МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук,
управління та адміністрування
Кафедра інформаційних технологій

Комплексна бакалаврська кваліфікаційна робота

на тему: «Розробка системи управління рухомим об'єктом на основі штучної нейронної мережі»

Склад:

Частина 1 «Розробка системи штучного зору для управління рухомим об'єктом»

Виконавець: Шпак Мирослав Ілліч
Керівник: к.т.н., доцент
Перелигін Борис Вікторович

Частина 2 «Розробка координатора для управління рухомим об'єктом»

Виконавець: Дьомін Олександр Володимирович
Керівник: к.т.н., доцент
Перелигін Борис Вікторович

Староста роботи: Дьомін Олександр Володимирович
Провідний керівник проекту: к.т.н., доцент Перелигін Борис Вікторович
Рецензент: Мещеряков Володимир Іванович, д. т. н, професор

Одеса 2019

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук,
управління та адміністрування
Кафедра інформаційних технологій

Бакалаврська кваліфікаційна робота

на тему Розробка системи штучного зору для управління рухомим об'єктом 52 ст

Виконав студент 4 курсу групи К-45
Спеціальність 122 комп'ютерні науки,
Шпак Мирослав Ілліч

Керівник к.т.н., доцент
Перелигін Борис Вікторович

Консультант

Рецензент д. т. н, професор
Мещеряков Володимир Іванович

Одеса 2019

МІНІСТЕРСТВО ОСВІТИ І НАУКИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет Комп'ютерних наук, управління та адміністрування
Кафедра Інформаційних технологій
Рівень вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
С.Д. Кузніченко
"03" 04 2019 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Шпак Мирослав Ілліч
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка системи штучного зору для управління рухомим об'єктом
керівник роботи к.т.н., доцент Перелигін Борис Вікторович,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від "01" квітня 2019 року № 49-с
2. Строк подання студентом роботи 28.05.2019 р.
3. Вихідні дані до роботи JavaScript, NodeJS, Electron, Cordova, Raspberry Pi, CSS

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1 Історія розвитку робототехніки
2 Розробка дистанційно-керованого робота

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання “ 18 ” квітня 2019 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Оцінка виконання етапу	
			у %	за 4-х бальною шкалою
1	Вступ	18.04 – 19.04		
2	1 Історія розвитку робототехніки	20.04 – 12.05		
3	Рубіжна атестація	13.05 – 19.05		
4	2 Розробка дистанційно-керованого робота	20.05 – 27.05		
5	Висновки	28.05		
6				
7				
8				
9				
10				
	Інтегральна оцінка виконання етапів календарного плану (як середня по етапам)			

Студент _____ **Шпак М.І.**
(підпис) (прізвище та ініціали)

Керівник роботи _____ **Перелигін Б.В.**
(підпис) (прізвище та ініціали)

ВСТУП

Кожен рік на ринку робототехніки обіг коштів складає 5 – 6 мільярдів доларів, і ця цифра постійно зростає. За останніми даними, сьогодні в світі працюють 1,8 млн. найрізноманітніших роботів – промислових, домашніх, роботів-іграшок то що. Експерти Міжнародної федерації робототехніки відзначають, що в промисловості використовується найбільше роботів – приблизно 770 тисяч. Причому половина з них – 350 тисяч працюють у Японії. У Європі використовується 233 тисяч, а в Північній Америці – 104 тисячі промислових роботів і використовуються вони, головним чином, на складальних конвеєрах. Також електронні помічники зайняті і при збиранні сміття або навантаженні. Серед європейських держав найбільше промислових роботів використовується в Німеччині – 105,2 тисячі, друге місце займає Італія – 46,8 тисячі, на третьому – Франція – 24,2 тисячі. В Росії працює 5 тисяч роботів, Швейцарія та Австрія використовують по 3,5 тисячі роботів, Фінляндія – 3 тисячі, Данія – 1,8 тисячі, Польща – 644 робота і Угорщина – 176.

1ІСТОРІЯ РОЗВІТКУ РОБОТОТЕХНІКИ

1.1ЗАГАЛЬНА ІСТОРІЯ РОБОТОТЕХНІКИ

Роботи навколо нас. Іноді вони маленькі і непомітні, іноді великі індустріальні. Ми їх сприймаємо частиною мрій про далеке безтурботне майбутнє людства або як загрозу його зникнення. Вони оживають у фільмах, коміксах, книгах, наукових лабораторіях та автоматизованих заводах і кожного разу вражають людей новими можливостями та перспективами. Коли з’явилася ідея робота? Хто створив перших роботів і як вони виглядають зараз?

Одного разу до мене в гості прийшов Йозеф. Він – робот і, як відома персона у середовищі комп’ютерних ігор, хотів з’ясувати для себе історію

свого роду.

Історія роботів розпочинається із ідеї про створення людиною живої істоти. Якби людина змогла це зробити, то вона би повторила акт деміурга – створення живого, навіть якщо «життя» при цьому дещо по-іншому розуміється.

Перші рецепти створення життя ґрунтувалися на міфах, магії та релігії. Довгий час у багатьох культурах переважало переконання, що людина була створена із глини, а отже найближчим шляхом буде саме глино-технологія. Але якщо із глиною проблем не виникало, то от із іншою складовою – душею, свідомістю, розумом, які гарантували певну самостійність існування живої істоти, розібратися не могли.

Навіть із розвитком техніки, повністю ідея міфічного акту створення життя не зникла повністю, а переселилася у готичні романи, а згодом у фантастику. Прикладами таких «експериментів» можна назвати ґолема та монстра Франкенштайна.

Наступний етап в історії роботів пов'язаний із розвитком механіки та застосуванням певних відкриттів для розваг аристократів. Поява легендарних автоматонів ще в Античності (хоч до нашого часу вони не дійшли) і їхній розвиток у Середньовіччі були революційними і настільки цікавими, що ми в одній із наступних статей напишемо про них докладніше.

Історія сучасних роботів пов’язана із кількома людьми. Сам термін «робот» з’явився завдяки творчості чеського письменника Карела Чапека і його фантастичній п’єсі «Rossumovi univerzální roboti» (1920). Якщо точніше, це були не зовсім роботи, а радше кіборги (навіпмеханічні і напіворганічні істоти). Однак в науці слово закріпилося і застосовується досі.

Термін «робототехніка» (robotics) вводить інший письменник – Айзек Азімов. Він також був першим автором, який описав правила робототехніки, які мали би гарантувати в майбутньому захист від неконтрольованої або непередбачуваної поведінки роботів стосовно людини.

3 закони робототехніки (Айзек Азімов)

-Робот не може заподіяти шкоду людині, або своєю бездіяльністю дозволити, щоб людині була заподіяна шкода;

-Робот повинен підкорятися наказам людини, за винятком тих, котрі суперечать першому пункту;

-Робот повинен захищати самого себе, якщо тільки його дії не суперечать першому і другому пунктам.

Серед вчених, які доклалися до еволюції ідеї самостійного, але не обов'язково людиноподібного робота, можемо згадати Алана Тюрінґа (відомий завдяки «тесту Тюрінґа») та Норберта Вінера (батька кібернетики – науки про комунікаційні системи і теорії штучного інтелекту).

А тепер хронологічно пробіжимося по біографіях найвідоміших представників виду «Роботи», які з'явилися на планеті Земля завдяки зусиллям науковців та інженерів.

40-ві рр. XX ст. – поява роботів-черепашок Елмера та Елсі, дітей Вільяма Грея Волтера. Ґрунтуючись на ідеях нейрофізіології, вчений вважав, що взаємодія навіть між кількома простими клітинами мозку породжує велику кількість сценаріїв поведінки. Нам залишається лише зімітувати ці «мозкові клітини» для роботів.

1954-56 рр. – Джордж Девол відкриває світу першого індустріального руко-робота і формує першу світову робототехнічну компанія Unimation.

1960-ті рр. – з'являється «Звір» (Beast) в Університеті Джона Гопкінза (США). Мобільний прото-робот, який мав зародки власного інтелекту і орієнтувався у просторі.

1965 рік – General Electric випускають «Квадропед» (Walking Truck). Вага – більше тонни. Основна місія цього робота – допомога солдатам долати пересічену місцевість.

1968-69 рік – поява «Щупальця» Марвіна Мінскі і «стенфордської руки»

1978 рік – на виробництві General Motors починають використовувати збирального робота PUMA (Programmable Universal Machine for Assembly).

1983 рік – під час аварії на АЕС Трі-Майл-Ійленд було застосовано мобільного робота-ремонтника, оскільки радіація перешкоджала роботі рятівників-людей.

1990 рік – на сцену виходить робот Амблер (Ambler), дуже повільно виходить, але ця хода багато в чому змінила подальшу еволюцію цілого напряму робототехніки.

1992-94 рік – роботи Данте почали вивчати кратери вулканів і готуються відправитися в космос. Хоч перша спроба завершилася невдало і робот не зумів дійти до місця призначення, але це не зупинило науковців.

1996-97 рік – на Марс було відправлено марсохід, який зумів дистанційно дослідити геологічні зразки, зазнимкувати поверхню і дещо наблизити людство до омріяної колонізації Сонячної системи.

2000 рік – корпорація Honda випускає Asimo – робота-андроїда, який легко пересувається у просторі, має складну і сучасну систему орієнтування та розпізнавання об’єктів навколо себе.

2004 рік – розважати людство починає Робосапієн, дітище Марка Тілдена.

2008 рік – робот вчиться грати на скрипці

2010 рік – опановує фортепіано.

2013 рік – Google купують компанію Boston Dynamics (Рис. 1.1), яка активно долучається до роботобудування. Основні робото-діти: BigDog, CHEETAH, LittleDog, Atlas та ін. Дехто із знавців передбачає, що саме із цих роботів або їхніх дітей почнеться кінець людства. Чому? Бо в роботів хороша пам’ять.

Еволюція роботів тісно пов’язана із загальним технічним прогресом людства, із пошуками супер-інтелекту, із побоюваннями щодо майбутнього і мріями трансгуманістів про нові можливості наших нащадків.

Що ми можемо гарантувати – це те, що навіть оподаткування роботів, запропоноване Білом Гейтсом, не зашкодить родичам Йозефа нарешті знайти свою Землю Обітовану. Чи існуватиме після появи розумного робота

людство – це питання ми залишимо відкритим.

Рисунок 1.1 - Найостаніші розробки компанії Boston Dynamics

1.2ЗАСТОСУВАННЯ РОБОТОТЕХНІКИ

Оскільки все більше і більше роботів призначено для виконання окремих завдань, спосіб їх класифікації, стає все більш потрібним. Наприклад, багато роботів призначено для праці з монтажу, і не можуть бути легко пристосовані для інших застосувань. Їх називають «складальними роботами». Для зварювання шва, деякі виробники постачають повні зварювальні системи з роботом, тобто зварювальне устаткування поряд з іншими зручностями обробки матеріалів, таких як, поворотні столи та інше, як єдине ціле. Така інтегрована роботизована система, називається «зварювальний робот». Деякі роботи, спеціально призначено для маніпулювання важкими навантаженнями і позначені як «важкі роботи службових обов'язків».

Поточні і можливі області застосування роботів, передбачають:

-Військові роботи.

-Caterpillar планує розробити дистанційно керовані машини та повністю автономних важких роботів до 2021 року. Деякі підйомні крани, вже керуються дистанційно.Було показано, що робот може виконувати скотарські завдання.

-Роботи все частіше використовуються у виробництві (з 1960 року). В автомобільній промисловості, вони можуть складати більше половини від загальної «праці». Є навіть фабрики «з вимкненим світлом», такі як завод з виробництва клавіатур IBM у Техасі, котрий на 100 відсотків автоматизовано.Роботи, такі як HOSPI, використовуються як кур'єри в лікарнях (лікарняний робот). Інші лікарняні завдання, виконують роботи рецепціонери, гіди і носильники – помічники.

-Роботи можуть служити офіціантами і кухарями, також і у домашніх умовах.

-Робот бойового спорту – хобі або спортивний захід, де два або більше роботів борються на арені, щоби вимкнути один одного.

Роботи також є єдиними Землянами, кто побував на Марсі, Венері, на поверхні комети, Титані, та навіть поза зоною сонячної системи. Коротка історія польотів роботів у космос:

-4 січня 1959 – станція «Луна-1» пройшла на відстані 6000 кілометрів від поверхні Місяця та вийшла на геліоцентричну орбіту. Вона стала першим у світі штучним супутником Сонця.

-14 вересня 1959 – станція «Луна-2» вперше в світі досягла поверхні Місяця в районі Моря Ясності поблизу кратерів Аристид, Архімедта Автолік, доставивши прапор із гербом СРСР.

-4 жовтня 1959 – запущена автоматична міжпланетна станція «Луна-3», яка вперше в світі сфотографувала невидиму з Землі сторону Місяця. Також

під час польоту вперше в світі був на практиці здійснений гравітаційний маневр.

-3 лютого 1966 – АМС «Луна-9» здійснила першу в світі м'яку посадку на поверхню Місяця, були передані панорамні знімки Місяця.

-1 березня 1966 – станція «Венера-3» вперше досягла поверхні Венери та доставила прапор СРСР. Це був перший у світі переліт космічного апарату з Землі на іншу планету.

-3 квітня 1966 – станція «Луна-10» стала першим штучним супутником Місяця.

-24 вересня 1970 – станція «Луна-16» провела забір та подальшу доставку на Землю (станцією «Луна-16») зразків місячного ґрунту. Вона ж — перший безпілотний космічний апарат, що доставив на Землю проби породи з іншого космічного тіла (тобто, у даному випадку, з Місяця).

-17 листопада 1970 – м'яка посадка і початок роботи першого в світі напівавтоматичного дистанційно керованого самохідного апарату, керованого із Землі – Луноход-1.

-15 грудня 1970 – перша в світі м'яка посадка на поверхню Венери — «Венера-7».

-13 листопада 1971 – станція «Маринер-9» стала першим штучним супутником Марсу.

-27 листопада 1971 – станція «Марс-2» вперше досягла поверхні Марсу.

-2 грудня 1971 – перша м'яка посадка АМС на Марс на «Марс-3».

-20 жовтня 1975 – станція «Венера-9» стала першим штучним супутником Венери.

-жовтень 1975 – посадка двох космічних апаратів «Венера-9» та «Венера-10» і перші в світі фотознімки поверхні Венери, та й досі - єдині фотографії (Рис. 1.2).

-7 грудня 1995 – станція «Галілео» стала першим штучним супутником Юпітеру.

-24 червня 2000 – станція «NEAR Shoemaker» стала першим штучним супутником астероїду (433 Ерос).

-30 червня 2004 – станція «Кассіні» стала першим штучним супутником Сатурну.

-15 січня 2006 – станція «Стардаст» доставила на Землю зразки комети Вільда 2.

-17 березня 2011 – станція «MESSENGER» стала першим штучним супутником Меркурію.

Рисунок 1.2 – Перші фото з поверхні Верени апарату Венера-13

На землі роботи також допомогають у дослідженнях. Велика частина досліджень в області робототехніки зосереджується не лише на конкретних виробничих завданнях, а й на дослідженнях в області нових типів роботів, новітніх методах проектування, нових способах їх виготовлення та інших дослідженнях, таких як проект cyberflora Массачусетського технологічного інституту, що майже повністю академічний.

Зокрема, першою новинкою в області дизайну роботів, є відкритий

сорсинг проектів-роботів. Для опису рівня просування робота, може бути використано термін «Покоління Роботи». Цей термін вигадано професором Hans Moravec, головним науковим співробітником в Університеті Карнегі-Меллона Інституту робототехніки під час змалювання розвитку робототехніки у найближчому майбутньому. З'явлення роботів першого покоління, Moravec передбачив 1997 року, і які матимуть інтелектуальний потенціал, котрий можна порівняти з можливостями ящірки та повинні були стати доступними до 2010 року. Але, робот першого покоління, буде нездатний до навчання, проте, Moravec пророкує, що друге поліпшене покоління роботів, з'явиться до 2020 року, з інтелектом, можливо порівняним, з мишаком. Робот третього покоління, повинен мати інтелект, який можна буде порівняти зі здібностями мавпи. Хоча професор Moravec, пророкує роботи четвертого покоління з людським інтелектом, але на його думку, це відбудеться не раніше 2040 або 2050 років.

По-друге, еволюційні роботи. Це методологія, яка використовує еволюційні обчислення, задля розробки роботів, особливо форм тіла, або контролерів руху та поведінки. Схожим чином до природної еволюції, великій популяції роботів, буде дозволено певною мірою, конкурувати, отже, їх придатність вимірюватиметься здатністю виконувати завдання. Ті, які працюють гірше, видалятимуться із популяції та їх буде замінено новим набором, який матиме кращі моделі поведінки, засновані на цих переможцях. Згодом, «населення роботів» поліпшиться, і у кінцевому підсумку, може з'явитися задовільний робот. Це буде відбуватися без будь-якого безпосереднього програмування роботів, дослідниками. Розробники використовують цей метод як для створення кращих роботів так і задля дослідження природи еволюції. Оскільки процес часто вимагає багатьох поколінь роботів, які будуть моделюватися, цей спосіб, може бути запущено повністю або в основному, у моделюванні, а відтак, випробувано на реальних роботах, як тільки виділиться досить хороший алгоритм. 2016 року, налічувалося близько 10 мільйонів промислових роботів, які працюють в

усьому світі, і Японія є передовою країною, що має високу щільність використання роботів в обробній промисловості.

Вивчення руху, можна розподілити на кінематику та динаміку. Пряма кінематика стосується розрахунку позиції кінцевого ефектору, орієнтації, швидкості та прискорення, коли відомі відповідні спільні значення. Зворотна кінематика, відноситься до протилежного випадку, у якому потрібно розраховувати спільні значення для заданих положень кінцевих ефекторів, як це було зазначено під час планування шляху. Деякі спеціальні аспекти кінематики містять обробку надмірності (різні можливості виконання того-ж руху), запобігання зіткнень і уникнення сингулярності. Після того, як усі відповідні позиції, швидкості і прискорення було розраховано з використанням кінематики, застосовуються методи з області динаміки, для вивчення впливу силна ці рухи. Безпосередньо динаміка, стосується обчислень прискорень у роботі, оскільки прикладені сили відомі. Пряма динаміка використовується у комп'ютерному моделюванні роботів. Зворотна динаміка відноситься до розрахунку сил виконавчих механізмів, потрібних для створення запропонованого прискорення кінцевих ефекторів. Ці дані, може бути використано для поліпшення алгоритмів управління роботом.

У кожній галузі, згаданій вище, дослідники прагнуть розробити нові концепції і стратегії, поліпшення дійсних, а також, покращення взаємодії між цими областями. Для цього, повинно бути розроблено і впроваджено. критерії для «оптимальної» продуктивності та способів оптимізації дизайну, структури та контролю роботів.

Оскільки все більше і більше роботів призначено для виконання окремих завдань, спосіб їх класифікації, стає все більш потрібним. Наприклад, багато роботів призначено для праці з монтажу, і не можуть бути легко пристосовані для інших застосувань. Їх називають «складальними роботами». Для зварювання шва, деякі виробники постачають повні зварювальні системи з роботом, тобто зварювальне устаткування поряд з

іншими зручностями обробки матеріалів, таких як, поворотні столи та інше, як єдине ціле. Деякі роботи, спеціально призначено для маніпулювання важкими навантаженнями і позначені як «важкі роботи службових обов'язків».

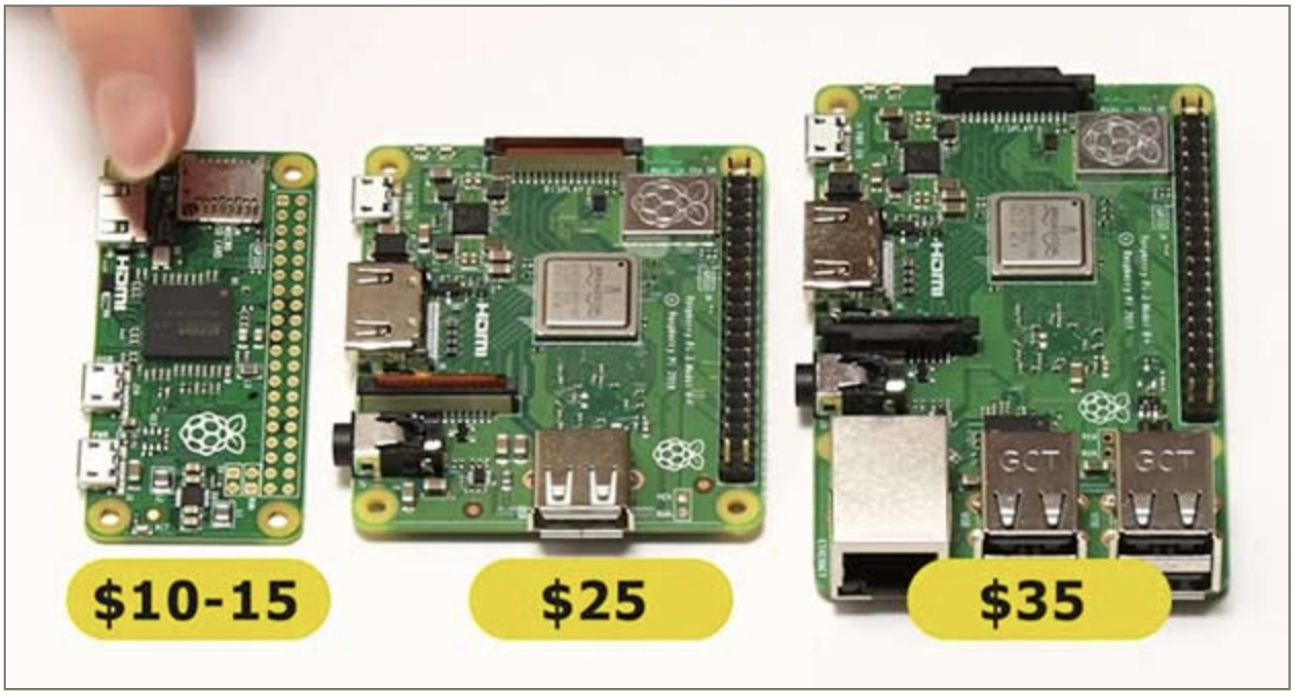
2РОЗРОБКАДИСТАНЦІЙНО-КЕРОВАНОГО РОБОТА

2.1 ПРОЕКТУВАННЯ ЕЛЕКТРОННОЇ СХЕМИ РОБОТА

Центральним мозком робота буде комп'ютер Raspberry Pi 3 Model A, потужності якого вистачає для обробки відео-потоку нашої нейромережі. Вибирали з наступного списку моделей (Рис 2.1):

-Raspberry Pi Zero – Дуже мало споживчий і мініатюрний комп'ютер який підходить саме по енергоспоживанню, але на жаль просто ніяк би не витримав очікуване навантаження вона графічну частину;

- Raspberry Pi – Середня модель, найменш підходяща, т до енергоспоживання на рівні побратима вище, але при цьому потужності значно менше.



- Raspberry Pi 3 – На жаль високе енергоспоживання, але і потужності найвищі серед усіх моделей.

Рисунок 2.1– МоделіRaspberry Pi

Проблему харчування я вирішив досить Костильна методом. Ззаду робота буде провід живлення для самого комп'ютера, але харчування

двигунів буде у вигляді окремого блоку живлення на борту. Для управління двигунами в обидві сторони я буду використовувати драйвер харчування L298N. Двітателі будуть звичайні «постійного струму».

Модуль L298N H-bridge (Рис 2.2) можна використовувати для двигунів, напруга живлення яких знаходиться в діапазоні від 5 до 35 вольт. Крім того, на багатьох подібних платах є вбудований 5В регулятор, який дає можливість живити ваші пристрої.

Рисунок [illegible] L298N

Raspberry Pi – Одноплатний комп'ютер розміром з банківську карту, спочатку розроблений як бюджетна система для навчання інформатиці, але пізніше отримав більш широке застосування. Розробляється Raspberry Pi Foundation. Всього за п'ять років було продано більше 12,5 мільйонів пристроїв Raspberry Pi; Raspberry Pi випускається в декількох комплектаціях: «A», «A +», «B», «B +», «2B», «Zero», «Zero W», «3B» і «3B +». Моделі «Zero W», «3B» і «3B +» підтримують Wi-Fi і Bluetooth. Перші три версії оснащені ARM11 процесором Broadcom BCM2835 з тактовою частотою 700 МГц і модулем оперативної пам'яті на 256МБ / 512МБ, розміщеними за технологією «package-on-packageruen» безпосередньо на процесорі. Модель «2B» оснащується процесором з 4 ядрами Cortex-A7 з частотою 1 ГГц і оперативною пам'яттю розміром 1 ГБ. Модель «A» оснащується одним USB 2.0 портом, модель «B» – двома, а моделі «B +» і «2B» – чотирма. Також в моделях «B», «B +», «2B» і «3B» присутній порт Ethernet. Крім основного ядра, BCM2835 включає в себе графічне ядро з підтримкою OpenGL ES 2.0, апаратного прискорення і FullHD-відео і DSP-ядро. Однією з особливостей є відсутність годин реального часу. Висновок відеосигналу можливий через композитний роз'єм RCA або через цифровий HDMI-інтерфейс. У версії «B +», «2B» і «3B» висновок можливий через аудіоразьем 3,5. Коренева файлова система, образ ядра і призначені для користувача файли розміщуються на карті пам'яті SD, MMC (в моделях A і B), в нових моделях, починаючи з «B

+», використовується microSD, в «3B» і «3B +» з'явилася можливість завантажуватися з USB носій або по мережі, також можна використовувати SDIOruen. Однією з найцікавіших особливостей Raspberry Pi є наявність портів GPIO (general purpose input / output). Завдяки цьому «малиновий» комп'ютер можна використовувати для управління різними пристроями. У моделі «B» плати присутній 26-піновий, а в моделі «B +» і «2 B» – 40-піновий роз'єм GPIO;

H-міст – це електронна схема, яка дає можливість прикласти напругу до навантаження в різних напрямках. Ця схема дуже часто використовується в робототехніці і іграшкових машинах, щоб змінювати напрямок обертання мотора. H-мости представлені у вигляді інтегральних схем, а також можуть бути побудовані з окремих радіодеталей.

Корпус виробу виготовлений з фанери, для пружності при русі я зробив корпус не цільним, а зі спеціальними вирізами (Рис 2.3, 2.4)

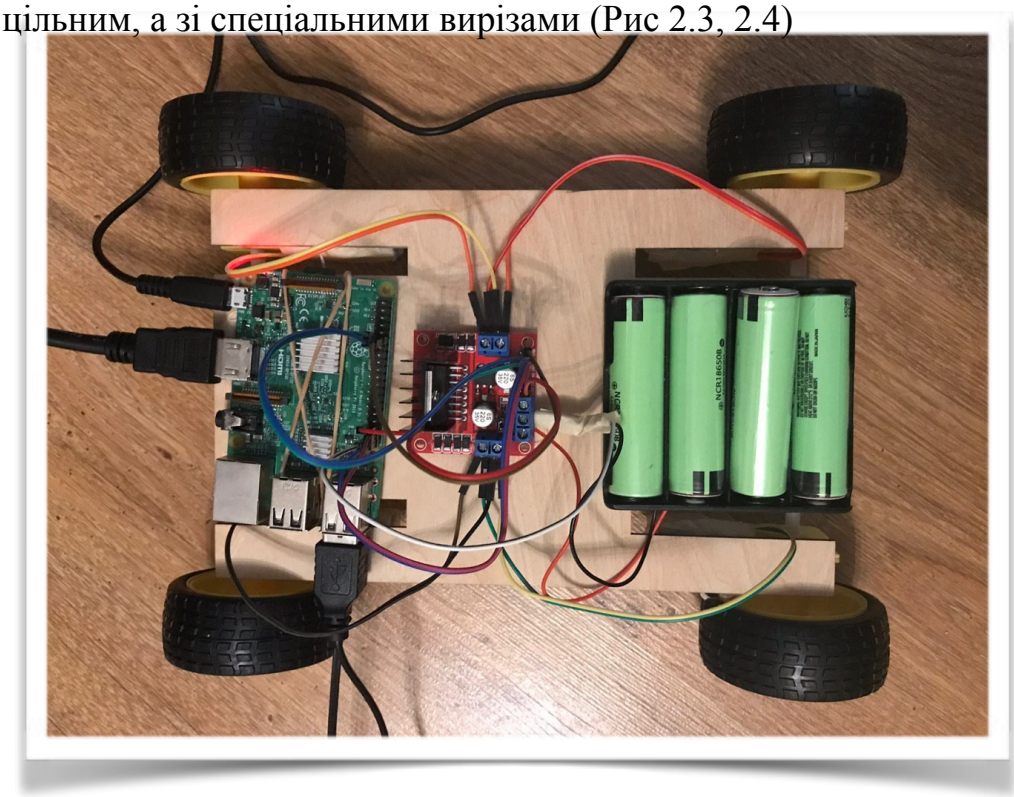
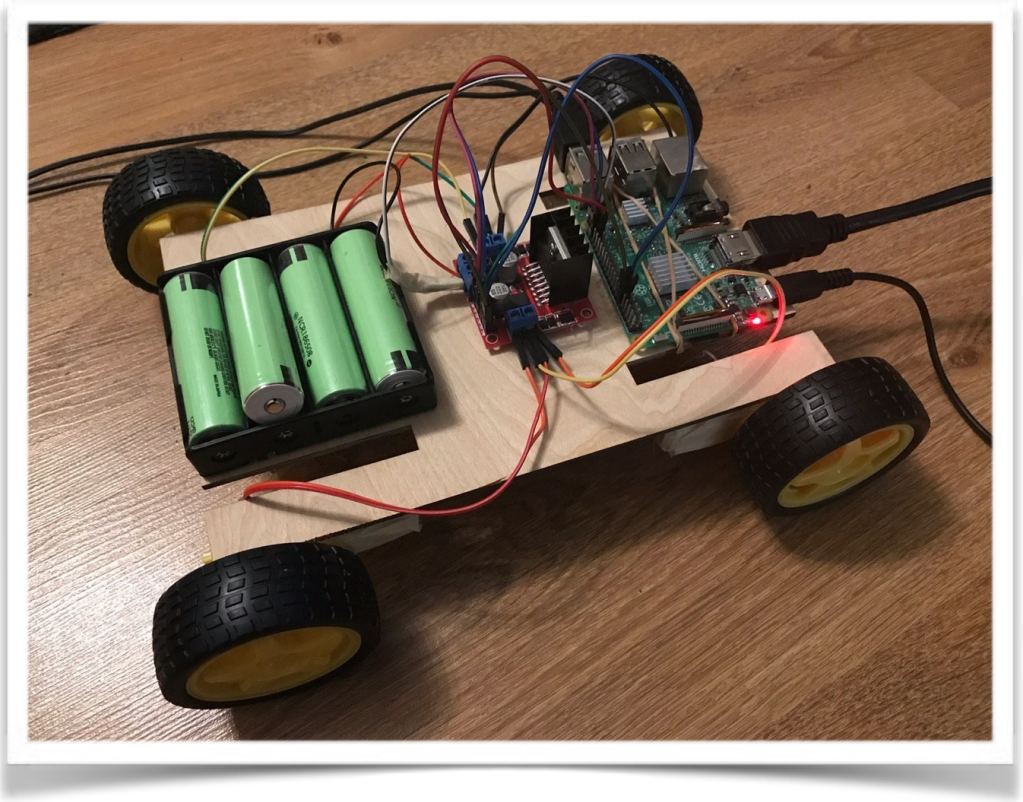


Рисунок 2.3–Робот вид зверху

Рисунок 2.4 –Робот вид збоку



Це по апаратної частини. Усередині машинки є природно програма, яка керує роботом і / або дає доступ до її управління. Ця програма називається сервером, написаний на мові JavaScript використовуючи платформу NodeJS, і все це реалізовано фреймворком Express.

2.2РОЗРОБКА СЕРВЕРНОЇ ЧАСТИНИ РОБОТА

Нейросеть буде звертатися до мого сервера з REST API, який знаходиться по url – localhost, з портом 3000. Сервер за отриманими даними думатиме куди рушити – вперед, назад, вліво, вправо і з якою потужністю. Запити до сервера йдуть по протоколу HTTP.

JavaScript ("JS"для стислості) – це повноцінний динамічний мову програмування, який застосовується до HTML документу, і може забезпечити динамічну інтерактивність на веб-сайтах. Його розробив Brendan Eich, співзасновник проекту Mozilla, Mozilla Foundation і Mozilla Corporation. JavaScript неймовірно універсальний. Ви можете почати з малого, з простих функцій, таких як каруселі, галереї зображень, що змінюються макети і відгук на натискання кнопок. Маючи великий досвід, ви зможете створювати гри, анімовану 2D і 3D графіку, повномасштабні програми з базами даних і багато іншого! JavaScript сам по собі досить компактний, але дуже гнучкий. Розробниками написано велику кількість інструментів поверх основного мови JavaScript, які розблокують величезна кількість додаткових функцій з дуже невеликим зусиллям. До них відносяться:

-Програмні інтерфейси додатка (API), вбудовані в браузери, що забезпечують різні функціональні можливості, такі як динамічне створення HTML і установку CSS стилів, захоплення і маніпуляція відеопотоком, робота з веб-камерою користувача або генерація 3D графіки і аудіо семплів.

-Програмні інтерфейси додатка (API), вбудовані в браузери, що забезпечують різні функціональні можливості, такі як динамічне створення HTML і установку CSS стилів, захоплення і маніпуляція відеопотоком,

робота з веб-камерою користувача або генерація 3D графіки і аудіо семплів.

-Сторонні API дозволяють розробникам впроваджувати функціональність в свої сайти від інших розробників, таких як Twitter або Facebook.

-Також ви можете застосувати до вашого HTML сторонні фреймворки і бібліотеки, що дозволить вам прискорити створення сайтів і додатків.

Node або Node.js – програмна платформа, заснована на движку V8 (здійснює трансляцію JavaScript в машинний код), що перетворює JavaScript з вузькоспеціалізованого мови в мову загального призначення. Node.js додає можливість JavaScript взаємодіяти з пристроями введення-виведення через свій API (написаний на C ++), підключати інші зовнішні бібліотеки, написані на різних мовах, забезпечуючи виклики до них з JavaScript-коду. Одним з найпоширеніших призначень API є надання набору широко використовуваних функцій, наприклад для малювання вікна чи іконок на екрані. Програмісти використовують переваги API у функціональності, таким чином їм не доводиться розробляти все з нуля. API є абстрактним поняттям — програмне забезпечення, що пропонує деякий API, часто називають реалізацією (англ. implementation) даного API. У багатьох випадках API є частиною набору розробки програмного забезпечення, водночас, набір розробки може включати як API, так і інші інструменти/апаратне забезпечення, отже ці два терміни не є взаємозамінювані.Високорівневі API часто програють у гнучкості. Виконання деяких функцій нижчого рівня стає набагато складнішим, або навіть неможливим.При використанні прикладного програмного інтерфейсу в контексті веб-розробки, як правило, ППІ визначається набором повідомлень запиту HTTP. Ми відійшли від теми. Node.js застосовується переважно на сервері, виконуючи роль веб-сервера, але є можливість розробляти на Node.js і десктопні віконні додатки (за допомогою NW.js, AppJS або Electron для Linux, Windows і macOS) і навіть програмувати мікроконтролери (наприклад, tessel і espruino). В основі Node.js лежить

подієво-орієнтоване і асинхронне (або реактивне) програмування з неблокуючим введенням / висновком.

Чому я використовую скризь Джаваскрипт ? Він майже не має недостатків, зручний , швидкий , найбільше у світі комьюніті. А все це можливе завдяки двигуну V8. Рушій JavaScript з відкритим сирцевим кодом. Розроблений данським відділенням компанії Google та розповсюджується за ліцензією BSD.V8 виконує JavaScript-сценарії в особливих «контекстах», які по суті є окремими віртуальними машинами. Але в одному процесі може працювати тільки одна віртуальна машина, незважаючи на можливість використання декількох потоків. У Chromium це обходиться мультипроцесовою архітектурою, підвищується також стабільність і безпека через реалізацію механізму «пісочниці».Рушій V8 відрізняється від інших рушіїв (JScript, SpiderMonkey, JavaScriptCore, Nitro) високою швидкодією та продуктивністю.

Express – це мінімалістичний і гнучкий веб-фреймворк для додатків Node.js, що надає великий набір функцій для мобільних і веб-додатків. Маючи в своєму розпорядженні безліч службових методів HTTP і проміжних оброблювачів, створити надійний API можна швидко і легко. Express надає тонкий шар фундаментальних функцій веб-додатків, які не заважають вам працювати з давно знайомими і улюбленими вами функціями Node.js.

Фреймворк, іноді фреймворк (англіцизм, неологізм від framework «остов, каркас, структура») – заготовки, шаблони для програмної платформи, що визначають архітектуру програмної системи; програмне забезпечення, що полегшує розробку і об'єднання різних модулів програмного проекту.

Доречно використання терміна «каркас». Деякі автори використовують його в якості основного, не спираючись на англомовний аналог. Можна також говорити про каркасному підході як про підхід до побудови програм, де будь-яка конфігурація програми будується з двох частин:

-Постійна частина – каркас, не змінний від конфігурації до конфігурації і несе в собі гнізда, в яких розміщується друга, змінна частина;

-Змінні модулі (або точки розширення).

REST (скорочення від англ. Representational State Transfer – «передача стану уявлення») – архітектурний стиль взаємодії компонентів розподіленого додатка в мережі. REST є узгоджений набір обмежень, що враховуються при проектуванні розподіленої гіпермедіа-системи. У певних випадках (інтернет-магазини, пошукові системи, інші системи, засновані на даних) це призводить до підвищення продуктивності і спрощення архітектури. У широкому сенсі [уточнити] компоненти в REST взаємодіють на зразок взаємодії клієнтів і серверів у Всесвітній павутині. REST є альтернативою RPC. В мережі Інтернет виклик віддаленої процедури може являти собою звичайний HTTP-запит (зазвичай «GET» або «POST»; такий запит називають «REST-запит»), а необхідні дані передаються в якості параметрів запиту. Для веб-служб, побудованих з урахуванням REST (тобто не порушують накладаються їм обмежень), застосовують термін «RESTful». На відміну від веб-сервісів (веб-служб) на основі SOAP, не існує «офіційного» стандарту для RESTful веб-API. Справа в тому, що REST є архітектурним стилем, в той час як SOAP є протоколом. Незважаючи на те, що REST не є стандартом сам по собі, більшість RESTful-реалізацій використовують стандарти, такі як HTTP, URL, JSON і XML.

HTTP (англ. HyperText Transfer Protocol – «протокол передачі гіпертексту») – протокол прикладного рівня передачі даних спочатку – у вигляді гіпертекстових документів в форматі «HTML», зараз використовується для передачі довільних даних. Основою HTTP є технологія «клієнт-сервер», тобто передбачається існування:

-Споживачів (клієнтів), які ініціюють з'єднання і надсилають запит;

-Постачальників (серверів), які очікують з'єднання для отримання запиту, виробляють необхідні дії і повертають назад повідомлення з результатом.

HTTP в даний час повсюдно використовується у Всесвітній павутині для отримання інформації з веб-сайтів. У 2006 році – в Північній Америці

частка HTTP-трафіку перевищила частку P2P-мереж і склала 46%, з яких майже половина – це передача потокового відео і звуку. HTTP використовується також як «транспорт» для інших протоколів прикладного рівня, таких як SOAP, XML-RPC, WebDAV. Основним об'єктом маніпуляції в HTTP є ресурс, на який вказує URI (Uniform Resource Identifier) в запиті клієнта. Зазвичай такими ресурсами є що зберігаються на сервері файли, але ними можуть бути логічні об'єкти або щось абстрактне. Особливістю протоколу HTTP є можливість вказати в запиті і відповіді спосіб представлення одного і того ж ресурсу за різними параметрами: формату, кодуванні, мови і т. Д. (Зокрема, для цього використовується HTTP-заголовок). Саме завдяки можливості вказівки способу кодування повідомлення, клієнт і сервер можуть обмінюватися двійковими даними, хоча даний протокол є текстовим. HTTP – протокол прикладного рівня; аналогічними йому є FTP і SMTP. Обмін повідомленнями йде по звичайній схемі «запит-відповідь». Для ідентифікації ресурсів HTTP використовує глобальні URI. На відміну від багатьох інших протоколів, HTTP не зберігає свого стану. Це означає відсутність збереження проміжного стану між парами «запит-відповідь». Компоненти, що використовують HTTP, можуть самостійно здійснювати збереження інформації про стан, пов'язаної з останніми запитами і відповідями (наприклад, «куки» на стороні клієнта, «сесії» на стороні сервера). Браузер, який посилає запити, може відстежувати затримки відповідей. Сервер може зберігати IP-адреси і заголовки запитів останніх клієнтів. Однак сам протоколу не обізнаний про попередні запити і відповідях, в ньому не передбачена внутрішня підтримка стану, до нього не пред'являються такі вимоги.

Почнемо з установки сервера. На роботі встановлена ОС Raspbian Stretch. Для початку нам треба встановити node і npm для роботи з сервером. Але для початку оновимо репаміторіі.

```
sudo apt update
```

```
sudo apt install npm
```

Разом з npm йде відразу залежність node, так що нам не треба буде встановлювати окремо. Після цього треба встановити opencv, zmq для нашої нейромережі. На Raspbian відразу встановлено python3.5, так що встановлювати нам його не треба буде.Доречі про «apt». Так як мі працюємо на Debian-дистрибутиві - я не можу не згадати про нього.

apt – програма для встановлення, оновлення і вилучення програмних пакунків в операційних системах Debian і заснованих на них (Ubuntu, Linux Mint тощо). Apt також застосовується в деяких дистрибутивах на основі пакетного менеджера RPM, таких як Mandriva і ALT Linux. Здатна автоматично встановлювати і налаштовувати програми UNIX-подібних операційних систем як із заздалегідь скомпільованих пакунків, так і з джерельного коду.

```
pip3 install opencv-pythonzmq
```

pip – система керування пакунками, яка використовується для встановлення та управління програмними пакетами, які написані на Python. Багато пакетів можна знайти в Python Package Index (PyPI).Починаючи з версій Python 2.7.9 та Python 3.4, вони містять пакет pip (або pip3 для Python 3) за умовчанням.pip є рекурсивним акронімом, що означає «Pip Installs Packages» або «Pip Installs Python».Тепер все що нам треба це написати код сервера. Почнемо з залежностей. Для ініціювання сховища npm давайте напишемо:

```
npm init
```

Скрізь натискайте Enter і у вас створиться файл package.json – це файл

який зберігає в собі всі залежності проекту (без вихідного коду), в цей файл давайте закинемо потрібні нам залежності:

```
{
"name": "test",
"version": "1.0.0",
"description": "",
"main": "index.js",
"scripts": {
"start": "nodemon"
  },
"author": "",
"license": "ISC",
"dependencies": {
"body-parser": "^1.19.0",
"cors": "^2.8.5",
"express": "^4.17.0",
"nodemon": "^1.19.0",
"path": "^0.12.7",
"raspi-gpio": "^6.2.1",
"rpi-gpio": "^2.1.3"
  }
}
```

Файл package.json містить в собі інформацію про вашому додатку: назва, версія, залежно тощо. Будь-яка директорія, в якій є цей файл, інтерпретується як Node.js-пакет, навіть якщо ви не збираєтеся публікувати його. Спосіб використання файлу package.json залежить від того, чи збираєтеся ви завантажувати пакет або публікувати його.

Після цього треба встановити всі ці залежності, для цього треба ввести

```
npm i
```

Ця команда встановить всі в папку node_modules. Тепер можемо приступити до логіки сервера. Точку входу, надоєм за стандартом index.js.

```
const express = require('express'),
  path = require('path'),
  bodyparser = require('body-parser'),
```

```
  cors = require('cors');

const raspi = require('raspi');
const gpio = require('raspi-gpio');
const pwm = require('raspi-pwm');

const app = express();
app.use(express.static('public'));
app.use(bodyparser.json());
app.use(cors());

raspi.init(() => {
  const pins = {
in_1: new gpio.DigitalOutput('P1-13'),
    in_2: new gpio.DigitalOutput('P1-11'),
    in_3: new gpio.DigitalOutput('P1-40'),
    in_4: new gpio.DigitalOutput('P1-38'),
    pwm_1: new pwm.PWM('P1-12'),
    pwm_2: new pwm.PWM('P1-35'),
  };

  app.use((req, res, next) => {
    req.payload = {
      pins: pins
    };
    next();
  });

  app.use(require('./routes'));

  var port = process.env.PORT || 3000;
  app.listen(port, () => {
    console.log('Listening on port ' + port);
  });
});
```

-raspi – бібліотека для роботи з пинами raspberry pi і все його модулі PWM відповідно.

-express – наш Фреймворк про який я писав вище

-bodyparser – це Middleware для парсинга тіла запиту з JSON в читається мовою об’єкт.

Тут ми створюємо сервер і включаємо його на порту 3000, так само включавем потрібні нам Піни.

Далі нам потрібно прописати логіку переміщення. Це у нас є в файлі controls.js в папці routes.

Термін middleware часто використовують для позначення інфраструктури: систем керування базами даних, веб-серверів, серверів застосунків, систем керування змістом, і тому подібних інструментів, які використовуються в процесі розробки й експлуатації застосунків. Проміжне програмне забезпечення складає ядро сучасних застосунків, заснованих на XML, SOAP, веб-сервісах і сервісно-орієнтованій архітектурі. Впровадженням концепції ППЗ активно займається консорціум «Інтернет2».

Так як NodeJS Express –асінхронний сервер, нам потрибно завжди чікаті видповіді від функції. Від цього код здається дуже дивним, але в асінхронності його плюс. У синхронному коді кожна операція чекає закінчення попередньої. Тому вся програма може «зависнути», якщо одна з команд виконується дуже довго.Асинхронний код прибирає операцію, яка блокує основний потік програми, так що основний потік не блокується, і програма може виконувати інші операції.Асинхронне програмування успішно вирішує безліч завдань. Одна з найважливіших – користувач може взаємодіяти з програмою поки виконується інше завдання.Візьмемо для прикладу додаток, яке підбирає серіали за зазначеними критеріями. Після того як користувач обрав параметри по якім буде відбуватися пошук, програма відправляє запит на сервер. А там здійснюється пошук вказаних фільмів. Обробка може тривати доволі тривалий час. Якщо додаток працює синхронно, то користувач не зможе взаємодіяти зі сторінкою, поки не прийде результат.В цьому випадку головний потік виконання поділяється на дві гілки. Одна з них продовжує займатися інтерфейсом, а інша виконує запит (Рис 2.5).

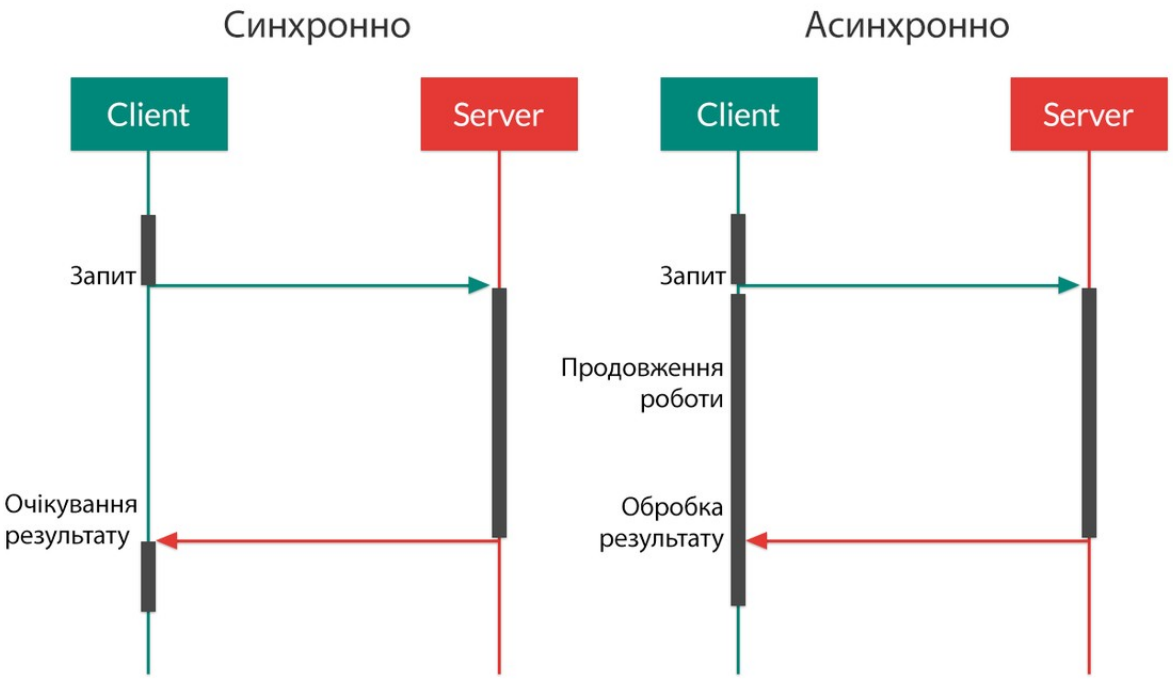


Рисунок 2.5 –Приклад роботи асинхронного та синхронного коду

```
const router = require('express').Router();
const gpio = require('raspi-gpio');

var timeout = null;

router.get('/', function (req, res, next) {
  res.send('Hello');
});

router.post('/joystick', function (req, res, next) {

  const pins = req.payload.pins;

  const x = req.body.x;
  const y = req.body.y;

  var force = {
    a: parseFloat(1 – Math.abs(x)),
    b: Math.abs(y)
  }

  console.log(force);

  if(x == 0 && y == 0) return res.end();

  if(x > 0 && y == 0) {
    pins.in_2.write(gpio.LOW);
```

```
  pins.in_1.write(gpio.HIGH);
  pins.in_3.write(gpio.LOW);
  pins.in_4.write(gpio.HIGH);
  pins.pwm_1.write(1);
  pins.pwm_2.write(1);
}
else if(x < 0 && y == 0) {
  pins.in_2.write(gpio.HIGH);
  pins.in_1.write(gpio.LOW);
  pins.in_3.write(gpio.HIGH);
  pins.in_4.write(gpio.LOW);
  pins.pwm_1.write(1);
  pins.pwm_2.write(1);
}


else if(x < 0 && y > 0) {
  pins.in_1.write(gpio.HIGH);
  pins.in_2.write(gpio.LOW);
  pins.in_3.write(gpio.HIGH);
  pins.in_4.write(gpio.LOW);
  pins.pwm_1.write(force.a);
  pins.pwm_2.write(force.b);
}
else if(x < 0 && y < 0) {
  pins.in_1.write(gpio.LOW);
  pins.in_2.write(gpio.HIGH);
  pins.in_3.write(gpio.LOW);
  pins.in_4.write(gpio.HIGH);
  pins.pwm_1.write(force.b);
  pins.pwm_2.write(force.a);
}
else if(x > 0 && y < 0) {
  pins.in_1.write(gpio.LOW);
  pins.in_2.write(gpio.HIGH);
  pins.in_3.write(gpio.LOW);
  pins.in_4.write(gpio.HIGH);
  pins.pwm_1.write(force.a);
  pins.pwm_2.write(force.b);
}
else if(x > 0 && y > 0) {
  pins.in_1.write(gpio.HIGH);
  pins.in_2.write(gpio.LOW);
  pins.in_3.write(gpio.HIGH);
  pins.in_4.write(gpio.LOW);
  pins.pwm_1.write(force.b);
  pins.pwm_2.write(force.a);
}

clearTimeout(timeout)

timeout = setTimeout(() => {
```

```
    pins.in_2.write(gpio.LOW);
    pins.in_1.write(gpio.LOW);
    pins.in_3.write(gpio.LOW);
    pins.in_4.write(gpio.LOW);
    pins.pwm_1.write(0);
    pins.pwm_2.write(0);
  }, 100);

  res.end();
});

module.exports = router;
```

Трохи про код – по суті тут знаходяться різні комбінації для двигунів, in_1, in_2 – це ліва сторона машинки, якщо подати на два входи 1 і 0 то ця сторона поїде вперед, якщо 0 і 1 то назад. Теж саме і для другої сторони зі входами in_3, in_4. Є 2 варіанти управління – Джойстик і Дискретне управління. Джойстик це плавне управління – на вхід сервера треба подавати координати від 0 до 1 по x і y в такому вигляді (формат JSON)

```
{ «x»: 0.5, «y»: 0.9 }
```

Raspbian (Рис 2.6) – заснована на Debian операційна система для Raspberry Pi. Існує кілька версій Raspbian, в тому числі Raspbian Stretch і Raspbian Jessie. З 2015 року Raspbian офіційно представлена Raspberry Pi Foundation в якості основної операційної системи для одноплатних комп'ютерів Raspberry Pi. Raspbian була створена Майком Томпсоном і Пітером Гріном в якості незалежного проекту. Первісна збірка була виконана в червні 2012 року. Операційна система знаходиться в стадії активної розробки. Raspbian оптимізована для нізкопроїзводітельних процесорів ARM, які використовуються в лінійці комп'ютерів Raspberry Pi. В якості основної середовища робочого столу в Raspbian використовується PIXEL (Pi Improved Xwindows Environment, Lightweight). Вона складається з модифікованої середовища робочого столу LXDE і менеджера вікон Openbox з новою темою і декількома іншими змінами. Дистрибутив поставляється з копією програми

комп'ютерної алгебри Mathematica, особливою версією Minecraft під назвою Minecraft Pi, а також з полегшеною останньою версією браузера Chrome.Система керування пакунками програмного забезпечення Debian (особливо APT) відома своєю суворою політикою стосовно пакунків та якості випусків. APT дозволяє легко оновлювати систему, а також автоматично розв'язувати залежності між пакунками. Debian використовує повністю прозорий процес розробки і тестування. Система розробляється волонтерами з усього світу і підтримується пожертвами, які збирає некомерційна організація «Програмне забезпечення в інтересах суспільства».

Unix сімейство переносите, багатозадачних і багатокористувацьких операційних систем, які засновані на ідеях оригінального проекту AT & T Unix, розробленого в 1970-х роках в дослідницькому центрі Bell Labs Кеном Томпсоном, Деннісом Рітчі та іншими. Операційні системи сімейства Unix характеризуються модульним дизайном, в якому кожна задача виконується окремою утилітою, взаємодія здійснюється через єдину файлову систему, а для роботи з утилітами використовується командна оболонка. Ідеї, закладені в основу Unix, справили величезний вплив на розвиток комп'ютерних операційних систем. В даний час Unix-системи визнані одними з найбільш історично важливих ОС. Операційні системи сімейства Unix характеризуються модульним дизайном, в якому кожна задача виконується окремою утилітою, взаємодія здійснюється через єдину файлову систему, а для роботи з утилітами використовується командна оболонка. Ідеї, закладені в основу Unix, справили величезний вплив на розвиток комп'ютерних операційних систем. В даний час Unix-системи визнані одними з найбільш історично важливих ОС.Основное відміну Unix-подібних систем від інших операційних систем полягає в тому, що це спочатку розраховані на багато користувачів багатозадачні системи. В Unix може одночасно працювати відразу багато людей, кожен за своїм терміналом, при цьому кожен з них може виконувати безліч різних обчислювальних процесів, які будуть використовувати ресурси саме цього комп'ютера.

Друга колосальна заслуга Unix – в її мультіплатформенності. Ядро системи розроблено таким чином, що його легко можна пристосувати практично під будь-який мікропроцесор.

Unix має і інші характерні особливості:

-використання простих текстових файлів для настройки та управління системою;

-широке застосування утиліт, що запускаються з командного рядка;

-взаємодія з користувачем за допомогою віртуального пристрою – терміналу;

-уявлення фізичних і віртуальних пристроїв і деяких засобів межпроцессового взаємодії у вигляді файлів;

-використання конвеєрів з декількох програм, кожна з яких виконує одну задачу.

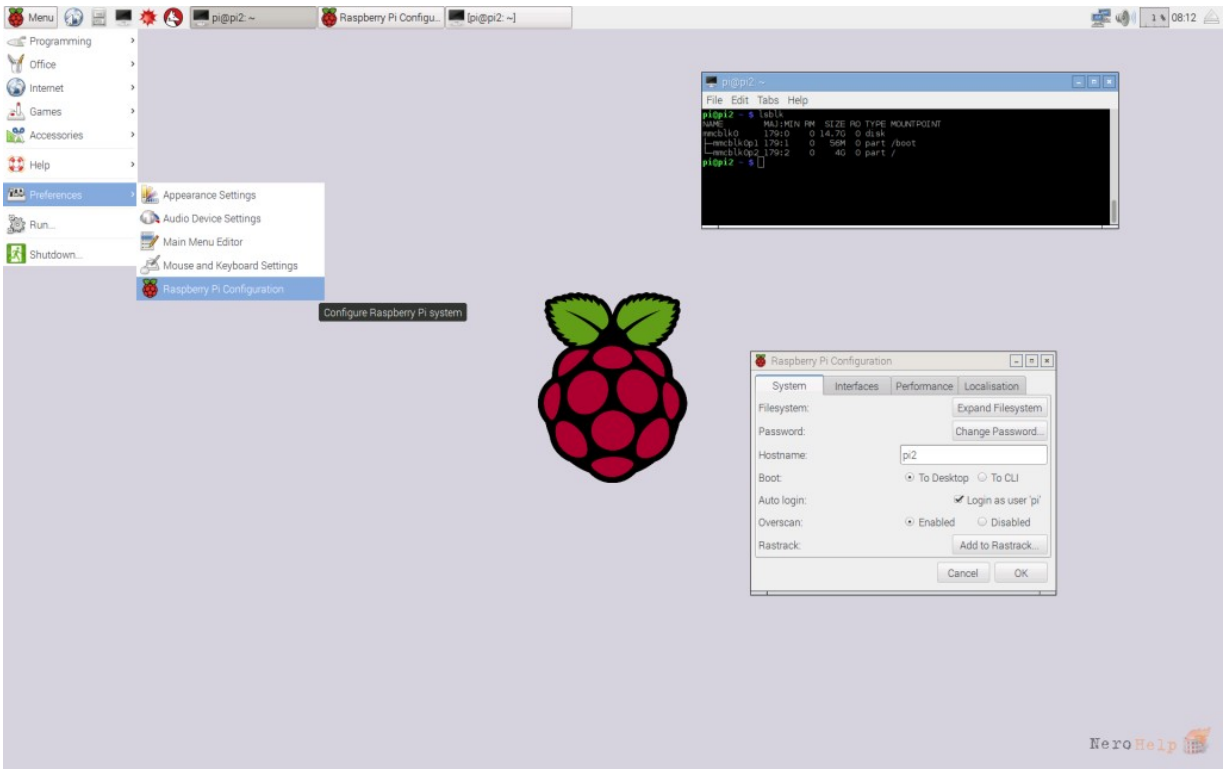


Рисунок 2.6– Графічна оболочка Raspbian

NPM – с допомогою npm можна встановлювати пакети локально або глобально. У локальному режимі пакети встановлюються в каталог node_modules батьківського каталогу. Власником каталогу є поточний користувач. Глобальні пакети встановлюються в каталог {prefix} / lib / node_modules /, власником якого є root (префіксом в даному випадку зазвичай є каталог / usr / або / usr / local). Це означає, що вам треба використовувати sudo для глобальної установки пакетів, що може спричинити помилки з повноваженнями при вирішенні сторонніх залежностей, а також створює проблему для безпеки.

Широтно-імпульсна модуляція (ШІМ, англ. Pulse-width modulation (PWM)) – процес управління потужності методом пульсірующёго включення і виключення приладу. Разлічаютают аналогову ШІМ і цифрову ШІМ, двійкову (дворівневу) ШІМ і трійкову (трирівневу) ШІМ. Основною причиною застосування ШІМ є прагнення до підвищення ККД при побудові вторинних джерел пітаніяелектронной апаратури і в інших вузлах,

наприклад, ШІМ використовується для регулювання яскравості підсвічування LCD-моніторів і дисплеїв в телефонах, КПК і т.п .

JSON – текстовий формат обміну даними, заснований на JavaScript. Як і багато інших текстові формати, JSON легко читається людьми. Формат JSON був розроблений Дугласом Крокфорд. Незважаючи на походження від JavaScript (точніше, від підмножини мови стандарту ECMA-262 1999 роки), формат вважається незалежним від мови і може використовуватися практично з будь-якою мовою програмування. Для багатьох мов існує готовий код для створення і обробки даних в форматі JSON.За рахунок своєї лаконічності в порівнянні з XML, формат JSON може бути більш підходящим для сериализации складних структур. Якщо говорити про веб-додатках, в такому ключі він доречний в задачах обміну даними як між оглядачем і сервером (AJAX), так і між самими серверами (програмні HTTP-сполучення). Оскільки формат JSON є підмножиною синтаксису мови JavaScript, то він може бути швидко десеріалізован вбудованою функцією eval (). Крім того, можлива вставка цілком працездатних JavaScript-функцій. У мові PHP, починаючи з версії 5.2.0, підтримка JSON включена в ядро у вигляді функцій json_decode () і json_encode (), які самі перетворять типи даних JSON в відповідні типи PHP і навпаки.

2.3РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ДЛЯ ТЕСТУВАННЯ

Для тестування машинки без нейромереж я зробив дві програми, одна – додаток на десктопний комп'ютер, іншу на телефони з системою iOS. На телефон додаток написано на мові Javascript з Vue.js використовуючи платформу Apache Cordova. Програма представляє собою Віртуальний джойстик який при використанні передає через AJAX дані на сервер. IP сервера можна вказувати зверху додатки (Рис 2.7).

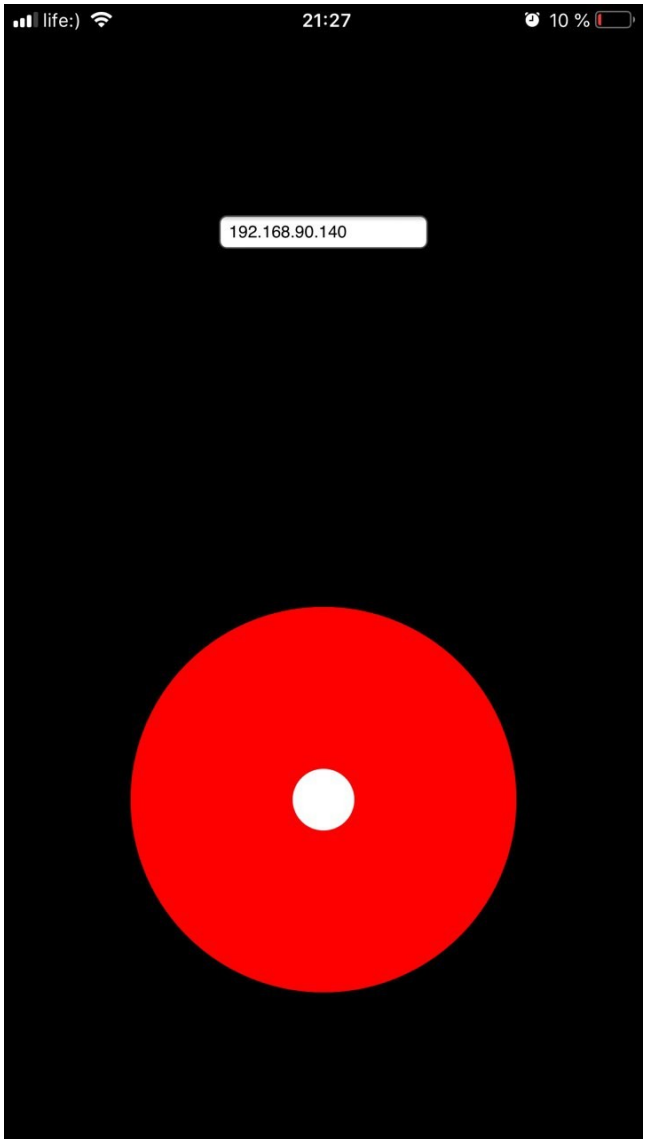


Рисунок 2.7–Додаток для управління роботом на iOS

Що б поставити Cordova: На комп'ютер треба мати NodeJS, npm. Далі просто введіть:

```
npm i -g cordova
```

i – shortcut «install»

-g – global (установка на весь комп’ютер)

Далі що б створити проект введіть:

```
cordova create my-app
```

Тепер приступимо до коду. Як завжди почнемо з npm. Встановлюємо Vue.js, на якому буде написано додаток.

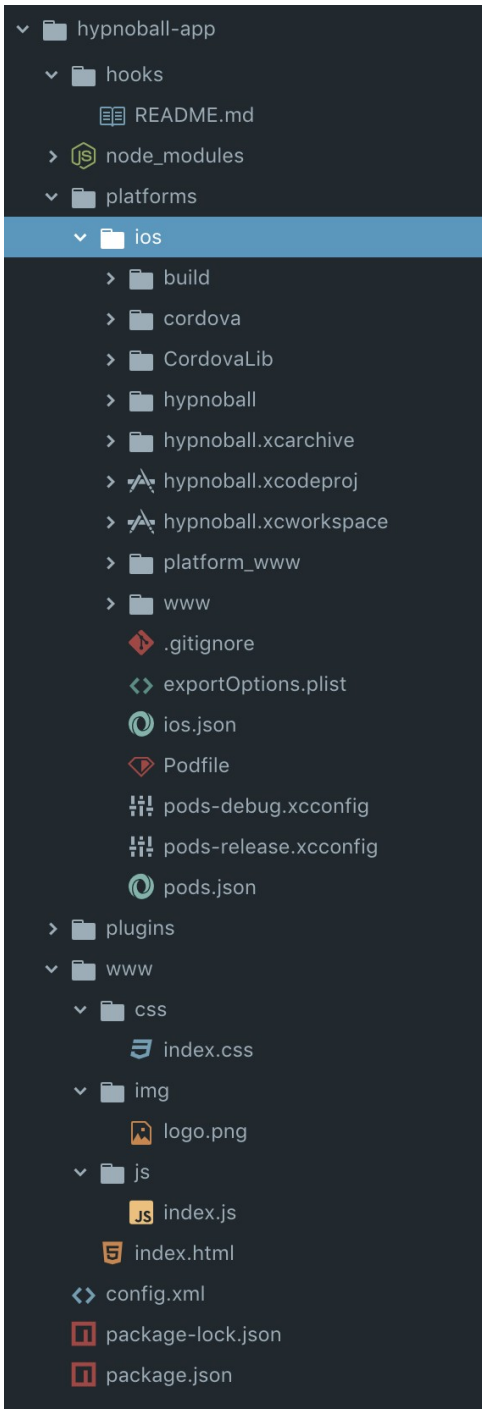
```
npm i vue
```

Наш package.json:

```
{
"name": "com.bytesbay.hypnoball",
"displayName": "hypnoball",
"version": "1.0.0",
"description": "A sample Apache Cordova application that responds to the deviceready event.",
"main": "index.js",
"scripts": {
"test": "echo \"Error: no test specified\"&& exit 1"
  },
"keywords": [
"ecosystem:cordova"
  ],
"author": "Apache Cordova Team",
"license": "Apache-2.0",
"dependencies": {
"cordova-ios": "^5.0.1",
"cordova-plugin-statusbar": "^2.4.2"
  },
"devDependencies": {
"cordova-plugin-whitelist": "^1.3.3"
  },
"cordova": {
"plugins": {
"cordova-plugin-whitelist": {},
"cordova-plugin-statusbar": {}
    },
"platforms": [
"ios"
    ]
  }
}
```

Тепер тут присутні ще й пакети Кордови, whitelist нам потрібен що б відправляти запити на інший домен, інакше програма не пустить. Тепер

index.js, index.css, index.html. А так же для наочності покажу файлову структуру програми (Рис 2.8). В папці platforms знаходяться проекти для різний платформ, але їх треба додавати командою:




```
cordova platform add ios
cordova platform add android
```

Рисунок 2.8– Файлова структура додатку

В папці plugins знаходяться різні плагіни які ми завантажуємо для додатка в ній нічого особливого.

```
Number.prototype.map = function (in_min, in_max, out_min, out_max) {
  return (this – in_min) * (out_max – out_min) / (in_max – in_min) + out_min;
}

var size = {
  width: 40,
  height: 40,
};

var radius = 250;

var pos = {
  x: radius / 2,
  y: radius / 2,
};

var starting = {
  x: 0,
  y: 0,
};

var flag = false;
var block = false;

var joystick = document.getElementById('joystick');

joystick.addEventListener('touchstart', function (e) {

  console.log('TAP');

  console.log(e.touches[0]);

  starting.x = (e.layerX ? e.layerX : e.touches[0].clientX);
  starting.y = (e.layerY ? e.layerY : e.touches[0].clientY);

  flag = true;

});
```

Після того як ми зробили прослуховування цьієї події - ми можемо відстежувати коли нажмуть на джойстик. Для відстежування руху по

джойстику ми будемо використовувати touchmove.

```
joystick.addEventListener('touchmove', function (e) {
  if(!flag) return;

  var rect = this.getBoundingClientRect();

  pos.x = (e.layerX ? e.layerX : (e.touches ? e.touches[0].clientX – rect.left : 0));
  pos.y = (e.layerY ? e.layerY : (e.touches ? e.touches[0].clientY – rect.top : 0));

  e.preventDefault();

  render();
});


joystick.addEventListener('touchend', function (e) {
  if(!flag) return;
  flag = false;

  pos.x = (radius / 2);
  pos.y = (radius / 2);

  render();
});

function render() {

  var el = document.getElementById('controller');

  var x = pos.x – (size.width / 2);
  var y = pos.y – (size.height / 2);

  el.style.transform = `translate(${x}px, ${y}px)`;

  x = (pos.x – (radius / 2)).map(0, radius / 2, 0, 1);
  y = (pos.y – (radius / 2)).map(0, radius / 2, 0, 1);

  var ip = document.getElementById('ip').value;

  if(!block) {

    block = true;

    fetch('http://' + ip + ':3000/controls/joystick', {
      method: 'POST',
      headers: {
        'Content-type': 'application/json',
      },
      body: JSON.stringify({
        x: x.toFixed(1), y: -y.toFixed(1),
```

```
        })
      }).then(res => {
        block = false;
      });
    }

  }

  render();
```

Для работи джойстика нам необхідно додати CSS , тому що він просто не зможе рухатися. Нижче - код для стілизації джойстика. Я зробив додаток максимально чорним задля того щоб на моему телефоні було чітко бачно джойстик який я зробив червоним , а сам стик - білим. Такий интерфейс не викликає непріємних відчуттів для очей.

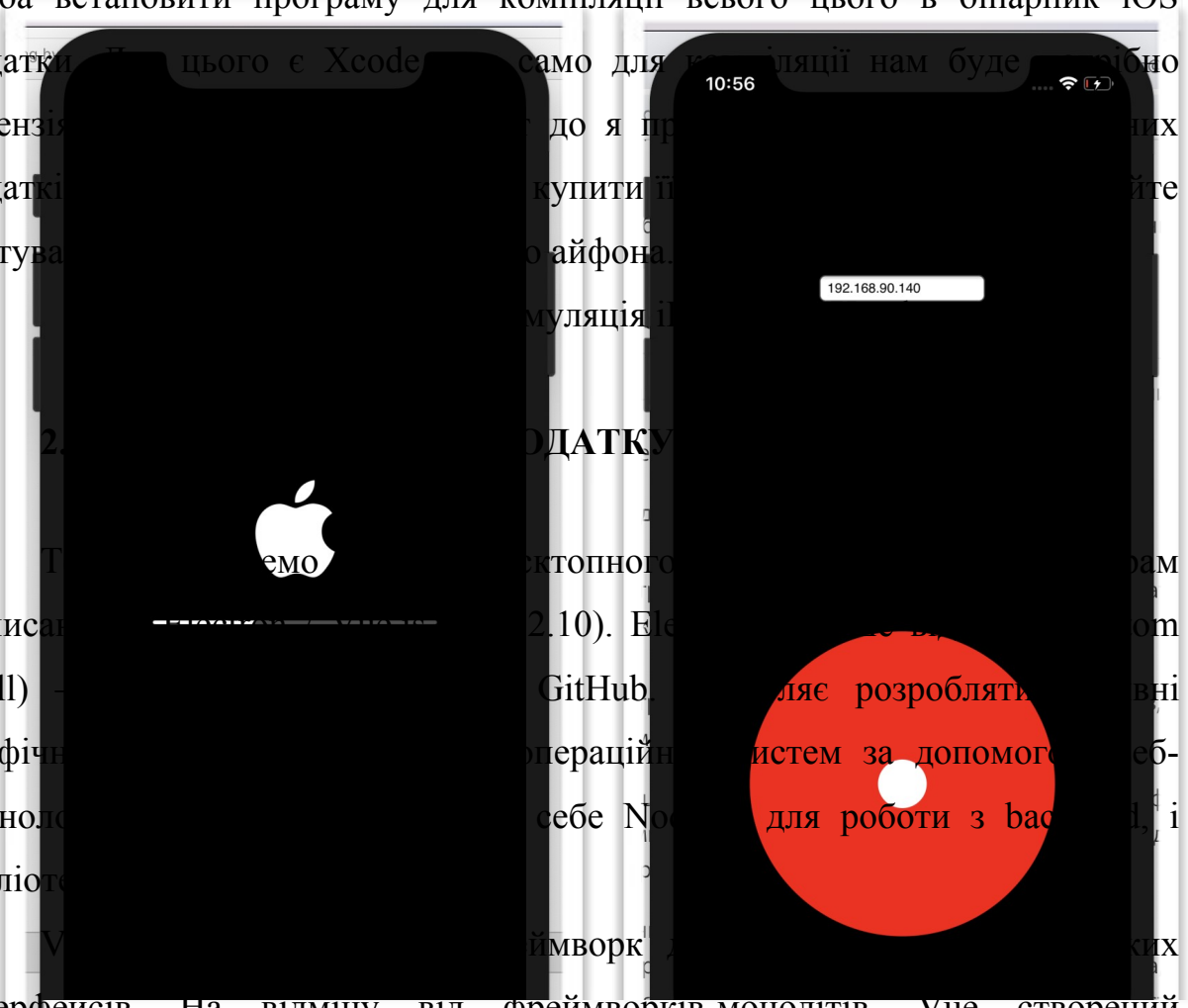
```
  * {
     -webkit-tap-highlight-color: rgba(0,0,0,0); /* make transparent link selection, adjust last value opacity 0 to 1.0 */
  }

  body {
    -webkit-touch-callout: none;                /* prevent callout to copy image, etc when tap to hold */
    -webkit-text-size-adjust: none;             /* prevent webkit from resizing text to fit */
     -webkit-user-select: none;                   /* prevent copy paste, to allow, change 'none' to 'text' */
    background-color: #000;
    background-image:linear-gradient(top, #A7A7A7 0%, #E4E4E4 51%);
     font-family: system-ui, -apple-system, -apple-system-font, 'Segoe UI', 'Roboto', sans-serif;
    font-size: 12px;
    height: 100vh;
    margin: 0px;
    padding: 0px;
    /* Padding to avoid the "unsafe" areas behind notches in the screen */
     padding: env(safe-area-inset-top, 0px) env(safe-area-inset-right, 0px) env(safe-area-inset-bottom, 0px) env(safe-area-inset-right, 0px);
    text-transform:uppercase;
    width: 100%;
    overflow: hidden;
  }

  #app {
    height: 100%;
    display: flex;
    flex-direction: column;
```

```
  align-items: center;
  justify-content: space-around;
}

#joystick {
  position: relative;
  width: 250px;
  height: 250px;
  border-radius: 50%;
  background: red;
}

#controller {
  width: 40px;
  height: 40px;
  border-radius: 50%;
  /* border: 2px solid #333; */
  background: #fff;
}
```

На початку ми підключаємо функцію математичного діапазону, вона нам потрібна для більш зручної роботи з анімацією, часто її використовують у великих проектах, вона дуже дуже проста і красива одночасно. Далі ми задаємо наші константи для подальшої роботи що б уникнути повторення коду, і потім вже починаємо роботу з документом, і елементом. Ми знаходимо елемент джойстика, де ми будемо вставляти наш «Стік» для управління і повісимо на нього слушаютель «Тач» події. Щоб надати якісну підтримку touch-based призначеного для користувача інтерфейсу, тач-події пропонують можливість інтерпретації впливу пальця (або стилуса) на тач-екрани або трекболи. Інтерфейси тач-подій є відносно низькорівневим API який може бути використаний для підтримки додатків зі специфічними мультитач взаємодіями, наприклад з жестом двох пальців. Мультитач взаємодія запускається коли палець (або стилус) вперше стосується контакний поверхні. Інші пальці можуть потім натиснути поверхню і опціонально рухатися по ній. Взаємодія закінчується коли пальці видаляються з поверхні. Під час взаємодії, додаток отримує тач події на початку, під час і наприкінці фаз. Тач-події подібні подіям миші за винятком

що вони підтримують одночасні торкання і різні розташування на поверхні торкання. Інтерфейс TouchEvent инкапсулирует все точки дотику які зараз активні. Інтерфейс Touch, який представляє одну точку дотику, включає інформацію таку як позиція точки дотику щодо viewport браузера. Беремо координати пальця і провертає з ними магію математики, з вище зазначеною функцією. Зрештою компілюємо і заливаємо на телефон через Xcode.

І так, у нас є код, тепер треба перевірити його роботу, для цього нам треба встановити програму для компіляції всього цього в бінарник iOS додатки. Для цього є Xcode, само для компіляції нам буде потрібно ліцензія до я пр них додатк купити і йте тестува айфона. муляція і



2. ОДАТКУ

Т ктопного ам написан 2.10). El om shell) GitHub. ляє розроблять вні графічн операційн истем за допомог еб-технол себе No для роботи з ba d, і бібліот

V ймворк ких інтерфейсів. На відміну від фреймворків-монолітів, Vue створений придатним для поступового впровадження. Його ядро в першу чергу вирішує завдання рівня уявлення (view), що спрощує інтеграцію з іншими бібліотеками та існуючими проектами. З іншого боку, Vue повністю підходить і для створення складних односторінкових додатків (SPA), якщо використовувати його спільно з сучасними інструментамії додатковими бібліотеками.

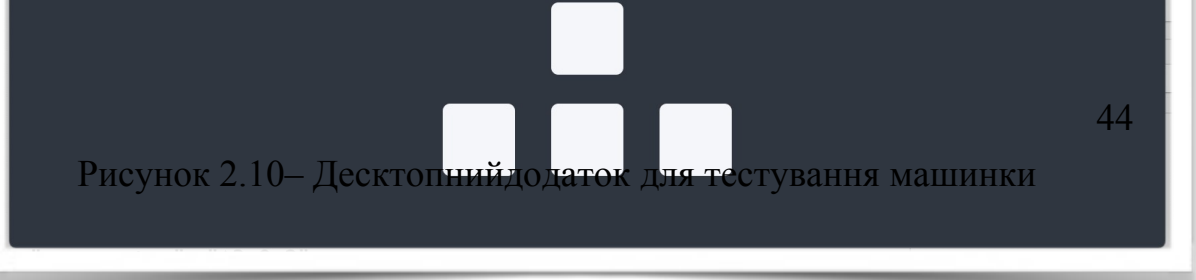


Рисунок 2.10– Десктопнийдодаток для тестування машинки

```
{
"name": "hypnoball",
"version": "1.0.0",
"description": "",
"main": "index.js",
"dependencies": {
"bootstrap": "^4.1.3",
"electron": "^3.0.10",
"vue-router": "^3.0.2"
  },
"devDependencies": {
"axios": "^0.18.0",
"cross-env": "^5.1",
"jquery": "^3.2",
"laravel-mix": "^2.0",

"lodash": "^4.17.4",
"vue": "^2.5.7"
  },
"scripts": {
"start": "electron .",
"dev": "npm run development",
"development": "cross-env NODE_ENV=development node_modules/webpack/bin/webpack.js --progress --hide-modules --config=node_modules/laravel-mix/setup/webpack.config.js",
"watch": "npm run development -– --watch",
"watch-poll": "npm run watch -– --watch-poll",
"hot": "cross-env NODE_ENV=development node_modules/webpack-dev-server/bin/webpack-dev-server.js --inline --hot --config=node_modules/laravel-mix/setup/webpack.config.js",
"prod": "npm run production",
"production": "cross-env NODE_ENV=production node_modules/webpack/bin/webpack.js --no-progress --hide-modules --config=node_modules/laravel-mix/setup/webpack.config.js"
  },
"author": "bytesbay",
"license": "ISC"
}
```

Це стандартний package.json для початкового проекту Vue.js. Декілька скриптів для компіляції коду і запуск дев-серверу. Далі йде основний файл додатку - в ньому ми створюємо вікно у якому ми виводимо html файл.

```
const electron = require('electron');
```

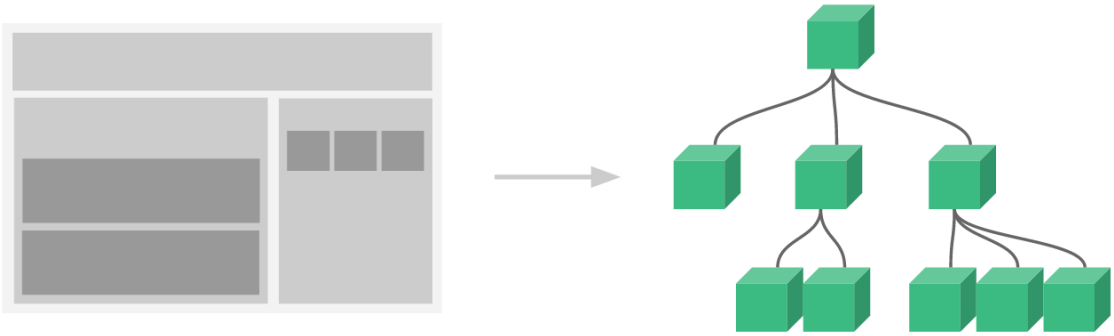
```
const path = require('path');
const url = require('url');

// SET ENV
process.env.NODE_ENV = 'development';

const {app, BrowserWindow, Menu, IPCMain} = electron;

let main_window;

// Listen for app to be ready
app.on('ready', function(){
  // Create new window
  main_window = new BrowserWindow({frame: false, titleBarStyle: 'hiddenInset'});
  // Load html in window
  main_window.loadURL(url.format({
    pathname: path.join(__dirname, 'index.html'),
    protocol: 'file:',
    slashes:true
  }));
  // Quit app when closed
  main_window.on('closed', function(){
    app.quit();
  });

  // Insert menu
  Menu.setApplicationMenu(Menu.buildFromTemplate(menu_template));
});

// Catch item:add
// ipcMain.on('item:add', function(e, item){
//   mainWindow.webContents.send('item:add', item);
//   addWindow.close();
//   // Still have a reference to addWindow in memory. Need to reclaim memory
(Grabage collection)
//   //addWindow = null;
// });

// Create menu template
const menu_template = [
  // Each object is a dropdown
  {},
  {
    label: 'Разработка',
    submenu: [
      {
        label: 'Инструменты разработки',
        click() {
          main_window.webContents.openDevTools()
        }
      }
    ]
```

```
  }
];
```

Ну і тут зовсім простий код в html файлі, по суті ми замінюємо #app на наше JS-додаток. Після рендеру сторінки в інструментах розробника ви побачите що там тепепрь знаходиться зовсім інший код.

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
<head>
<meta charset="utf-8">
<title>Hypno Ball</title>
<link rel="stylesheet" href="static/css/app.css">
</head>
<body>
<div id="app">
<header></header>
<router-view></router-view>
</div>
<script src="static/js/app.js" charset="utf-8"></script>
</body>
</html>
```

Це називається root-компонент , від нього йдуть усі дочерні компоненти. Важливою концепцією Vue є компоненти. Ця абстракція дозволяє збирати великі програми з маленьких «шматочків». Вони являють



собою придатні до повторного використання об'єкти. Якщо подумати, майже будь-який інтерфейс можна уявити як дерево компонентів (Рис. 2.11). У Vue компонент - це, по суті, екземпляр Vue з попередньо встановленими опціями.

Рисунок 2.11 – Дерево компонентів Vue.

```
import Vue from 'vue';
import VueRouter from 'vue-router';

Vue.use(VueRouter);

import Index from './views/Index.vue';

const router = new VueRouter({
  routes: [
          { path: '/', component: Index, name: 'index' }
        ]
});

const app = new Vue({
        el: '#app',
        router,
        data: {
                hello: 'Hello'
        },
});
```

```
<template lang="html">
<main>
<controls/>
</main>
</template>

<script>
import Controls from './../components/Controls.vue';

export default {
  components: {
    Controls
  }
}
</script>

<style lang="scss" scoped>

</style>
```

Дуже маленький компонент , це зроблено задля того щоб покращити майбутню скалізацію проекта. Фішка досвіту. Завжди роблять маленькі компоненти якщо є шанс того що цей компонент буде рости з проектом.

```
<template lang="html">
<div class="controls">
<button class="top" :class="{ focus: keys.includes(38) }"></button>
<button class="left" :class="{ focus: keys.includes(37) }"></button>
<button class="bottom" :class="{ focus: keys.includes(40) }"></button>
<button class="right" :class="{ focus: keys.includes(39) }"></button>
</div>
</template>
```

Це був HTML код компонента, як ві бачіте код дуже нестандартний , це тому що це спеціалізований сиснтаксис для Vue.js. Далі йде логика компонента.

```
<script>
import controls from '../api/controls.js';

export default {
  name: 'controls',
```

```
data() {
  return {
    keys: [],
  };
},
methods: {

},
created() {
  window.onkeydown = e => {

    let code = e.which;
    if(!this.keys.includes(code)) {
      this.keys.push(code);
    }

    if(this.keys.length > 1) {
      if(this.keys.includes(38) && this.keys.includes(37)) {
        controls.post('up-left');
      }
      else if(this.keys.includes(38) && this.keys.includes(39)) {
        controls.post('up-right');
      }
      else if(this.keys.includes(40) && this.keys.includes(37)) {
        controls.post('down-left');
      }
      else if(this.keys.includes(40) && this.keys.includes(39)) {
        controls.post('down-right');
      }
    } else {
      if(code == 38) {
        controls.post('up');
      }
      else if(code == 40) {
        controls.post('down');
      }
      else if(code == 37) {
        controls.post('left');
      }
      else if(code == 39) {
        controls.post('right');
      }
    }
  };
  window.onkeyup = e => {
    let index = this.keys.indexOf(e.which);
    if (index > -1) {
      this.keys.splice(index, 1);
    }
  };
},
destroyed() {
```

```
    window.onkeydown = function () {};
    window.onkeyup = function () {};
  }
}
</script>
```

У логіці ми проверяємо на які кнопки було натиснуто, и на основі цього ми відсилаємо HTTP-запит на сервер, який буде рухати нашого робота. Далі це все треба покрасити за допомогою SASS.

```
<style lang="scss" scoped>
.controls {
  position: absolute;
  left: 50%;
  width: 200px;
  height: 120px;
  margin-left: -100px;
  bottom: 50px;
  button {
    border: 0;
    background: #f5f6fa;
    border-radius: 5px;
    padding: 0;
    width: 50px;
    height: 50px;
    position: absolute;
&.top {
      top: 0;
      right: 0;
      margin: 0 auto;
      left: 0;
    }
&.left {
      left: 0;
      bottom: 0;
    }
&.bottom {
      right: 0;
      margin: 0 auto;
      left: 0;
      bottom: 0;
    }
&.right {
      right: 0;
      bottom: 0;
    }
&.focus {
      background: #7f8fa6;
```

```
    }
  }
}
</style>
```

Sass (англ. Syntactically Awesome Stylesheets) — скриптова метамова, яка інтерпретується в каскадні таблиці стилів (CSS). Спроектована Гемптоном Кетліном та розроблена Наталі Вейзенбаум. Sass призначений для підвищення рівня абстракції коду та спрощення файлів CSS.Мова Sass має два синтаксиси:

-sass (оригінальний) – відрізняється відсутністю фігурних дужок, в ньому вкладені елементи реалізовані за допомогою відступів, а правила відокремлюються переведенням рядка;

-scss (новий) – використовує фігурні дужки (подібно до CSS). Файли sass-синтаксису мають розширення .sass, scss-синтаксису – .scss.

Sass розширює CSS, надаючи кілька механізмів, доступних в більш традиційних мовах програмування, зокрема об'єктно-орієнтованих мовах, але недоступних для CSS. Інтерпретатор Sass транслює SassScript у блоки правил CSS. По суті, Sass — це синтаксичний цукордля CSS.Специфікації CSS були створені та розвиваються Консорціумом Всесвітньої мережі.CSS має різні рівні та профілі. Наступний рівень CSS створюється на основі попередніх, додаючи нову функціональність або розширюючи вже наявні функції. Рівні позначаються як CSS1, CSS2 та CSS3. Профілі — сукупність правил CSS одного або більше рівнів, створені для окремих типів пристроїв або інтерфейсів. Наприклад, існують профілі CSS для принтерів, мобільних пристроїв тощо.CSS (каскадна або блочна верстка) прийшла на заміну табличній верстці веб-сторінок. Головна перевага блочної верстки — розділення змісту сторінки (даних) та їхньої візуальної презентації.CSS використовується авторами та відвідувачами веб-сторінок, щоб визначити кольори, шрифти, верстку та інші аспекти вигляду сторінки. Одна з головних переваг — можливість розділити зміст сторінки (або

контент, наповнення, зазвичай HTML, XML або подібна мова розмітки) від вигляду документу (що описується в CSS).Таке розділення може покращити сприйняття та доступність контенту, забезпечити більшу гнучкість та контроль за відображенням контенту в різних умовах, зробити контент більш структурованим та простим, прибрати повтори тощо. CSS також дозволяє адаптувати контент до різних умов відображення (на екрані монітора, мобільного пристрою (КПК), у роздрукованому вигляді, на екрані телевізора, пристроях з підтримкою шрифту Брайля або голосових браузерах та ін.).

Тепер підключаємо API для нашого застосування. Я розбиваю це на настільки малі файли через те що я навчився так робити через 4 роки досвіду веб розробки, це обов'язково робити в будь-якому проекті, від малого до великого, всюди. Це слід робити через майбутнього скалірованія. API визначає функціональність, яку надає програма (модуль, бібліотека), при цьому API дозволяє абстрагуватися від того, як саме ця функціональність реалізована. Якщо програму (модуль, бібліотеку) розглядати як чорний ящик, то API – це безліч «ручок», які доступні користувачеві даного ящика і які він може крутити і смикати. Програмні компоненти взаємодіють один з одним за допомогою API (Рис 2.12). При цьому зазвичай компоненти утворюють ієрархію – високорівневі компоненти використовують API низькорівневих компонентів, а ті, в свою чергу, використовують API ще більш низькорівневих компонентів. За таким принципом побудовані протоколи передачі даних через Інтернет. Стандартний стек протоколів (мережева модель OSI) містить 7 рівнів (від фізичного рівня передачі біт до рівня протоколів програм, подібних протоколів HTTP і IMAP). Кожен рівень користується функціональністю попереднього («нижчого») рівня передачі даних і, в свою чергу, надає потрібну функціональність наступного («вищерозміщений») рівню. Важливо зауважити, що поняття протоколу близьке за змістом до поняття API. І те, і інше є абстракцією функціональності, тільки в першому випадку мова йде про передачу даних, а в другому – про взаємодію додатків. API бібліотеки функцій і класів включає

в себе опис сигнатур і семантики функцій.

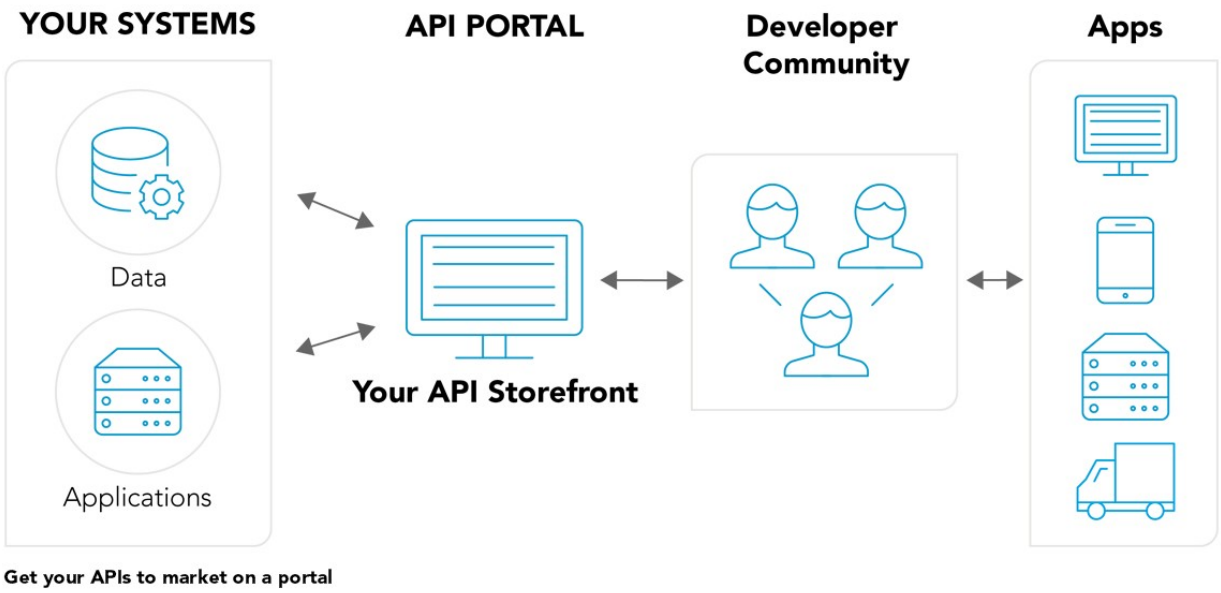


Рисунок 2.12– Подання схеми API в сучасних додатках

```
import axios from 'axios';
import config from '../config.js';

export default axios.create({
  baseURL: 'http://' + config.ip + '/controls',
  timeout: 100,
  headers: {
    'X-Custom-Header': 'foobar',
  }
});
```

ВИСНОВОК

Я получив досвід у роботі з мікрокоп’ютерами , UNIX-системами , та взагалі у розробци програмного продукту. Було вітрачено десяткі часів на розробку з моїм товаришем, та зараз я відчвую що час пройшов з великою пользою. Робота з Javascript не припиняє мене поражати , мени не набридає на ньому писати, я вважаю що це найкраща скриптова мова, яка підходить як для новачків так і для професіоналів.

My balls

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Мала гірнича енциклопедія / за ред. В. С. Білецького. — Д. : Східний видавничий дім, 2004—2013.

1. Синтез робототехнічних систем в машинобудуванні: підруч. для студентів вищ. техн. навч. закл., які навчаються за спец. 015 «Проф. освіта. Машинобудування»: присвяч. 100-річчю Вєтрова Ю. О., ректора Київ. інж.-буд. ін-ту, зав. каф. буд. машин / Л. Є. Пелевін, К. І. Почка, О. М. Гаркавенко та ін. ; М-во освіти і науки України, Київ. нац. ун-т буд-ва і архітектури. — Київ: ТОВ НВП «Інтерсервіс», 2016. — 258 с. : іл. — Бібліогр.: с. 257 (16 назв). — ISBN 978-617-696-447-6

2. http://www.raspberrypi.org/archives/2180

3. Raspberry Pi Foundation. Raspberry Pi Foundation. Архіворигіналу за 2013-07-16. Процитовано 2011-07-02.