Felhő alapú elosztott vezérlés, yakinduban tervezett állapotgépet használó beágyazott robot rendszeren.

**Készítette**

* **Kovács Levente Ákos - CM6UKU**
* **Tóth Krisztián Dávid - J38GIK**

Tartalomjegyzék

[Bevezetés 1](#_Toc437437359)

[A feladat: 1](#_Toc437437360)

[Program futása: 2](#_Toc437437361)

[Felhasznált Technológiák: 2](#_Toc437437362)

[A cloud rendszerhez: 2](#_Toc437437363)

[A robothoz: 2](#_Toc437437364)

[Technológiák: 2](#_Toc437437365)

[Node-red 2](#_Toc437437366)

[Bevezetés: 2](#_Toc437437367)

[Telepítése: 3](#_Toc437437368)

[Nodeok: 3](#_Toc437437369)

[Nodeok készítése/felépítése: 7](#_Toc437437370)

[Yakindu 10](#_Toc437437371)

# Bevezetés

A feladat: Egy labirintusban közlekedő buta(magától csak jobbra és felfele közlekedő) robot szimulálása beágyazott rendszeren(Raspberry Pi), ami adott kezdeti állapotból egy vég állapotba próbál eljutni, egy pc-n futó cloud rendszer segítségével. Továbbá az ehhez tartozó technológiák megismerése(Node-red, Yakindu).

Program futása: A Robot a pályát, a vég céljának és a saját poziciójának koordinátáit, a cloud rendszertől kapja meg az inicializációs szakaszban. Ez után a robot addig megy felfele ameddig csak tud, és ha elakad akkor jobbra kerül. Amennyiben se jobbra se fel nem tud lépni a robot(szenzorai falat érzékelnek mind 2 irányban), jelez a cloud rendszernek, hogy szüksége van segítségre és eküldi az aktuális koordinátáit, ezek után pedig vár a külső vezérlés válaszára. A cloud rendszer lefuttat egy A\* út kereső algoritmust a robot és a cél koordinátáival. Majd vissza küldi a robotnak. A gyakorlati megvalósitásokról külön fejezetekben részletesebben írunk.

## Felhasznált Technológiák:

### A cloud rendszerhez:

A kommunikációs logika Node Red-ben került implementálásra MQTT (Message Queueing Telemetry Transport) protokoll felhasználásával, amit a mosquitto(mqtt broker) valósít meg. A kereső motornak IMOR nevű felhasználó pathfindig Node package-ét használja.

### A robothoz:

Yakinduban tervezett állapotgépből generált C++ kódot futtat, továbbá szintén node red fut a komunikáláshoz.

# Technológiák:

## Node-red

### Bevezetés:

A Node-RED egy grafikus felület hardwarek-ek, Apik, és online szolgáltatások összekötésére, és az esemény vezérelt kommunikációs modell létrehozására. A Node-red alapja Node.js ami megkönnyíti a fejlesztést a több mint 120 000 szabad forrású moduljával ami gyorsan elérhető a npm-en keresztül(1 parncs cmd-ből), továbbá optimálissá teszi cloudon és raspberry Pi on való felhasználásra.

A flow ((„esemény”) folyam):az összeköttetést és a hozzá tartozó logikát megvalósító esemény vezérelt modell. A node red-ben ilyen flow-kat alakítunk ki, innentől kezdve a futó modellünkre/programunkra is flowként fogok utalni.

Egy flow kialakítását web browser-ben(firefox,chrome…) a node red portjára(sajátip:1880) csatlakozással lehet végrehajtani miután fut a Node-red. Minden flow-t automatikussan ment a node-red, de csak ha deploy-oltuk. A flow-k között a képernyő tetején lévő sheet(munkalap)-ek kiválasztásával tudunk váltogatni, és itt tudunk ujjat sheet et létrehozni a + gombbal.

A flow–kat JSON ben tárolja a Node-red megkönyítve az importálás/exportálás –ukat, továbbá a megosztásukat az „online flow library”-ban.

### Telepítése:

(Megj: Jól érthető angol leírás található a nodered.org-on. Ennek az alfejezetnek csak a windows-os telepítésnek általános ismertetése a célja.)

Le kell tölteni és telepíteni egy 0.10.x nél későbbi Node.js-t.

Majd a legegyszerűbb módszer, globálissan telepíteni a nodered-et, a node Package manager(npm)-en keresztül az alábbi parancsal:

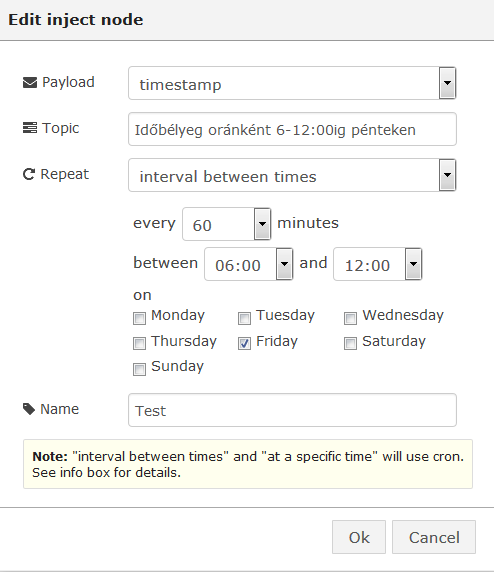
npm install -g --unsafe-perm node-red

Ez a parancs a C:\Users\”felhasználó”\AppData\Roaming\npm mappába fogja globálissan telepíteni a node-red-et , ami után bárhonnan futtatni lehet parancssorból a node-red parancs kiadásával.

Nodeok:

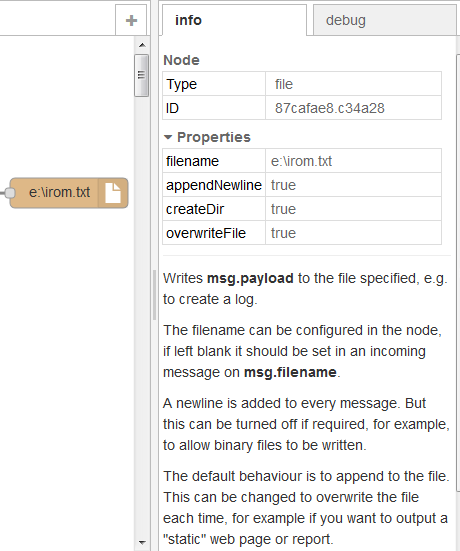
A node-ok, a folyamjainknak(flow) az építő elemei. Ezekből huzalozzuk össze a grafikus felületen a megvalósitandó rendszerünket. A node-ok müködését úgy tudjuk beállítani hogy 2x klikkelünk a modul inkonjára és a felugró ablakon beállítjuk a megfelelő konfigurációs értékeket.

PL: Inject node beálltása



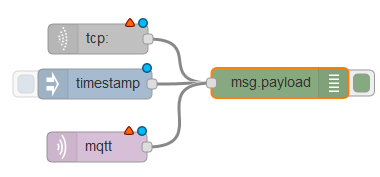
Minden node hoz tartozik egy leírás, ezt a grafikus felület jobb oldalán található info ablakban olvashatjuk el, ez tartalmazza a típusát, a program által generált egyedi azonosítoját, a Properties-t ami a node konfigurációs értékeit tartalmazza, és egy álltalánost leírást a node müködéséről.

PL: file(iró) node info palettája.



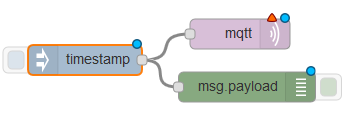
Megkülönböztethetjük őket :  
Funkció szerint:

* Input(Bemeneti): Esemény generálásra használatos, ez lehet különböző interface-ről (Tcp/Udp/Mqtt) beérkező üzenet-ből létrehozott node msg vagy csak egyszerű esemény injektálás. Az esemény injektáló nodeot be lehet úgy állítani, hogy ismételje az esemény injektálást bizonyos időközönként (pl mp-enként), megadott időpillanatban(pl péntek 18:00-kor), vagy megadott intervallumon belül időközönként(pl szombaton és pénteken 19:00-20:00 között percenként ). Ez széleskörű polling lehetőségeket biztosít a felhasználó számára.



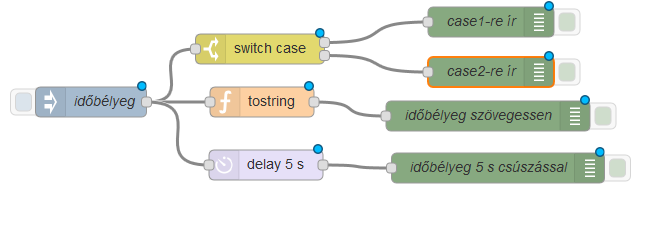
(Bejövő üzenetek payloadját kiirja a debug ablakban)

* Output(Kimeneti): Csak torkollanak bele folyamok, ki nem indulnak belőle. Főleg üzenet továbbitásra használjuk, pl udp,mqtt csomag küldése vagy debug node-ot(zöld) a message payload-gyának logolásra.



Időbélyeget küld ki mqtt-n keresztül, továbbá kiirja a debug ablakba.

* + - * Function(függvény):Olyan nodeok amik valamilyen adatmanipulációt (xml,Json convert to/from …), késleltetést(delay), vagy valamilyen kapcsolat felvételt(tcp/http request) hajtanak végre. Ezeknek a nodeoknak tipikusan kimenetet és bemenete is van. Itt található a saját kódot futató function node is, de ez saját sandbox környezetbe fut, aminek eredményeként nem hivatkozhat más package-re csak a saját magunk általt Java scriptben megírt kód futhat benne. Ez főleg egyszerű de egyedi adatmanipulációra jó, de a komplexebb feladatokhoz érdemes saját nodeot készíteni (lsd később).

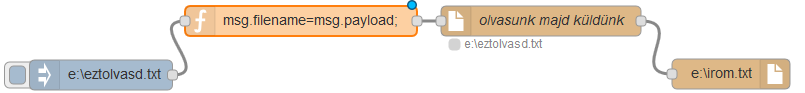


* + - * Social: Twitter/Email üzenetek lekérésére és küldésére használt nodeok.

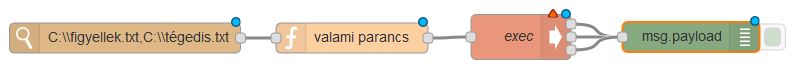
(tweetekből emailt-küld , emaileket tweeteli)



* + - * Storage: File kezelést megvalósító 2 node tartozik alá, az 1. beolvassa azt a file-t aminek a nevét kapta msg payloadban a bemenetén majd tovább küldi a tartalmát. A másik csak file ba írást végez.



* + - * Advanced: Ide kerülnek a különleges funkciókat ellátó node-ok, továbbá általában az internetről letöltött nodeok. Gyárilag a rendszerhívást megvalosítto exec(3 output stderr stdout return), és a fájl változást figyelő watch található itt.



#### Kimenetek/bemenetek száma szerint:

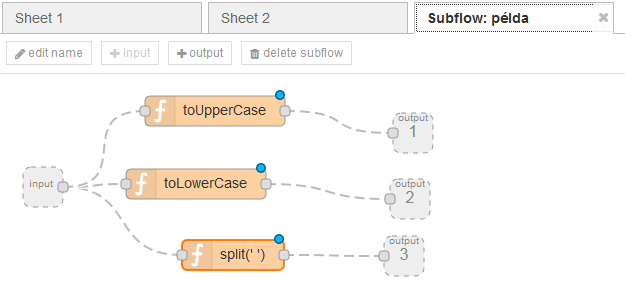
Egy nodenak lehet tetszőleges számú kimeneti és 0 vagy 1 bemeneti socket-je is (ezek kezeléséről lsd később). Egy bemenet socket-be tetszőleges számú event folyhat be (pl : Advanced példa-nál execből mind a 3 kimeneti socket a debug 1 bemenetével van összekapcsolva), és egy kimeneti socket-ből akár több írányba is továbbithatunk egyszerre msg-t (pl Output példánál). Több kimeneti socket-el, külön választhatjuk az adatfolyamunkat több párhuzamos végrehajtási irányban, vagy feltételhez köthetjük az üzenet útját/tartalmát .

### Nodeok készítése/felépítése:

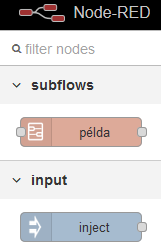
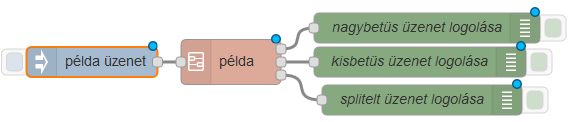
Egy node-ot 3 féle képen lehet létrehozni,

1. Internetről npm-en keresztül : Ez a legegyszerűbb módszer, keresünk az npmjs.com oldalon található node-ok küzül egy olyat amire nekünk szükségünk van majd kiadjuk a npm install node-red-node-{filename} parancsot és telepitjük (Megj:minden node red node neve node-red-node-\* kezdödik, a többi sima node.js package ezeket a 3. modszerben ismertetett modon lehet felhasználni.)
2. Subflow:

Lehetőségünk van a meglévő node-jaink-ból egy új node létrehozására, ez főleg az átláthatóságot és a hierarchikus tervezést hivatott megkönnyíteni. A node-red web browser-es kezelő felületén a jobb felső sarokban található 3 egymás alatti vizszintes vonalra klikkelve(  ), kiválasztjuk a create subflow opciót. Ezek után hozzá adhatunk maximum 1 bemenetet, és tetszőleges számú kimenetet a subflow-nk hoz. És kialakíthatunk egy flow-t a hozzátartozó logikával. PL:



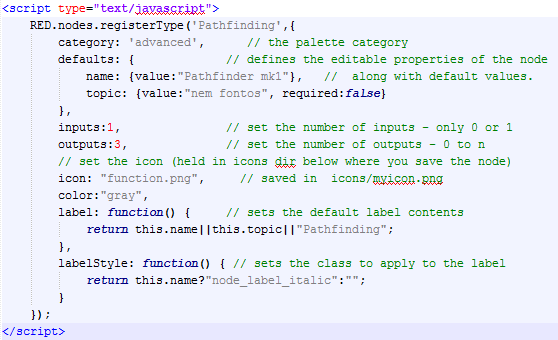
Ha ezzel készen vagyunk meg fog jelenni a node-listában a subflow menü pont alatt az új nodeunk. Innentől pedig használhatjuk, csak be kell huzni egy flowba.

Saját node készítése/node felépítése:  
Minden node 2 fájlból felépíthető manuálisan is. Egy Html és egy Javascript fileból. Ezek nevének meg kell egyeznie, és az alábbi formátumra illeszkedni : az első karakterek ’–’ ig egy számkódot reprezentálnak, ezt követi a név.html/js attól függően hogy melyik file(pl 99-pelda.js , 99-pelda.html). Ezeket a fileokat a C:\Users\”felhasználó név”\AppData\Roaming\npm\node\_modules\node-red\nodes mappába kell elhelyezni ha globálissan lett a node red telepítve. De a node-red működését és elérési útvonalait személyre lehet szabni (elég hozzáértéssel), a settings.js file manipulálásával (ami a node red könyvtárban található).( A nodeok készítéséről jó leírás és példa található a node-red weboldalán )

A Html file: Ez a node konfigurációjáért felel, innen fogja tudni a browser-es grafikus felületünk, hogy hogyan kell kezelnie a nodeot, mi a bemenetek illetve kimenetek száma, milyen konfigurációs ablakot dobjon fel mikor ráklikkelünk, melyik tipusba sorolodik a node listában, hogy néz ki az ikonja, milyen leírást tartalmaz az info ablaka stb.

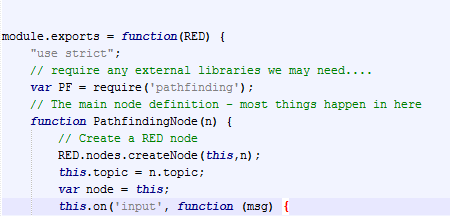
pl (html részlet):



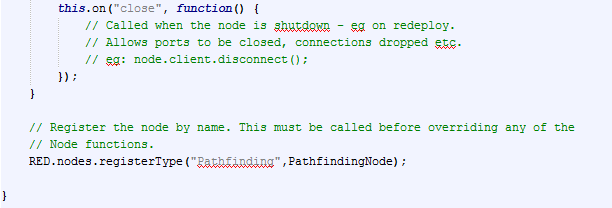
##### A Javascript file:

A node viselkedését egy javascript fileban tudjuk megirni. A function node-hoz képest meg van az az előnye hogy itt includeolhatunk( javascriptben requre) más library-ket . (Megj: Nagy könnyítést jelent hogy rengeteg problémára található library az npmjs.com-on amit egy npm install parancsal már használatra késszé is tehetünk. )

Alapvető js program keret:



Valamilyen logika amit csinál a node.



### Message kezelés:

Node red-ben message-ekkel kommunikálnak a node-ok, ezek JSON objectumok amikre msg-ként hivatkozhatunk. Minden msg-nek van egy msg.payload property-je , ebben tároljuk a hasznos adatokat, ezen felül még msg.topic property-vel is rendelkezik sok default node által generált msg, ennek az oka a node red mqtt-s származásban keresendő. Vannak olyan nodeok amik több property-t is használnak, ilyen például a file („in”) node ami ha nem égetve kapja meg az olvasni kívánt file elérési utvonalát, akkor a msg.filename property-től várja ezt. De a saját nodejainkban is tetszőleges számu property-t bevezethetünk.(Megj: a msg-k 2 node közötti property manipulációjára nagyon hasznos a function node, igy pl az msg.filename=msg.payload kód beiktatásával a file in számára feldolgozhatóvá lehet tenni egy sima msg-t)

#### Példa kódok üzenet manipulációra:

* + - * Üzenet tovább küldése:
      * Új üzenet létrehozása és továbbküldése:
      * Üzenet szétválasztása (2 output közül csak az egyikre rakja ki a másikra nem (nem küld NULL üzenetet, megszakítja a folyamot))
      * Több üzenet egyszerre kiküldése/üzenet sorosítás

## Yakindu

Források:

<https://www.npmjs.com/package/pathfinding>

<http://www.rs-online.com/designspark/electronics/eng/blog/building-distributed-node-red-applications-with-mqtt>

<http://nodered.org/>