пази тази връзка (фиг. 8).



Фигура 8 Sequence диаграма на регистрацията на токен

3. Указания за инсталация

3.1. Клиентско приложение

Към настоящия момент не е предвидено изтеглянето на приложението от някакъв Store. За да се пусне проекта на устройство е необходимо да се отвори папката /application/android в AndroidStudio и след това да се свърже физическо или виртуално устройство (*[15]*, *[16]*) и да се пусне самото приложение.

3.2. Сървър

За сървърното приложение трябва да има предварително налична MySql база данни на компютъра.

При наличие на такава сървърът се пуска като се влезе в директорията му, зареди се в IntelliJ Idea и се избере Run.

3.3. Хардуерно устройство

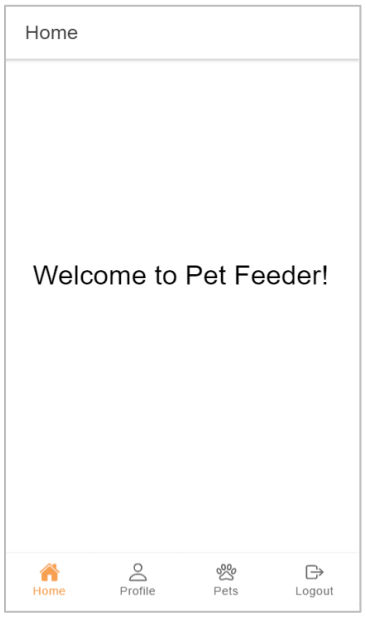
За да се пусне са нужни следните стъпки:

1) Захранване и пускане на платката (Raspberry Pi).
2) Локализиране на директорията, в която се намира кода.
3) Пускане на PiGPIO демона, чрез командата: sudo pigpiod
4) Изпълняване на програмата: python3 request_handler.py

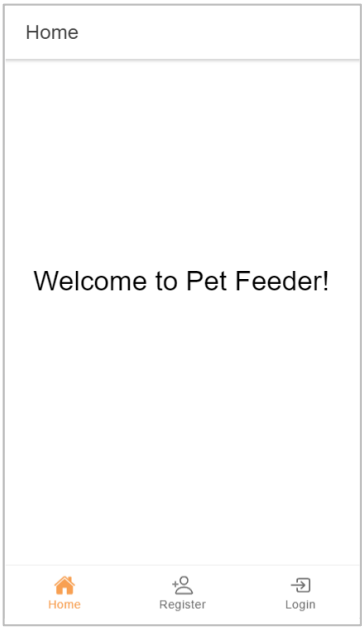
Има зависимости от следните библиотеки: *RPi.GPIO, pigpio, picamera, matplotlib, Adafruit_MCP3008, Adafruit_GPIO, socketserver, http*.

4. Потребителско упътване

4.1. Начална страница



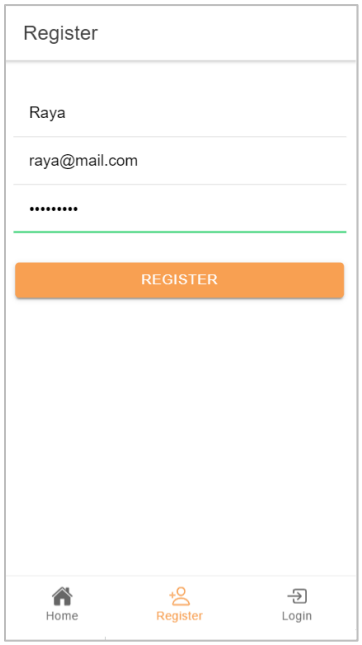
Фигура 10 Начална страница на приложението влязъл потребител



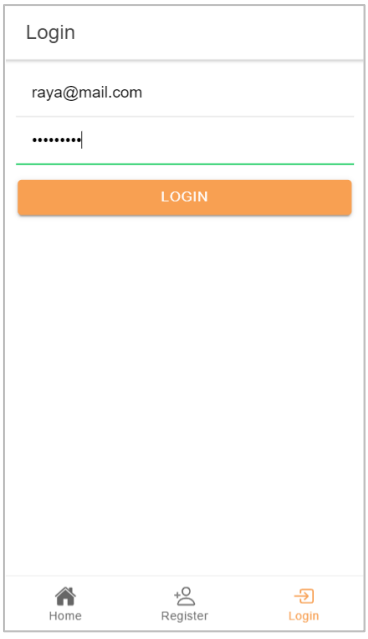
Фигура 9 Начална страница на приложението за нерегистриран потребител

4.2. Регистрация

От началната страница се избира опция Register от табовете долу и се попълва формата, при успешно попълване бутонът Register най-долу става активен (фиг. 12)



Фигура 12 Форма за регистрация



Фигура 11 Форма за вход

4.3. Вход

При избор на таб „Login” най-вдясно се отваря формата за вход. Бутонът „Login” първоначално не е активен, но при попълване на данните се активира. При несъвпадащи имейл и парола се показва съобщение за грешка.

4.4. Списък животни

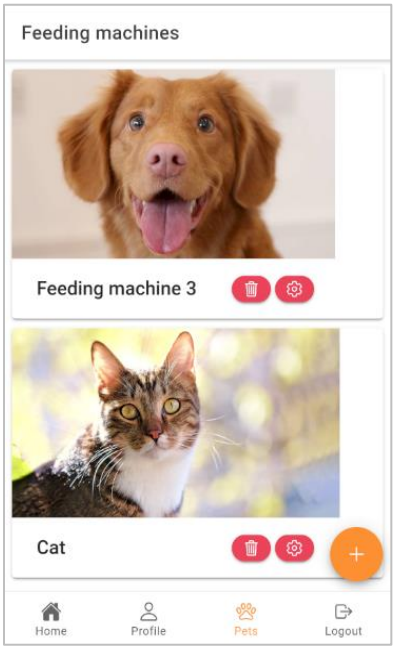
При избор на таб „Pets” се отваря списък с домашни любимци, като всеки от тях може да се изтрие, редактира или отвори. Най-долу вдясно има бутон +, от който се добавя ново животно (фиг. 14).

4.5. Добавяне на животно

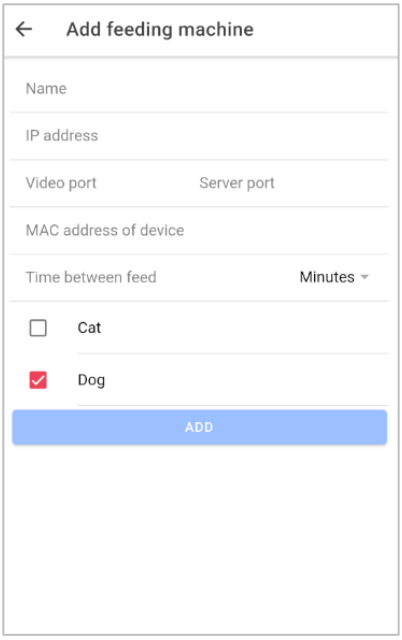
Попълва се формата с данните за хранилката: име, публичен IP адрес, порт, през който се достъпва камерата и порт, през който се достъпва сървъва на Raspberry Pi платката. Отбелязва се желания период между храненията като интервал в минути, часове или дни. Накрая се отбелязва типът на животното: коте или куче (фиг. 13).

4.6. Хранене на животно и връзка на живо

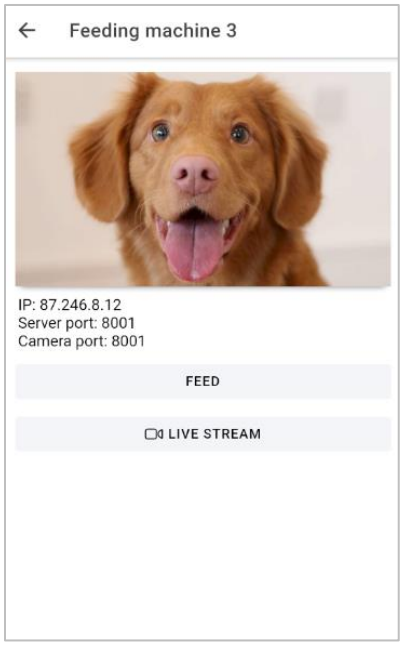
При избор на съответна хранилка от списъка се отваря нейната страница (фиг. 16). В нея към момента се виждат снимка и два бутона: „Feed” и “Live stream”. При избор на „Feed” се предизвиква ръчно хранене. При избор на бутона “Live stream” се отваря страничка, на която се вижда живата връзка с устройството (фиг. 15).



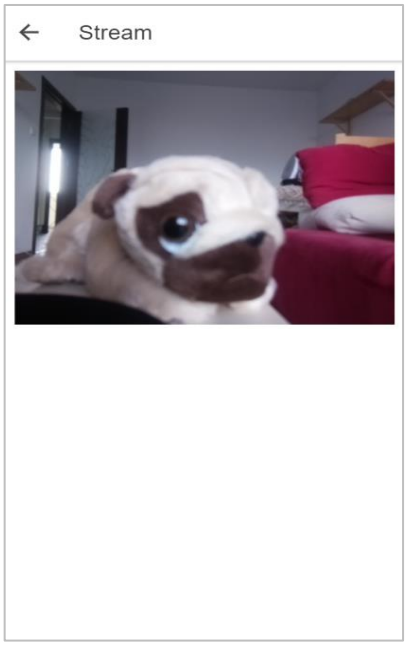
Фигура 14 Списък животни



Фигура 13 Форма за добавяне на животно



Фигура 16 Страница на хранилка



Фигура 15 Страница за жива видео връзка

5. Бъдещо развитие

Не успях да реализирам първоначалната идея за разпознаване на животното и като за начало това е нещо, върху което бих искала да се съсредоточа за бъдеще. Други идеи за развитие са свързване на няколко физически хранилки, например в рамките на един и същи дом, за една плата. Също така подобряване на имплементацията на сървърната част, тъй като останаха необработени грешки, както и на клиентската част, тъй като смятам, че може да се подобри значително потребителския интерфейс. За финал, проект от такъв тип би могъл да се внедри в по-голям мащаб в земеделски предприятия и да улесни работата на фермерите, но може да си остане и в домовете на хората, помагайки им в грижата за техните любимци.

5. Източници:

[1] SG90 servo motor datasheet (http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf)

*[2] Force Sensitive Resistor Datasheet (*https://cdn.sparkfun.com/assets/c/4/6/8/b/2010-10-26-DataSheet-FSR406-Layout2.pdf)

*[3] PIR motion sensor datasheet (*https://www.sparkfun.com/datasheets/Sensors/Proximity/SE-10.pdf)

*[4] Continuous rotation on SG90 servo (*https://www.youtube.com/watch?v=zV_5wUo7Kxs)

*[5] Raspberry Pi Camera Video Streaming (*https://picamera.readthedocs.io/en/latest/recipes2.html#web-streaming)

*[6] MCP3008 datasheet (*https://cdn-shop.adafruit.com/datasheets/MCP3008.pdf)

*[7] MCP3008 tutorial (*https://learn.adafruit.com/raspberry-pi-analog-to-digital-converters/mcp3008)

*[8] 3D model (*https://gmail414454.autodesk360.com/g/shares/SH919a0QTf3c32634dcf3cf581a2a09c229f)

*[9] PGPIO library (*http://abyz.me.uk/rpi/pigpio/python.html)

*[10] Ionic Docs (*https://ionicframework.com/docs)

*[11] Code from lectures (*https://github.com/iproduct/course-social-robotics/tree/master/iot-demo-springboot)

*[12] Firebase in Spring Boot (*https://medium.com/@maheshwar.ligade/spring-boot-firebase-crud-b0afab27b26e)

*[13] Firebase Documentation (*https://firebase.google.com/docs)

*[14] Capture images Raspberry Pi Camera (*https://picamera.readthedocs.io/en/latest/recipes2.html#rapid-capture)

*[15] Connect Hardware Device in Android Studio (*https://developer.android.com/studio/run/device)

*[16] Create and Manage Virtual Device in Android Studio (*https://developer.android.com/studio/run/managing-avds)

*[17] Setup Firebase App (*https://firebase.google.com/docs/cloud-messaging/android/client)

[18] RPi GPIO basic input/output (https://raspi.tv/2013/rpi-gpio-basics-6-using-inputs-and-outputs-together-with-rpi-gpio-pull-ups-and-pull-downs)

*[19] Dogs vs Cats Model (*https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-to-classify-photos-of-dogs-and-cats/)