Arduino

Arduino е електронна платформа с отворен код, базирана на лесен за използване хардуер и софтуер. Arduino дъските могат да четат входове - светлина върху сензор, пръст върху бутон или съобщение в Twitter - и го превръщат в изход - активиране на двигател, включване на светодиод, публикуване на нещо онлайн. Можете да кажете на вашия съвет какво да прави, като изпратите набор от инструкции до микроконтролера на дъската. За целта използвате езика за програмиране на Arduino (базиран на Wiring) и Arduino Software (IDE), базиран на Processing.

През годините Arduino е мозъкът на хиляди проекти, от ежедневни обекти до сложни научни инструменти. Световна общност от производители - студенти, хобисти, художници, програмисти и професионалисти - се е събрала около тази платформа с отворен код, техните приноси са добавили към невероятно количество достъпни знания, които могат да бъдат от голяма полза както за начинаещи, така и за експерти.

Ардуино е роден в Ivrea Interaction Design Institute

като лесен инструмент за бързо създаване на прототипи, насочен към студенти без опит в електрониката и програмирането. Веднага след като достигна до по-широка общност, бордът на Arduino започна да се променя, за да се адаптира към новите нужди и предизвикателства, диференцирайки своето предложение от прости 8-битови дъски до продукти за IoT приложения, носими, 3D печат и вградени среди. Всички дъски на Arduino са напълно отворен код, даващи възможност на потребителите да ги изграждат независимо и в крайна сметка да ги адаптират към техните специфични нужди. Софтуерът също е с отворен код и се разраства чрез приноса на потребителите по целия свят.

Благодарение на своето просто и достъпно потребителско изживяване, Arduino е използван в хиляди различни проекти и приложения. Софтуерът Arduino е лесен за използване за начинаещи, но достатъчно гъвкав за напреднали потребители. Работи се на Mac, Windows и Linux. Учителите и студентите го използват за изграждане на нискотарифни научни инструменти, за доказване на принципите на химията и физиката или за започване на програмиране и роботика. Дизайнерите и архитектите изграждат интерактивни прототипи, музикантите и художниците го използват за инсталации и да експериментират с нови музикални инструменти. Създателите, разбира се, го използват за изграждането на много от проектите, изложени на Faire Maker. Arduino е ключов инструмент за научаване на нови неща. Всеки - деца, любители, артисти, програмисти - може да започне да се занимава само следвайки стъпка по стъпка инструкциите на комплект или да споделя идеи онлайн с други членове на общността на Arduino.

Има много други микроконтролери и микроконтролерни платформи, достъпни за физически изчисления. Parallax Basic Stamp, BX-24 на Netmedia, Phidgets, MIT's Handyboard и много други предлагат подобна функционалност. Всички тези инструменти вземат разхвърляните подробности за програмирането на микроконтролери и ги увиват в лесен за използване пакет. Arduino също опростява процеса на работа с микроконтролери, но предлага известно предимство за учители, студенти и заинтересовани любители пред други системи:

* Евтин - Arduino платките са сравнително евтини в сравнение с други платформи за микроконтролери. Най-малко скъпата версия на модула Arduino може да бъде сглобена на ръка и дори предварително сглобените Arduino модули струват по-малко от $ 50
* Крос-платформа - Софтуерът Arduino (IDE) работи на операционни системи Windows, Macintosh OSX и Linux. Повечето микроконтролери са ограничени до Windows.
* Проста, ясна програмна среда - Софтуерът Arduino (IDE) е лесен за използване за начинаещи, но достатъчно гъвкав и за напреднали потребители, за да се възползва от тях. За преподавателите удобно се основава на програмната среда за обработка, така че студентите, които се учат да програмират в тази среда, ще бъдат запознати с начина, по който работи Arduino IDE.
* С отворен код и разширим софтуер - Софтуерът Arduino се публикува като инструменти с отворен код, достъпни за разширение от опитни програмисти. Езикът може да бъде разширен чрез C++ библиотеки и хората, които искат да разберат техническите подробности, могат да направят скока от Arduino към езика за програмиране на AVR C, на който е базиран. По същия начин можете да добавите AVR-C код директно във вашите програми на Arduino, ако искате.
* Отворен код и разтегателен хардуер - Плановете на платките на Arduino са публикувани под лиценз Creative Commons, така че опитни дизайнери могат да направят своя собствена версия на модула, като го разширят и подобрят. Дори сравнително неопитни потребители могат да създадат версията на модула, за да разберат как работи и да спестят пари.

В проекта исползван Arduino Uno R3

A circuit board

Description automatically generated

Arduino Uno е микроконтролерна платка, базирана на ATmega328P. Той има 14 цифрови пина за вход / изход (от които 6 могат да се използват като PWM изходи), 6 аналогови входа, 16 MHz керамичен резонатор (CSTCE16M0V53-R0), USB връзка, захранващ жак, ICSP заглавка и бутон за нулиране.

Информационен лист

|  |  |
| --- | --- |
| Microcontroller | [ATmega328P](http://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061A.pdf) |
| Operating Voltage | 5V |
| Input Voltage (recommended) | 7-12V |
| Input Voltage (limit) | 6-20V |
| Digital I/O Pins | 14 (of which 6 provide PWM output) |
| PWM Digital I/O Pins | 6 |
| Analog Input Pins | 6 |
| DC Current per I/O Pin | 20 mA |
| DC Current for 3.3V Pin | 50 mA |
| Flash Memory | 32 KB (ATmega328P) of which 0.5 KB used by bootloader |
| SRAM | 2 KB (ATmega328P) |
| EEPROM | 1 KB (ATmega328P) |
| Clock Speed | 16 MHz |
| LED\_BUILTIN | 13 |
| Length | 68.6 mm |
| Width | 53.4 mm |
| Weight | 25 g |

FireBase realtime database

React Native

React Native е рамка за мобилни приложения с отворен код, създадена от Facebook. Използва се за разработване на приложения за Android, iOS, Web и UWP, като дава възможност на разработчиците да използват React заедно с възможностите на родната платформа.

React Native позволява на разработчиците да създават кросплатформени приложения, които изглеждат и се чувстват изцяло Native, тъй като използва JavaScript компоненти, които са изградени както на iOS, така и на Android компоненти. Това спестява много време и пари, тъй като разработчиците не трябва да създават едно и също приложение за множество платформи. Освен това е по-лесно да се поддържа приложение, създадено с React Native, тъй като има само една кодова база. React Native използва основни блокове за Android и iOS, за да компилира Native приложения и за двете платформи в JavaScript. Това прави работата с кодовата база по-лесна, а добавянето на нови функции също е опростено, дори докато приложението работи.

A picture containing monitor

Description automatically generated

Нека да разгледаме работния процес на React Native, както следва:

Първоначално UI нишката започва да зарежда JS пакети и изпълнения в приложението. Тези две нишки никога не комуникират директно и блокират взаимно пътя си в резултат на синхронизация.

UI нишката и JS нишката никога не споделят и не работят върху едни и същи данни. Те предпочитат да обменят сериализирани съобщения.

Докато започва визуализацията, Reconciler започва "диференциация", когато генерира нов V-DOM и модифицираното оформление се изпраща към shadow нишка, която генерира shadow nodes.

След изчисляване на оформлението, той изпраща параметъра към нишката на потребителския интерфейс, защото единственият основен поток може да изобрази нещо на екрана.

Потребителският интерфейс ще се представи, когато нишката на сенките ще изпрати генерирано оформление към нишката на потребителския интерфейс.

Има 4 вида нишки в реагиращите местни приложения:

В реагиращото местно приложение нишките играят основна роля, като позволяват отделни задачи да се изпълняват заедно, докато една нишка изпълнява една задача, друга нишка може да изпълнява някаква друга отделна задача по паралелен начин.

* Основна нишка: Известена е също като UI нишка, която се използва за естествено изобразяване на iOS или Android UI. Той се създава автоматично в началото на приложенията, тъй като основната нишка ще породи всички долни нишки. UI нишката е и последната нишка, която завършва изпълнението, като изпълнява различни задачи за затваряне.
* JavaScript нишка: Той ще стартира основния логически код на JavaScript, включително сензорни събития, API обаждания и много други. Всички актуализации на собствените изгледи се пакетират и в края на всеки цикъл на събитието в нишката на JavaScript ще бъдат изпратени директно към основната страна. За по-добра ефективност JavaScript нишката изпраща пакетните актуализации в потребителския интерфейс, преди следващия кадър да представи крайния срок. Има и едно изключение от JS нишката, че родният изглед, който работи в нишката на потребителския интерфейс, не е блокиран поради бавна js нишка. напр .: ScrollView работи в UI нишка.
* Native Modules нишка: Native модулите обикновено се разпространяват като npm пакети, с изключение на типичните JavaScript файлове и се използват за достъп до специфичния за платформата API.
* Render нишка: Това е специална нишка, която изпраща команди към графичния процесор и използва за генериране на действителни OpenGL команди за изготвяне на потребителски интерфейс.

Нов проект React Native може да бъде лесно започнат как с помоща на Expo CLI, так и с React Native CLI.

Expo е рамка и платформа за универсални React приложения. Това е набор от инструменти и услуги, изградени около React Native и родните платформи, които ви помагат да развивате, изграждате, разгръщате и бързо да итератирате в iOS, Android и уеб приложения от същата JavaScript / TypeScript кодова база.

Тази дипломна работа исползва React Native CLI.

Трябва ни предварително да имаме Node.js, Watchman, React Native CLI, Xcode, JDK и Android Studio (проектът разработен на macOS за iOS и Android).

Инсталиране.

Homebrew менеджер на пакетите за macOS и Linux.

/bin/bash -c "$(curl -fsSL <https://raw.githubusercontent.com/Homebrew/install/master/install.sh>)"

Node.js е JavaScript среда, изградена на двигателя на JavaScript Chrome V8.

brew install node

Watchman е инструмент на Facebook за гледане на промени във файловата система.

brew install watchman

Xcode – интегрирана среда за разработка софтуер за macOS, iOS, padOS, AppleTV.

Най-лесният начин за инсталиране на Xcode е чрез Mac App Store. Инсталирането на Xcode ще инсталира и iOS Simulator и всички необходими инструменти за изграждане на приложението ви за iOS.

CocoaPods е изграден с Ruby и той ще бъде инсталируем с стандартния Ruby, наличен на macOS. CocoaPods е мениджър на зависимости за проекти на Swift и Objective-C Cocoa.

A screenshot of a computer

Description automatically generated

React Native има вграден интерфейс за команден ред. С npx react-native <command> текущата стабилна версия на CLI ще бъде изтеглена и изпълнена по време на изпълнение на командата.

npx react-native init SmartFarmMobile

Java Development Kit

npx react-native init SmartFarmMobile

Android Studio

<https://developer.android.com/studio/index.html>

A screenshot of a computer screen

Description automatically generated

HTML

HTML е език за маркиране на хипертекст. Той позволява на потребителя да създава и структурира секции, параграфи, заглавия, връзки и блокове за уеб страници и приложения. HTML не е език за програмиране, което означава, че той няма способността да създава динамични функции. Вместо това ви позволява да организирате и форматирате документите си по подобен начин на Microsoft Word. Когато работим с HTML, ние използваме прости кодови структури (тагове и атрибути), за да маркираме страницата на уебсайта. Например, можем да създадем абзац, като поставим прикачения текст в оригиналния <p> таг и затварящия </p> таг.

HTML е изобретен от Тим ​​Бърнърс-Ли, физик в изследователския институт на CERN в Швейцария. Той дойде с идеята за интернет хипертекстова система. Хипертекст означава текст, който съдържа връзки към други текстове, до които зрителите могат да получат незабавен достъп. Той публикува първата версия на HTML през 1991 г., състояща се от 18 HTML тагове. Оттогава всяка нова версия на HTML езика се появява с маркиране на нови тагове и атрибути (модификатори на тагове). Според справочника за HTML елементи от мрежата за разработчици на Mozilla, в момента има 140 HTML тагове, въпреки че някои от тях вече са оттеглени (не се поддържат в съвременните браузъри). Поради бързия си растеж на популярност, HTML вече се счита за официален уеб стандарт. Спецификациите на HTML се поддържат и разработват от World Wide Web Consortium (W3C). Можете да проверите най-новото състояние на езика по всяко време на уебсайта на W3C.

Най-голямата актуализация на езика беше въвеждането на HTML5 през 2014 г. Няколко нови семантични маркера са добавени към маркирането, които показват значението на тяхното собствено съдържание, като <article>, <header> и <footer>.

CSS

Как работи HTML?

HTML документите са файлове, които завършват с разширение .html или .htm. Можете да го видите с всеки уеб браузър (като Google Chrome, Safari или Mozilla Firefox). Браузърът чете HTML файла и показва неговото съдържание, така че потребителите в Интернет да могат да го преглеждат.

Обикновено средният уебсайт включва няколко различни HTML страници. Например: началните страници, обикновените страници и страниците за контакти ще имат отделни HTML документи.

Всяка HTML страница е съставена от набор от тагове (също наречени елементи), които можете да наречете градивните елементи на уеб страниците. Те създават йерархия, която структурира съдържанието в секции, параграфи, заглавия и други блокове от съдържание.

Повечето HTML елементи имат отворени и затворени, които използват синтаксиса <tag> </tag>.

По-долу можете да видите пример на кода, с който можете да структурирате HTML елементи:

<div>

<h1> Основна позиция </h1>

<h2> Подпозиция </h2>

<p> Точка 1 </p>

<img src = "/" alt = "Изображение">

<p> Елемент втори с хипервръзка <a href="https://example.com"> </a> </p>

</ div>

Най-важният елемент е просто разделение (<div> </div>), което можете да използвате, за да маркирате големи секции от съдържание. Той съдържа заглавие (<h1> </h1>), подзаглавие (<h2> </h2>), два абзаца (<p> </p>) и изображение (<img>).

Вторият параграф съдържа връзка (<a> </a>) с атрибут href, който съдържа целевия URL адрес.

Името на изображението също има два атрибута: src за пътя на изображението и alt за описанието на изображението.

Javascript

Преглед на най-използваните HTML маркери

HTML маркерите са от два основни типа: блокове и вградени маркери.

Елементите на ниво блок заемат цялото свободно пространство и винаги започват нов ред в документа. Заглавията и параграфите са чудесни примери за блокови тагове.

Вградените елементи заемат толкова място, колкото е необходимо, и не стартират нов ред на страницата. Те обикновено се използват за форматиране на вътрешното съдържание на елементи на ниво блок. Връзки и подчертани линии са добри примери за вградени маркери.

Тагове на ниво блок

Три тагове на ниво блок, които всеки HTML документ трябва да съдържа: <html>, <head> и <body>.

<html> </html> тагът е елемент от най-високо ниво, който обхваща всяка HTML страница.

Тегът <head> </head> съдържа мета информация като заглавие на страницата и кодиране.

И накрая, тагът <body> </body> съдържа цялото съдържание, показано на страницата.

<html>

<head>

<! - МЕТА ИНФОРМАЦИЯ ->

</ head>

<body>

<! - СЪДЪРЖАНИЕ НА СТРАНИЦА ->

</ body>

</html>

Заглавията са 6 нива в HTML. Те варират от <h1> </h1> до <h6> </h6>, където h1 е заглавието с най-високо ниво и h6 е най-ниското.

Параграфите са приложени <p> </p>, докато коментарите за блокове използват маркера <blockquote> </blockquote>.

Разделите са по-големи секции от съдържание, които обикновено съдържат множество абзаци, изображения, понякога блокират записи и други по-малки елементи. Можем да ги маркираме с помощта на <div> </div> тага. Елемент div може да съдържа друг div tag вътре в него.

Можете също да използвате маркерите <ol> </ol> за подредени списъци и <ul> </ul> за списъци, които не са подредени. Отделните елементи от списъка трябва да бъдат затворени в маркера <li> </li>. Например, това изглежда основен неупореден списък в HTML:

<ul>

<li> Списък с елемент 1 </li>

<li> Списък на списък 2 </li>

<li> Списък на списък 3 </li>

</ul>

Вградени (inline) маркери

Много вградени маркери се използват за форматиране на текст. Например, етикетът <strong> </strong> превръща маркирания елемент с удебелен шрифт, докато маркерите <em> </em> го правят курсив.

Хипервръзките също са вградени елементи, които изискват <a> </a> тагове и href атрибути, за да посочат дестинацията на връзката:

<a href="https://example.com/"> Кликнете тук! </a>

Изображенията също са вградени елементи. Можете да добавите такъв с <img> без затварящ маркер. Но също така ще трябва да използвате атрибута src, за да определите пътя на изображението, например:

<img src = "/ images / example.jpg" alt = "Примерно изображение">

CSS

Javascript

React

Литература

<https://www.arduino.cc/>

<https://firebase.google.com/>

<https://reactnative.dev/>