**ТЕХНИЧЕСКИ УНИВЕРСИТЕТ- СОФИЯ, ФИЛИАЛ ПЛОВДИВ**

ФАКУЛТЕТ ПО ЕЛЕКТРОНИКА И АВТОМАТИКА

КАТЕДРА СИСТЕМИ ЗА УПРАВЛЕНИЕ

Дата на задаване: април 2019 г. Утвърждавам:

Дата на предаване: юни 2019 г. Декан: …………………………

(проф. д-р Г. Спасов)

**ЗАДАНИЕ**

**ЗА ДИПЛОМНА РАБОТА**

на студента Мустафа Хасанов Алиджиков**,** фак.№. 364906

Образователна степен: ОКС бакалавър, Специалност: АИУТ

Тема: Проектиране на надграждаща система за управление на мобилен робот “IRobot Create” на базата на микроконтролер NodeMCU.

1. Описание на конкретната инженерна задача и вида на крайните резултати, които трябва да се решат в оригиналната част от дипломната работа: Дипломантът следва да се запознае с възможностите на вградения контролер на робот “IRobot Create”, да проучи функционалните му и комуникационни възможности и да предложи решение за създаване на надграждаща система за управление на мобилния робот. Предложеното решение следва да е в посока на увеличаване на възможностите за изграждане на автономно поведение на робота и разширяване на потенциалните начини за комуникации с други устройства.

2. Изходни цифрови данни за изчислителната част. Функционални изисквания:

На дипломанта са предоставени мобилен робот “IRobot Create” с техническо описание.

3. Изходни литературни и други източници:

1) iRobot® Create Owner’s guide, <https://www.irobot.com/filelibrary/create/Create%20Manual_Final.pdf> , April 20, 2018

2) iRobot® Create OPEN INTERFACE, <https://www.irobot.com/filelibrary/pdfs/hrd/create/Create%20Open%20Interface_v2.pdf>

3) R. Siegwart I. Nourbakhsh, Introduction to Autonomous Mobile Robots, The MIT Press 2004

4. Съдържание на дипломната работа

4.1. Заглавна страница по образец

4.2. Оригинал на завереното дипломно задание

4.3. Съдържание

4.4. Увод

4.5. Глава 1 - Технически характеристики на мобилен робот “IRobot Create” – преглед;

4.6. Глава 2 - Проектиране на надграждаща система за управление;

4.7. Глава 3 - Разработка на програмно осигуряване;

4.8. Глава 4 - Получени резултати;

4.09. Глава 5 - Изводи;

4.10. Използвана литература

4.11. Приложения

Научен ръководител: ………………………..

( доц. д-р Н. Шакев )

Студент:………………………. Ръководител катедра: …………………….

( доц. д-р Ив. Костов )

# Съдържание

1. Увод………………………………………….………………………………………..…………………………………….4
2. Технически характеристики на мобилен робот “IRobot Create” – преглед ………....5
   1. Развитие на iRobot през годините…………………………………………………………………....5
   2. Роля на iRobot в производството на роботи………………….…………………………...…..6
   3. За iRobot Create………………………………………………………………………………………………...6
   4. Мобилна платформа…………………………………………….………………………………….…...…..7
      1. Държачи за колела……………………………………………………………………......8
      2. Добавяне на четвъртото колело…………………………………………….........8
      3. Захранване на iRobot Create………………………………………………….…......9
      4. Добавяне на тегло към капака………………………………….……………….....9
      5. Използване на пакета с алкални батерии……………………..……….......10
      6. Конектор в товарния сектор (Cargo Bay Connector)……………..........10
   5. Описание на функционалните възможности и потребителския интерфейс на IRobot Create……………………………………………………………………………………….……………11
      1. Потребителски интерфейс(ПИ)........................................................11
      2. Физически връзки………………..………………………….……………………………12
      3. Настройки на серийния порт………………………………………………………..14
      4. Режими на отворения интерфейс на iRobot Create……………………..15
      5. Описание на командите на интерфейса………………………………..….…16
      6. Команди за режим на работа……………………………..………………………..18
      7. Демонстрационни команди………………………………………………...………18
      8. Команди за задвижване. …………………………………………………...………..22
      9. Команди за вход................... ………………………………………………………..29
      10. Команди за скриптиране………………………………..…………………………....32
      11. Команди за изчакване ………………………......…………………………….......33
      12. Сензорни пакети на интерфейса на iRobot Create….........…………..35
3. Проектиране на надграждаща система за управление..……………………………………39

2.1 Как се програмира NodeMCU?..........................................................................40

2.2 Захранване на NodeMCU ……..……………………………………………………………………….42

2.3 Какво представлява ESP8266?...……………………………………………………………….……43

2.4 Хардуер на ESP8266…………………………………………………..…..……………………………...43

2.4.1 Цифрови входове и изходи……………………………………………………....….…43

2.4.2 Напрежение и ограничения на ток……………….………………………………...43

2.4.3 Използваеми пинове ……………………………………………………………………….44

2.4.4 ШИМ(широчинно импулсна модулация)..…….………………...…………….44

2.4.5 Аналогов вход…………………………..……………………………………...……………..44

2.4.6 Комуникация …..……………………………………………………………...……………...44

2.4.6.1 Серийно……………………………………………………………………………..44

2.4.6.2 I2C……………………………………………………………………………….……..45

2.4.6.3 SPI………………………………………………………………………………………45

2.5 Софтуер на ESP8266………………………………….....……………………………………………...46

2.5.1 Цифрови входове и изходи………………….………………..……………………..46

2.5.2 Аналогов вход…………………………………………………………………………………46

2.5.3 Комуникация…………………………………………………………………………………47

2.5.3 Споделяне на времето на CPU с RF частта………………………………………47

2.6 Wi-Fi………………………..………………………………………..…………………………………………...48

2.7 ESP8266 Уеб сървър……………………………………..……………………………………………...52

1. Разработка на програмно осигуряване…………………………………………………...............57

3.1 Използване на уеб сървъра за превключване на LED диод………….........…...…59

3.1.1 СПИ(серийен периферен интерфейс) Флаш файлова система.........62

3.1.2 СПИ(серийен периферен интерфейс) Флаш файлова система.........62

3.2 Реализация на управление на IRobot Create чрез платформата NodeMcu.....64

3.2.1 Начин на работа.....................................................................................65

3.3 Roomba в съответствие с отворения интерфейс на Create..............................66

1. Анализ на получените експериментални резултати….....…………………………………..69
2. Заключение………………………………………………………………………………………………………….70
3. Използвана литература…………………………………………………………………………………..….71
4. Приложения...............................................................................................................72

***Увод***

В дипломната работа е разгледано управлението на мобилен робот IRobot Create чрез интернет връзка. За целта се налага разширяване функционалността на робота.Това се постига като се добави модул за връзка с интернет и микроконтролер който да изпраща команди към робота. Има доста начини за постигане на тази цел, но един доста евтин и бърз начин е с помощта на платформата NodeMCU. При наличие на безжична интернет връзка платката създава уеб сървър към който могат да се изпращат команди които впоследтсвие се изпращат към IRobot Create. Връзката между платката и робота е осъществена чрез серийна комуникация.

В глава 1 е разгледана мобилната платформа на iRobot, разработката й през годините и ролята на фирмата в производството на роботи. iRobot Create е разработка от вече създаденият робот - прахосмукачка iRobot Roomba, но в Create са премахнати частите на прахосмукачката и той е повече за научни цели. В тази глава разглеждаме платформата на робота – шасито, колелата, захранването на робота, пакета с батерии и добавяне на товар върху шасито.

В глава 2 са разгледани функционалните възможности и потребителския интерфейс на Create. В тази глава се разглеждат подробно функционалните възможности, настройката и свързването на серийния порт, режимите на работа и командите за управление на робота. Всяка една от командите е номерирана и описана и с тях може да се напише код за управление на робота.

Глава 3 е за платформата NodeMCU и микрочипа който стои в основата и ESP8266. Разглеждат се подробно характеристиките и възможностите на платките. Разгледан е и OSI модела в сбит вариант и уеб технологиите защото са необходими за работата с интернет модула. Дадени са примери за създаване на уеб сървър към който могат да се изпращат заявки.

Глава 4 съдържа анализ на получените експериментални резултати и оценка надеждността на създадените приложения.

Глава 5 обобщаваме изпълнената задача и наученото от процеса

Глава 6 е списък с използваната помощна литература

Глава 7 съдържа двете приложения създадени за изпълнение на зададената тема

***Глава 1 Технически характеристики на мобилен робот “IRobot Create” – преглед***

* 1. **Развитие на IRobot през годините**

През 1990г. е основано дружеството IRobot от 3-ма вишисти на технологичен институт MIT за създаване на роботи за космически проучвания и за военните. Първият конструиран робот през 1991г. е робота IRobot Genghise който е предназначен за космически проучвания. Със седалище в Делауеър компанията проектира и разработва вътрешни и външни автономни роботи за къщата като прахосмухачката Roomba и Braava за почистване на подове.

През 1998 година фирмата получава изследователски договор с DARPA което води до разработката на серията PackBot. Това са военни роботи като повече от 2000 са били използвани в Ирак и Афганистан за обезвреждане на бомби. Използвани са за преттърсване на развалините след атентатите на 11 септември. През 2011 те са първите в ядрената централа във Фукушима след земетресението и тсунамито в Тохоку. Намират и приложение в разработките на НАСА.

Прес Септември 2002 е пуснат на пазара IRobot Roomba който до 2004г. достига до 1 милион продажби. През същата година е пуснат и робот за изследване на шахтите в египетските пирамиди.

Някои от продуктите които не са пуснати за продажба са IRobot Verro робот за почистване на басейни и IRobot Looj за почистване на канавки.

Уникална разработка на IRobot e Seaglider автономен подводен робот който с едно зареждане на батерията може да работи в продължение на месеци и да изминава хиляди километри. Той предоставя океанографски данни в реално време. Робота ChemBot който може да променя формата и големината си.

В процес на разработка е и робот за косене на трева който използва подобна на Roomba безжична технология. Също и медицински робот-асистент RP-VITA който ще има връзка към база данни с информация за пациентита и ще бъде оборудван с редица устройства като например стетоскопи, отоскопи и др.

* 1. **Роля на IRobot в производстото на роботи**

Като пионер в производството на роботи, целта на iRobot е да управлява новостите, да служи като катализатор за индустрията и да промени света, чрез започване на ерата на роботите. За да подкрепя и насърчава за развитието на роботизираните технологии, iRobot предлага ресурси за други разработчици, предоставя информация и продукти, които улесняват създаването и лесната интеграция на допълнителни товари, поведения и способности на iRobot платформите. iRobot си сътрудничи с външни разработчици от държавни агенции, академични институции и малки и големи фирми за създаване и нововъведения на пазара, които помагат на войници и професионалисти за обществената безопасност, да се справят с опасни мисии с по-малък риск.

Изследователската група на iRobot изпълнява авангардни изследвания, които да отговарят на нуждите на спонсори със сложни роботизирани решения. Изследователската група изследва възможности с водещи академични изследователски функции, фирми и други иновативни технологии, деблокирането на опит като системен интегратор и сформирането на най-добър екип от партньори от широк обхват от технологични области.

iRobot STEM е уникална и многопластова образователна инициатива. Проектирана да вдъхнови учениците да изберат кариера в рамките на STEM, включително и роботиката, програмата помага за събуждане на интерес към науката, технологията, инженеринга и математиката. В допълнение, програмируемия робот iRobot осигурява на преподавателите, студентите и разработчиците достъпна платформа, предварително монтирана за програмиране и развитие.

С повече от две десетилетия на водеща фирма в производството на роботи, iRobot се ангажира да предоставя платформи за изобретения и открития, развитие на ключови партньорства, за да насърчават технологичните проучвания и проектиране на роботи, които подобряват качеството на живот и стандартите за безопасност в света.

* 1. **За IRobot Create**

iRobot предприема важна стъпка за насърчаване на развитието на автономни битови роботи със създаването на Create. Create включва шасито и интелекта на iRobot Roomba, както и интерфейс устройството, което позволява комуникирането с външния свят. Частите на прахосмукачката са премахнати, за да се направи място за други устройства. Робота, който е показан на международното събитие, вече е достъпен за разработчиците, които могат да свалят малки скриптове с помощта на терминална програма; да пишат програми за автономно управление с използване на С или С++; или да използват други методи.

iRobot Create включва отворен интерфейс [1], който осигурява достъп до роботизирани задвижвания и сензори, както и почти всичко останало, което може да бъде монтирано върху шасито на робота и управлявано от интерфейсното устройство. Стандартният интерфейс е наличен като опция.

Продуктът също включва редица монтирани точки за подсигуряване на устройствата на робота, както и стандартни за индустрията електрически връзки за управление и добавяне на сензори.

iRobot Create е готов за работа с изваждането си от кутията. Начинаещите могат да наблюдават поведението на робота в някой от десетте демонстрационни режими или програмиране на робота, чрез изтегляне на скриптове с основни програми.

Напредналите потребители могат да пишат потребителски софтуер, чрез използване на различни методи, като се възползват от „потока на данни от сензорите” за управление на робота. А доста напредналите потребители могат да пишат програми за напълно автономно поведение.

Create е само една част от задълженията на iRobot в подкрепа на научното, технологичното, инженеринговото и математическото образование. iRobot е създал портал Начална програма за развитието на знанията по роботика, за да засили интереса на деца, учители и родители към роботиката.

* 1. **Мобилна платформа**

Роботът IRobot Create е мобилна платформа, създадена по подобни критерии като вакуумната прахусмухачка Roomba (но без вакуум). Тя е разработена за изследователски и учебни цели и е лесно програмируема за тази цел. СЪдържа няколко вградени сензори, както и два входно/изходни серийни портове, които позволяват да бъдат добавени повече.

IRobot Create е напълно роботизиран развит комплект, който позволява програмирането на нов робот, без да се тревожите за механичния монтаж и програмирането на ниско ниво. Отворения интерфейс на робота предоставя набор от команди, като управляващи команди, демонстрационни команди, команди за звук и команди за сензори, които насърчават находчивостта.

Когато започнете да работите с IRobot Create ще намерите десет вградени демонстрации, които са полезни за вдъхновение на вашите идеи. Когато се запознаете по-задълбочено с възможностите и интерфейса на роботам ще искате да напишете свои програми за управление на робота, с помощта на серийна връзка от всяко устройство което има възможност за такава комуникация.

Можете да прикачите и управлявате и други хардуерни и електронни устройства за iRobot Create; например, роботизирана ръка, дисплей или вариращ сензор. Може също да бъде и музикален инструмент! Чрез използване на командата за песен можете да запишете и запазите до 16 песни в робота.

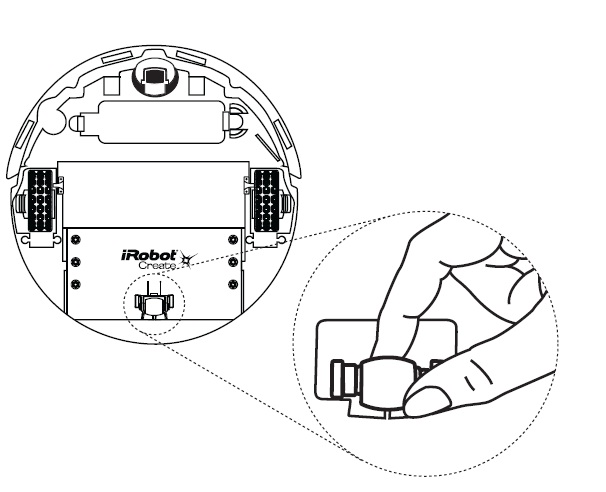
* + 1. **Държачи за колела**

Робота идва оборудван с държачи за колела, които държат основните колела в прибрана позиция. Можете да премахнете държачките, което автоматично поставя колелата в свободна позиция. Запазете ги в случай, че поискате по – късно да приберете колелата. Има предимства и в двете позиции на колелата:

* Позицията на прибраните колела (държачите са поставени) осигурява по-голяма стабилност на робота. Това е полезно ако има товар върху робота.
* Позицията на свободните колела (държачите са свалени) осигурява голяма мобилност на робота. Това е полезно ако искате робота да се движи по неравни повърхности.
  + 1. **Добавяне на четвъртото колело**

iRobot Create идва с допълнително непоставено четвърто колело, което позволява по – голяма стабилност и предпазва от хлъзгане, когато сте добавили товар. Четвъртото колело не е необходимо, освен ако има добавен товар.

За да прикачите четвъртото колело, премахнете всички тежести от робота и го обърнете обратно, така че да можете да виждате долната страна. Поставете четвъртото колело във вдлъбнатините в задната част на робота, с колелото под ъгъл към задната част (фиг. 1).



Фиг. 1 Четвъртото колело на iRobot Create

* + 1. **Захранване на IRobot Create**

IRobot Create може да се захранва с пакет вградени алкални батерии, като приема „АА“ алкални батерии или с презареждащи се акумулаторни батерии. Индикаторът за захранване на робота показва състоянието на батериите. Свети зелено, когато батериите са напълно заредени или нови и постепенно започва да свети червено когато заряда в батериите намалява.

|  |  |
| --- | --- |
| **Цвят на индикатора** | **Състояние на батерията** |
| Бавно пулсиращо оранжево | Зареждане (само презареждащи се батерии) |
| Бързо пулсиращо оранжево | Възстановяване на зареждането |
| Зелено | Напълно заредени батерии |
| Кехлибар | Частично заредени |
| Червено | Почти без заряд |
| Мигащо червено | Не заредени батерии |

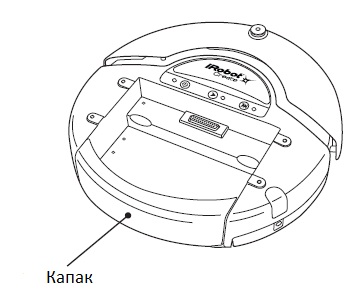
Таблица 1. Състояние на батерията на iRobot Create

* + 1. **Добавяне на тегло към капака**

Робота идва с капак, който позволява да променяте центъра на тежестта на робота чрез добавяне на тегло към задната част на робота. Това е полезно когато върху робота има товар.

**За добавяне на тегло:**

* Премахване на 2 винта, които ръжат капака за робота
* Премахване на 5 винта, които държат покритието на задния капак
* Добавяне на тежести в слотовете на задния капак.
* Завиване на покритието обратно за капака и капака отново за робота.

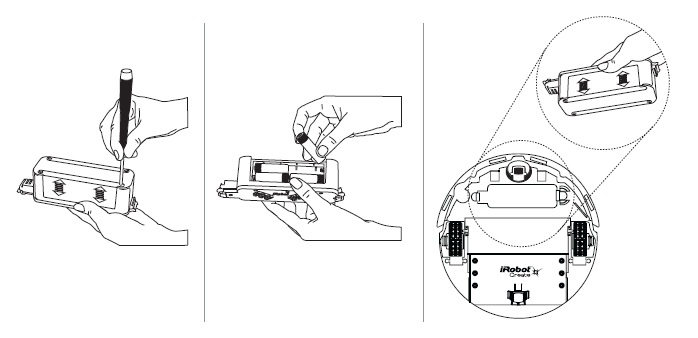


Фиг. 2 Капак на робота

* + 1. **Използване на пакета с алкални батерии**

Използвайте кръстосаната отвертка за да отслабите четирите винта от долната страна на зеленият пакет с алкални батерии. Махнете капачетата и поставете 6 непрезареждаши се алкални батерии „АА“ във всяка страна на отделението за батерии и затворете капачетата отново. Завъртете винтовете плътно и го поставете в робота на мястото си.

Забележка : Алкалните батерии не се препоръчват за използване в iRobot Roomba.



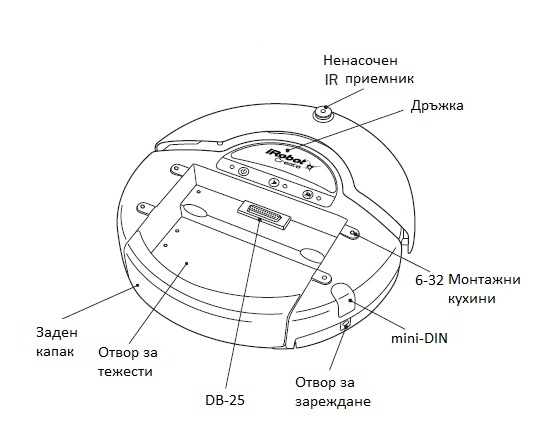
Фиг. 3 Поставяне на батериите на робота

* + 1. **Конектор в товарния сектор (Cargo Bay Connector)**

Отвора за добавяне на товар е разположен върху робота и в средата и съдържа 25 означени извода които могат да се използват за прикачване на електроника и други периферни устройства като допълнителни сензори, лампи или двигатели към робота. Отвора предлага четири цифрови входа, аналогов вход, три цифрови изхода, три изхода на ниско нив, индикатор за зареждане, мощност на превклчючване, сериен Tx и Rx, заземяване на батерията и напрежение на батерията.

* 1. **Описание на функционалните възможности и потребителския интерфейс на IRobot Create**
     1. **Потребителски интерфейс(ПИ)**

Интерфейса на робота се състои от електронен интерфейс и софтуерен интерфейс за управление на робота и прочитане на сензорите му. Електронния интерфейс съдържа 7-пинова min-DIN връзка и DB-25 връзка в отвора за товар за свързване на хардуера и електрониката за сензорите и допълнителния като ръка на робота или светлинен сензор. Софтуерния интерфейс ви позволява да управлявате поведението на робота и да четете сензорите му чрез серия от команди, които пращате към серийния порт на робота чрез път от компютъра или микроконтролер който е свързан чрез mini-DIN връзка или връзка за допълнителен товар.



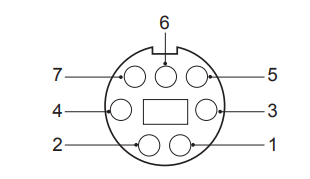
Фиг. 4. Скелет на робота

* + 1. **Физически връзки**

За да използвате потребителския интерфейс(ПИ), ви трябва процесор, който може да генерира серийни команди като компютър или микроконтролер който трябва да бъде свързан към външния мини-DIN конектор или към конектора в товарния сектор. Тези конектори осигуряват двупосочна, серийна комуникация при нива TTL (0 - 5V). Конекторите също осигуряват нерегулирана директна връзка към батерията на iRobot Create, която можете да използвате за захранване устройствата включени към ПИ. Кабелният конектор "Cargo Bay" също осигурява регулирано 5V захранване и няколко входно-изходни пина. Mini-DIN конекторът се намира в задната дясна част на Create, под пластмасова защита, докато конекторът в товарния регион се намира отпред в средата на товарния регион.

**mini-DIN връзка**

Тази диаграма показва изходите на горния изглед на робота. Имайте предвид, че пиновете 5, 6 и 7 са към външния периметър на Create.



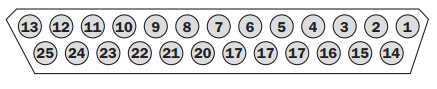
Фигура 5

|  |  |  |
| --- | --- | --- |
| Пин | Наименование | Описание |
| 1 | Vpwr | Батерията на Create + (нерегулиран) |
| 2 | Vpwr | Батерията на Create + (нерегулиран) |
| 3 | RXD | 0-5V Сериен вход към Create |
| 4 | TXD | 0-5V Сериен изход към Create |
| 5 | BRC | Зарежда бодовете |
| 6 | GND | Заземяване н батерията на Create |
| 7 | GND | Заземяване н батерията на Create |

Таблица 1. Описание на пиновете

Тъй като пиновете RXD и TXD използват 0 - 5V логическо нива и PC серийните портове използват различни напрежения (rs232 нива), това е необходимо да се промемят на нивата на напрежение. За да направите това, използвайте iRobot Create сериен кабел, а не обикновен сериен кабел, като серийният кабел iRobot Create съдържа всички необходими хардуерни свойства за промяна на нивата на напрежение, докато нормалния сериен кабел не.

**Конектор в товарната част на робота (КТЧ)**



фигура 6 – пинове на КТЧ

|  |  |  |
| --- | --- | --- |
| Пин | Име | Описание |
| 1 | RXD | 0 - 5V сериен вход за Create |
| 2 | TXD | 0 - 5V Сериен изход от Create |
| 3 | Превключвател за управление на захранването | Включва или изключва Create с преход от ниско към високо |
| 4 | Аналогов вход | 0 - 5V аналогов вход за Create |
| 5 | Цифров вход 1 | 0 - 5V цифров вход за Create |
| 6 | Цифров вход 3 | 0 - 5V цифров вход за Create |
| 7 | Цифров изход 1 | 0 - 5V, 20 mA цифров изход от Create |
| 8 | Превключване 5V | Осигурява регулирано 5V 100 mA захранване и аналогово напрежение когато робота е включен |
| 9 | Vpwr | Напрежение на заряда (нерегулиран), 0.5A |
| 10 | Превключен Vpwr | Осигурява захранване на батерията при 1,5 A когато Create е включен. |
| 11 | Превключен Vpwr | Осигурява захранване на батерията @ 1,5 А когато Create е включен. |
| 12 | Превключен Vpwr | Предоставя мощност на батерията @ 1,5 A когато Create е включен. |
| 13 | Зареждане на робот | Когато Create се зарежда, този пин е висок (5V) |
| 14 | GND | Заземяване на батерията |
| 15 | BRC | Използва за промяна на бодовете до 19200 |
| 16 | GND | Заземяване на батерията |
| 17 | Цифров вход 0 | 0 - 5V цифров вход на Create |
| 18 | Цифров вход 2 | 0 - 5V цифров вход на Create |
| 19 | Цифров изход 0 | 0 - 5V, 20 mA цифров изход от Create |
| 20 | Цифров изход 2 | 0 - 5V, 20 mA цифров изход от Create |
| 21 | GND | Заземяване на батерията |
| 22 | Драйвер на ниско ниво 0 | 0.5A драйвер на ниско ниво от Create |
| 23 | Драйвер на ниско ниво 1 | 0.5A драйвер на ниско ниво от Create |
| 24 | Драйвер на ниско ниво 2 | 1.5A драйвер на ниско ниво от Create |
| 25 | GND |  |

Таблица 2. Описание на пиновете на КТЧ

* + 1. **Настройки на серийния порт**

Битове за секунда (baud rate): 57600 или 19200

Данни: 8B

Съответствие: Няма

Стоп битове: 1

Контрол на потока: Няма

По подразбиране iRobot Create комуникира на 57600 baud. Ако използвате микроконтролер, който не поддържа 57600 baud, има два начина да принудим Create да превключи до 19200:

**Метод 1:**

Когато включите Create, задръжте бутона за възпроизвеждане. След около 4 секунди, Create възпроизвежда мелодия на низходящо ниво. Create ще комуникира при 19200 baud докато захранването се изключи, батерията е свален и повторно поставена, напрежението на батерията спадне под под минимума, изискван за операцията на процесора или скоростта на предаване на битове се изменя изрично чрез ПИ.

**Метод 2:**

Използвайте пина "Промяна на битовете" (пин 15 на КТЧ / пин 5 на mini-DIN конектора) за промяна номера на битовете. След като включите Create, изчакайте 2 секунди и след това пуснете на промяната на битовете. Всеки импулс трябва да трае между 50 и 500 милисекунди. Create ще комуникира на 19200 baud докато процесора загуби захранването на батерията или скоростта на предаване на данни се изменя изрично чрез ПИ.

* + 1. **Режими на отворения интерфейс на IRobot Create**

ПИ Create има четири режима на работа: Изключен, Пасивен, Безопасен и Пълен. След смяната на батерията или когато робота е доствен, той е в режим "изключен". Когато е изключен, ПИ слуша при стандартна скорост на предаване (57600 или 19200 ) за командата за стартиране на робота. След като получи командата "Старт", можете да влезете в някоя от четирите режими на работа чрез изпращане на команда за режима до ПИ. Можете също така да превключвате между режимите на работа по всяко време чрез изпращане на команда до ПИ за режима на работа който искате да използвате.

**Пасивен режим**

При изпращане на командата "Старт" или някоя от демонстративните команди(което също така стартира конкретната демонстрация, напр. откриване ,откриване и прибиране или други), интергейса преминава в пасивен режим. Когато е в пасивен режим, вие можете да изисквате и получавате данни от сензорите, но не можете да промените параметрите за задвижванията (мотори, високоговорители, светлини, ниски странични драйвери, цифрови изходи) към нещо друго. За да промените начина, по който един от задвижванията работи, трябва да превключете от пасивен режим в пълен или в безопасен режим. Докато сте в Пасивен режим, можете да прочетете сензорите на Create, да гледате робота да изпълнява всеки един от десетте си вградени демонстрации, и да заредите батерията.

**Безопасен режим**

Когато изпращате команда за безопасен режим към ПИ, Create преминава в Безопасен режим. Безопасният режим ви дава пълен контрол върху Create, с изключение на следните условия, свързани с безопасността:

• Откриване на наклон при движение напред (или преместване назад с малък радиус на завиване, по-малко от един робот радиус).

• Откриване на пропаднало колело (което и да е колело).

• Зарядно устройство е включено.

Ако възникне някое от горепосочените състояния, свързани с безопасността докато робота е в безопасен режим, Create спира всички мотори и се връща в пасивния режим.

Ако не се изпращат команди към ПИ, когато се намира в Безопасен режим, Create изчаква с изключени двигатели и светодиодит и не реагира на Play или Advance натискания на бутони или други. Имайте предвид, че зареждането се прекратява, когато влезете в безопасен режим.

**Пълен режим**

Когато изпратите командата пълен режим, Create влиза в пълен режим. Пълен режим ви дава пълен контрол над Create на всички задвижвания и всички свързани с безопасността условия, които са ограничени, когато робота е в безопасен режим, тъй като режимът "пълен" изключва инфрачервения сензор против падане, сензора за пропадане на колелото и вътрешното зарядно устройство. За да върнете ПИ обратно в безопасен режим, трябва да изпратите командата за безопасен режим. Ако не се изпращат команди към ПИ, когато е в пълен режим, Create чака с изключени двигатели и светодиоди не реагира на Play или Advance натискания на бутони или друг входен датчик. Имайте предвид, че зареждането се прекратява, когато влезете в пълен режим.

* + 1. **Описание на командите на интерфейса**

Тук е списъка на всички команди от интерфейса на робота използвани в дипломната работа. Всяка команда започва с 1-битов код. Някои от командите могат да бъдат проследени чрез данни от байта. Всички команди, включително и техните данни и байтове са описани, Размерността на допълнителните параметри е дадена в байтове.

**ЗАБЕЛЕЖКА**: Винаги изпращайте необходимия брой байтове данни за командата, в противен случай процесорът ще влезе и остане в "чакащо" състояние докато всички необходими байтове пристигнат.

**Команди за започване**

Следните команди стартират ПИ и получават и го правят готов за употреба. Start

|  |  |  |
| --- | --- | --- |
| **Старт** | **Код: 128** | **Доп. параметри: 0** |

Тази команда стартира интерфейса. Трябва винаги да изпращате старт, преди да изпратите други команди.

• Серийна последователност: [128].

• Предлага се в режими: пасивен, безопасен или пълен

• Променя режима на: Пасивен. Робота издава звук, за да покаче, че започва от режим "изключен".

|  |  |  |
| --- | --- | --- |
| **Бауд** | **Код: 129** | **Доп. параметри: 1** |

Тази команда определя скоростта на предаване в битове за секунда (bps) на командата и данните от ПИ се изпращат съгласно кода на бита, изпратен данните на бита. Стандартната скорост на предаване 57600 бауда, но началната скорост на предаване може да се промени на 19200, като задържите бутона докато включвате Create докато не чуете последователност от низходящи тонове. След като се промени скоростта на предаване, тя не се променя докато робота е захранен чрез натискане на бутона за мощност или премахване на батерията или когато напрежението на батерията пада под минимума, необходим за работата на процесора. Трябва да изчакате 100ms след изпращане на тази команда преди изпращане на допълнителни команди при новата скорост на предаване.

Забележка: при скорост на предаване 115200, трябва да има поне 200 μs между началото на всеки знак или някои от знаците може да не се приемат.

• Серийна последователност: [129] [Baud Code]

• Предлага се в режими: Пасивен, Безопасен или Пълен

• Променя режима на: Без промяна

• Байт данни за предаване 1: Код на предаване (0-11)

Baud Code Baud Rate in BPS

|  |  |
| --- | --- |
| Бауд код | Бауд код в bps |
| 0 | 300 |
| 1 | 600 |
| 2 | 1200 |
| 3 | 2400 |
| 4 | 4800 |
| 5 | 9600 |
| 6 | 14400 |
| 7 | 19200 |
| 8 | 28800 |
| 9 | 38400 |
| 10 | 57600 |
| 11 | 115200 |

* + 1. **Команди за режим на работа**

Create има четири режима на работа: Изключен, Пасивен, Безопасен и Пълен. Робота се включва в пасивен режим. Следните команди променят интерфейса на робота.

|  |  |  |
| --- | --- | --- |
| **Безопасен** | **Код: 131** | **Доп. параметри: 0** |

Тази команда поставя интерфейса в Безопасен режим, позволявайки на потребителя да контрол на Create. Изключва всички светодиоди. Интерфейса може да бъде в Пасивен, Безопасен или Пълен режим, за да приемете тази команда.

• Серийна последователност: [131]

• Предлага се в режими: Пасивен, Безопасен или Пълен

• Променя режима на: Безопасен

|  |  |  |
| --- | --- | --- |
| **Пълен** | **Код: 132** | **Доп. параметри: 0** |

Тази команда ви дава пълен контрол върху Create чрез поставяне на ПИ в Пълен режим и изключвайки сензорите против падане, сензорите за пропадане на колелата и вътрешното зарядно за безопасност. Тоест, в Пълен режим, Create изпълнява всяка команда, която изпращате дори ако вътрешното зарядно устройство е включено или роботът има активиран сензор против падане или сензор за падане на колело.

• Серийна последователност: [132]

• Предлага се в режими: Пасивен, Безопасен или Пълен

• Променя режима на: Full

Забележка: Използвайте командата "Старт" (128), за да промените режима

към Пасивен.

* + 1. **Демонстрационни команди**

Следващите команди стартират вградените демонстрации на iRobot Create.

|  |  |  |
| --- | --- | --- |
| **Демонстрация** | **Код: 136** | **Доп. параметри: 1** |

Тази команда стартира заявената вградена демонстрация.

• Серийна последователност: [136] [Кое-демо]

• Предлага се в режими: Пасивен, Безопасен или Пълен

• Променя режима на: Пасивен

• Байт за демонстрационни данни 1: Демонстрационен номер (-1 - 9)

Demo имена, описания и номера

Описание на демонстрационния номер

|  |  |  |
| --- | --- | --- |
| Номер | Демонстрация | Описание |
| -1 (255) | Прекъсване | Прекъсва демонстра-цията, която се изпълня-ва в момента. |
| 0 | Покритие | Create прави опити за покриване на цяла стая, използва комбинация от поведения, като случайно скачане, следене на стената, и спирали. |
| 1 | Покритие и прибиране | Иидентично с демонстрацията за покритие, с едно изключение. Ако Create вижда инфрачервен сигнал от iRobot Home Base, той използва този сигнал, за да се докосне до нея и да започне да се презарежда. |
| 2 | Откриване на област | Create покрива област около своята изходна позиция чрез спирала навън, после навътре. |
| 3 | Следене | Create се движи в търсене на стена. Веднъж намерена, Create се движи по дължината на стената и около обиколката на стаята. |
| 4 | Движение във фигура на осмица | Create непрекъснато се движи във фигура на осмица |
| 5 | Wimp | Create се движи напред и ако се удари в препятствие се отдалечава от него |
| 6 | База презареждане | Create се движи към iRobot Virtual Стена докато гърба и страните на приемника на виртуалната стена са прикрити от черна електрическа лента. Виртуалната стена излъчва инфрачервени сигнали, която Create вижда с неговия универсален инфрачервен приемник, разположен в горната част на броня. Ако искате да робота да се прибере във виртуалната стена, покрийте всичко, но оставете малко пространство отпред на инфрачервения приемник с черна електрическа лента.  Create се завърта, за да намери виртуалната стена и тръгва срещу нея.  Ако удари стена или друго препятствие спира. |
| 7 | Следене | Подобно за демонстрацията за база за презареждане, Create се движи към няколко виртуални стени като се удари в една се завърта, продължава към следващата удря се и в нея завърта се и продължава към следващата. |
| 8 | Pachelbel | Create играе нотите на Pachelbel's Canon в съответствие, когато е включен сензора против падане. |
| 9 | Banjo | Create изпълнява ноти на акорд за всеки от четирите си сензора против падане. Избира акорд, използвайки сензора за удар, както следва:  • Няма удар :G major.  • Десен / ляв удар :D major 7  • Двата сензора (в средата) :C major |

Можете също да извикате демонстрациите, за Покритие, Покритие и прибиране, откриване на област чрез използване на кодовете описани долу.Това е предпоставка за осигуряване на обратна съвместимост с интерфейса на Roomba.

|  |  |  |
| --- | --- | --- |
| **Покритие** | **Код: 135** | **Доп. параметри: 0** |

Тази команда стартира демонстрацията за покритие.

• Серийна последователност: [135]

• Предлага се в режими: Пасивен, Безопасен или Пълен

• Променя режима на: Пасивен

|  |  |  |
| --- | --- | --- |
| **Покритие и прибиране** | **Код: 143** | **Доп. параметри: 0** |

Тази команда стартира демонстрацията за покритие и прибиране.

• Серийна последователност: [143]

• Предлага се в режими: пасивен, безопасен или пълен

• Променя режима на: Пасивен

|  |  |  |
| --- | --- | --- |
| **Откриване на област** | **Код: 134** | **Доп. параметри: 0** |

Тази команда стартира демонстрацията за откриване на област.

• Серийна последователност: [134]

• Предлага се в режими: Пасивен, Безопасен или Пълен

• Променя режима на: Пасивен

* + 1. **Команди за задвижващия**

Следните команди управляват задвижванията на iRobot Create:

колела, високоговорители, светодиоди, цифрови изходи и драйвери от ниската страна.

|  |  |  |
| --- | --- | --- |
| **Управление** | **Код: 137** | **Доп. параметри: 4** |

Тази команда управлява колелата на Create. Използва четири байта, интерпретирани като две 16-битови стойности използвайки две допълнителни. Първите два байта определят средната стойност скоростта на задвижващите колела в милиметри в секунда (мм / сек), като първо се изпраща високия байт. Следващите две байта определят радиуса в милиметри, на който робота се завърта. По-дългият радиус прави Create да се движи по-изправен, докато по-късите радиуси карат Create да се завърта повече. Радиусът се измерва от центъра на ъгъла на завъртане до центъра на Create. Управляващата команда с положителна скорост и положителен радиус прави Create да се движи напред докато не завие наляво. Отрицателният радиус кара робота да се завърти надясно. Специфични случаи за радиуса правят Create на завива на място или да се движи направо, както е посочено по-долу. Негативно скоростта създава задвижване назад.

• Серийна последователност: [137] [Скорост на висок байт] [Скорост на нисък байт] [Радиус висок байт] [Радиус нисък байт]

• Предлага се в режими: Безопасен или Пълен

• Променя режима на: Без промяна

• Байтове за управление 1: Скорост (-500 - 500 mm / s)

• Байтове за управление 2: Радиус (-2000 - 2000 mm)

Специални случаи:

В права посока = 32768 или 32767 = hex 8000 или 7FFF

Завъртете по посока на часовниковата стрелка = hex FFFF

Завъртете обратно на часовниковата стрелка = hex 0001

Пример:

За движeние в обратна посока при скорост от -200 мм / сек при радиус на завъртане 500мм изпратете следната последователност от байтове:

[137] [255] [56] [1] [244]

Скорост = -200 = шестнадесетичен FF38 = [hex FF] [hex 38] = [255] [56]

Радиус = 500 = хекс 01F4 = [hex 01] [hex F4] = [1] [244]

|  |  |  |
| --- | --- | --- |
| **Директно управление** | **Код: 145** | **Доп. параметри: 4** |

Тази команда ви позволява да контролирате движението напред и назад на задвижващите колела на Create независимо. Използва четири байта данни, които се интерпретират като два 16-битови стойности, използва и две допълнения. Първите два байта определят скоростта на дясното колело в милиметри за секунда (мм / сек), с изпратения висок байт първо. Следващите два байта задават скоростта на лявото колело в същия формат. Положителната скорост кара това колело да се движи напред, а а отрицателната скорост го кара да се движи назад.

• Серийна последователност: [145] [Скорост на дясно на висок байт] [Скорост на дясно на нисък байт] [Скорост на ляво на висок байт] [Скорост на ляво на нисък байт]

• Предлага се в режими: Безопасен или Пълен

• Променя режима на: Без промяна

• Байт за директно задвижване 1: Скорост на дясно колело (-500 - 500 мм / сек)

• Байт за директно задвижване 2: Скорост на лявото колело (-500 - 500 мм / сек)

|  |  |  |
| --- | --- | --- |
| **Светодиоди** | **Код: 139** | **Доп. параметри: 3** |

Тази команда управлява светодиодите на Create. Състоянието на лампичките Play и Advance се определят от два бита в първия байт данни. Светодиода за зареждане е определена от два байта данни: един за цвят друг за интензивност.

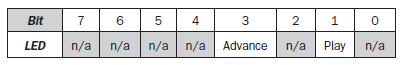
• Серийна последователност: [139] [LED бита] [Цвят на зареждане][Интензитет на зареждане]

• Предлага се в режими: Безопасен или Пълен

• Променя режима на: Без промяна

• Байт данни за светодиоди 1: LED бита (0 - 10)

Advance и Play използват зелени светодиоди. 0 = изключен, 1 = включен



Мощността използва двуцветен (червен / зелен) светодиод. Интензитетът и цветът на този светодиод може да се управлява с 8-битова разделителна способност.

• Данни за байт на светодиоди 2: Цвят на захранващия LED (0 - 255) 0 = зелена, 255 = червена. Средните стойности са междинни цветове (оранжево, жълто и др.).

• Данни за байт на светодиоди 3: Интензивност на захранването с LED (0 - 255) 0 = изключен, 255 = пълен интензитет. Средните стойности са междинни интензитети.

Пример:

За да включите светодиода Advance и да светне зеленият светодиод за захранване на половин интензитет, изпраща серийния байт последователност [139][8][0] [128].

|  |  |  |
| --- | --- | --- |
| **Цифрови изходи** | **Код: 147** | **Доп. параметри: 1** |

Тази команда контролира състоянието на 3 пина цифрови изходи от на 25 пиновия конектор в товарния сектор. Цифровите изходи могат да осигурят до 20 mA ток.

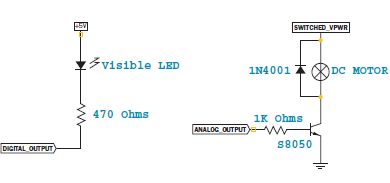
• Серийна последователност: [147] [Битове за изход]

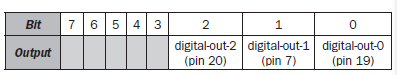
• Предлага се в режими: Безопасен или Пълен

• Променя режима на: Без промяна

• Цифрови изходи за байт данни 1: Изходни битове (0-7); 0 = ниска (0V); 1 = висока (5V).

Примерни схеми:





|  |  |  |
| --- | --- | --- |
| **ШИМ Управление на ниско ниво** | **Код: 144** | **Доп. параметри: 3** |

Тази команда ви позволява да контролирате трите драйвера на ниско ниво с променлива мощност. С всеки байт данни задавате работен цикъл на ШИМ за управлението на ниско ниво (макс. 128). Например, ако искате да управлявате драйвер с 25% от напрежението на батерията, изберете работен цикъл от 128 \* 25% = 32.

• Серийна последователност: [144] [Цикъл на управление на ниско ниво 2] [Цикъл на управление на ниско ниво 1] [Цикъл на управление на ниско ниво 0]

• Предлага се в режими: Безопасен или Пълен

• Променя режима на: Без промяна

• Данни за байт на драйвери на ниско ниво 1: Работен цикъл на управление на ниско ниво 2 (0 - 128)

• Данни за байт на драйвери на ниско ниво 2: Работен цикъл на управление на ниско ниво 1 (0 - 128)

• Данни за байт на драйвери на ниско ниво 3: Работен цикъл на управление на ниско ниво (0 - 128)

**Пример:**

За да включите управление на ниско ниво 2 на 25% и управление на ниско ниво 0 на 100%, изпратете последователност от серийни байтове [144] [32] [0] [128]

|  |  |  |
| --- | --- | --- |
| **Управление на ниско ниво** | **Код: 138** | **Доп. параметри: 1** |

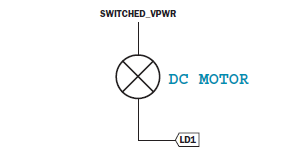
Тази команда ви позволява да контролирате трите драйвера на ниско ниво. Състоянието на всеки драйвер се определя от един бит в битовите данни. Управление на ниско ниво 0 и 1 могат да осигурят до 0.5А ток. Управление на ниско ниво 2 може да осигури до 1,5 А ток. Ако се изисква твърде много ток, токът е ограничен и е зададен флаг за свръхнапрежение (сензорен пакет 14).

• Серийна последователност: [138] [Шофьорски бийтове]

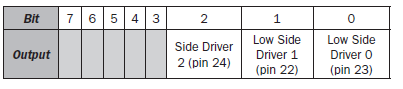
• Предлага се в режими: Безопасено или Пълен

• Променя режима на: Без промяна

• Байт данни за управление на ниско ниво 1: Битове за управление(0 - 7)



Фиг.7 0=изключен, 1=включено на 100% ШИМ-цикъл



Пример:

За да включите само управление на ниско ниво 1, изпратете серийната последователност [138] [2].

|  |  |  |
| --- | --- | --- |
| **Изпрати IR(Инфрачервен приемник)** | **Код: 151** | **Доп. параметри: 1** |

Тази команда изпраща искания байт от управление на ниско ниво 1 (pin 23 на конектора в товарния сектор), като използвате формат, очакван от IR приемника на iRobot Create. Трябва да използвате презареждащ се резистор (предложена стойност: 100 ома) паралелно с инфрачервения светодиод и неговия резистор, за да го включите.

• Серийна последователност: [151] [Байт стойност]

• Предлага се в режими: Безопасен или Пълен

• Променя режима на: Без промяна

• Данни за байт от IR 1: Байт стойност за изпращане (0 - 255)

|  |  |  |
| --- | --- | --- |
| **Песен** | **Код: 140** | **Доп. параметри: 2N+2** |

(където N e номера на нотите в песента)

Тази команда ви позволява да зададете до шестнадесет песни в ПИ и да можете да ги изпълните по-късно. Всяка песен е свързана с номер на песента. Командата "Play" използва номера на песента, за да определи избора си на песен. Всяка песен може съдържа до шестнадесет ноти. Всяка нота е свързана с номера на нотата, която използва определене на нотите MIDI и продължителност която се посочено във фракции от секундата. Броят на байтовете варира в зависимост от дължината на определената песен. Една нота за песен се определя от четири байта за данни. За всяка допълнителна нота без песен, се добавят два байта.

• Серийна последователност: [140] [Номер на песента] [Дължина на песента] [Нота номер 1] [Продължителност на нота 1] [Нота номер 2] [Продължителност на нота 2] и т.н.

• Предлага се в режими: Пасивен, Безопасен или Пълен

• Променя режима на: Без промяна

• Байт данни за песни 1: Номер на песен (0 - 15)

Номерът на песента, е свързан със специфична песен. Ако изпращате втора команда за песен, като използвате същия номер на песен, старата песен се презаписва.

• Байт данни за песни 2: Продължителност на песента (1 - 16)

Продължителност на песента, според броя на музикални ноти в рамките на песента.

• Байтове данни за песни 3, 5, 7 и др .: Номер на нота (31 - 127)

Степента на нотите в Create ще зависи от MIDI на номериране на нотите. Най-ниската нота която Create може да изпълни е #31. Create счита всички ноти от диапазона 31-127 като ноти за почивка и не издава никакъв звук по време на тези ноти.

• Данни за байт на песен 4, 6, 8 и др .: Продължителност на нота (0 - 255)

Продължителността на една нота, в стъпки от 1/64-та от секундата.

Пример: половин секунда дълга музикална нота е с продължителност 32

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Номер | Нота | Честота | Номер | Нота | Честота | Номер | Нота | Честота |

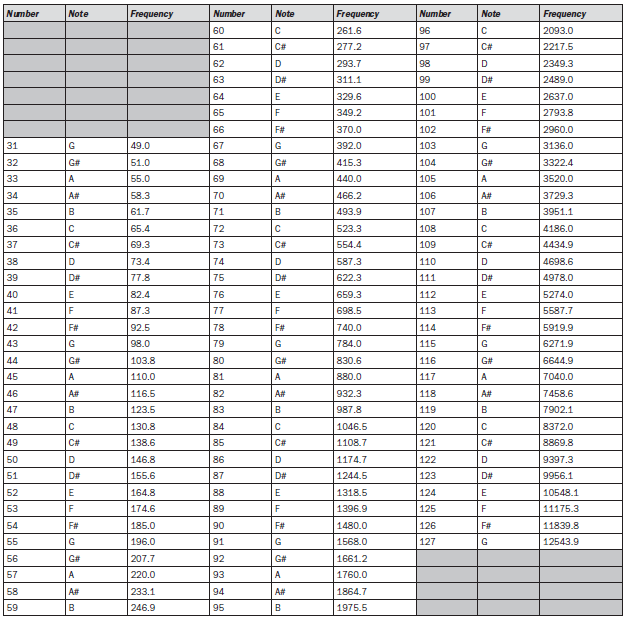


Таблица 6 продължителност на нотите

|  |  |  |
| --- | --- | --- |
| **Изпълняване на песен** | **Код: 141** | **Доп. параметри: 1** |

Тази команда ви позволява да изберете песен, която да се възпроизвежда от песните добавен към iRobot Create чрез командата "Песен". Трябва добавите една или повече песен, с помощта на командата "Песен" за да може командата "Play" да работи. Също така, тази команда

не работи, ако дадена песен вече се изпълнява. Изчакайте до изпълняваната в момента песен се завърши преди изпращането на тази команда. Имайте предвид, че може да използвате сензорния пакет за изпълнение на песен за да проверим дали робота е готов да приеме тази команта.

• Серийна последователност: [141] [Номер на песента]

• Предлага се в режими: Безопасено или Пълен

• Променя режима на: Без промяна

• Данни за байт за възпроизвеждане на песен 1: (0 - 15) Номерът на създадената песен която Create трябва да изпълни

* + 1. **Команди за вход**

Следните команди позволяват да прочетете състояието на вградените сензори на Create, цифрвои и аналогови входове и състоянието на някои вътрешни променливи. Робота актуализира вътрешно тези стойности на всеки 15ms. Не изпращайте тези команди по-рано от това.

|  |  |  |
| --- | --- | --- |
| **Сензори** | **Код: 142** | **Доп. параметри: 1** |

Тази команда изисква от интерфейса да изпрати данни за пакет от сензори. Има данни за 43 различни сензорни пакета. Всеки предоставя стойност от определен сензор или група сензори.

• Серийна последователност: [142] [Пакет ID]

• Предлага се в режими: Пасивен, Безопасен или Пълен

• Променя режима на: Без промяна

• Данни за байт на сензор 1: Идент. № на пакета (0 - 42)

Идентифицира кои от 43-те сензорни пакета данни трябва да бъдат изпратени обратно от интерфейса. Стойност 6 показва пакет с данни за всички сензори. Стойности от 0 до 5 показват определени сензорни данни.

|  |  |  |
| --- | --- | --- |
| **Списък заявка** | **Код: 149** | **Доп. параметри: N + 1** |

Тази команда позволява да поискате списък на сензорните пакети. Резултатът се връща веднъж, като в командата за сензори. Робота връща пакетите в последователността в която сте определили

• Серийна последователност: [149] [Брой пакети]

[Идент. Пакет 1] [Пакет ID 2] ... [Идент. Пакет N]

• Предлага се в режими: Пасивен, Безопасен или Пълен

• Променя режима на: Без промяна

• Данни за байт на заявения списък 1: Брой на поискани пакети (0-255)

• Данни за байт на заявения списък 2 - N: ИД на поисканите пакети (0 - 42)

Пример:

За да получите състоянието на левия сензор против падане (пакет 9) и индикатора за виртуална стена (пакет 13), изпратете последователността: [149] [2] [9] [13]

|  |  |  |
| --- | --- | --- |
| **Поток** | **Код: 148** | **Доп. параметри: N + 1** |

N е броя на поисканите пакети. Тази команда стартира непрекъснат поток от пакети данни. Списъка на поисканите пакети се изпраща на всеки 15 ms, което е стойността, която робота използва за да актуализира данните.

Това е най-добрият метод за поискване на сензорни данни ако управлявате робота чрез безжична мрежа със софтуер, работещ на настолен компютър.

• Серийна последователност: [148] [Брой пакети] [Пакет ID 1] [пакет ID 2] [пакет ID 3] и т.н.

• Предлага се в режими: Пасивен, Безопасен или Пълен

• Променя режима на: Без промяна

• Данни за байт на потока 1: Брой искани пакети (0 - 43)

• Данни за байт на потока 2 - N: ID на поисканите пакети (0 - 42)

Форматът който даанните връщат е:

[19] [N-байтове] [Пакет ID 1] [Данни за пакети 1 ...][Пакет ID 2] [Данни за пакети 2 ...] [Проверка]

N-байт е число между n-байт и контролната сума.

Контролната сума е със стойност 1-байт. Това е 8-битовото допълнение на всички байтове между header-а и контролната сума.Това означава, че ако добавите всички байтове след контролната сума, и контролната сума, ниският байт на резултата ще бъде 0.

Пример:

За да получите данни за сигнала от левия сензор против падане (пакет 29) и индикатора за виртуална стена (пакет 13), изпратете следната последователност от стрингове:

[148] [2] [29] [13]

Create започва потока данни който изглежда така:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 19 | 5 | 29 | 2 25 | 13 | 0 | 182 |
| Header | n-байт | Пакет ID 1 | Данни за пакет 1(2 байта) | Пакет ID 2 | Данни за пакет 2 (1 байт) | Контролна сума |

Таблица 7. Поток данни

**ЗАБЕЛЕЖКА**: Изчисление на контролната сума: (5 + 29 + 2 + 25 + 13 + 0+ 182) = 256 and (256 & 0xFF) = 0.

В горния сегментен поток стойността на сигнала от левия сензор против падане на Create е 549 (0x0225) и няма сигнал от виртуалната стена. От вас зависи да не поискате повече данни, отколкото могат да се изпратят в текущата скорост на предаване в интервал от време 15 ms. Например, на 57600 бода, може да се изпрати максимум 86 байта за 15 ms:

15 ms / 10 бита (8 данни + старт + стоп) \* 57600 = 86.4

Ако се изискват повече данни, в крайна сметка потокът от данни ще се развали. Това може да се потвърди чрез проверка на контролната сума. Горният байт и контролната сума могат да се използват за подравняване на вашите данни с програмата. Всички късове данни започват с 19 и завършва контролна сума с 1-байт.

|  |  |  |
| --- | --- | --- |
| **Пауза/възобнови потока** | **Код: 150** | **Доп. параметри: 1** |

Тази команда ви позволява да спрете и да рестартирате потока без изчистване на списъка с исканите пакети.

• Серийна последователност: [150] [Състояние на потока]

• Предлага се в режими: Пасивен, Безопасен или Пълен

• Променя режима на: Без промяна

• Данни за байт на потока за Пауза / Възобновяване 1: Състояние на поискания поток (0-1)

Аргумент от 0 спира потока, без да изчиства списъка от исканите пакети. Аргумент от 1 започва потока използвайки списъка с последно поискани пакети.

* + 1. **Команди за скриптиране**

Следващите команди позволяват да зададете скрипт за iRobot Create за изпълнение по-късно.

|  |  |  |
| --- | --- | --- |
| **Скрипт** | **Код: 152** | **Доп. параметри: N+1** |

N е номера на байта в скрипта. Тази команда определя скрипт, който да бъде възпроизведен по-късно. Скрипта се състои от интерфейс команди и може да бъде с дължина до 100 байта .Няма контрол на потока, но има команди за изчакване за да може да задържа текущото си състояние докато определеното събитието е открито.

• Серийна последователност: [152] [Дължина на скрипта] [Код 1][Код 2] [Код 3] и т.н.

• Предлага се в режими: Пасивен, Безопасен или Пълен

• Променя режима на: Без промяна

• Данни за байт на скрипта 1: Дължина на скрипта (0-100)

Указва дължината на скрипта по отношение на брой команди. Определя дължина 0, за да изчисти текущия скрипт.

• Данн за байт на скрипта 2 и нагоре: Команди на интерфейса и данни

Примерни скриптове:

Задвижване 40 см и спиране:

152 13 137 1 44 128 0 156 1 144 137 0 0 0 0

Превключване на светодиод за удар:

152 17 158 5 158 251 139 2 0 0 158 5 158 251 139 0 0 0 153

Движение в квадрат:

152 17 137 1 44 128 0 156 1 144 137 1 44 0 1 157 0 90 153

|  |  |  |
| --- | --- | --- |
| **Изпълнение на скрипт** | **Код: 153** | **Доп. параметри: 0** |

Тази команда зарежда предварително определени от интерфейса скриптове в последователността за възпроизвеждане на серийния вход.

• Серийна последователност: [153]

• Предлага се в режими: Пасивен, Безопасен или Пълен

• Променя режима на: Без промяна

|  |  |  |
| --- | --- | --- |
| **Показване на скрипт** | **Код: 154** | **Доп. параметри: 0** |

Тази команда връща стойностите на предварително съхранения скрипт, започвайки с броя на байтовете в скрипта последвани от командите на скрипта и байтовете за данни. На първо място спира сензора за потока , ако е започнал с команда Поток или Пауза / Възобновяване на потока. За да пуснете потока, изпратете Pause / Resume Stream (код 150).

• Серийна последователност: [154]

• Предлага се в режими: Пасивен, Безопасен или Пълен

• Променя режима на: Без промяна

* + 1. **Команди за изчакване**

Следните команди карат робота да изчака за определено време, разстояние, ъгъл на завъртане или каквото и събитие да се случи. Докато чака, състоянието на тобота не се променя, не реагира на никакви входови, серийни и каквито и да било въздействия. Тези команди са предназначени само за използване в скриптове.

|  |  |  |
| --- | --- | --- |
| **Време за изчакване** | **Код: 155** | **Доп. параметри: 1** |

Тази команда кара робота да изчака за определено време. През това време, състоянието на робота не се променя, не реагира на никакви входови серийни и каквито и да било въздействия.

• Серийна последователност: [155] [време]

• Предлага се в режими: Пасивен, Безопасен или Пълен

• Променя режима на: Без промяна

• Време на изчакване Час 1: Време (0 - 255)

Задава време за изчакване в десети от секундата с разделителна способност от 15мс.

|  |  |  |
| --- | --- | --- |
| **Разстояние на изчакване** | **Код: 156** | **Доп. параметри: 2** |

Тази команда кара iRobot Create да изчака, докато има изминало определеното разстояние в mm. Когато Create пътува напред, разстоянието нараства. Когато Create пътува назад, разстоянието намалява. Ако колелата се въртят в двете посоки, разстоянието нараства. Докато Create се движи на определено разстояние, неговото състояние не се променя, нито реагира на каквито и да е входове, серийни или по друг начин.

• Серийна последователност: [156] [Висок байт за разстояние][Нисък байт за разстояние]

• Предлага се в режими: Пасивен, Безопасен или Пълен

• Променя режима на: Без промяна

• Данни за байт за изчакване на разстояние 1-2: 16-битово определено разстояние в mm, висок байт първо (-32767 -32768)

|  |  |  |
| --- | --- | --- |
| **Ъгъл на изчакване** | **Код: 158** | **Доп. Параметри: 2** |

Тази команда кара Create да изчака, докато не се завърти на определен ъгъл в градуси. Когато Create се завърти обратно на часовниковата стрелка, ъгълът се увеличава. Когато Create се завърти по посока на часовниковата стрелка, ъгълът се намалява. Докато Create се завърти на определен ъгъл, неговото състояние не се променя, нито реагира на никакви входове - серийни или други.

• Серийна последователност: [157] [Висок байт за ъгъл] [Нисък байт за ъгъл]

• Предлага се в режими: Пасивен, Безопасен или Пълен

• Променя режима на: Без промяна

• Данни за байт за ъгъл на изчакване 1-2: 16-битов подписан ъгъл в градуси, първо висок байт (-32767 -32768)

|  |  |  |
| --- | --- | --- |
| **Резултат на изчакване** | **Код: 158** | **Доп. параметри: 2** |

Тази команда кара Create да изчака, докато достигне определен резултат. До откриването на зададения резултат, състоянието на Create не се променя, нито реагира на никакви входове - серийни или други.

• Серийна последователност: [158] [Номер на резултат]

• Предлага се в режими: Пасивен, Безопасен или Пълен

• Променя режима на: Без промяна

• Данни за байт за изчакване на резултат 1: Номер на зададено събитие (1 до 20 и -1 до -20)

За да изчакате обратния на даден резултат, изпратете отрицателният му номер му, използвайки две допълнения. Например, за да изчака за липса на сензори за удар, изпратете последователност от серийни байтове [158][-5], което е еквивалентно на [158] [251].

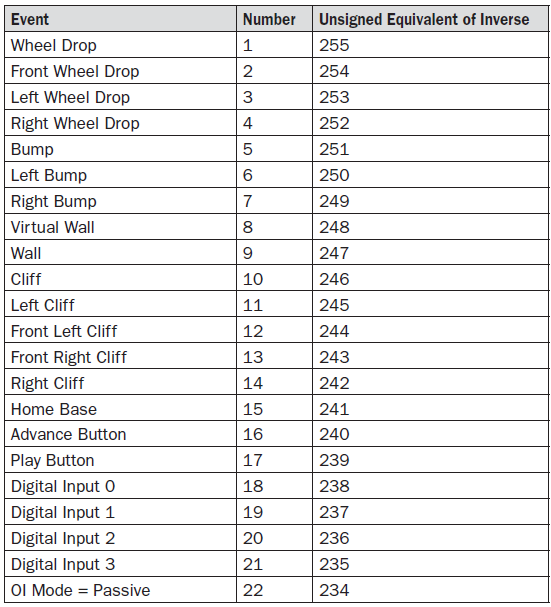


Таблица 8. Резултат на изчакване. Модулен еквивалент на обратното

* + 1. **Сензорни пакети на интерфейса на iRobot Create**

Create изпраща един от 43-те различни пакета данни на сензор, в зависимост от стойността на пакетните данни, когато отговаря на изискването за сензорните команди, списъка със заявени команди или командите за поток от пакета данни. Някои пакети съдържат групи от други пакети. Някои от стойностите на сензорните са 16-битови стойности.

Повечето от пакетите (номера 7 - 42) съдържат стойността на единичен сензор или променлива, която може да бъде 1 байт или 2 байта. Двубайтовите пакети съответстват на 16-битовите стойности, като изпращат високия байт първо. Някои от пакетите (0-6) съдържат групи от пакети с единичната стойност.

Събитие на изчакване: Модулен еквивалент на обратното

|  |  |  |
| --- | --- | --- |
| Групов пакет | Размер на пакета | Съдържание на пакета |
| 0 | 26 байта | 7-26 |
| 1 | 10 байта | 7-16 |
| 2 | 6 байта | 17-20 |
| 3 | 10 байта | 21-26 |
| 4 | 14 байта | 27-34 |
| 5 | 12 байта | 35-42 |
| 6 | 52 байта | 7-42 |

Таблица 9. Пакет данни

|  |  |  |
| --- | --- | --- |
| **Удар и пропадане на колелата** | **Пакет ID: 7** | **Доп. параметри: 1** |

Състоянието на сензора за удар (0 = няма удар, 1 = удар) и на сензорите за пропадане на колело (0 = колелата са вдигнати, 1 = колелата са пропаднали) са изпратени като отделни битове.

Обхват 0 – 31

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Бит | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Сензор | n/a | n/a | n/a | Пропадане на предното колело | Пропадане на лявото колело | Пропадане на дясното колело | Ляв сензор за удар | Десен сензор за удар |

Таблица 10. Номер на сензор

|  |  |  |
| --- | --- | --- |
| **Стена** | **Пакет ID: 8** | **Доп. параметри: 1** |

Състоянието на сензора за стена е изпратен като стойност 1 бит(0 = няма стена, 1 = видяна е стена). Обхват: 0-1

|  |  |  |
| --- | --- | --- |
| **Ляв сензор против падане** | **Пакет ID: 9** | **Доп. параметри: 1** |

Състоянието на левия сензор против падане от лявата страна на робота е изпратено като стойност 1 бит( 0 = сензора не е включен, 1 = сензора е включен).Обхват: 0-1

|  |  |  |
| --- | --- | --- |
| **Преден ляв сензор против падане** | **Пакет ID: 10** | **Доп. параметри: 1** |

Състоянието на сензор против падане отляво отпред на робота е изпратено като стойност 1 бит( 0 = сензора не е включен, 1 = сензора е включен).Обхват: 0-1

|  |  |  |
| --- | --- | --- |
| **Преден десен сензор против падане** | **Пакет ID: 11** | **Доп. параметри: 1** |

Състоянието на левия сензор против падане отдясно отпред на робота е изпратено като стойност 1 бит( 0 = сензора не е включен, 1 = сензора е включен).Обхват: 0-1

|  |  |  |
| --- | --- | --- |
| **Десен сензор против падане** | **Пакет ID: 12** | **Доп. параметри: 1** |

Състоянието на сензора против падане от дясната страна на робота е изпратено като стойност 1 бит. Обхват 0-1

|  |  |  |
| --- | --- | --- |
| **Виртуална стена** | **Пакет ID: 13** | **Доп. параметри: 1** |

Състоянието на индикатора за виртуална стена е изпратен като стойност 1 бит (0 = не е засечена виртуална стена, 1 = засечена е виртуална стена. Обхват: 0-1

|  |  |  |
| --- | --- | --- |
| **Разстояние** | **Пакет ID: 19** | **Доп. параметри: 2** |

Разстоянието, което Create e изминал в милиметри след разстоянието, от последното поискано разстояние, изпратено като 16-битова стойност, висок байт първо. Това е същото като сумата на разстоянието изминато от двете колела разделени на две. Положителните стойности показват движение в посока напред; отрицателните стойности показват движение в обратната посока.

Обхват: -32768 - 32767

|  |  |  |
| --- | --- | --- |
| **Ъгъл** | **Пакет ID: 20** | **Доп. параметри: 2** |

Ъгълът в градуси, на който робота се е завъртял ит оискедното поискване, е изпратен като 16- битова стойност. Ъглите обратно на часовниковата стрелка са положителни, а по часовниковата стрелка са отрицателни.

Обхват: -32768 - 32767

|  |  |  |
| --- | --- | --- |
| **Сигнал от левия сензор против падане** | **Пакет ID: 28** | **Доп. параметри: 2** |

Силата на сигнала от левия сензор пртив падане е 16-битова стойност.

Обхват: 0-4095

|  |  |  |
| --- | --- | --- |
| **Сигнал от предния ляв сензор против падане** | **Пакет ID: 29** | **Доп. параметри: 2** |

Силата на сигнала от предния ляв сензор против падане е 16-битова стойност.

Обхват: 0-4095

|  |  |  |
| --- | --- | --- |
| **Сигнал от предния десен сензор против падане** | **Пакет ID: 30** | **Доп. параметри: 2** |

Силата на сигнала от предния десен сензор против падане е 16-битова стойност.

Обхват: 0-4095

|  |  |  |
| --- | --- | --- |
| **Сигнал от десния сензор против падане падане** | **Пакет ID: 31** | **Доп. параметри: 2** |

Силата на сигнала от десния сензор против падане е 16-битова стойност.

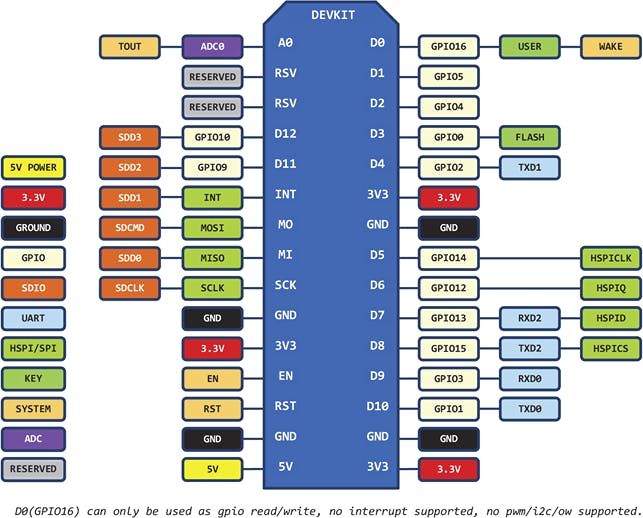
Обхват: 0-4095

***Глава 2 – Проектиране на надграждаща система за управление;***

1. **NodeMCU**

NodeMCU е платформа с отворен код за IoT(Интернет от нещата). Той включва фърмуер, който работи с ESP8266 Wi-Fi SoC(виж по долу) от Espressif Systems и хардуер, който се основава на модула ESP-12. Терминът "NodeMCU" по подразбиране се отнася за фърмуера, а не за комплектите за разработка. Фърмуерът използва програмния език Lua, но може да се ползват и други. В тази дипломна работа използваме Arduino IDE и процеса на работа е същия както при Ардуино.

NodeMCU е създаден скоро след излизането на ESP8266. На 30 декември 2013 г. Espressif Systems започва производството на ESP8266. ESP8266 е Wi-Fi SoC, интегриран с ядро ​​Tensilica Xtensa LX106, което се използва широко в приложенията на Интернет на нещата (вижте по долу). NodeMCU стартира на 13 октомври 2014 г., когато Хонг ангажира първия файл с nodemcu-фърмуер към GitHub. Два месеца по-късно проектът се разшириява, за да включи отворена хардуерна платформа, когато разработчикът Huang R предава графичния файл на ESP8266 платка, наречена devkit v0.9. По-късно през този месец Tuan PM пренеся MQTT(Message Queuing Telemetry Transport - протокол за съобщения, работещ на принципа публикуване-абониране. Той работи над TCP / IP протокола. Той е предназначен за свързване с отдалечени места, където е необходим "дребен код" или честотната лента на мрежата е ограничена.) клиентската библиотека от Contiki на платформата ESP8266 SoC и се ангажира с проекта NodeMCU, след което NodeMCU успява да поддържа протокола MQTT IoT, като използва Lua за достъп до брокера MQTT. Друга важна актуализация е направена на 30 януари 2015 г., когато Devcasurus пренеся u8glib в проекта NodeMCU, позволявайки на NodeMCU лесно да управлява LCD, екран, OLED и дори VGA дисплеи.

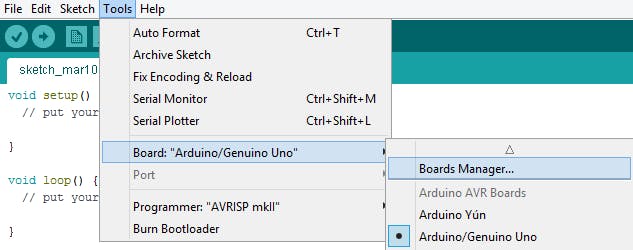
През лятото на 2015 г. създателите са изоставили проекта и група от независими, но посветени сътрудници го поемат. До лятото на 2016 г. NodeMCU включва повече от 40 различни модула. Поради ограничения на ресурсите, потребителите трябва да избират модулите, подходящи за техния проект, и да изградят фърмуер, съобразен с техните нужди.

* 1. **Как се програмира NodeMCU?**

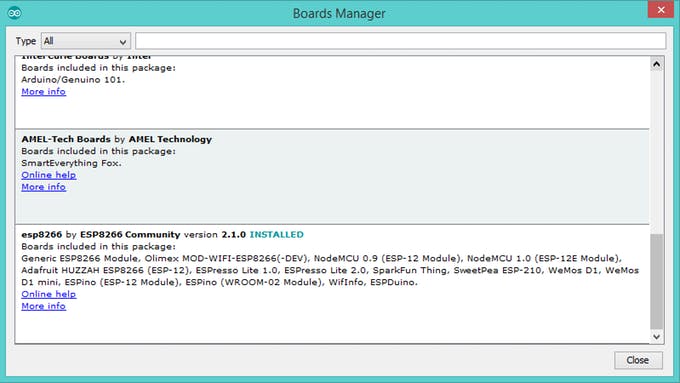
Първо трябва да изтеглите всички файлове от този адрес: <https://github.com/AritroMukherjee/ESP8266-NodeMcuV3>. Файлът съдържа всички необходими драйвери за този модул. Това ще ви спести доста време.

Трябва да имате инсталирана среда за програмиране на Ардуино. Ако нямате можете да си изтеглите от <https://www.arduino.cc/en/Main/OldSoftwareReleases>.

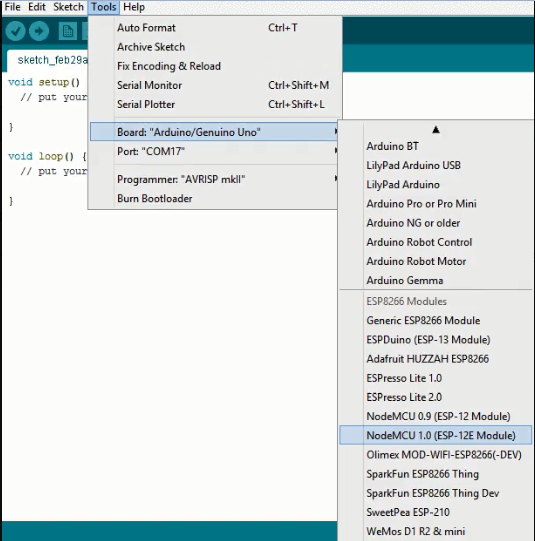
След като имате инсталиран Arduino IDE трябва да настроите програмата да може да работи с NodeMCU. За да направите това изпълнете указанията паказани на примера долу:



В отворения прозорец открийте раздела за ESP и инсталирайте.



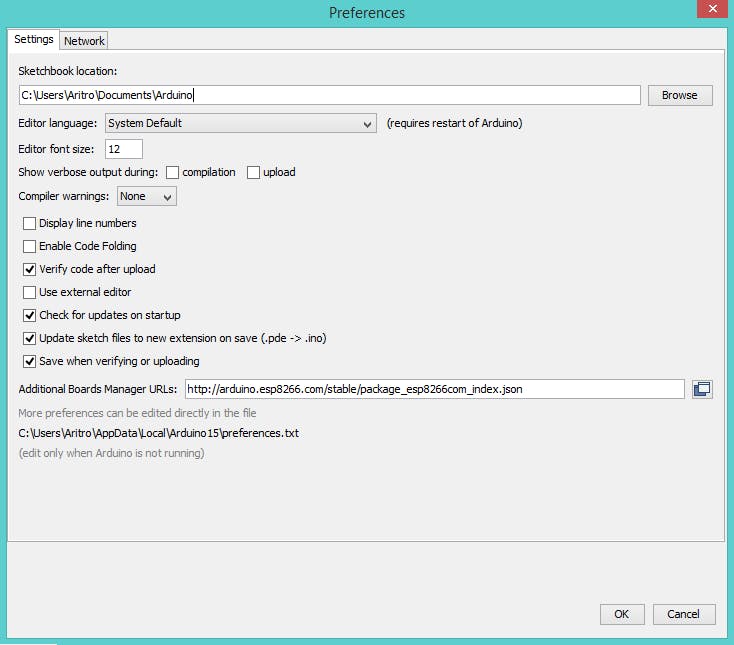
След тази стъпка вече може да видите че в (Tools > Board > ESP8266 Modules) има нови избори на платки. За нашата цел ни трябва:



След това от същото падащо меню изберете порта. В нашия пример сме избрали Port: “COM7”.

И последната стъпка която трябва да направите е да отворите (File > Preferences) и да копирате адреса: <http://arduino.esp8266.com/stable/package_esp8266com_index.json>

и да го поставите в полето Additional Boards Manager URLs.



Готово! Вече можем да пишем програми за NodeMCU така както за Ардуино.

* 1. **Захранване на NodeMCU**

Има три варианта за захранване:

1. **Използвайте USB захранване** – най-удобно за зареждане на програми, но не толкова добре ако искате действително да изключите проекта си от компютъра.
2. **Осигурете директно 3.3V** - това е една добра опция. С вашия собствен регулатор извън борда, можете да осигурите надежден източник на захранване за вашето устройство.
3. **Осигурете захранване чрез VIN порта** - Регулаторът е с номинал до 800 mA. В много случаи това е повече от достатъчно. Трябва да се внимава да следите натоварването, ако имате намерение да захранвате други устройства от пина 3.3.V.

**2.3 Какво представлява ESP8266?**

ESP8266 е едночипова система (на англ.: system on a chip, SoC, произнася се „ес-оу-си“), произведена от китайската компания Espressif. Състои се от 32-битов микроконтролер Tensilica L106 (MCU) и Wi-Fi приемно-предавателно устройство. Той има 11 GPIO пина \* (входно-изходни изходи за обща предназначение) и аналогов вход. Това означава, че можете да го програмирате като всеки нормален Arduino или друг микроконтролер. Освен това получавате Wi-Fi комуникация, така че да можете да я използвате, за да се свържете с Wi-Fi мрежата, да се свържете с интернет, да хоствате уеб сървър с реални уеб страници, да свързвате смартфона с нея и т.н. .. Възможностите са безкрайни! Не е чудно, че този чип се е превърнал в най-популярното устройство за IOT.

Има много различни модули, самостоятелни модули като ESP - ## серията от AI Thinker или пълни платки за развитие като NodeMCU DevKit или WeMos D1. Различните платки могат да имат различни разклонения, да имат различни Wi-Fi антени или различно количество флаш памет на борда. В тази дипломна работа ще използваме NodeMCU.

**2.4 Хардуер на ESP8266**

ESP8266 често се използва като сериен мост към WiFi, но също така той е и много мощен микроконтролер. В тази точка ще разгледаме специфичните за ESP8266 функции които не се отнасят за Wi-Fi.

* 1. **Цифрови входове и изходи**

Също като нормално Arduino, ESP8266 има цифрови входно-изходни пинове (I / O или GPIO, входно-изходни пинове за общо предназначение). Както подсказва името, те могат да се използват като цифрови входове за четене на цифрово напрежение, или като цифрови изходи 0V (заземяванек) или 3.3V (източник).

* + 1. **Напрежение и ограничения на ток**

ESP8266 е 3.3V микроконтролер, така че неговият вход / изход работи при 3.3V. Пиновете не са толерантни на 5V, прилагане на повече от 3.6V на всеки ПИН ще изгори чипа. Максималният ток, който може да се извлече от един GPIO пин е 12mA.

* + 1. **Използваеми пинове**

ESP8266 има 17 GPIO пина (0-16), но можете да използвате само 11 от тях, защото 6 пина (GPIO 6 - 11) се използват за свързване на флаш памет чип. Това е малкият чип с 8 крака точно до ESP8266. Ако се опитате да използвате някой от тях, може да сринете програмата си.

GPIO 1 и 3 се използват като TX и RX на хардуерния сериен порт (UART), така че в повечето случаи не можете да ги използвате като нормални I / O докато изпращате / получавате серийни данни.

* + 1. **ШИМ(широчинно импулсна модулация)**

За разлика от повечето Atmel чипове (Arduino), ESP8266 не поддържа хардуерен ШИМ, но софтуерен ШИМ се поддържа на всички цифрови пинове. Стандартният обхват на ШИМ е 10-битов @ 1kHz, но той може да бъде променен (до> 14 бита @ 1kHz).

* + 1. **Аналогов вход**

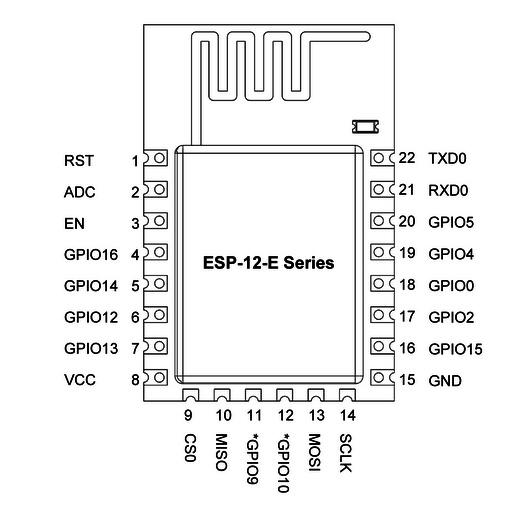
ESP8266 има един аналогов вход с входен диапазон 0 - 1.0V. Ако доставите 3.3V, например, ще повредите чипа. Някои платки като NodeMCU имат бордов резистивен делител на напрежението, за да получите по-лесен обхват от 0 до 3.3V. Можете също така просто да използвате малък потенциометър като делител на напрежение. ADC (аналогово-цифров преобразувател) има резолюция от 10 бита.

* + 1. **Комуникация**
       1. **Серийно**

ESP8266 има два хардуерни UARTS (серийни порта):

UART0 на пин 1 и 3 (TX0 и RX0 съответно) и UART1 на пин 2 и 8 (TX1 и RX1 resp.), Обаче, GPIO8 се използва за свързване на флаш чипа. Това означава, че UART1 може само да предава данни. UART0 също има контрол на хардуерния поток на пинове 15 и 13 (RTS0 и CTS0 resp.). Тези два пина също могат да се използват като алтернативни пинове TX0 и RX0.

* + - 1. **I2C**

ESP не разполага с хардуерен TWI (двужилен интерфейс), но се прилага в софтуера. Това означава, че можете да използвате почти всеки два цифрови пина. По подразбиране библиотеката I²C използва пин 4 като SDA и пин 5 като SCL. (Информационният лист определя GPIO2 като SDA и GPIO14 като SCL.) Максималната скорост е приблизително 450kHz.

фигура 3 - пинове на използвания в NodeMcu ESP модел

* + - 1. **SPI**

ESP8266 има една SPI връзка на разположение на потребителя, наречена HSPI. Той използва GPIO14 като CLK, 12 като MISO, 13 като MOSI и 15 като Slave Select (SS). Той може да се използва както в режим Slave, така и в режим Master (в софтуера).

|  |  |  |  |
| --- | --- | --- | --- |
| GPIO | Функция | Състояние | Ограничения |
| 0 | Избор на начин на пускане | 3.3V | Без висок импеданс |
| 1 | TX0 | - | Не може да бъде използван по време на серийна комуникация |
| 2 | Избор на начин на пускане TX1 | 3.3V (само при вкл.) | Да не се заземява по време на пускане. Изпраща данни за дебъгване по време на пускане |
| 3 | RX0 | - | Не може да се използва по време на серийна комуникация |
| 4 | SDA(I2C) | - |  |
| 5 | SCL(I2C) | - |  |
| 6-11 | Флаш връзка | x | Не може да се използва |
| 12 | MISO (SPI) | - |  |
| 13 | MOSI(SPI) | - |  |
| 14 | SCK(SPI) | - |  |
| 15 | SS(SPI) | 0V | Резистор за високо ниво. Не може да се използва |
| 16 | Събуждане от сън | - | Без резистор за високо ниво а за ниско.Трябва да бъде свързан към RST за да бъде събуден. |

Таблица3. Описание на пиновете на ESP8266

* 1. **Софтуер на ESP8266**

По-голямата част от функционалността на микроконтролера на ESP използва точно същия синтаксис като нормалния Arduino, което го прави много лесен за начинаещите.

* + 1. **Цифрови входове и изходи**

Също като с обикновен Arduino, можете да зададете функцията на пин, като използвате pinMode (pin, mode); където pin е GPIO номер \*, а режимът може да бъде или INPUT(вход), който е по подразбиране, OUTPUT(изход) или INPUT\_PULLUP, за да активирате вградените резистори за високи нива за пинове GPIO 0-15. За да активирате резистор за ниско ниво на пин GPIO16, трябва да използвате INPUT\_PULLDOWN\_16.

(\*) NodeMCU използва различно картографиране, вижте по долу. За адресиране на щифт NodeMCU, напр. pin 5, използвайте D5: например: pinMode (D5, OUTPUT);

За да зададете изходен пин висок (3.3V) или нисък (0V), използвайте digitalWrite (pin, value); където пин е цифровият пин, и стойността е 1 или 0 (или HIGH и LOW). За да прочетете вход, използвайте DigitalRead (pin);

За да активирате ШИМ на определен пин, използвайте analogWrite (pin, value); където pin е цифровият пин, и стойност между 0 и 1023.

Можете да промените обхвата (битовата дълбочина) на изхода на PWM, като използвате analogWriteRange (new\_range);

Честотата може да се промени, като се използва analogWriteFreq (new\_frequency) ;. new\_frequency трябва да бъде между 100 и 1000Hz.

* + 1. **Аналогов вход**

Също като на Arduino, можете да използвате analogRead (A0), за да получите аналогово напрежение на аналоговия вход. (0 = 0V, 1023 = 1.0V).

ESP може да използва и ADC(АЦП) за измерване на захранващото напрежение (VCC). За да направите това, включете ADC\_MODE (ADC\_VCC); в горната част на скицата и използвайте ESP.getVcc (); за да получите действително напрежение.

Ако го използвате за четене на захранващото напрежение, не можете да свържете нищо друго с аналоговия пин.

* + 1. **Комуникация**

**Серийна комуникация**

За да използвате UART0 (TX = GPIO1, RX = GPIO3), можете да използвате Serial обекта, точно както при Arduino: Serial.begin (baud).

За да използвате алтернативните пинове (TX = GPIO15, RX = GPIO13), използвайте Serial.swap () след Serial.begin.

За да използвате UART1 (TX = GPIO2), използвайте обекта Serial1.

Всички функции на Arduino Stream, като четене, писане, отпечатване, println, ... също се поддържат.

**I²C и SPI**

Можете просто да използвате библиотеката на Arduino по подразбиране, както обикновено.

* + 1. **Споделяне на времето на CPU с RF частта**

Едно нещо, което трябва да имате предвид при писането на програми за ESP8266, е, че вашата скица трябва да споделя ресурси (време на процесора и паметта) със стекове Wi-Fi и TCP (софтуера, който работи във фонов режим и се справя с всички Wi-Fi и IP връзки).

Ако кодът Ви отнема много време, за да се изпълни и не позволявате на TCP стаковете да правят нещо, то програмата може да се срине или да загубите данни. Най-добре е да задържите времето за изпълнение на цикъла под няколко стотици милисекунди.

Всеки път, когато основният цикъл се повтаря, скицата ви дава на Wi-Fi и TCP да обработват всички заявки за Wi-Fi и TCP.

Ако цикълът Ви отнема повече време, ще трябва изрично да дадете CPU време на стека на Wi-Fi / TCP, като използвате командата delay(0); или yield(); Ако не го направите, мрежовата комуникация няма да работи както се очаква, а ако е по-дълго от 3 секунди, софтуерният WDT (Watch Dog Timer) ще нулира ESP. Ако софтуерният WDT е деактивиран, след малко повече от 8 секунди хардуерният WDT ще нулира чипа.

От гледна точка на микроконтролера, обаче, 3 секунди са много дълги времена (240 милиона цикъла на часовника), така че ако не правите някои изключително тежки изчисления или ще изпращате изключително дълги текстове серийно, няма да бъдете засегнати от това. Просто имайте предвид, че трябва добавяте yield(); в рамките на while цикли, които може да отнемат повече от, да речем 100ms

* 1. **Wi-Fi**

### Използването на ESP8266 като елементарен микроконтролер е страхотно, но причината, поради която повечето хора го използват, е възможностите му за Wi-Fi. В тази точка разгледаме мрежови протоколи като Wi-Fi, TCP, UDP, HTTP, DNS TCP/IP стек

Системата, която повечето хора наричат ​​"Интернет", не е само един протокол: това е целият стек от слоеве от протоколи, често наричани TCP / IP стек. Ще преминем през тези различни слоеве, за да разберем как нашият ESP8266 комуникира с други устройства в мрежата.

| **Слой** | **Протоколи** |
| --- | --- |
| Application | HTTP, FTP, mDNS, WebSocket, OSC ... |
| Transport | TCP, UDP |
| Internet | IP |
| Link | Ethernet, Wi-Fi ... |

### The Link layer (Слой за връзка)

Слоят за връзка съдържа физическата връзка между две устройства, например Ethernet кабел или Wi-Fi връзка. Това е слоят, който е най-близо до хардуера.

За да свържете ESP8266 към мрежата, трябва да създадете Wi-Fi връзка. Това може да се случи по два различни начина:

* ESP8266 се свързва с безжична точка за достъп (WAP или просто AP). AP (Acces point – точка за достъп) например може да бъде вграден в модема или маршрутизатора ви. В тази конфигурация системата ESP действа като безжична станция.
* ESP8266 действа като точка за достъп и безжичните станции могат да се свързват с нея. Тези станции могат да бъдат вашият лаптоп, смартфон или дори друг ESP в станция.

След като се установи връзка Wi-Fi, ESP8266 е част от локална мрежа (LAN). Всички устройства в LAN могат да комуникират помежду си.

По-голямата част от времето AP е свързан и с физическа Ethernet мрежа, което означава, че ESP8266 може да комуникира и с устройства, свързани към AP (модем / рутер) чрез кабелна Ethernet връзка (настолни компютри, конзоли за игри и декодери, например).

Ако ESP8266 е в режим на точка за достъп, той може да комуникира с която и да е станция, която е свързана с него, и две станции (например лаптоп и смартфон) могат да комуникират един с друг.

ESP може да се използва само в AP-режим, само за станция или AP + станция.

### The Internet or Network layer(Мрежов слой)

Въпреки че устройствата вече са физически свързани (чрез действителни кабели (Ethernet) или чрез радиовълни (Wi-Fi)), те всъщност не могат да говорят помежду си, защото нямат начин да знаят къде да изпратят съобщението ,

Тук е идва и ролята на Интернет протоколът (IP). Всяко устройство в мрежата има личен IP адрес. DHCP сървърът (Dynamic Host Configuration Protocol Server) гарантира, че тези адреси са уникални.

Това означава, че вече можете да изпратите съобщение до определен адрес.

Има две версии на интернет протокола: IPv4 и IPv6. IPv6 е подобрена версия на IPv4 и има много повече адреси от IPv4 (защото има много повече устройства, отколкото наличните IPv4 адреси). В тази дипломна ще имаме впредвид само за IPv4 адреси, тъй като повечето LAN все още ги използват.

IP адресът се състои от 4 номера, например 192.168.1.5 е валиден IPv4 адрес. Всъщност тя се състои от две части: първата част е 192.168.1, това е адреса на локалната мрежа. Последната цифра, 5 в този случай, е специфична за устройството.

С помощта на IP адреси можем да намерим ESP8266 в мрежата и да изпратим съобщения до него. ESP може да намери нашия компютър или нашия телефон, ако знае съответните си IP адреси.

#### Връзка с интернет

Както може би сте забелязали, споменахме само локалната мрежа, това са компютрите във вашата собствена мрежа. ESP8266 да комуникира с интернет. Цялата мрежова инфраструктура, включена в "Интернет" се подчинява на правилата за IP, за да се уверите, че повечето от вашите пакети пристигат там където трябва.

**Транспортният слой**

Различните устройства в мрежата правят всичко възможно да доставят тези IP пакети на адреса, но не е необичайно пакетът да се загуби, така че никога да не пристигне. Или пакетът може да се развали по пътя: данните вече не са правилни. IP също не може да гарантира, че пакетите пристигат в същия ред, в който са изпратени. Това означава, че все още не можем да изпращаме надеждни съобщения, като използваме само връзката и интернет слоя, тъй като никога не можем да разберем кога и дали пакетът ще пристигне, или знаем със сигурност, че полученият пакет е правилен.Нуждаем се от трети слой върху слоя "Интернет": слоя "Транспорт".

Съществуват предимно два протокола, които съставят този трети слой: протокола за контрол на предаването (TCP) и протокола User Datagram Protocol (UDP).

* TCP гарантира, че всички пакети са получени, че пакетите са в ред и че повредените пакети са изпратени отново. Това означава, че може да се използва за комуникация между множество приложения, без да се притеснявате за целостта на данните или загубата на пакети. Ето защо се използва за неща като изтегляне на уеб страници, изпращане на имейл, качване на файлове и др
* UDP, от друга страна, не гарантира, че всеки пакет достига до целта си, той проверява за грешки, но когато открие такава, просто унищожава пакета, без да го изпраща отново. Това означава, че той не е толкова надежден, колкото TCP, но е по-бърз и има много по-ниска латентност, защото не изисква открита връзка за изпращане на съобщения, като TCP. Ето защо се използва при гласови и видео разговори, например в онлайн игри.

### Приложен слой

#### Вече имаме надеждна комуникация чрез TCP, но все още има един проблем. Помислете си по този начин: изпращате писмо и TCP гарантира, че ще стигне до местоназначението си, но ако получателят не разбира езика, на който е написан, той няма да знае какво да прави с него.

#### С други думи, имаме нужда от четвърти слой протоколи, за да могат две програми да комуникират помежду си.

#### Има много различни протоколи, но най-вече ще се съсредоточим върху протоколите за уеб сървъри и браузъри.

#### HyperText Transfer Protocol (Протокол за прехвърляне на HyperText)

#### Протоколът за прехвърляне на HyperText или HTTP е протоколът (език), който се използва от уеб сървърите и уеб клиентите за комуникация. Той използва текст, за да изпълнява заявки за изпращане и отговори от клиента на сървъра и обратно.

#### Например, когато въведете http://www.google.com в адресната лента на уеб браузър (клиент), той ще изпрати заявка HTTP GET на уеб сървъра на Google. Сървърът разбира тази HTTP заявка и ще изпрати уеб страницата на Google като отговор. Или когато качите изображение в Instagram, вашият браузър изпраща заявка за HTTP POST с вашето изображение, свързано към сървъра на Instagram. Сървърът разбира искането, запазва изображението и го добавя в базата данни, изпраща URL адреса на новото изображение обратно в браузъра ви и браузърът ще добави изображението в уеб страницата.

#### Както можете да видите, нито клиентът, нито сървърът трябва да се притесняват за целостта на изпратените от тях съобщения и знаят, че получателят разбира техния език и че ще знае какво да прави с определена HTTP заявка.

#### Повечето съвременни сайтове използват защитена версия на HTTP, наречена HTTPS. Защитената връзка криптира данните от съображения за сигурност.

#### WebSocket

#### HTTP е чудесен за неща като изтегляне на уеб страници, качване на снимки и т.н., но е доста бавен: всеки път, когато изпращате HTTP заявка, трябва да стартирате нова TCP връзка към сървъра, след това да изпратите заявката си, да изчакате сървъра да отговори, и изтеглите отговора. Не би ли било чудесно, ако

#### не е нужно да откриваме нова връзка всеки път, когато искаме да изпратим някои данни и ако можем да изпращаме и получаваме данни по едно и също време по всяко време, което бихме искали? Това е мястото, където WebSocket идва на помощ: можете да поддържате TCP връзката със сървъра отворена по всяко време, получавате добра TCP надеждност и е доста бързо.

#### Open Sound Control(Отворен звуков контрол)

#### HTTP и WebSocket използват TCP връзки. Какво ще стане, ако искате по-ниска латентност? Е, Open Sound Control или OSC, използва UDP за изпращане на малки парчета данни, като инчове, плувки, кратък текст и т.н. ... с много ниска латентност. Първоначално е проектиран за управление на аудио приложения с ниска латентност, но е много гъвкав протокол, така че често се използва за задачи с ниска латентност, различни от аудиоконтрола.

#### Система за имена на домейни

Както вече споменахме, можете да изпратите съобщение само на друг компютър, ако знаете неговия IP адрес. Но когато сърфирате в интернет, знаете само името на домейна на уебсайт (например www.google.com). Вашият компютър използва системата за имена на домейни, за да преведе това име на домейн на правилния IP адрес.

* 1. **ESP8266 Уеб сървър**

### В тази точка ще покрием основите на уеб сървърите и как се създават чрез ESP

### Уеб сървъри

### Уеб сървърът е свързано с интернет устройство, което съхранява и обслужва файлове.

### Клиентите могат да поискат такъв файл или други данни, а след това сървърът ще изпрати данните и файловете на клиента. Исканията се извършват чрез HTTP.

### HTTP

### HTTP или Hypertext Transfer Protocol е текстовият протокол, използван за комуникация с (уеб) сървъри. Има няколко метода за заявка по HTTP, но ще покрием само двете най-широко използвани: GET и POST.

### HTTP GET

### GET заявки се използват за извличане на данни от сървър, например уеб страница. Тя не трябва да променя нищо на сървъра, а просто да получава данните от сървъра, без странични ефекти.

### Когато отворите уеб страница в браузъра си, тя ще вземе URL адреса и ще го постави в заявка HTTP GET. Това е просто текст. След това тя ще изпрати заявката на правилния сървър, като използва TCP. Сървърът ще прочете заявката, ще провери URL адреса и ще изпрати правилния HTTP отговор за този URL обратно в браузъра.

### Анатомията на GET заявка

### Най-важните части от искането за GET са редът за заявка и заглавката на хоста. Нека да разгледаме един пример:

### Ако кликнете върху следната връзка: https://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html, вашият браузър ще изпрати следната HTTP заявка:



Първият ред е заявката: съдържа метода на искане: GET, в този случай URI или Uniform Resource Identifier: /Protocols/rfc2616/rfc2616-sec5.html и HTTP версията: 1.1.

Вторият ред е заглавието на хоста, той определя името на домейна на хоста (сървъра).

Има и много други заглавки, но те не са наистина важни, когато използвате ESP8266.

Повечето сървъри ще проверят дали URI файлът е във файловата си система и ако случаят е такъв, те ще изпратят този файл като отговор.

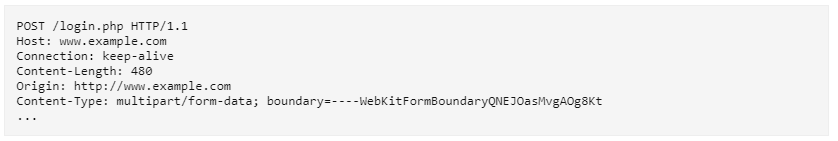
**HTTP POST**

POST заявките се използват за изпращане на данни до сървъра, например, за да изпратите вашето потребителско име и парола на сървъра, когато влезете или когато качите снимка. За разлика от GET, POST може да променя данните на сървъра или състоянието на сървъра.

Тялото на POST заявката може да съдържа данни, които се изпращат на сървъра.

**Анатомията на заявката за POST**

Например, страницата за вход на любимия ви сайт може да изпрати нещо подобно, когато въведете идентификационните си данни и кликнете върху бутона за вход:

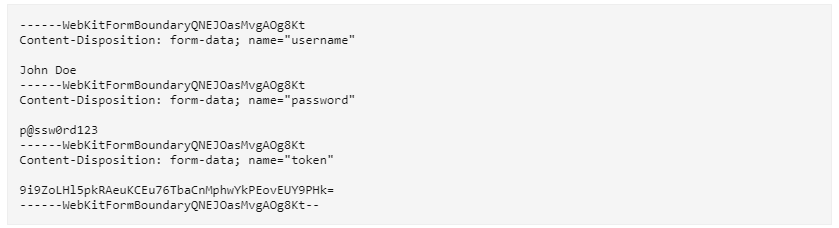


Както можете да видите, редът за заявки вече има метода POST в него и все още е последван от URI, /login.php и HTTP версията, 1.1. Главният хост все още съдържа само името на домейна.

Реалната разлика е тялото на заявката: заявката GET няма изпратени данни, докато можете да добавите много данни към тялото на заявка за POST. Те могат да бъдат нормални двойки ключ-стойност, като потребителско име и парола или действителни файлове, които се качват.

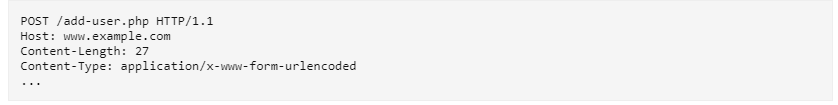
Също така имайте предвид заглавката Content-Type: тя казва на сървъра какви данни могат да бъдат намерени в тялото на заявката POST.

Нека да разгледаме тялото на примера за вход:



Както можете да видите, в тялото има три параметъра, всеки параметър има име (например потребителско име) и стойност (например John Doe).

Можете също да използвате същия синтаксис, който използвахме преди, когато добавяме параметри към заявка GET:



И изпратените данни:



Както можете да видите, заглавието на Content Type е различно, което показва, че кодирането на стойностите в изпратените данни е различно.

**Кодове на състоянието на HTTP**

Сървърът трябва да отговаря на всички заявки с код за състоянието на HTTP. Това е трицифрено число, което показва дали заявката е била успешна или да уведоми клиента какво се е объркало. Ето една таблица с някои от най-важните и полезни.

| **Статус код** | **Значение** |
| --- | --- |
| 200 | OK: заявката е успешна |
| 303 | See Other: използва се за пренасочване например след POST заявка |
| 400 | Bad Request: сървърът не успя да интерпретира заявката |
| 401 | Unauthorized: нужна е ауторизация |
| 403 | Forbidden: сървърът отказва да изпълни заявката ауторизацията няма да помогне |
| 404 | Not Found: адресът не е намерен |
| 500 | Internal Server Error: Сървърът срещна неочаквано условие и заявката не беше изпълнена |

**TCP и UDP портове**

В повечето случаи едно устройство предлага много различни услуги, например уеб сървър, имейл сървър, FTP сървър, Spotify стрийминг услуга, ...

Ако устройството има само IP адрес, би било невъзможно да се знае кое приложение е изпратил пакет. Ето защо всяка услуга има номер на порт. Това е идентификатор за всички различни услуги или приложения на едно устройство. В горния пример уеб сървърът ще слуша само заявки на порт 80, сървърът за електронна поща само на порт 25, FTP сървърът само на порт 20, Spotify ще получава потоци само на порт 4371 …

За да определите даден порт, можете да използвате двоеточие след IP адреса след името на домейна. Но през повечето време не е нужно да го добавяте изрично. Например, всички уеб сървъри слушат на порт 80, така че уеб браузър винаги ще се свързва към порт 80.

***Глава 3 –*** ***Разработка на програмно осигуряване***

Разработени са две приложения - сървър и клиент които комуникират помежду си.

Сървърното приложение е създадено чрез програмна среда Ардуино и C/C++. Задачата на това приложение е да получава и отговаря на уеб заявки. В зависимост от получената заявка програмата изпраща серийни команди към iRobot Create. Робота и платката са хардуерно свързани и серийната комуникация става чрез пиновете RX и TX на платката и робота. RX – пинът който чете изпратената команда от пин TX. Следователно за да работи правилно комуникацията трябва да свържем RX с ТХ и обратното. Заявките могат да бъдат изпратени от всяко приложение което знае URL-адреса на сървъра и е свързан към същата мрежа.

С цел по-удобна работа е написано и второ приложение – уеб интерфейс който може да изпраща заявки и да обработва получените отговори. Той предоставя разнообразни уеб елементи с които можем лесно и интуитивно да настроим параметрите на командите които ще се изпратят.

Сървъра е запеметен в паметта за програмен код в чипа на NodeMCU – Esp8266 а уеб приложението в неговата файловата система. Разликата между двете приложения е и в това къде те се изпълняват – сървърното приложение се изпълнява в средата NodeMCU, а уеб клиента се изпълнява в броузъра(или друго подобно приложение) на потребителя.

*Пример за създаване на сървър:*

#include <ESP8266WiFi.h>

#include <WiFiClient.h>

#include <ESP8266WiFiMulti.h>

#include <**ESP8266mDNS**.h>

#include <**ESP8266WebServer**.h> // Включва библитеката за уеб сървър ESP8266WiFiMulti wifiMulti; // Създава обект от тип ESP8266WiFiMulti

**ESP8266WebServer** server(80); // Създава уеб сървър обект който слуша на порт 80

void handleRoot(); // Функция за обработка на заявки

void handleNotFound();

void setup(void){ **Serial**.begin(115200); // Старт на серийна комуникация

delay(10);

**Serial**.println('\n');

wifiMulti.addAP("ssid\_from\_AP\_1", "your\_password\_for\_AP\_1"); // Добавя мрежи към които искаме да се свържем

wifiMulti.addAP("ssid\_from\_AP\_2", "your\_password\_for\_AP\_2"); wifiMulti.addAP("ssid\_from\_AP\_3", "your\_password\_for\_AP\_3"); **Serial**.println("Connecting ...");

int i = 0;

while (wifiMulti.run() != WL\_CONNECTED) { // Изчакваме връзка с мрежа и избираме тази с най силен сигнал

delay(250);

**Serial**.print('.');

}

**Serial**.println('\n'); **Serial**.print("Свързан към: "); **Serial**.println(**WiFi**.SSID()); // Изписва мрежата към която сме се свързали **Serial**.print("IP адрес:\t");

**Serial**.println(**WiFi**.localIP()); // Изпраща ип адреса на esp серийно

server.on("/", handleRoot); // Извиква функцията handleRoot когато клиента направи заявка към съответния URL адрес в случая „/“

server.onNotFound(handleNotFound); // Когато клиента направи заявка към несъществуваща страница извикай функцията handleNotFound

server.begin(); // Реален старт на сървъра

**Serial**.println("HTTP сървърът работи"); }

void loop(void){

server.handleClient(); // Слуша за HTTP заявки от клиенти

}

void handleRoot() {

server.send(200, "text/plain", "Здравей свят!"); // Изпраща HTTP заявка със статус 200 (Ok) и примерен текст

}

void handleNotFound(){

server.send(404, "text/plain", "404: Not found"); // Изпрати HTTP заявка със статус 404 (Not Found) когато адреса не е намерен

}

Първо, създаваме сървърен обект, който слуша HTTP заявки на порт 80. Това е портът по подразбиране за уеб сървъри. В настройката разясняваме на сървъра какво да прави с определени HTTP заявки. При заявка с URI '/', сървърът трябва да отговори с HTTP код на 200 (Ok) и след това да изпрати отговор с текст „Здравей свят!“. Поставяме кода за генериране на отговор в отделна функция и казваме на сървъра да го изпълни, когато се поиска "/", като се използва функцията server.on.

Не сме посочили какво трябва да направи сървърът, ако клиентът поиска URI, различно от "/". Той трябва да реагира със състояние HTTP 404 (Не е намерено) и съобщение за потребителя. Включихме и тази функция и използвахме server.onNotFound, за да кажем, че трябва да я изпълни, когато получи заявка за URI, която не е посочена със сървъра.

След това започваме да слушаме HTTP заявки, като използваме server.begin.

По време на цикъла непрекъснато проверяваме дали е получена нова HTTP заявка при стартиране на server.handleClient. Ако handleClient открие нови заявки, автоматично ще изпълни правилните функции, които сме посочили в настройката.

За да го изпробвате, качете скицата, отворете нов раздел на браузъра и прегледайте http: //esp8266.local. Трябва да получите уеб страница, казвайки "Здравей свят! , Ако се опитате да отидете на друга страница, http: //esp8266.local/test, например, трябва да получите грешка 404: 404: Не е намерена.

**3.1** **Използване на уеб сървъра за включване и изключване на LED диод**

Можем да използваме уеб сървъра, за да обслужваме интерактивни страници и да реагираме на определена заявка за POST. В следващия пример ESP8266 е домакин на уеб страница с бутон. Когато бутонът бъде натиснат, браузърът изпраща POST заявка / LED. Когато ESP получи такава POST заявка в / URI LED, той ще включи или изключи светодиода, след което ще пренасочи браузъра обратно към началната страница с бутона.

За да извършите това пренасочване трябва да добавите хедър за местоположение към отговора и да използвате 303 HTTP код на състоянието.



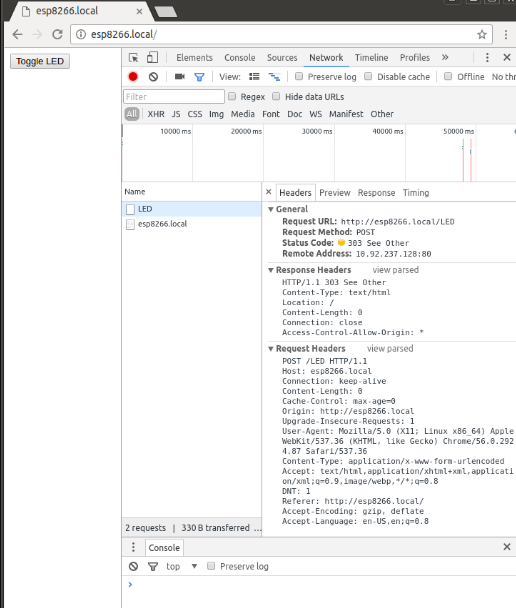
Бутонът за изпращане на заявката за POST в браузъра е част от HTML формуляр. Трябва да посочите целевия URI, за да изпратите искането, и метода на заявката, в този случай това е "/ LED" и POST съответно.

Имайте предвид, че променихме типа съдържание на отговора от "text / plain" на "text / html". Ако го изпратите като обикновен текст, браузърът ще го покаже като текст, вместо да го интерпретира като HTML и да го показва като бутон.

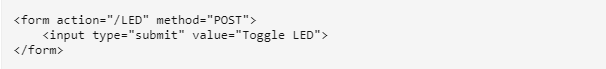
Както можете да видите, функцията server.on има три параметъра: URI, метода на заявката (GET или POST) и функцията, която трябва да се изпълни.

Свържете LED към GPIO2 и качете скицата. След това отидете на http: //esp8266.local/ и кликнете върху бутона, за да включите или изключите светодиода.

Можете да отворите опциите за програмисти в Chrome (F12), за да проверите HTTP заявката, която се прави, когато кликнете върху бутона: ще видите, че първо изпраща заявка за POST и след това получава статус HTTP 303 като отговор. Отговорът има и заглавка на местоположението, съдържащо URI "/", така че браузърът ще изпрати заявка GET на URI на това ново местоположение:



Ако проверите сорс кода на страницата (CTRL + U), можете да видите простия HTML формуляр, който се използва:



**3.1.1** **СПИ(серийен периферен интерфейс) Флаш файлова система**

### В предишните пример винаги включвахме HTML кода на нашите уеб страници като низове в нашата скица. Това прави нашия код много труден за четене и води до бързо изчерпване на паметта.

### Serial Peripheral Interface(СПИ) Flash файлова система, или SPIFFS за кратко. Това е лека файлова система за микроконтролери с SPI флаш чип. Вграденият флаш чип на ESP8266 има достатъчно място за вашите уеб страници, особено ако имате версия с памет 1MB, 2MB или 4MB.

### SPIFFS ви дава достъп до флаш паметта, сякаш е нормална файлова система като тази на вашия компютър (но много по-опростена разбира се): можете да четете и пишете файлове, да създавате папки и т.н.

### Най-лесният начин да научите как да използвате SPIFFS е да разгледате някои примери. Но файлова система без файлове, които да се показват, е доста безсмислена, затова първо ще разгледаме как можем да качваме файлове на SPIFFS.

* + 1. **Качване на файлове на SPIFFS**

За да изберете правилните файлове, които да бъдат качени, трябва да ги поставите в папка, наречена "data", в папката на вашия проект: Отворете wa[ata скица в IDE на Arduino и натиснете CTRL + K. Изчакайте да се отвори прозорец на файловия прозорец и създайте нова папка, наречена “data”. Копирайте файловете си в тази папка. (Използвайте само малки файлове като текстови файлове или икони. Няма достатъчно място за големи снимки или видеоклипове.)

След това изберете всички файлове в папката (CTRL + A) и проверете размера на всички комбинирани файлове (не забравяйте подпапките). Отидете отново на IDE на Arduino и под Tools> Flash Size, изберете опция с правилния размер на флаш панела и размер SPIFFS, който е по-голям от размера на папката с данни.

След това качете скицата. Когато това приключи, се уверете, че серийният монитор е затворен, след това отворете менюто "Инструменти" и кликнете върху качването на данни за скициране на ESP8266. Ако вашият ESP има автоматично нулиране и автоматично програмиране, той трябва да работи автоматично, ако нямате автоматична програма, трябва ръчно да влезете в режим на програмиране, преди да качите данните на SPIFFS. Процедурата е точно същата като влизането в програмен режим, преди да качите скица.

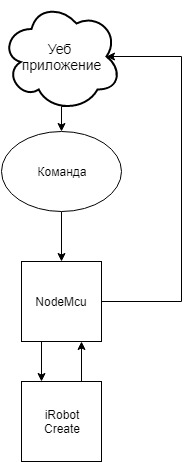
Ако получите грешка, казвайки SPIFFS\_write грешка (-10001): Файловата система е пълна, това означава, че вашите файлове са твърде големи, за да се поберат в паметта SPIFFS. Изберете по-голям размер SPIFFS под Инструменти> Флаш размер или изтрийте някои файлове.

Дори ако компютърът ви казва, че файловете са по-малки от избрания размер SPIFFS, все пак можете да получите тази грешка: това е свързано с размерите на блоковете и мета данните като имената на файлове и папки, които заемат място.

Ако промените размера на SPIFFS, трябва да презаредите скицата си, защото когато промените размера на SPIFFS, мястото на паметта ще бъде различно. Програмата трябва да знае актуализирания SPIFFS адрес, за да може да прочете файловете.

* 1. **Реализация на управление на IRobot Create чрез платформата NodeMcu**

Потребителите могат да използват робота чрез изпращане на серийни изречения. Това може да се имплементира с NodeMcu. За целта може да създадем уеб приложение с удобен за използване потребителски интерфейс. От този интерфейс можем да изращаме команди за управление на робота чрез интернет връзка.



**Изисквания**

**Робот:** Трябва да имате робот iRobot Create. Версията на физичесия робот няма голямо значение, но toolbox - а е разработен за работа с отворен интерфейс, създаден във версия 1.0. Той не е напълно тестван със серийния интерфейс на Roomba, въпреки това е доста сходен.

**Свързване:** Робот а и платформата NodeMcu трябва да могат да комуникират помежду си. Най-елементарния начин за постигане на това е чрез серийна комуникация.

**NodeMcu:** Тук съхраняваме уеб приложението с потребителски интерфейс и изпращаме серийни команди към робота

**Интернет връзка:** За да се свържем към уеб приложението ни трябва връзка с мрежата.

**Захранване:** Можете да използвате батерии или usb свързване за да захраните NodeMcu и робота.

***3.2.1*** ***Начин на работа***

**Хардуерно свързване**

1. Свържете робота с NodeMcu използвайки пиновете RX (D9) и ТX(D10) на NodeMcu свързани съответно към пинове TX(2) и RX(1);
2. Свържете захранването към робота и към NodeMcu. Пиновете за захранване на NodeMcu са Vin и Gnd(заземяване) няма значение кой от пиновете за заземяване използвате. В тази дипломна свързваме батерия 9 волта към Vin пина. Това не е препоръчителен начин за захранване на платката понеже батерията бързо се изхабява и се използва доста неефективно. По-добрия вариант е да подадете регулирано напрежение 5 волта към някой от пиновете за 5 волта.

**Софтуерно свързване**

За да работи приложението правилно трябва да отворим кода на програмата заредена на NodeMcu и да попълним полетата за връзка с интернет. Трябва да укажем към коя връзка искаме да се свържем и каква е паролата на тази мрежа. Използваме тези две променливи за конфигуриране на връзката.

const char\* ssid = ""; - Име на мрежата

const char\* password = ""; - Парола

Можем да настроим платката да търси свободни мрежи и да се свързва към тях ако е необходимо, но в тази дипломна този вариант не е разгледан.

След като сме записали данните за мрежата трябва да разберем какъв е URL адреса на уеб приложението. В зависимост от това към каква мрежа е свързано устройството адреса може да бъде различен. За да укажете правилния адрес следвайте стъпките долу:

1. Свръжете NodeMcu към компютър който има инсталирано Arduino и отворете приложението.
2. Отворете серийния монитор и вижте адреса който е изписан. Адреса се появява след като платката се е свързала към мрежата и може да приема команди от уеб приложението.
3. Копирайте адреса и го заместете с URL адреса въведен в двата файла на уеб приложението.
4. Запишете направените промени и качете файловете в файловата система на NodeMcu. За да направите това ...

**3.3** **Roomba в съответствие с отворения интерфейс на Create**

Отворения интерфейс на Roomba е подмножество на отворения интерфейс на Create. Това означава, че всички команди за Roomba работят върху Create; но не всички команди за Create работят върху Roomba. Всяка функция в кутията с инструменти с думата „Roomba” в името си работи и на Roomba и на Create. Но всяка функция с „Create” в името си работи само на Create.

Освен това има много команди в отворения интерфейс на Create, които не са включени в кутията с инструменти. Изглежда, че някои от т.нар. Create Scripting команди са излишни ако използвате Matlab. Също така ако се интересувате от управление на напрежението на изводите на товара (защото искате да добавите сензори или двигатели на робота) ще трябва да се консултирате с ръководството.

**Поток на данни (Стрийминг)** : Обърнете внимание, че робота има режим, където потоците данни от сензорите минават през серийния кабел непрекъснато. Трябва да изпратите команда до сензора всеки път, когато искате да прочете данните.

**Време :** PC и NodeMcu комуникират на 115200 бит/сек. Това означава, че има ограничение за това колко бързо робота може да изпраща, получава и отговаря на команди.

Съвет : Ако програмите ви спират да работят доста често, може да добавите малко време закъснение след всяка команда, за да дадете време на робота да обработи инструкциите.

**Уеб приложение:** Приложението което използваме за управление на робота е Single Page Application(приложение с една страница) или SPA т.е. всички действия в страницата като например задаване на посока на движение които правят уеб заявки не презареждат цялата страница както е при обикновените уеб страници. Пример за такова приложение е Gmail. Това е важна черта на потребителския интерфейс която ни дава следните предимства:

* Бързодействие – това че страницата не се презарежда изцяло означава че правим значително по-малко заявки следователно чакаме по-малко. Това ни позволява да изпращаме команди по-често към робота и да реагираме по-бързо на неочаквани обстоятелства. Изискванията от силна интернет връзка също намаляват.
* Възможност за работа в реално време.
* Подобрено потребителско изживяване – SPA приложенията ни придават чувството че работим с настолно приложение и възможностите за изграждане на по-добър интерактивен и външен вид са доста повече.

Използваната библиотека за SPA е Vue.js тя е с отворен код и се използва за изграждане на потребителски интерфейси.

**Зареждане на допълнителни файлове към приложението:** за да работи правилно потребителксия интерфейс използва външни библиотеки като например Vue.js За да включим тези файлове към нашия проект използваме файловата система на NodeMcu или зареждаме файловете от други интернет сайтове. Vue.js и Bootstrap(библиотека за придаване на по-добър външен вид на страницата) зареждаме през интернет, а пример за библиотека която зареждаме от файловата система на NodeMcu е:

**Nipple.js:** Това е библиотеката която използваме за управление на посоката на робота. Контролера ни позволява да вземем 4-те необходими параметъра за управлението на робота по интуитивен и интерактивен начин. Това са параметрите за скорост нисък и висок байт и ъгъл нисък и висок байт. В зависимост от координатите на контролера изчисляваме тези параметри в допустимия диапазон на стойностите. Това е пример за предимството на SPA понеже без този подход няма как да постигнем същия ефект.

*Пример*: След като разберем URL адреса на приложението влизаме в страницата и натискаме бутона за включване на робота. Това изпраща команда за стартиране на Create. Вече можем да видим всички команди които можем да изпратим към робота. Сменяме режима на безопасен и използваме контролера за движение за да зададем посока на движение на робота. Той започва да се движи по указаната посока.

**Забележка**: всяка команда изпратена от контролера за движение пристига със закъснение до робота поради времето което изпратената заявка от приложението отнема. Затова е препоръчително да изчакате около секунда за всяка следваща команда.

След като сме се уверили че контролера за движение работи можем да включим робота в един от режимите на работа. Избираме квадратно движение и наблюдаваме как робота се движи във формата на квадрат.

***4. Анализ на получените експериментални резултати***

В дипломната работа е поставена задача за разработване на софтуер за управление на мобилния робот iRobot Create чрез комуникация и обмен на данни между робота и платформата NodeMcu. Експерименталната част включва тестване на командите от потребителския интерфейс на приложението.

Направените опити показват че приложението успешно изпраща команди към робота. Потребителския интерфейс е лесен за разбиране и работа. Получихме и забавяне в някои от заявките. От направените опити нямаше грешки в заявките. Първоначалното зареждане на страницата е най-бавната част от целия процес. Отнема около 1.2 секунди.

Елементите за задаване на скорост на робота правилно показват предела на стойностите за линейна и ъглова скорост които могат да се изпратят към iRobot. Вижда се че колкото линейната скорост е по-голяма толкова по-ниски стойности за ъглова скорост са възможни и обратната зависимост също важи.

Командата за движение в квадрат коректно се изпълнява от iRobot. Тректорията на джижение не е идеална фигура но е приемлива. Грешките идват от несъвършената одометрия на робота.

Командите за управление на диодите на iRobot работят по очаквания начин.

Добавихме и сензор за разстояние към робота който е насочен напред. В програмата на сървъра е зададена стойност от 25см. Ако сензорът отчете по-малка стойност от нея се изпраща команда за спиране към iRobot. Първоначално измерванията се отчитаха на всяка секунда и робота спираше по-близо от зададената стойност до предпядствието, но така или иначе ударът биваше предотвратен. След като увеличихме броя на измерванията iRobot реагира п-добре на предпятсвията и спира на безопасно разстояние. Следователно ако искаме робота да спира максимално близо до предпядствието трябва да увеличим честотата на измерванията. Натоварването на процесора от тези увеличени измервания е минимално.

1. ***Заключение***

За осъществяването на задачата са написани две приложения за сървър и за уеб страница. Сървъра има задачата да комуникира с iRobot. Това става чрез изпращане и получаване на серийни команди с библиотеката Serial която е включена към средата за разработка на сървъра – Arduino. Тя сама по себе си не е достатъчна за работа с платформата NodeMcu затова и включихме 2 външни библиотеки.

Другата основна задача на сървъра е да изпраща отговори на заявките които са отправени към нея. За да изпращаме тези заявки създадохме второто приложение. Те се изпращат през TCP-IP стека по конкретно използваме HTTP протокола, който се оказа достатъчно добър за нашите цели. Като алтернатива може да се използва MQTT протокола с който бихме постигнали по-бърза комуникация или новата технология Websockets която има предимството да изпраща заявки без те да бъдат изрично заявени от клиента.

В уеб приложението потребителя може да използва интерактивен и удобен за използване интерфейс за изпращане на команди. За стилизиране на външния вид на страницата е използвана библиотеката Bootsrap 4 която има доста готови за използване уеб елементи, а за интерактивната част Vue.js – джаваскрипт библиотека която премахва неудобството да се презарежда страницата след всяка отправена заявка и придава на уеб приложението усещане за настолно приложение.

Проведените експерименти потвърдиха коректната комуникация между робота и NodeMcu. Успешното решение на зададената задача показва полезността на платформата NodeMcu за управление на робота.iRobot Create е робот за хора, които искат да тестват възможностите си за програмиране. Напредналите могат да пишат софтуер за управление с помощта на различни методи, като се възползват от режима „поток на данни“ за по – голям контрол на работа. А експертите в програмирането могат да пишат програми за напълно автономно управление.

1. ***Използвана литература***
2. iRobot Create Owner’s Guide, 2006 IRobot Corporation.
3. iRobot Create Open Interface, 2006 IRobot Corporation.
4. Matlab Toolbox for IRobot Create, 2011.
5. Tod E. Kurt, Hacking Roomba, 2007.
6. <https://www.hackster.io/Aritro/getting-started-with-esp-nodemcu-using-arduinoide-aa7267>
7. <https://tttapa.github.io/ESP8266/Chap01%20-%20ESP8266.html>
8. ***Приложения***