Ismerkedés a JavaScript Elektronikával

Arduino fejlesztés JavaScript és Node.js segítségével

Marschalkó Máté

Ismerkedés a JavaScript Elektronikával,
írta **Marschalkó Máté**

Publikálta a **Web on Devices**,
London, Egyesült Királyság, 2015.

Szerkesztés és lektorálás: **Nicholas Headlong**.
Programtechnikai lektorálás: **James Miller**.
Villamosmérnöki lektorálás: **Szőke Gábor.**

Írta

Marschalkó Máté

Szenior Kreatív Technológus
és Web Programozó

Máté jelengleg Szenior Kreatív Technológus és Web Fejlesztő munkakörben dolgozik londoni kreatív ügynökségeknek. Az elmúlt 10 évet web fejlesztéssel töltötte, ami immár a hobbijává is vált. Szereti felfedezni a legújabb technikai vívmányokat, eszközöket, azokat JavaScript-tel meghekkelni. A barkácsasztalán mindig találni valami játékszert, amit Arduino-val, Raspberry PI-al vagy más hasonló fejlesztő eszközzel épített.

Programtechnikailag lektorálta

James Miller

Vezető Programozó
és Kreatív Technológus

James 10 év tapasztalattal rendelkező Vezető Programozó és Kreatív Technológus. Gyakori vendég írója a Net Magazin és a Smashing Magazin újságoknak. Újabban a Hibrid Web Applikációk kötik le az idejét, mellette saját hardvereit építgeti és szurkol imádott focicsapatának, a Luton Town-nak.

Web on Devices

Elektromos eszközök hekkelése JavaScript és más Webes Technológiák alkalmazásával

www.webondevices.com

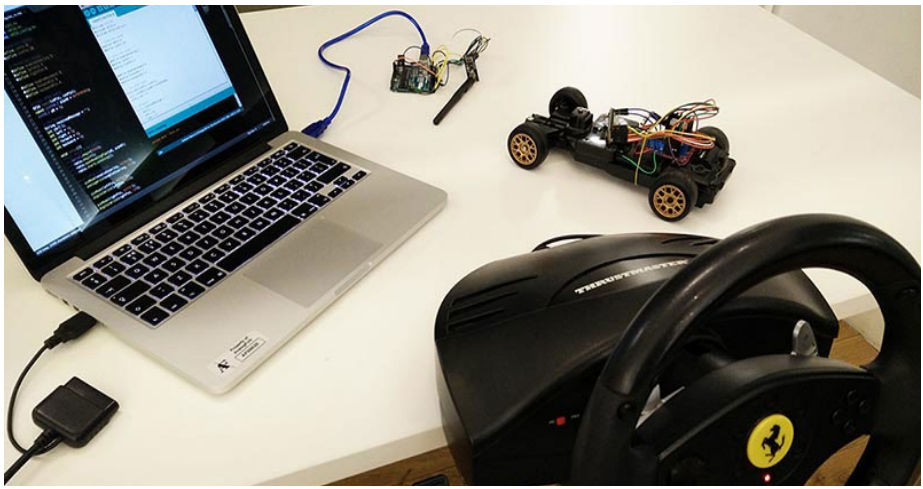
TARTALOM

BEVEZETÉS

Az elmúlt évek során a JavaScript programozási nyelv rengeteget fejlődött és változott, immár nem csupán DOM manipulációra használjuk, jelenléte és szerepe pedig tullépett a böngésző ablak keretein. Manapság azt látjuk, hogy a JavaScript megjelenik, mint operációs rendszer (Chrome OS, Firefox OS), a szerver oldalon (Node.js), nem is beszélve az apró mikrokontroller csippekről. Lassan teljesen általánossá válik, hogy fejlesztők hardver eszközöket, elektromos játékokat építenek JavaScript nyelven olyan platforomokon, mint például az Arduino. Ezek a platformok és fejlesztő panelek lehetővé tették, hogy olyan emberek is belefogjanak az elektronikai munkába, akiknek nincs mérnöki előképzettségük.

Az ‘Internet of Things’, vagyis az Internettel kommunikálú egyszerű használati tárgyak hálózata, új lehetőségeket kínál a marketing számára, hogy egyes termékeket kézzelfogható reklámokon

keresztül népszerűsítsen, illetve a fejlesztőknek, hogy tudásukat eziránya szélesítve magasabb jövedelemre tegyenek szert hardver prototipizálással. A jövőbetekintő reklámügynökségek már ma olyan webfejlesztőket keresnek, akiknek van tapasztalatuk Kreatív Technológus munkakörben. Ezek a fejlesztők a weblap készítés mellett jártasak interaktív kioszkok tervezésében, összeszerelésében, virtuális valóság élmények programozására és más hardver alapú prototípusok elkészítésében. Nem csak, hogy ez a tudás magasabb fizetést jelent sok esetben sima webfejlesztő állásokkal szemben, de a legújabb tech eszközökkel kísérletezgethetnek, játszhatnak a napi munkájuk során!



Játék egy Arduino távirányítós autóval

Ez a rövid köny segít majd Neked abban, hogy meg tedd az első lépést a Kreatív Technológus világ felé, és az elektromos eszközök építésébe az Arduino UNO fejlesztő panel és a JavaScript segítségével. A bevezető után átnézzük az elektromosság alapjait, az UNO-t és az alaktrészeit, LED égő kapcsolgatását és egy elektromos hőmérő beolvasását. Mindeközben megtunljuk a JavaScript kód futtatását a böngészőn kívül szerver környezetben a Node.js használatával, egyszerű áramkörök építését, majd megnézzük a Johnny-Five JavaScript könyvtárat, amit az Arduino UNO és komponenseivel való kommunikációra foguk használni.

Arduino Macintoshoz csatlakoztatva

A BÖNGÉSZŐ KORLÁTOZÁSAI

Asztali számítógép és mobil alkalmazásokhoz hasonlítva a JavaScript a böngésző környezetben meglehetősen limitált lehetőségekkel bír. A hozzáférés a kommunikációs portokhoz, hardver komponensekhez és a fájlrendszerhez mind korlátozottak. Ezek a szintű biztonsági korlátozások szükségesek az Interneten, ugyanis nem minden weblap bízható meg teljesen, így meg kell győződnünk, hogy a számítógépünket nem érhetik nemkívánatos támadások.

Egy módja annak, hogy ezeket a korlátozásokat áthidaljuk az, ha az applikációnkat a szerveren futtatjuk, ahonnan később minden adatot a böngésző oldalra küldhetünk. A szerver megbízhat a programkódban, amit futtat, így engedélyezheti a hardver eszközök elérését. Nagyon sok módja van annak, hogy szerver oldali programot írjunk, például futtathatunk PHP kódot egy Apache Linux szerveren amit egy adatbázis szolgál ki (pl. MySQL, Postgres), ezt néha LAMP-nek is hívják.

E helyett a felállás helyett JavaScriptet is használhatunk, a szerver pedig lehet a saját számítógépünk.

A szerverrel való kommunikáció hagyományosan HTTP kérések küldését és válaszokra várást jelentett. 2005 környékén az AJAX megjelenésével ez a folyamat kicsit dinamikusabbá vált. Az AJAX még mindig csupán szerver kérések küldését jelenti új adat reményében, de ezúttal anélkül történik mindez, hogy a weboldalt újra kellene töltenünk. Ez széleskörűbb kezelhetőséget jelent, hisz a kapcsolat és a válasz kezelése már JavaScript felhasználásával zajlik.

A WebSocket-ok használata azonban egy sokkal hatékonyabb megközelítést nyújt számunkra a szerverről érkező adatok kezelésére. A protokoll lehetőséget biztosít, hogy megnyissink egy interaktív, két irányú kommunikációs csatornát a felhasználó böngészője és a szerver között. Ezzel az API-val üzeneteket tudunk küldeni a szervernek és esemény alapú válaszokat kezelhetünk anélkül, hogy a választ folyamatos időközönként le kellene kérnünk.

A következő lépés az lesz, hogy megírjuk a szerver programot, ami az USB porton keresztül képes kommunikálni az Arduinónkkal, ami pedig tökéletes feladat a Node.js számára.

JAVASCRIPT A SZERVER OLDALON

A Node.js a Google V8 nevű JavaScript motorjára épül, ami képes futni a böngészőn kívül is, akár a szerveren is.

Node.js szerverről vezérelt AR Drone

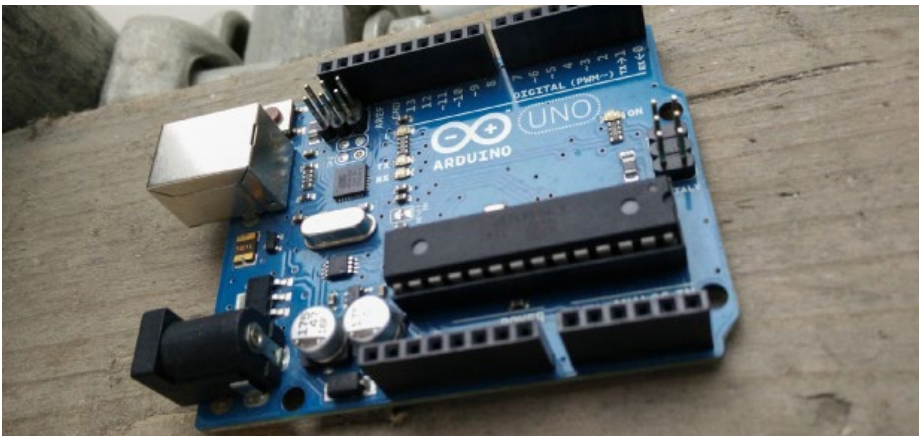
A javascript és a Node.js használata elektromos projektek megvalósítására nem egy teljesen új kezdeményezés. A programozó közösség már használja ezt a módszert olyan projektekhez, mint pl. a nodebots.io, Firmata, Cylon vagy a Johnny-Five. A Node.js

környezet még ettől is nagyobb népszerűségnek örvend a több, mint százezer kiegészítőnek köszönhetően, ami a node kiegészítő manageren (node package manager, www.npmjs.org) keresztül elérhető. Sebesség és teljesítmény szempontjából a JavaScript drámaian sokat fejlődött az elmúlt évtizedben, így ez a kérdés sem probléma többé.

Az esemény alapokon nyugvó rendszer, az aszinkron tulajdonságok szintén tökéletesek az elektromos szenzorokkal való munkára. Lehetőség van ezen felül olyan applikációk írására, amik NFC, RFID vagy Bluetooth kapcsolaton keresztül kommunikálnak, ami jelenleg mind elképzelhetetlen a böngészőben. A Node.js használata új lehetőségeket nyit, hogy olyan eszközökhöz fejlesszünk programot, mint az Xbox Kinect, a Leap Motion, Midi kontrollerek, drónok vagy okos otthoni kiegészítők, pl. A Nest termosztát vagy a Philips HUE okos villanykörte.

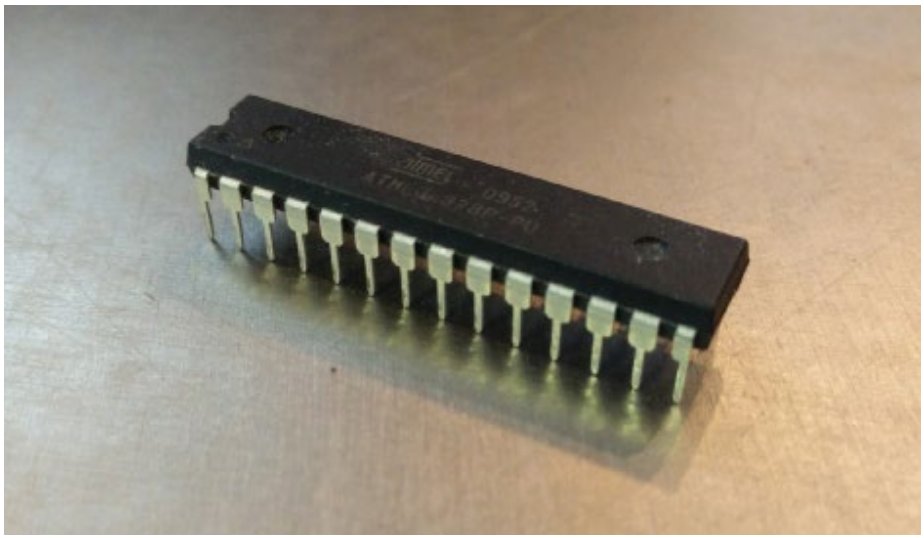
FEJLESZTŐ PANELEK

A fejlesztő panelek forradalma 2005 körül kezdődött, amikor az első Arduino piacra került. Az Arduino projektet néhány Olasz egyetemi tanár kezdeményezte azzal a céllal, hogy diákjaiknak megkönnyítsék az elektronikai eszközökkel való munkát és a mikrokontroller chipek programozását. Az Arduino a mai napig messze a legnépszerűbb platform.



Arduino UNO fejlesztő panel

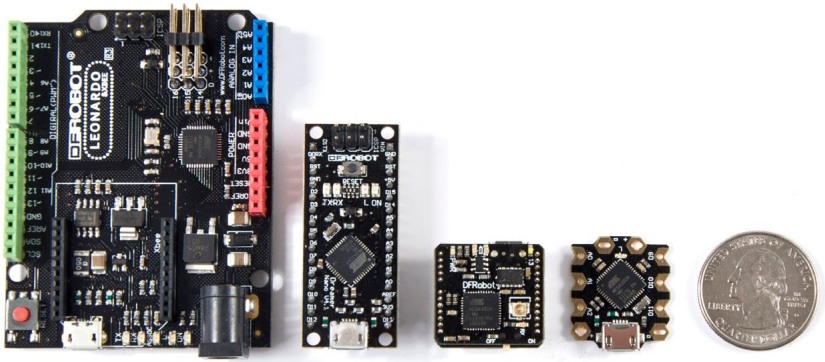
A mikrokontroller chip a fejlesztő panel agya, ami mindent irányít és vezérel. Ez a chip egy rendkívül olcsó, alacsony fogyasztású és teljesítményű processzor, ami egyszerre egy bizonyos program futtatását végzi. A mikrokontroller mindössze annyit feladatot lát el, hogy egyszerű elektromos kimeneti és bemeneti jeleket kezeljen, ezek függvényében néhány számítást végezzen. A chip képes a csatlakoztatott eszközök elektromos áram ellátásának ki- és bekapcsolására, szenzorokból érkező elektromos jelek feszültségének mérésére, valamint más komponensekkel való kommunikációra.



Az Arduino UNO mikrokontroller chipje

Ezek a chipek önállóan is képesek működni, azonban a fejlesztő panel megkönnyíti a prototípus építés folyamatát. Segítenek a chip energiaellátásban és a vele való kommunikációban. Ezenfelül kapunk egy USB csatlakozót is, amivel programkódot tudunk feltölteni az UNO-ra, egy hengeres csatlakozót 9V-os elemről való üzemeltetéshez, amik mind nagyon hasznos funkciók.

Jelenleg hatalmas a választék a fejlesztő panelek piacán. A legtöbbjük Arduino kompatibilis, de akkor mégis miben különböznek?

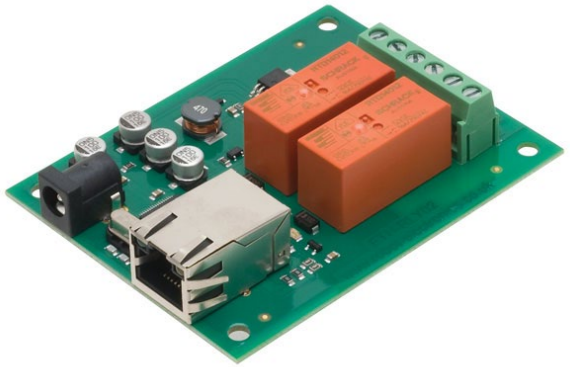


Különböző méretű, Arduino kompatibilis panelek a DFRobot-tól

Először is nagyban különbözhet a méretük. Az egyik legkisebb elérhető panel a DFRobot Beetle és a Tinyduino, amik tökéletesek lehetnek drónok vagy

viselhető eszközök készítéséhez, ahol a méret és a súly nagyon fontosak. A kisebb mérettel kevesebb funkció és kevesebb komponens is jár. Mindez kisebb energiafogyasztást is jelent, így elemről is nagyon könnyen üzemeltethetők ezek a projektek.

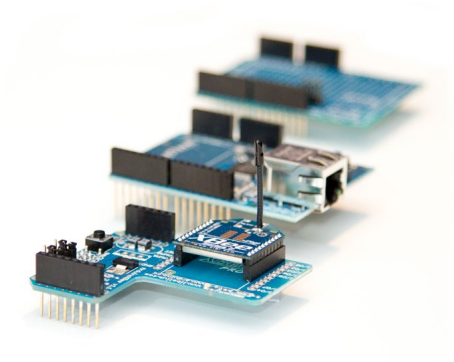
Ezen kívül vannak panelek olyan funkciókkal, mint a Wi-Fi, Ethernet vagy Bluetooth kapcsolat, mások képesek egér, billentyűzet vagy más USB eszköz szimulálására, megint mások pedig otthon-automatizálásra vagy biztonsági rendszerek kiépítésre specializálódtak.



Fejelsztő panel otthon-automatizáláshoz kettő relével magasfeszültségű áramforrások kapcsolásához

AZ ARDUINO UNO

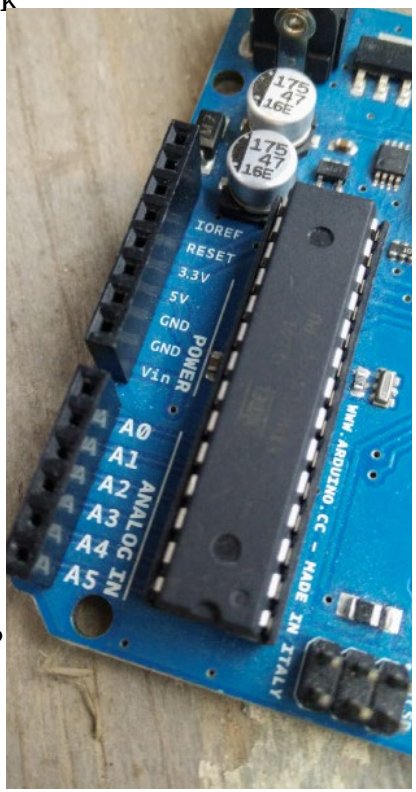
A különböző fejlesztő panelek mind hasznosak és olcsóak, de az Arduino UNO még mindig az elsőszámú panelek az elektromos fejlesztéssel való ismerkedéshez. Ennek az egyik oka valószínűleg az, hogy a csatlakozó kiosztása rendezett és tágas, ami könnyebbséget jelent a kezdőknek. A fent már említett extra funkciók, mint a vezetéknélküli kapcsolat mind lehetségesek az UNO-val, azonban ezekhez külső kiegészítők, vagy úgynevezett shieldek szükségesek.



Kiegészítő shieldek az Arduino UNO-hoz

Az Arduino felépítése

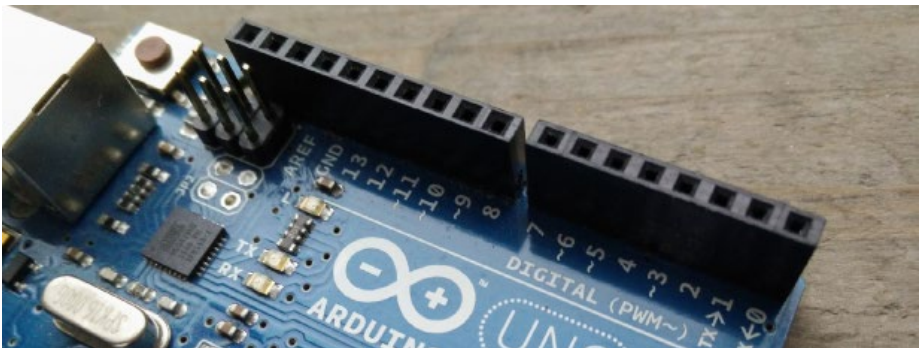
Az Arduino UNO legfontosabb alkatrésze az ATmega328 mikrokonrtoller chip. Ez a chip felelős a ki- és bemeneti csatlakozók (GPIO) vezérléséért, amik a panel felső és alsó oldala mentén helyezkednek el. Röviden, a bemeneti csatlakozók (ANALOG IN) képesek elektromos áramot mérni, amelyek különböző szenzorok (fény, hang, hőmérséklet) felől érkeznek. A kimeneti csatlakozók (DIGITAL) ezzel szemben képesek az áram kapcsolására lámpák, motorok és más elektromos komponensek számára. Mind a kimeneti, mind a bemeneti csatlakozók elérhetőek és irányíthatóak a programkódból, amit írsz és feltöltesz a fejlesztő panelre.



Áramforrás és kimeneti csatlakozók az UNO-n a hosszú mikrokontroller chip mellett

A csatlakozók ezen felül egy folyamatos 5V illetve 3,3V (5 volt és 3,3 volt) áramforrást rejtenek. Ezek lehetővé teszik, hogy az UNO-t úgy használjuk, mintha egy akkumulátor lenne, amit a komponenseik energiaellátásra használhatunk. Hagyományos elemeken két pólust találunk: pozitív és negatív. Az 5V és a 3,3V csatlakozók az Arduinón a pozitív pólusként múködnek, a GND (ground, föld) csatlakozó pedig a negatívként.

A csatlakozók mellett találhatjuk a nagy, B típusú USB aljzatot, amin keresztül a számítógéppel kumminkálhatunk és a programkódunkat feltölthetjük az Arduinóra. A bal alsó sarokban található hengeres csatlakozón keresztül csatlakoztathatunk egy 9 voltos elemet vagy más egyéb egyenáramú áramforrást 6 és 20 volt között (az ajánlott 7 és 12 volt), ez lehet bármilyen adapter a megfelelő méretű csatlakozóval és feszültséggel.

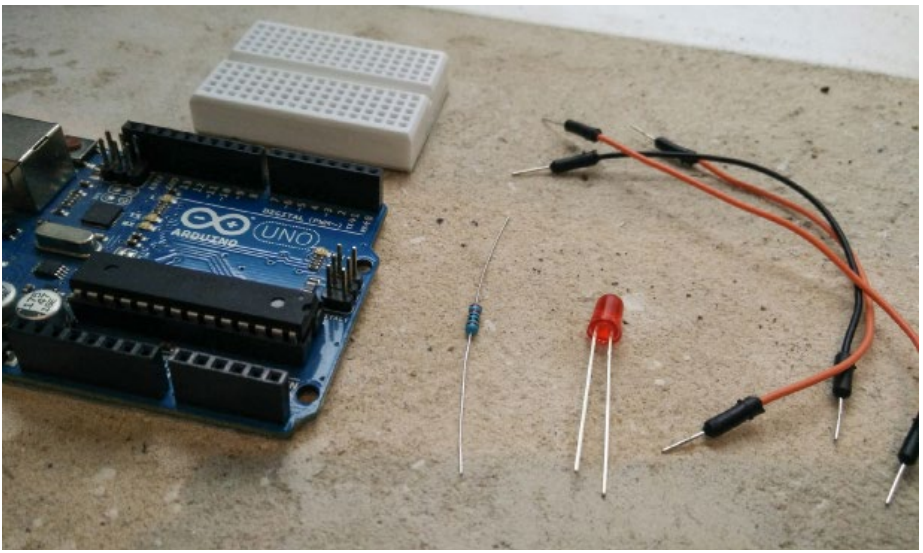


Digitális kimeneti csatlakozók

A következő példák során az Arduinonem lesz külső energiaforrásról üzemeltetve, helyette az USB kábellel a számítógéphez csatlakoztatjuk.

A kimeneti csatlakozók szintén képesek energiaforrásként üzemelni, csak úgy, mint az 5V-os. A különbség az, hogy a kimeneti csatlakozót a mikrokontroller chip ki be képes kapcsolgatni a rátöltött programkódon keresztül.

Ennek a funkciónak a demonstrálásához építsünk egy egyszerű áramkört, amit egy JavaScript programmal fogunk irányítani. A hardwer fejlesztés „Hello World" példája egy LED, vagyis világító dióda, kapcsolgatása

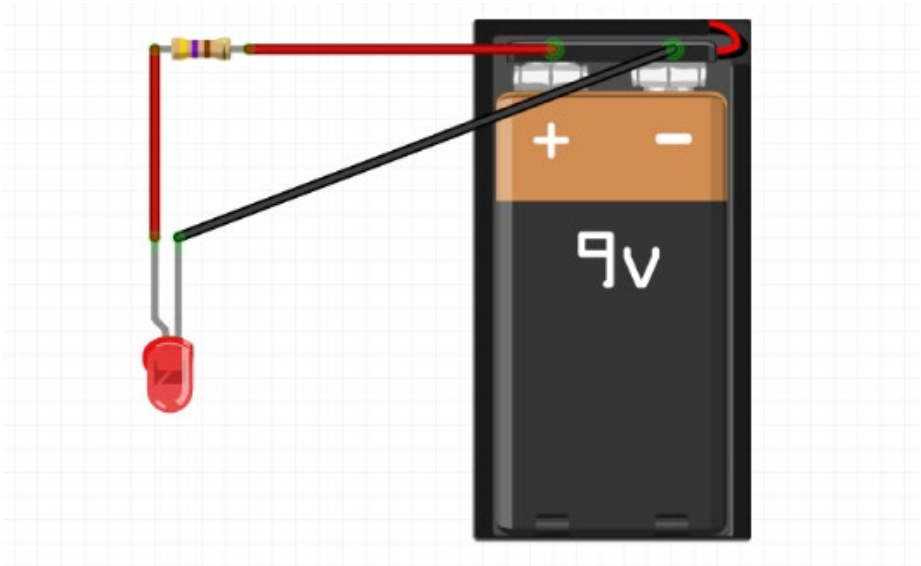


Alkatrészek a villogó LED projekthez

és villogtatása, így mi is ezzel fogjuk kezdeni. Ehhez a projekthez szükséged lesz egy Arduino UNO-ra, egy sima LED-re, egy 150 – 1k ohm közötti ellenállásra, egy forrasztásmentes próbanyákra és néhány próbakábelre.

Az ellenállásokat a rajtuk található színes csíkokról tudjuk megkülönböztetni. Az Interneten rengeteg segítséget[1], táblázatot találunk ezekhez a színkódokhoz.

A lehető legegyszerűbb áramkör amire gondolni tudunk az egy LED, amit egy elem működtet. Ez az áramkör az elem pozitív pólusát a LED pozitív lábával, a negatív pólust a negatív lábbal köti össze. Amint a

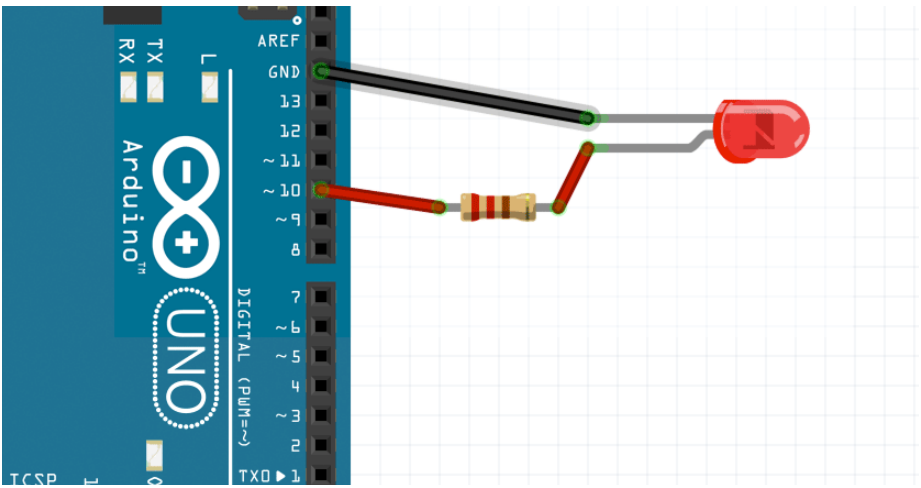


LED elemről működtetve (Fritzing.org)

[1] http://www.dannyg.com/examples/res2/resistor.htm

kapcsolat létrejött az elektromos áram el kezd folyni a LED-en keresztül, ami az rögtön fénnyel reagál.

Ennek az egyszerű áramkörnek a megépítése az Arduinón tulajdonképpen abból fog állni, hogy az elem pozitív pólusát az Arduino 5V csatlakozójával, a negatív pólusát pedig a GND csatlakozóval váltjuk fel. Következő lépésként a példánkhoz a 10-es kimeneti csatlakozót választottam véletlenszerűen.



LED az Arduino kimeneti csatlakozójához kötve (Fritzing.org)

Ezzel az áramkörünk kész is; ha a programkódunkkal a 10-es kimeneti csatlakozót felkapcsoljuk, a LED elkezd világítani.

Valószínűleg feltűnt, hogy egy kis ellenállás is

került az áramkörbe, és most azon tűnődsz, erre vajon mi szükség volt. Nos, van egy kis problémánk: a LED nem korlátozza az elektromos áramot az áramkörünkben, így e nélkül túl sokat próbálna belőle felhasználni. Ellenállás nélkül a LED-ünket túlhajtanánk, ami rövid időn belül károsodást szenvedne. Valószínűleg néhány perc működés után szétégne.

Ennek elkerülése érdekében korlátozzuk a LED-en áthaladó áramot az ellenállásal. Az Ohm törvény [1] felhasználásával (az ellenállás egyenlő a feszültség és az áram hányadosával, másképp R = V / I) ki tudjuk számolni az áramkörhöz pontosan szükséges ellenállás mértékét.

A miénkben az Arduino 5 voltot szolgáltat, de ebből nem mind fog az ellenállásra esni: a LED ebből 2 voltot fog felhasználni, ami számára a szükséges feszültség a működéshez. Ezen kívül azt is tudjuk, hogy a megfelelő mennyiségű áram a LED számára 20mA (milliamper) vagy 0,02 amper. Most számoljuk ki, hogy mekkora ellenálás szükséges, hogy 2 volt feszültség essen a LED-re 20mA esetén. Mindezt az Ohm törvénnyel kapjuk meg:

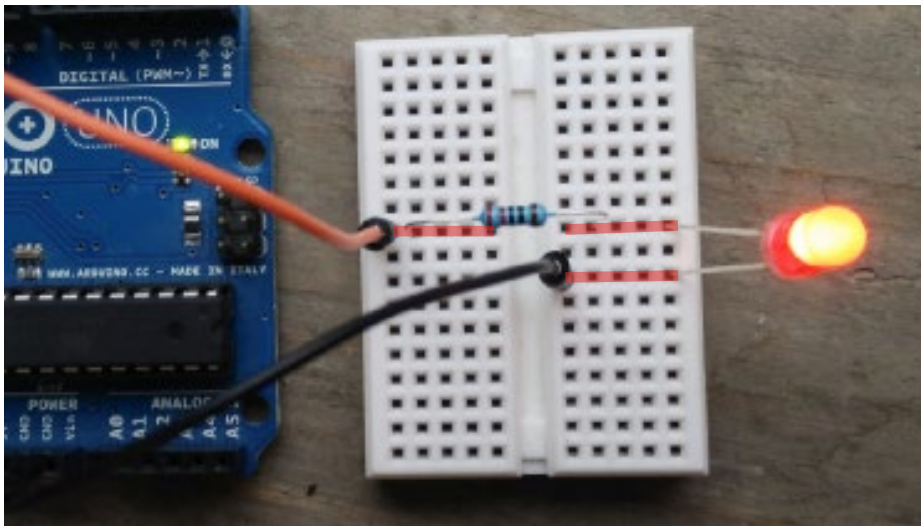
```
R = V / I
R = (5V - 2V) / 0.02A
R = 150Ω
```

[1] https://youtu.be/-mHLvtGjum4

Ez azt jelenti, hogy körülbelül 150 ohmra lesz szükségünk. 150 ohmnál kisebb ellenállás nem lesz jó, de egy nagyobb, például 220Ω, 500Ω vagy 1000Ω nem fog gondot okozni, egész egyszerűen jobban korlátozná az áramot, aminek hatására a LED kissé sötétebben világítana, cserébe viszont megnövelné annak élettartamát.

Forrasztásmentes próbanyák

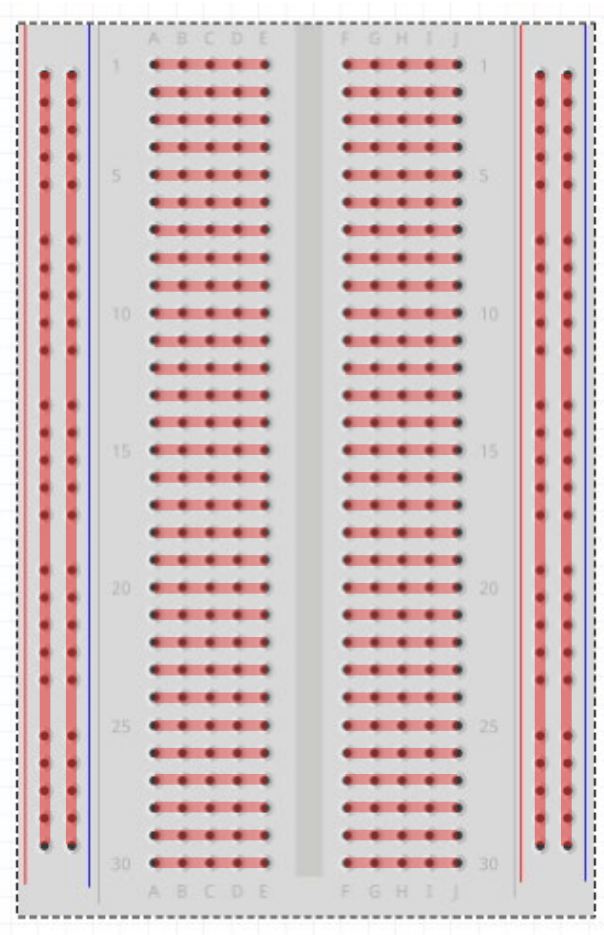
A próbanyákok megkönnyítik és felgyorsítják az áramkör prototípusok készítését. Hagyományos, tüskés, hím, próbakábelek, vagy egyszerűbb elektromos komponensek lábainak befogadására tervezték őket, mint pl. LED-ek, ellenállások, diódák vagy kondenzátorok. A próbanyák gyakorlatilag ezen komponensek összekapcsolásában segít nekünk. Amikor bedugod egy LED lábát a próbanyákon akárhova, azok a



LED áramkör kapcsolat a próbanyák belsejében

komponensek, amelyek ugyanabba csatlakozósorba kerülnek, mind összeköttetésbe kerülnek a LED-del.

Itt egy nagyobb méretű próbanyák, amelyen minden összeköttetésben álló sínt kiemeltünk:



Összeköttetésben álló csatlakozók egy nagyobb próbanyákon

ISMERKEDÉS A NODE.JS KÖRNYEZETTEL

A LED készen áll arra, hogy ki be kapcsoljuk a JavaScript programunkkal. Ehhez a JavaScript kódot a szerver oldalon kell futtatnunk, ugyanis csak így lehetséges az USB csatlakozóval való kommunikáció, ahova az Arduinónkant fogjuk dugni. Az első lépés ezen az úton a Node.js környezet telepítése a számítógépünkre.

A Node.js telepítése gyors és egyszerű a hivatalos telepítőcsomaggal. Ezeket a https://nodejs.org/download/ oldalról tudjuk letölteni.

Amint a telepítés kész, az új node parancs elérhetővé válik a Terminálban OSX-en, illetve a DOS Parancssorban Windows alatt. Egy egyszerű "Hello World" teszthez nyissunk meg egy új, helloworld.js nevű fájlt, majd adjuk hozzá az alábbi sor JavaScript kódot:

```
console.log("Hello World");
```

A parancssori ablakba beírt `node helloworld.js` parancs lefuttatja a JavaScript programunkat és kiírja a Hello World üzenetet (a program futtatásához a fájllal megegyező mappába kell navigálnunk a paraccsort).

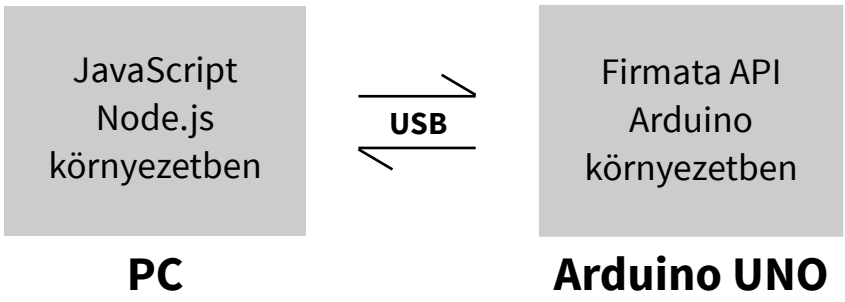
Node.js “Hello World”

Mindez önmagában nem tűnhet nagy lépésnek, de így, hogy a Node.js-t telepítettük, készen állunk arra, hogy saját hardver projektekbe kezdjünk a JavaScript nyelvvel. Ebből a környezetből már elérhető számunkra az USB port, valamint szenzorok és más komponensek a számítógépünkön.

Az Arduino – Node.js kapcsolat kiépítése

Ehhez az applikációhoz a Johnny-Five JavaScript könyvtárat[1] fogjuk használni, amit kifejezetten Node.js-hez terveztek. Ez a könyvtárat a Bocoup[2] vállalatnál dolgozó csapat készítette, hogy a hardver prototipizálást megkönnyítsék a web fejlesztők számára.

JavaScript kód futtatása egyenesen a fejelsztő panelek chipjein mindössze néhány változaton működik, és az Arduino sajnos nem egyike ezeknek. Az Arduinóra csak Arduino nyelven íródott programkódot tudunk feltölteni és futtatni és ezen nem könnyű változtatni. A Johnny-Five és még néhány Node könyvtár a Firmata



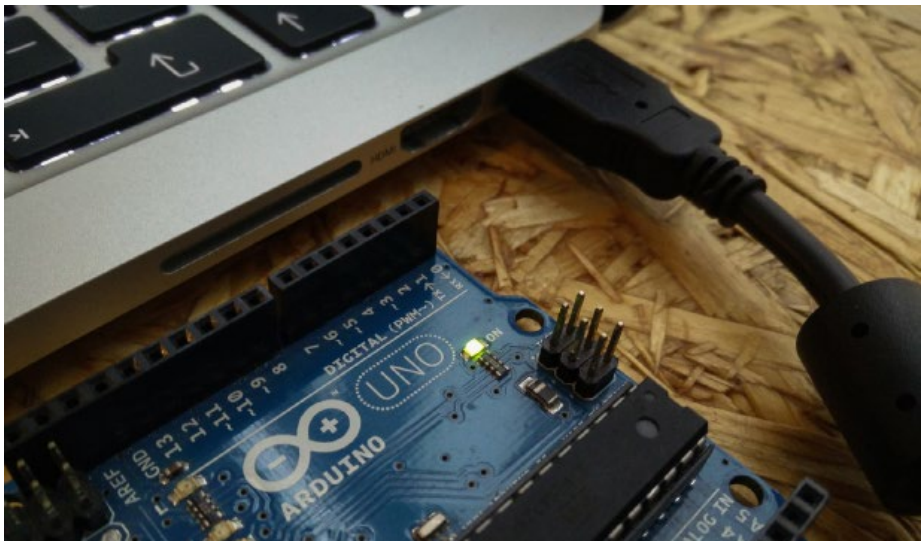
Firmata munkafolyamat

[1] https://github.com/rwaldron/johnny-five

[2] https://bocoup.com/

Arduinóra való telepítésével lép túl ezen a korláton. A Firmata tulajdonképpen egy API interfészt tesz elérhetővé az Arduinon, hogy azt külső eszközök ezen keresztül irányíthassák.

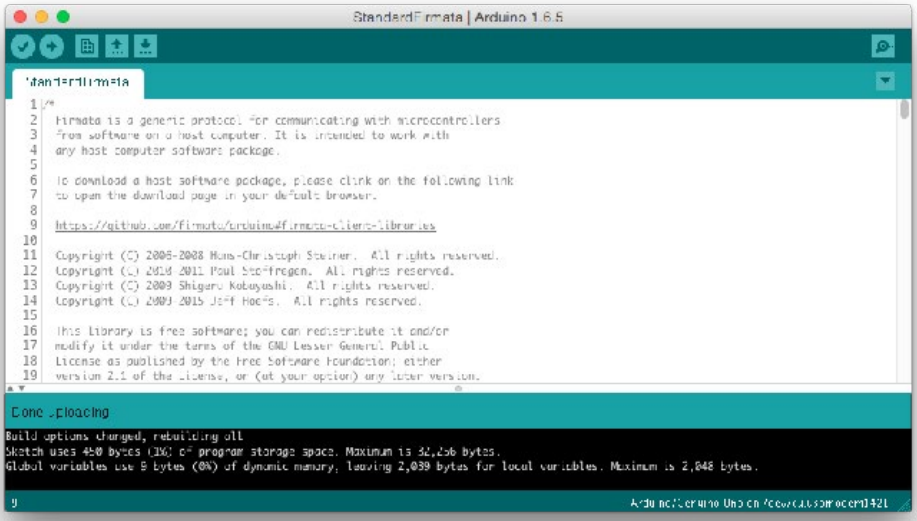
Ahhoz, hogy a LED-ünk végre villogjon először fel kell töltenünk a Firmata programot az Arduinóra, ahhoz pedig, hogy akármilyen kódot fel tudjunk tölteni, le kell töltenünk és telepítenünk a hivatalos Arduino IDE programot az Arduino weboldalról.[1] Az Arduino IDE a neve annak a programnak, ahol a kódunkat megírjuk, és ahonnan azt feltöltjük a fejlesztő panelekre.



Az Arduino UNO áramot kap az USB portból

[1] https://www.arduino.cc/en/Main/Software

Amikor ezt feltelepítettük, csatlakoztassuk az Arduinót az USB portra, ami után a zöld, ON felirattal ellátott LED-nek világítania kell. Most nyisd meg a frissen telepített IDE programot és győződj meg, hogy a Tools/Board menu alatt az Arduino UNO van kiválasztva. Ezután azt is ellenőrizzük le, hogy a Tools/Serial Port menüpont alatt az az USB port van megjelölve, amelyikre az Arduinót csatoltuk. A port neve csak akkor fog megjelenni, ha az Arduino panelt már csatlakoztattuk. Ha problémá áll fel a környezet telepítése és beállítása közben, próbáld meg a [hivatalos telepítési útmutatót](https://www.arduino.cc/en/Guide/HomePage)[1] és



Az Arduino IDE program a Firmata kóddal

[1] https://www.arduino.cc/en/Guide/HomePage

a hibaelhárító oldalt[2] átnézni.

Fentebb látható, hogy az IDE program hogyan néz ki az indítás után. Ezen a ponton nem fogunk hozzá Arduino nyelvű program írásához. Csak azért nyitottuk meg a programot, hogy feltöltsük az alap Firmata programot. Válaszd ki a File/Examples/Firmata/Standard Firmata menüpontot, ami után megnyílik a programkód a szerkesztőben, majd kattints a feltöltés gombra (upload gomb, zöld, jobbra mutató nyíllal), ez feltölti nekünk a megnyitott kódot az UNO-ra.

Ha minden sikeresen lefutott a "Done uploading" üzenetet olvashatjuk a zöld státusz csíkban. Ha hibaüzenet jelenik meg, bizonyosodj meg, hogy az Arduino rendben csatlakozott a számítógéphez, áram alatt áll, és hogy a menüben a helyes panel és port beállítások szerepelnek.

Ha így sem sikerül a Firmata telepítése, olvasd át ezt a hasznos Instructables leírást[1].

[1] http://www.instructables.com/id/Arduino-Installing-Standard-Firmata

[2] https://www.arduino.cc/en/Guide/Troubleshooting

LED KAPCSOLGATÁSA

Az áramkörrel már végeztünk, az Arduinóra pedig feltöltöttük a Firmata programot. Az UNO készen áll, hogy Node.js-től kapott parancsokat hajtson végre.

Nyugodtan töltsd le az összes kész projekt programkódját a www.webondevices.com/download-source/ címről, ha inkább így szeretnéd a példákat végigkövetni.

Mivel a Johnny-Five könyvtárat fogjuk használni a LED kapcsolgatásához, telepsítsük azt fel. Először készítsünk egy új mappát, és lépjünk bele a parancssorunkból, mielőtt beírjük a telepítéshez szükséges parancsot:

```
npm install johnny-five --save
```

Most készítsünk egy üres fájlt is a Node.js programunknak és nevezzük el blink.js-nek. Az első dolog, amit meg kell tennünk, az a könyvtárak

beolvasása és mentése külön-külön változókba a programunk számára. A mi esetünkben csak a Johnny-Five könyvtárról van szó:

```
var five = require("johnny-five");
```

A következő lépés a Board osztály inicializálása, ami különböző hasznos eljárásokat fog nekünk biztosítani az Arduinóval való interakcióhoz:

```
var arduino = new five.Board();
```

Amikor jQuery-vel dolgozunk, gyakran használjuk a `$(document).on("ready", callback);` eseménykezelőt, hogy megvárjuk, hogy a dokumentumunk befejezze a betöltést, mielőtt bármi másba kezdenénk. Az Arduinónak és az USB kapcsolatnak is van egy kis időre szüksége, hogy elinduljon, így itt is egy hasonló eseménykezelőt fogunk használni, mielőtt parancsokat kezdenénk küldeni:

```
arduino.on(”ready”, function(){
    // A panel készen áll
    // A LED itt már kapcsolható
});
```

Ahhoz, hogy kapcsolni tudjuk a LED-et, szükségünk van egy LED példányváltozóra az eseménykezelő meghívott függvényében:

```
// LED hozzáadása a 10-es csatlakozóhoz
var led = new five.Led(10);
```

Ezután pedig egyszerűen meghívjuk a blink() eljárást:

```
// Kapcsold ki-be fél másodpercenként
led.blink(500);
```

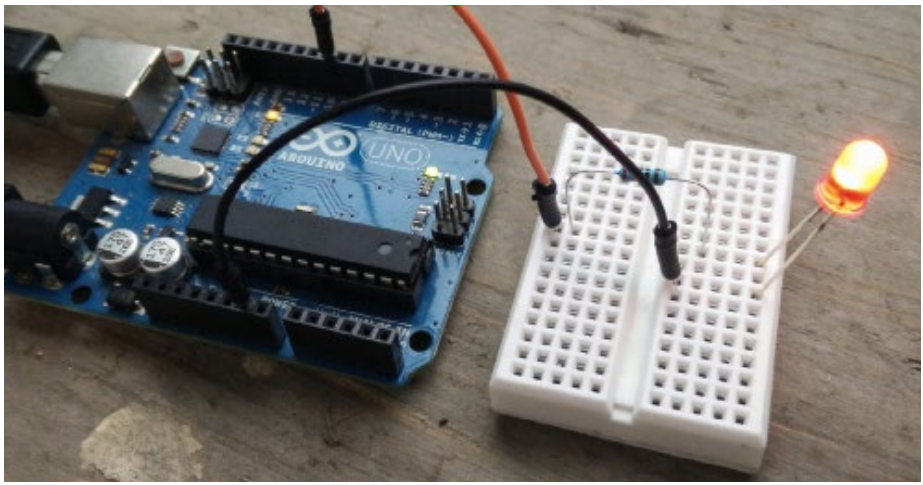
Íme a kész villogó LED kód egyben:

```
var five = require("johnny-five");
var arduino = new five.Board();

arduino.on("ready", function(){
    var led = new five.Led(10);
    led.blink(500);
});
```

Most, hogy a LED működik, megpróbálhatunk más eljárásokat is a példányváltozóról: led.pulse(); fokozatmentesen fogja ki- és beúsztatni a LED-et átmenet nélküli, sima ki-be kapcsolás helyett. Ezen felül külön ki- és beúsztatás eljárások is elérhetőek:

```
// Úsztasd be a LED-et
led.fadeIn();

// Várj 3 másodpercet,
// majd úsztasd ki
```

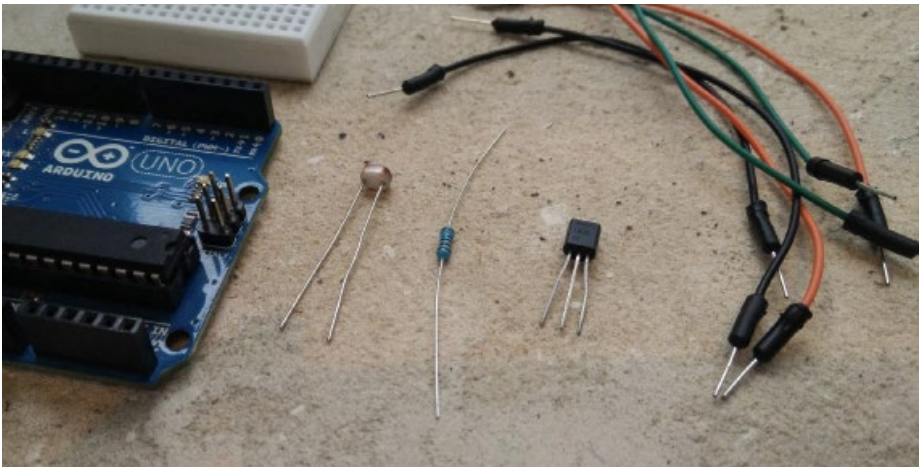
```
this.wait(3000, function(){
    led.fadeOut();
}
```



LED kapcsolása a 10-es kimeneti csatlakozón

A VILÁG ÉRZÉKELÉSE

Ebben a fejezetben kiszélesítjük a tudánsunkat a bemeneti csatlakozók megismerésével. Azt már tudjuk, hogy ezek a csatlakozók képesek elektromos feszültséget mérni, így most olyan egyszerű, analóg szenzorokat fogunk használni, amik a mért környezeti értékek függvényében befolyásolják a kimeneti feszültségüket.



A szeznzor példákhoz használt alkatrészek

Ezekhez a projektekhez szükséged lesz egy Arduino UNO-ra, LM35 vagy TMP36 jelű hőmérő szenzorra, fotoellenállásra (LDR), próbanyákra, két 1k ohmos ellenállásra és néhány próbakábelre.

Hőmérséklet mérés

A hőmérésre a nagyon elterjedt LM35 szenzort választottam. Ezt az olcsó szezort -55 és 150°C közötti mérésre tervezték +/- 0.5°C-os pontossággal (habár negatív értékeket nem annyira egyszerű mérni vele). Ez az analóg szenzor egy elég egyszerű működési elven alapszik sok másik analóg szenzorhoz hasonlóan. Először is két csatlakozójukon keresztül (+ és -) folyamatos energiaellátást kapnak, a harmadik lábon kereszült pedig a mért környezeti értékkekel arányosan csökkentett feszültség értéket mérhetünk.

A mi hőmérőnknek az energiaellátást az Arduino UNO 5V csatlakozója fogja szolgáltatni, kimenetén a feszültség pedig 0 és 2 volt között fog változni a hőmérséklettel arányosan. Az LM35 skálázási együtthatója 0,01V/°C, ami azt jelenti, hogy 1 celsius fok változás a levegő hőmérsékletében 0,01 volt mértékben fogja a kimeneti csatlakozó feszültségét változtatni.

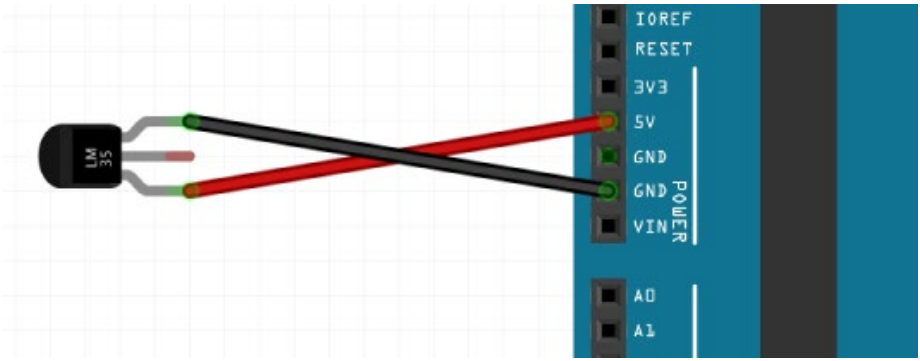
Kézenfekvő, hogy az Arduino bementi csatlakozóit ilyen feszültség értékek mérésére és általunk felhasználható információva alakítására tervezték. Konkrét számokkal kifejezve a bemeneti csatlakozók teljes, 0-tól 5 voltig terjedő méréstartományát az Arduino UNO egy 0 – 1024 közötti skálára vetíti rá a programunk számára.

Tegyük fel, hogy az egyik analóg szenzorunk 1,5 voltot küld vissza a bementi csatlakozók felé a kapott 5 voltból. A programunkban megjelenő érték ez esetben 307 lesz a 0-tól 1024-ig terjedő skálán, mivel (1,5 / 5) * 1024 az 307,2.

A Johnny-Five könyvtár ezek után a 0 – 1024 skálán kapott értéket rávetíti a szenzor -55 - 150°C-ig terjedő tartományára és ebből megadja számunkra a kapott hőmérsékleti értéket.

Lépjünk most vissza a hőmérőnkhöz és építsük be az áramkörbe. Mielőtt elkezdenénk a vezetékek kötögetését, bizonyosodjunk meg arról, hogy az Arduinót kihúztuk az USB csatlakozóból vagy bármilyen más energiaforrásból, így ugyanis fenn áll az esélye annak, hogy véletlenül rossz helyre dugunk egy próbakábelt, amitől sérülést szenvedhet az Arduino, a szenzor, de akár a számítógépünk is.

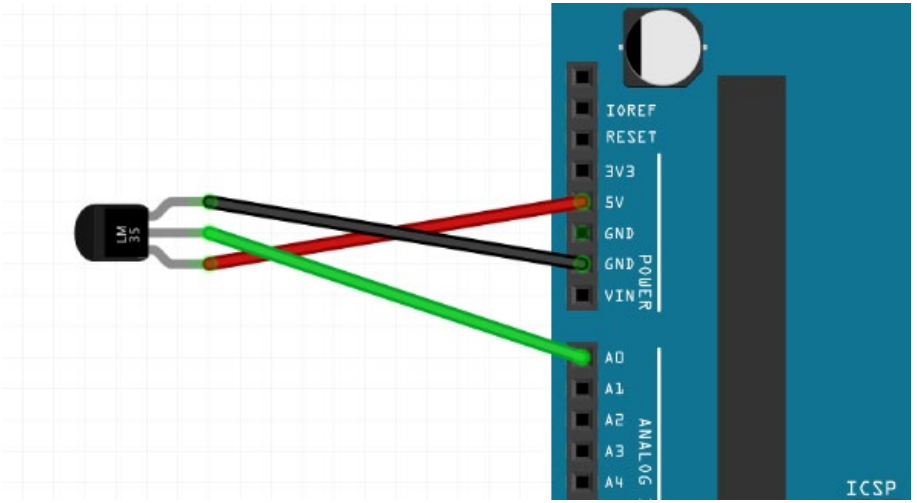
A három láb közül, ami a hőmérő szenzoron található, a bal oldali a pozitív, a jobb oldali pedig a negative pólus (a szenzor lapos oldala az eleje). Az áramot az Arduino UNO 5V (pozitív) és a GND (negatív) csatlakozóiról fogjuk bekötni.



A hőmérő áram alatt áll (Fritzing.org)

A szenzort ezzel áram alá helyeztük, már csak a kimeneti jelért felelős láb van hátra. Ezt az UNO öt rendelkezésre álló analog bemenete közül az egyikre kötjük. Ezek azok a csatlakozók, amiket A0-tól A5-ig jelöl a panel.

Válasszuk az elsőt az A0 felirattal:



A hőmérő az Arduino UNO analóg bemenetére kötve (Fritzing.org)

Most, hogy az áramkör elkészült, írjuk meg a programot, ami beolvassa a szenzor méréseit. Először készítsünk egy új projektet és egy új fájlt temp.js névvel. Ezt a programot ugyanúgy kezdjük, mint a villogó LED projektet:

```
var five = require("johnny-five");
var arduino = new five.Board();

arduino.on("ready", function(){
    // El Arduino está listo
});
```

Most készítünk egy új Temperature osztály példányváltozót, amit néhány beállítással fogunk inicializálni. Ezek a szenzor neve és a csatlakozó száma, amihez kötöttük.

```
var tempSensor = new five.Temperature({
    controller: "LM35",
    pin: "A0"
});
```

Az LM35 szenzor helyett a TMP36 is ugyanúgy használható. Egyszerűen a beállításoknál cseréld ki a nevet, az áramkör változatlan marad.

A hőmérő inicializálás után rendelkezésünkre áll az adat eseménykezelő, ami a bejövő méréseket hívatott elérhetővé tenni a program számára. A Johnny-Five könyvtár kiaknázza a Node.js aszinkron képességeit, ami azt jelenti, hogy a szenzor mérések azonnal megjelennek a beérkező adatok eseménykezelőjének meghívott függvényében. Most írjuk meg ezt az eseménykezelőt:

```
tempSensor.on("data", function(er, data){
    console.log(data.celsius + "°C");
    console.log(data.fahrenheit + "°F");
});
```

Most is várnunk kell, még a panel elindul így mindkét blokk, amit eddig írtunk az `arduino.on("ready", callback);` függvény belsejébe kerül.

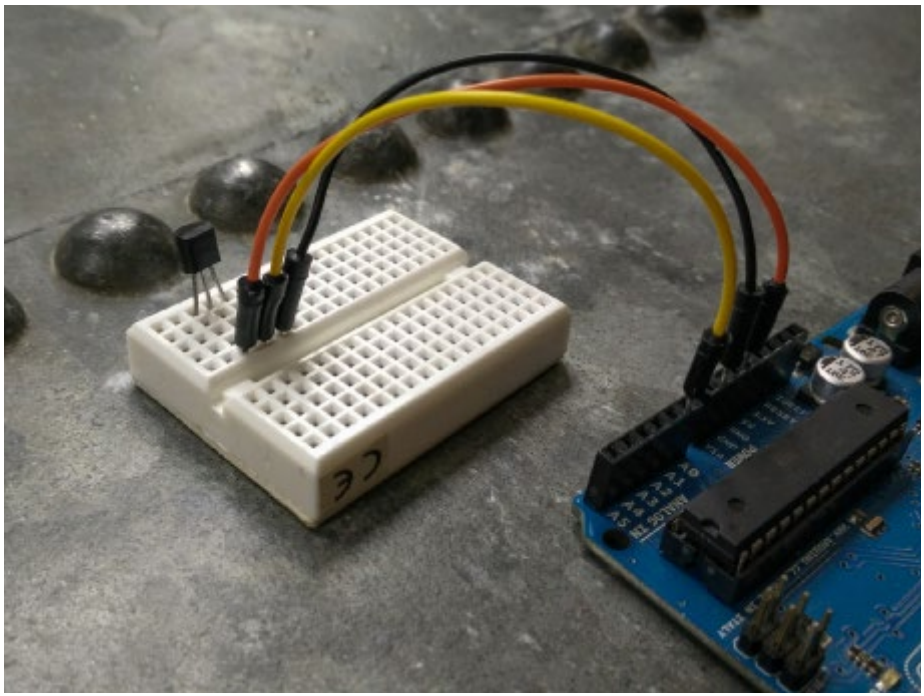
A program végleges változatában mind a Celsius és Fahrenheit értékek egy tizedesjegyig kerekítettek, hogy a számok jobban olvashatóak legyenek.

```
var five = require("johnny-five");

var arduino = new five.Board();

var celsius = 0;
var fahrenheit = 0;

arduino.on("ready", function(){
```

```
var tempSensor = new five.Temperature({
    controller: "LM35",
    pin: "A0"
});

tempSensor.on("data", function(er, data){
    celsius =
        data.celsius.toFixed(1);
    fahrenheit =
        data.fahrenheit.toFixed(1);
    console.log(celsius + "°C");
    console.log(fahrenheit + "°F");
});

});
```

Futtassuk le a programot a `node temp.js` paranccsal. Ellenőrizzük le a méréseket, ha valami nincs rendben, ellenőrizd a kapcsolást, győződj meg, hogy az Arduino az USB-hez van kötve, és, hogy a zöld LED világít rajta. Ha minden megfelelő, folytathatjuk is a munkát a fényérzékelővel.

Végül itt az LM35 hőmérő áramkör egy apró próbanyákon:



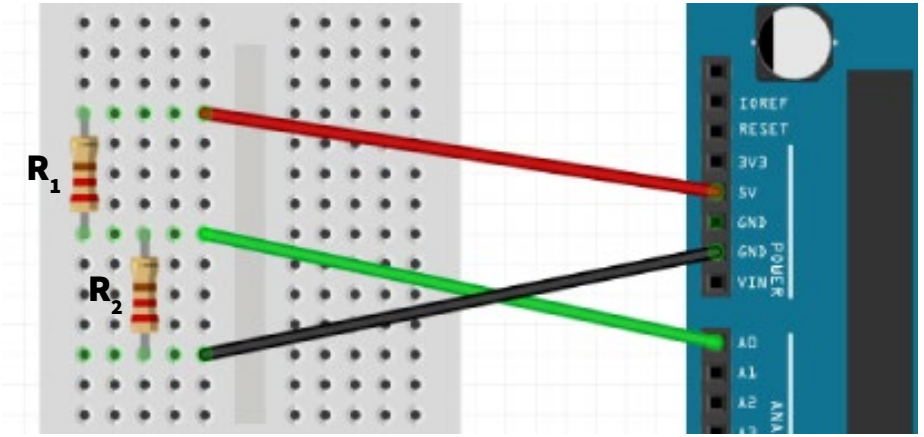
A hőmérő áramkör egy kisebb próbanyákon

Fénymérés

A fénymérő szenzorunk, ami ezúttal egy fotoellánállás, egyike a legegyszerűbb szenzoroknak. A fotoellenállás úgy viselkedik, mint egy hagyományos ellenálls a rajta keresztülfolyó árammal. A legnagyobb különbség az, hogy a fotoellánállás ellenállás értéke változik a környezete fényerejétől.

Sajnos az ellenállás mérése az Arduino analóg bemenetei számára nem túl egyszerű. Ahhoz, hogy ezt az ellánállás váltakozást feszültség váltakozássá alakítsuk, amit a bemenetek már tudnak mérni, szükségünk van egy egyszerű feszültségosztó áramkör építésére. A feszültségosztó egy nagyobb feszültséget képes két kisebbre osztani a benne felhasznált két ellenállás arányában.

Az 5 volt az áramkörbe az Arduinóból érkezik a piros vezetéken keresztül, a kimeneti feszültség pedig a zöldön jelentkezik, ami arányos a bemeneti feszültség és az ellenállások arányával (R1 , R2).



Feszültségosztó (Fritzing.org)

$$V_{out} = V_{in} * (R_2 / (R_1 + R_2))$$

Ebben az áramkörben két 1000Ω-os ellenállást használtunk fel. Nézzük meg, hogy néz ki az egyenletünk ezekkel az értékekkel:

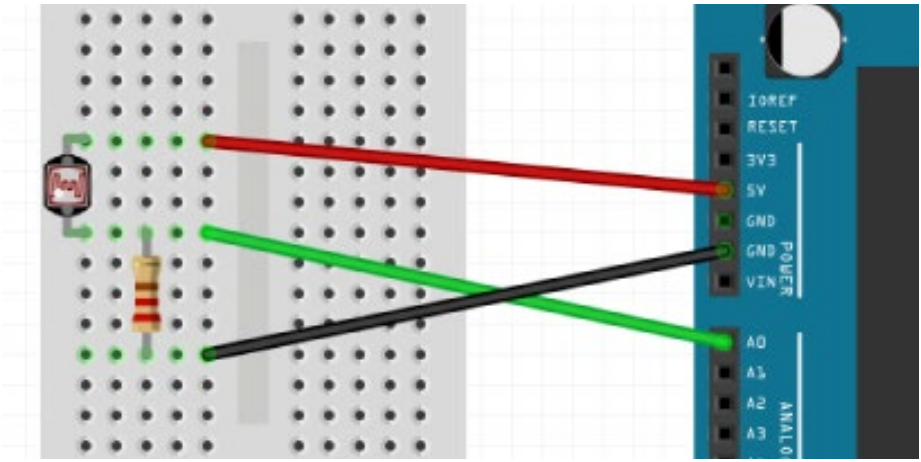
$$V_{out} = 5V * (1000 / (1000 + 1000))$$
$$V_{out} = 5V * 0.5$$
$$V_{out} = 2.5V$$

Ez azt jelenti, hogy ha két egyforma ellenállást használunk a feszültség osztó áramkörben, a kimeneti feszültség fele lesz a bemenetének. 2,5 volt a mi esetünkben. Az egyenlettel azt is látjuk rögtön, hogyha

az egyik ellenállást változtatjuk, a kimeneti feszültség is fel és le fog ingadozni.

A fénymérő áramkörben az egyik hagyományos ellenállást a fotoellenállásra cseréljük. A fotoellenállás ellenállás értéke változni fog a fényviszonyok hatására a feszültségosztó áramkörben, ez pedig a kimeneti águnkon jelentkező feszültség értéket fogja változtatni az Arduino bemenete számára.

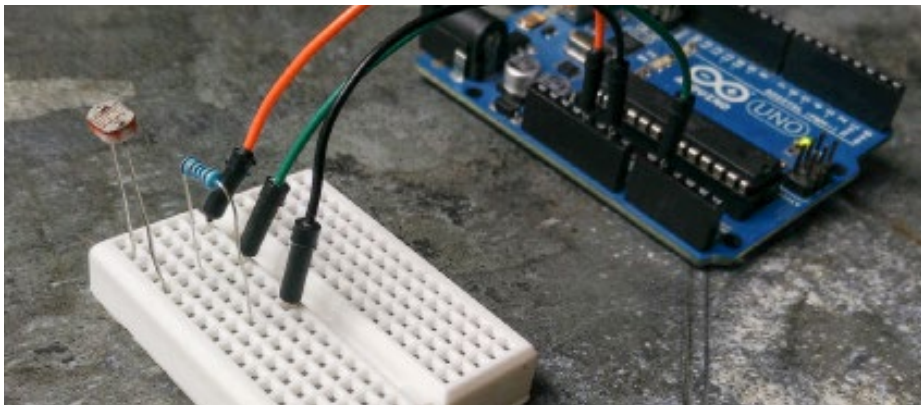
Ha a fotoellenállást fénynek tesszük ki, ellenállása csökkenni kezd, ezzel a kimeneti feszültség növekszik. Fordított esetben, kevesebb fény hatására a feszültség alacsonyabb lesz. Ez a feszültség változás az, amit az analóg bementi csatlakozó az Arduinón mérni képes.



Fénymérő áramkör (Fritzing.org)

A fotoellenállás ellenállása nagy fénynél 150Ω körül kezdődik, sötétben pedig 9000Ω körüli értékig csökken. Ha megtartjuk az eredeti 1000Ω-os ellenállást, a feszültségváltó áramkörünkben kiszámolhatjuk, hogy a kimeneti feszültség 4,35V és 0,5V között fog változni, ami egy tökéletes tartomány az Arduino bemenetei számára.

Így néz ki a kész áramkör a próbanyákon:



Fénymérő áramkör

Az áramkör készen áll, adjuk most hozzá a fénymérő szenzort a JavaScript programunkhoz az Arduino "on ready" függvényébe. Először inicializálunk egy új Sensor osztály példányváltozót, majd hozzáadunk egy új beérkező adat eseménykezelőt:

```
var lightSensor = new five.Sensor({
    pin: "A1",
    freq: 250
});

lightSensor.on("data", function(){
    console.log(this.value);
});
```

Ez a program blokk nagyon hasonlít arra, ahogy az előzőekben a hőmérőnket a Johnny-Five könytár kezelte. Először az új szenzort inicializálnunk kell néhány beállítással, majd az új példányváltozó eseménykezelője várja a beérkező adatokat.

A fénymérő program végleges változata a következő képpen néz ki:

```
var five = require("johnny-five");

var arduino = new five.Board();

var light = 0;
```

```
arduino.on("ready", function(){

var lightSensor = new five.Sensor({
    pin: "A1",
    freq: 250
});

lightSensor.on("data", function(){
    light = this.value;
    console.log(light);
});

});
```

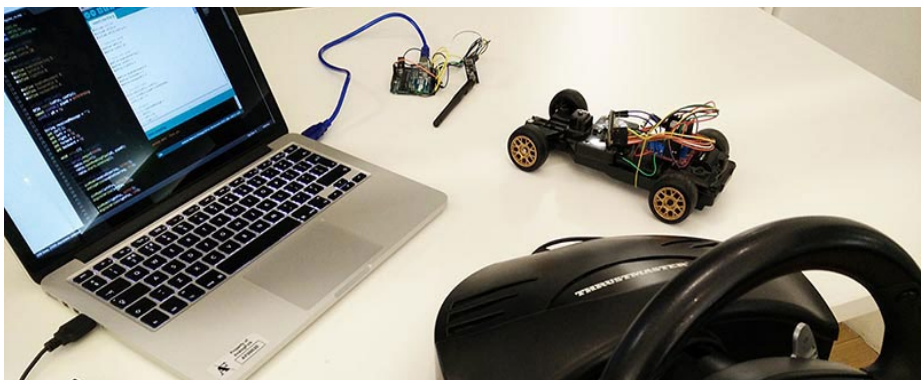
Mentsük mindezt el egy új, light.js fájlba, majd adjuk ki a `node light.js` parancsot a parancssorba. Ezután a fénymérő áramkörünkből érkező mérések meg fognak jelenni minden negyed másodpercben. A freq (frekvencia) beállításnak megadott milliszekundum érték a kódban az, ahol a frissítés sűrűségét meg tudjuk változtatni.

Az alábbi linkről letöltheted az egész project forráskódját: www.webondevices.com/download-source

TOVÁBBI LEHETŐSÉGEK

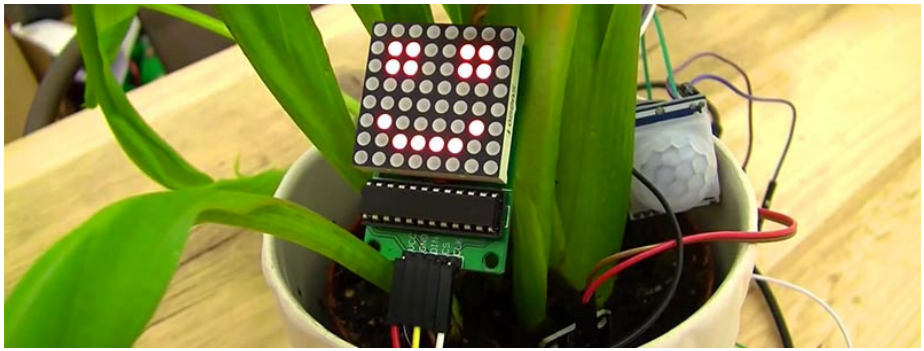
Ebben a könyvben megismertük az Arduino UNO és a Node.js közti kommunikáció alapjait, amit az USB porton keresztül menedzseltünk. A Web on Devices weboldalon, www.webondevices.com, fogsz találni néhány projektet, amelyek hasonló technikák felhasználásával készültek.

Olvass utána a távirányítós autónak,[1] amit az alapoktól építettem újra Arduinókkal. Az autó vezeték nélkül kapcsolódik a számítógéphez, amit a felhasználó a Gamepad API-n keresztül egy USB-s kormányról tud vezetni.

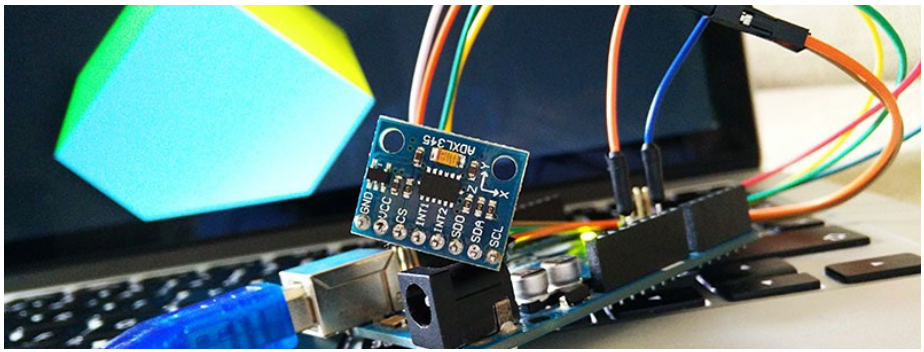


[1] http://www.webondevices.com/arduino-nodejs-rc-car-driven-with-html5-gamepad-api/

Vagy ott van George, a beszélő növény.[1] Szenzorain keresztül képes a hőmérsékletet, fényt, mozgást és a talaj nedvességét érzékelni. Panaszkodik, ha nem tetszik neki valami a mérések közül, és egyszerűbb kérdésekre is tud válaszolni. A növény a WebSpeech API segítségével hallgat és beszél.



Ezenkívül van egy Arduino giroszkóp projekt [2], amivel egy 3D CSS kockát tudunk mozgatni a képernyőn egy kontrollerrel.



[1] http://www.webondevices.com/the-arduino-plant-with-javascript-voice-recognition/

[2] http://www.webondevices.com/rotate-a-css-3d-cube-with-an-arduino/

A Node.js-el való Arduinó kommunikáció csak egy módja annak, hogy a JavaScriptet elektronikai projektekhez használjuk fel. Ezekben a példákban a JavaScript kód valójában a számítógépünk processzorán futottak ahonnan vezérlő parancsokat küldtünk az Arduinónak. Más fejlesztő panelek, mint például a Raspberry PI, az Arduino Yun, a Tessel és az Espruino képesek JavaScript futtatására a saját processzorukon. Az Arduino kompatibilis Particle panelek egy REST API-t tesznek elérhetővé, amivel egy Node.js könyvtáron keresztül tudunk kommunikálni. A Particle panelek vezetéknélkül csatlakoznak az Internetre, így a Node könyvtárnak nincs szüksége USB kapcsolatra.

A Web on Devices projekt azzal a céllal jött létre, hogy feszegesse az okos eszközök és fejlesztő panelek főleg web technológiákkal történő együttműködésének határait. A következő projektekben minden említett panelt és technológiát alaposabban felfedezünk.

Kövesd a Web on Devices-t Facebook-on, Twitter-en és Instagram-on, hogy ne marajd le a következő projektekről:

www.facebook.com/webondevices
www.twitter.com/web_on_devices
www.instagram.com/web_on_devices

Web on Devices

Elektromos eszközök hekkelése JavaScript és más Webes Technológiák alkalmazásával

www.webondevices.com