|  |  |  |
| --- | --- | --- |
|  | | |
| GRBL vezérlő felhasználói interfész fejlesztése REACT.jS technológiával | | |
| Razvoj GRBL upravljačkog interfejsa REACT.JS tehnologijom | | |
| Grbl control interface development with REACT.JS | | |
|  |  |  |
| Hallgató |  | Mentor |
| Kovács Árpád |  | dr. Pintér Róbert |
|  | Szabadka, 2019 |  |

Absztrakt

A feladat egy olyan CNC vezérlés létrehozása mellyel akár a programot távolról is lehet futtatni. A programnak JavaScript alapúnak kell lennie, valamint tartalmazza, ábrázolja, és engedélyezze a kód szerkesztését.

A feladatot úgy kell megoldani, hogy Raspberry PI mikroszámítógépen keresztül futtatható legyen.

A program céljai egy olyan böngésző alapú alkalmazás létrehozása melyen keresztül a GRBL CNC vezérlőt lehessen irányítani akár mobil telefonról is, valamint ábrázolja a kívánt kódot és szerkeszthető is legyen, mutatja az aktuális program állását és az adott pillanatban leállítható is legyen.

A feladat eredménye egy olyan alkalmazás létrejötte mely távolról is engedi a felhasználónak, hogy távolról is dolgozhasson.

Tartalomjegyzék

[Absztrakt 2](#_Toc11774397)

[Jelmagyarázat 4](#_Toc11774398)

[Köszönetnyilvánítás 5](#_Toc11774399)

[A szakdolgozat témája 6](#_Toc11774400)

[1. Bevezető 7](#_Toc11774401)

[1.1. A probléma leírása 7](#_Toc11774402)

[1.2. A szakdolgozat céljai 7](#_Toc11774403)

[2. Szakirodalmi áttekintés 8](#_Toc11774404)

[2.1. Node.JS 8](#_Toc11774405)

[2.2. Serialport 8](#_Toc11774406)

[2.3. Express.JS 8](#_Toc11774407)

[2.4. Electron 8](#_Toc11774408)

[2.5. Socket.IO 8](#_Toc11774409)

[2.6. Three.JS 9](#_Toc11774410)

[2.7. React.JS 9](#_Toc11774411)

[2.8. GRBL Vezérlő 9](#_Toc11774412)

[2.9. Fejlesztőkörnyezet - Visual Studio Code 9](#_Toc11774413)

[2.10. Arduino 10](#_Toc11774414)

[2.11. Raspberry PI 3 10](#_Toc11774415)

[2.12. DB25 Shield 11](#_Toc11774416)

[3. Problémamegoldás 12](#_Toc11774417)

[3.1. A vezérlés lépései 13](#_Toc11774418)

[3.2. Web szerver 13](#_Toc11774419)

[3.3. Soros kommunikáció 14](#_Toc11774420)

[3.4. Fájlbeolvasás 17](#_Toc11774421)

[3.5. Az objektumok megjelenítése a képernyőn 18](#_Toc11774422)

[3.6. Program futtatása 21](#_Toc11774423)

[3.7. Kézi vezérlés 22](#_Toc11774424)

[3.8. A szoftver telepítése 24](#_Toc11774425)

[3.9. A felhasználói felület 26](#_Toc11774426)

[4. Összegzés 27](#_Toc11774427)

[Irodalom 28](#_Toc11774428)

[Mellékletek 29](#_Toc11774429)

Jelmagyarázat

| **Jelzés/Rövidítés** | **Jelmagyarázat** |
| --- | --- |
| **Node.JS** | JavaScript alapú web szerver oldali környezet |
| **Socket.IO** | Egy valós idejű támogatást biztosíŧó könyvtár |
| **JavaScript** | Egy böngészőben futó szkript nyelv |
| **HTML** | HyperText Markup Language |
| **CSS** | Cascading Style Sheets |
| **Bootstrap** | CSS könyvtár |
| **G kód** | A G kód az a nyelv ahol, az emberek meghatározzák, hogy az automatizált gép mit csináljon [1] |

Köszönetnyilvánítás

Köszönettel tartozom Dr. Pintér Róbert tanár úrnak, a mentorom volt a feladat megvalósítása során. Amikor felötlött bennem a szakdolgozat témája, hozzá fordultam és a véleményét kértem róla. Miután elmagyaráztam, hogy mit is szeretnék akkor alaposan átbeszéltük a részleteket és még bővítettük a már amúgy is vaskos teendők listáját. Megbeszéltük, hogy az elkészült projektnek minden olyan eszközön működnie kell, amelyen a vezető modern böngészők valamely változata fut.

A Rehák kollégium volt és jelenlegi lakóinak.

Valamint köszönettel tartozom még a családomnak, akik nagyban támogattak és hozzájárultak ahhoz, hogy eljuthattam idáig és megírhatam a szakdolgozatomat.

A szakdolgozat témája

A szakdolgozatnak témája egy marógép vezérlő megszerkesztése és megvalósítása. Egy meglévő marógéphez kell készíteni olyan szoftveres-hardveres rendszert, amely lehetővé teszi:

* maróminták számítógépről való letöltését
* a letöltött minták helyességének ellenőrzését
* marási folyamat vizualizálását a számítógépen
* a marási folyamat távvezérlését mobil eszközről.

1. Bevezető
   * 1. A probléma leírása

A technológia fejlődése és a tömegtermelés lehetővé teszik, hogy korábban nehezen beszerezhető és drága eszközöket bárki megvásárolhatja, most már elfogadható áron. Egy ilyen eszköz a szakdolgozathoz használt marógép is. Ez a készülék is ugyanúgy, mint sokmás hasonló eszköz, vezérlőegységgel érkezik. A vezérlés megvalósítása, a programok fejlesztése és telepítése mind a gyártótól függ. Sokszor találkozhatunk limitált képességű vezérlőegységekkel, olyanokkal például ahol nincs vizualizálva a megmunkálás folyamata. Ezek az "apró" hiányosságok kihatnak az eszköz hatékonyságára, a termelés folyamatára.

Mivel legtöbb ilyen eszköz szabványos protokollokkal kommunikál a vezérlőegységgel, ebben a szakdolgozatban egy olyan megoldás kifejlesztésén lesz a hangsúly, amely helyettesítené egy marógép vezérlőjét, ezzel biztosítva olyan lehetőségeket, amelyek az eredeti vezérlőben nem szerepeltek.

* + 1. A szakdolgozat céljai

A szakdolgozat célja elkészíteni egy rendszert, amely helyettesítené egy marógép vezérlőjét. Ehhez egy GBRL vezérlőt szeretnénk implementálni, illetve kezelőfelületet fejleszteni. A GBRL-ről annyit kell tudni, hogy ez egy ingyenes, nyílt forráskódú, nagyteljesítményű szoftver a mozgó gépek mozgásának vezérlésére, Arduino segítségével. Az új vezérélés, amely azon felül, hogy tartalmazza a G kód kezelőt a marógép vezérléséhez, lehetővé teszi a gép vezérlését desktop és web alapú alkalmazásokkal. Maga a vezérlés legfontosabb szempontja, hogy kielégítse a leendő felhasználó, a rendszerrel szemben támasztott elvárásait.

1. Szakirodalmi áttekintés

A szakdolgozat megvalósításához a következő technológiákat használtam fel:

* HTML5
* JavaScript
* Jquery
* Ajax
* Node.js
* Css
* Three.JS
  + 1. Node.JS

A Node.js egy szoftverrendszer, melyet skálázható internetes alkalmazások, mégpedig web szerverek készítésére hoztak létre. A programok JavaScript-Ben írhatók, eseményalapú, aszinkron I/O-val a túlterhelés minimalizálására és a skálázhatóság maximalizálására.[2]

* + 1. Serialport

A Serialport könyvtár egy C++-ban íródott könyvtár a Node.JS-hez mely multi platform alkalmazásoknál UNIX, Windows és OSX alapú rendszerek közötti soros kommunikációt támogatja.

* + 1. Express.JS

Az Express.JS ami web szervert támogatja, az alkalmazásban melyen keresztül megjeleníti a HTML weboldalat.

* + 1. Electron

Az Electron egy olyan könyvtárra Node.JS-nek mely egy Chromium böngészőbe belefordítja az aktuális programot. S azon keresztül a használatát vezérli.

Az Electron alapú alkalmazások multi platform támogatottságúak.

* + 1. Socket.IO

A Socket.IO egy olyan esemény hajtott könyvtár melyen keresztül a program (node alapú szerver) kapcsolódik hasonlóan az AJAX alapú rendszerekhez a kívánt weboldalhoz melyen keresztüli vezérlés valós időben történik. Magyarul, ha a felhasználó rákattint egy gombra, akkor a Socket.IO elküldi ezt a szervernek, melynek a kívánt eseményre válaszol. Ezzel a technológiával lett megírva a szünet gomb is.

* + 1. Three.JS

A Three.JS könyvtár egy 3D megjelenítő könyvtár melyen keresztül az adott modellt megjelenítjük a böngészőben.

* + 1. React.JS

Facebook által fejlesztett könyvtár-csomag melyet kifejezetten egyoldalas applikációk számára fejlesztettek ki.

* + 1. GRBL Vezérlő

A GRBL vezérlő program az ATMEL ATMEGA328P vezérlőre írt CNC vezérlő program. Egy nagy teljesítményű CNC marógép program. Mely G kódokat végrehajtja.

* + 1. Fejlesztőkörnyezet - Visual Studio Code

Ez egy a Microsoft által fejlesztett IDE. Amely integrált fejlesztő környezetet jelent. A Visual Studio Code talán napjaink egyik legfejlettebb IDE-je, minden eszköz a fejlesztő rendelkezésére áll.

Található benne:

* Debugger
* Automatikus kód kiegészítés
* Különféle fejlesztői tesztek
* Szintaxis kiemelés
* Automatikus kódformázás

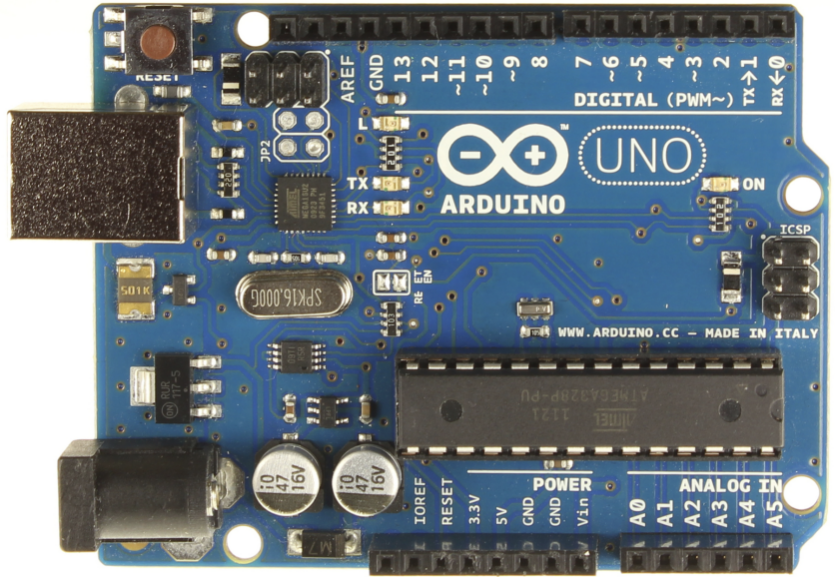
A Visual Studio Code azért is rendkívül jó IDE mert rengeteg programozási nyelvet támogat. Csak, hogy néhányat említsek:

* JavaScript
* Node.js
* HTML
* CSS
* C#
* C++
* F#
* CoffeScript
* TypeScript

A szakdolgozat megvalósításánál ebben készítettem a JavaScript, Node.js, HTML és CSS részeket. Azért is esett a Visual Studio -ra a választásom mert a Főiskola tanulóinak ingyenesen beszerezhető.

* + 1. Arduino

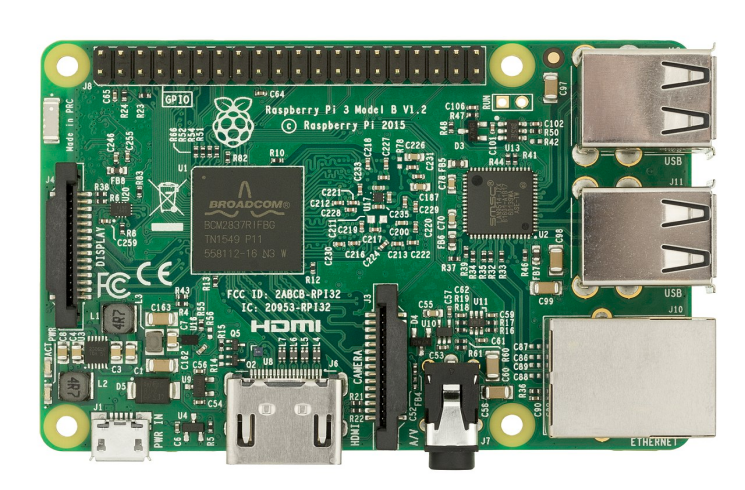
Az Arduino egy szabad szoftveres, nyílt forráskódú elektronikai fejlesztőplatform, arra tervezve, hogy a különböző projektekben az elektronikus eszközök könnyebben hozzáférhetőek, kezelhetőek legyenek. Széles tömegek számára elérhető, mivel olcsó, könnyen beszerezhető, egyszerűen programozható, és csatlakoztatható más eszközökhöz. [3]



1. ábra Az Arduino

* + 1. Raspberry PI 3

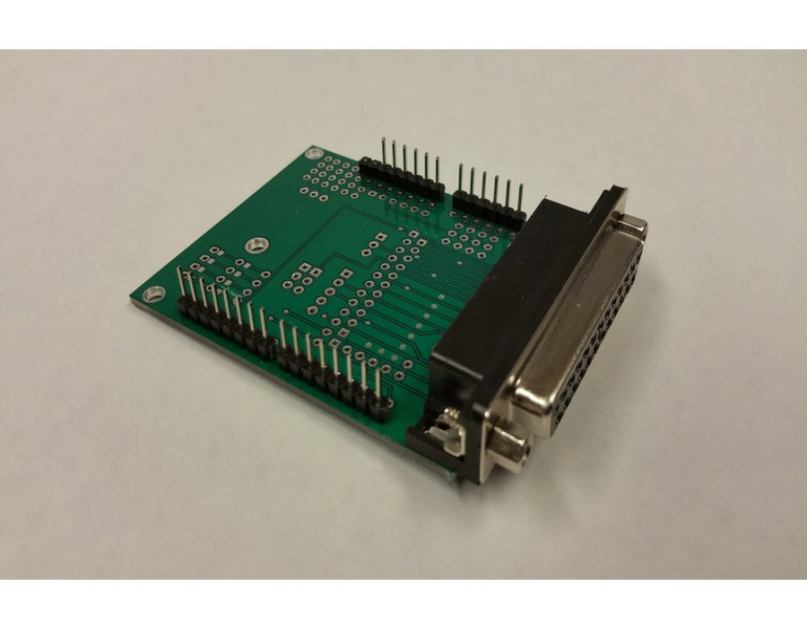
A Raspberry Pi egy bankkártya méretű, egyetlen áramköri lapra/kártyára integrált BCM2835 alapú egykártyás számítógép, amelyet az Egyesült Királyságban fejlesztettek oktatási célokra. Az eredeti két változat (A és B) kiadása óta már több továbbfejlesztése is kiadásra került. AA hivatalosan ajánlott operációs rendszer a laphoz a Raspbian, ami a Debian Linux kifejezetten Raspberry Pi-re optimalizált változata. A Legfrissebb modellje a 3. generációs Raspberry Pi 3, de időközben kiadtak még kisebb méretű Zero változatokat is. [4]



. ábra Raspberry PI 3B

* + 1. DB25 Shield

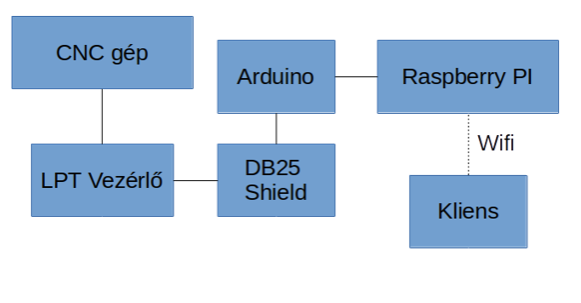
A DB25 Shield kifejezetten Arduino vezérlőhöz és CNC-gépnek a párhuzamos portját összekötő shield. Kis kereskedelmi forgalomban van és zárt hozzáférésű LightElectronics cég forgalmazza.



3. ábra DB25 Shield

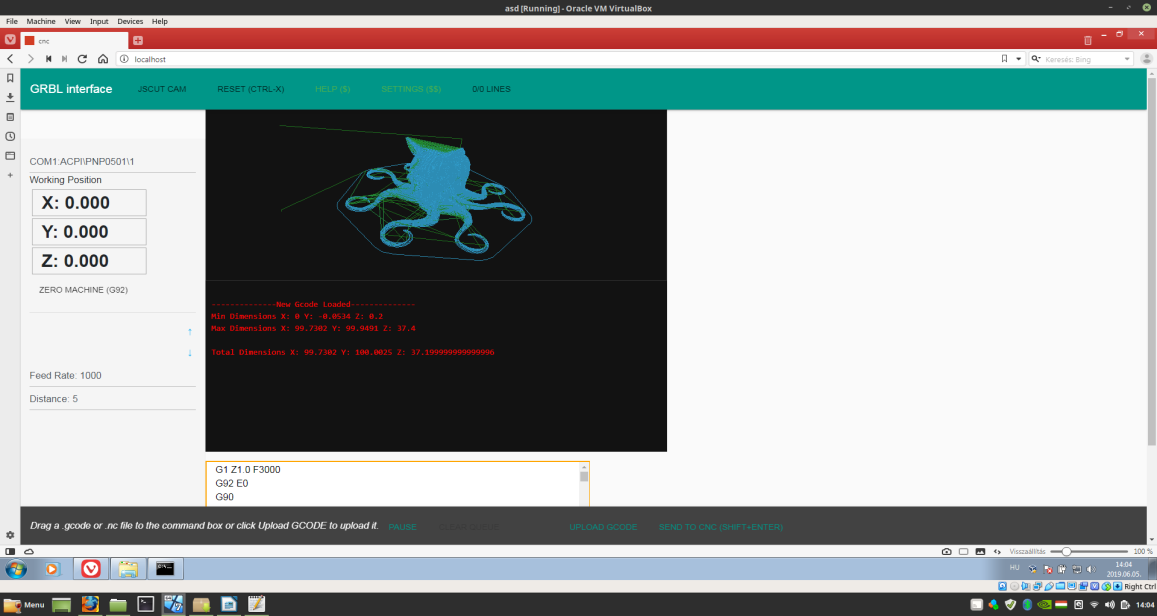
1. Problémamegoldás

Ebben a fejezetben fogom ismertetni a szakdolgozat megvalósítását. A szakdolgozat két részből áll. Az első maga a szerver elkészítése, a második a pedig a kliens alkalmazás elkészítése. Mivel mind a két rész igen terjedelmes és összetett ezért csak a fontosabb részeket fogom ismertetni. A többi rész megtalálható forráskóddal együtt a CD mellékletben. Valamint röviden bemutatom a fejlesztésre használt eszközöket is. Először is talán kezdjük a rendszer blokk vázlatával:



4. ábra A rendszer blokkvázlata

A marógép vezérlését biztosító Arduino-hoz csatlakozik a Raspberry PI 3 mikroszámítógép. A Raspberry szerepe a web server futtatása. Erre a web szerverre csatlakozik a felhasználó számítógépe vagy mobil eszköze, és ezzel lehetővé válik marógép távoli vezérlése is. A Raspberry mikroszámítógépen Linux Manjaro fut. A marógépet a /dev/ttyACM0 portra lett csatlakoztatva.



5. ábra A program működés közben

* + 1. A vezérlés lépései

A szerver az említett *Node.JS* technológiával fut az eszközön, vagyis a Raspberry Pi-n. Ez lesz a rendszer web szervere. Az Arduino a web szerverrel a *Serialport* nevű könyvtár segítségével, a soros porton keresztül kommunikál. Mi esetünkben ez *USB port*. Az eszköztől érkező adatokat, a Raspberry web szerverre egy statikus *HTML* weboldalon jeleníti meg. A kommunikációhoz a Socket.IO könyvtárat használjuk. A Socket.IO a kommunikáció adatait és a parancsokat is továbbítja az applikációnak, kliensnek. Ehhez hasonló megoldásokat a Real Time alkalmazásoknál szoktak használni.

A vezérléshez használt G kódot a számítógépről olvassuk be. A beolvasott G-fájlt ellenőrizni szükséges, ezért először egy G-code ellenőrző függvénnyel futtatjuk át. Helyes kód esetén a desktop alkalmazás kezelőfelületén megjelenik, a G-code viewer segítségével a kidolgozandó objektum 3D modellje, valamint maga a G kód is. A G-code ellenőrzés a kliensen-n és GRBL vezérlőn is végrehajtódik.

A G kódot tartalmazó szövegszerkesztő, a Socket.IO-n keresztül csatlakozik a marógéphez, ami annyit jelent, hogy a TextArea-ban szerkesztett kóddal tudjuk vezérelni a gépet. Ezzel megvalósul a marógép valósidőben történő vezérlése. Programon belül lehetőség van kézi irányításra is. A desktop alkalmazásban kurzor nyilakkal képesek vagyunk a mikro vezérlőt irányítani.

A gép mozgásának aktuális koordinátái a Three.JS-ben jelenik meg. Kód futtatásakor megjelenik a program aktuális állása is, valamint az összes parancs száma (hány százaléknál tart a megvalósítás).

* + 1. Web szerver

Web szerver a Raspberry Pi mikroszámítógépen fut. Segítségével jelennek meg a marógép fontosabb adatai, a vezérlés lépései, a program állapota a kliensen (PC-n), és ezen keresztül lehet táv vezérelni a gépet.

A web szerver felülete Node.JS-el dolgozik, amely egy statikus web szerver felületet add a rendszernek. Az alábbi kód fogadja a klienstől a G kódot, leellenőrzi, majd visszaküldi és megjeleníti magában a kliensben.

app.listen(config.webPort);

var fileServer = new static.Server('./fsserver');

function handler (req, res) {

console.log(req.url);

if (req.url.indexOf('/api/uploadGcode') == 0 && req.method == 'POST') {

// this is a gcode upload, probably from jscut

console.log('new data from jscut');

var b = '';

req.on('data', function (data) {

b += data;

if (b.length > 1e6) {

req.connection.destroy();

}

});

req.on('end', function() {

var post = qs.parse(b);

io.sockets.emit('gcodeFromJscut', {'val':post.val});

res.writeHead(200, {"Content-Type": "application/json"});

res.end(JSON.stringify({'data':'ok'}));

});

} else {

fileServer.serve(req, res, function (err, result) {

if (err) console.log('fileServer error: ',err);

});

}

}.

A kommunikáció párhuzamos a kliens és a szerver között: az elküldött kód ellenőrzött része megjelenik a kliensen.

* + 1. Soros kommunikáció

Az Arduino és a Raspberry között megvalósított soros kommunikáción keresztül töltődik fel a vezérlőkód a marógépre. A kapcsolat létesítéséhez, először is szükséges kilistázni a rendszerben található aktuális kommunikációs portokat.

serialport.list(function (err, ports) {

allPorts = ports;

for (var i=0; i<ports.length; i++) {

!function outer(i){

sp[i] = {};

sp[i].port = ports[i].comName;

sp[i].q = [];

sp[i].qCurrentMax = 0;

sp[i].lastSerialWrite = [];

sp[i].lastSerialReadLine = '';

sp[i].handle = new serialport.parsers.Readline({delimiter: '\r\n'});

// 1 means clear to send, 0 means waiting for response

sp[i].port = new serialport(ports[i].comName, {

baudRate: config.serialBaudRate

});

// write on the port

sp[i].port.pipe(sp[i].handle);

sp[i].sockets = [];

sp[i].port.on("open", function() {

console.log('connected at '+config.serialBaudRate, sp[i].port.path);

// loop for status ?

setInterval(function() {

//console.log('writing ? to serial');

sp[i].port.write('?');

}, 1000);

});

// line from serial port

sp[i].handle.on("data", function (data) {

//console.log('got data', data);

serialData(data, i);

});

}(i)

}

});

A G program sorainak küldéséhez szükséges szkript:

function sendFirstQ(port) {

if (sp[port].q.length < 1) {

// nothing to send

return;

}

var t = sp[port].q.shift();

// remove any comments after the command

tt = t.split(';');

t = tt[0];

// trim it because we create the \n

t = t.trim();

if (t == '' || t.indexOf(';') == 0) {

// this is a comment or blank line, go to next

sendFirstQ(port);

return;

}

//console.log('sending '+t+' ### '+sp[port].q.length+' current q length');

// loop through all registered port clients

for (var i=0; i<sp[port].sockets.length; i++) {

sp[port].sockets[i].emit('serialRead', {'line':

'<span class="color-write">SEND: '+t+'</span>'+"\n"});

}

sp[port].port.write(t+"\n")

sp[port].lastSerialWrite.push(t);

}

Mivel egyes karakterek (pl. #, “, ‘ stb.) HTML környezetben másképp szerepelnek ezért szükséges az átalakítás ANSII karakterekből HTML karakterekbe.

function ConvChar( str ) {

c = {'<':'&lt;', '>':'&gt;', '&':'&amp;', '"':'&quot;', "'":'&#039;',

'#':'&#035;' };

return str.replace( /[<&>'"#]/g, function(s) { return c[s]; } );

}

* + 1. Fájlbeolvasás

A fájlbeolvasást a kliens kezeli. A fájl beolvasását az alábbi szkript végzi el. A technológia jQuery támogatottsággal rendelkezik ez miatt a böngésző drag and drop funkcióval van ellátva.

if (window.FileReader) {

var reader = new FileReader ();

// drag and drop

function dragEvent (ev) {

ev.stopPropagation ();

ev.preventDefault ();

if (ev.type == 'drop') {

reader.onloadend = function (ev) {

document.getElementById('command').value = this.result;

openGCodeFromText();

};

reader.readAsText (ev.dataTransfer.files[0]);

}

}

document.getElementById('command').addEventListener ('dragenter', dragEvent, false);

document.getElementById('command').addEventListener ('dragover', dragEvent, false);

document.getElementById('command').addEventListener ('drop', dragEvent, false);

// button

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

fileInput.addEventListener('change', function(e) {

reader.onloadend = function (ev) {

document.getElementById('command').value = this.result;

openGCodeFromText();

};

reader.readAsText (fileInput.files[0]);

});

} else {

alert('your browser is too old to upload files, get the latest Chromium or Firefox');

}

});

Majd ha beolvasás megtörtént, akkor meghívásra kerül az OpenGcodeFromText függvény. Ez az a szkript mely meghívja Three.JS G-code viewer G kód megjelenítő szkriptjét. A createObject függvényt.

function openGCodeFromText() {

var gcode = $('#command').val();

if (document.hasFocus()) {

createObject(gcode);

//console.log('adding object with existing focus');

} else {

// wait for focus, then render

//console.log('waiting for focus');

$(window).bind('focus', function(event) {

createObject(gcode);

//console.log('focus exists');

// unbind for next object load

$(this).unbind(event);

});

}

}

* + 1. Az objektumok megjelenítése a képernyőn

A G kód megjelenítése a Three.JS könyvtár segítségével történik meg. Ehhez az előbb említett createObject függvényt használjuk. E parancs folyamán ellenőrzésre kerül a G kód szintaktikai helyessége. Ezt a G Code library segítségével valósítjuk meg. Ez mellett megjelenítjük a minta modelljét is.

var lastLine = {

x: 0,

y: 0,

z: 0,

e: 0,

f: 0,

// set to true for cnc, no extruder

extruding: true

};

var layers = [];

var layer = undefined;

var bbbox = {

min: {

x: 100000,

y: 100000,

z: 100000

},

max: {

x: -100000,

y: -100000,

z: -100000

}

};

function newLayer(line) {

layer = {

type: {},

layer: layers.length,

z: line.z,

};

layers.push(layer);

}

function getLineGroup(line) {

if (layer == undefined)

newLayer(line);

var speed = Math.round(line.e / 1000);

var grouptype = (line.extruding ? 10000 : 0) + speed;

var color = new THREE.Color(line.extruding ? 0x33aadd : 0x228B22);

if (layer.type[grouptype] == undefined) {

layer.type[grouptype] = {

type: grouptype,

feed: line.e,

extruding: line.extruding,

color: color,

segmentCount: 0,

material: new THREE.LineBasicMaterial({

opacity: line.extruding ? 0.8 : 0.8,

transparent: true,

linewidth: 2,

vertexColors: THREE.FaceColors

}),

geometry: new THREE.Geometry(),

}

}

return layer.type[grouptype];

}

function addSegment(p1, p2) {

var group = getLineGroup(p2);

var geometry = group.geometry;

group.segmentCount++;

geometry.vertices.push(new THREE.Vector3(p1.x, p1.y, p1.z));

geometry.vertices.push(new THREE.Vector3(p2.x, p2.y, p2.z));

geometry.colors.push(group.color);

geometry.colors.push(group.color);

//if (p2.extruding) {

// do this for all lines, not just extruding on cnc

bbbox.min.x = Math.min(bbbox.min.x, p2.x);

bbbox.min.y = Math.min(bbbox.min.y, p2.y);

bbbox.min.z = Math.min(bbbox.min.z, p2.z);

bbbox.max.x = Math.max(bbbox.max.x, p2.x);

bbbox.max.y = Math.max(bbbox.max.y, p2.y);

bbbox.max.z = Math.max(bbbox.max.z, p2.z);

//}

}

var relative = false;

function delta(v1, v2) {

return relative ? v2 : v2 - v1;

}

function absolute(v1, v2) {

return relative ? v1 + v2 : v2;

}

Természetesen a program nem támogatja az összes parancsot, köztük a görbére alkalmazott kódok, valamint a több tengely ábrázolását se.

* + 1. Program futtatása

A program a beolvasás után futtathatóvá válik, ahol soron lehet követni az aktuális sorszámot. A programot kliensben szerkeszteni is lehet.

A marási folyamat gyorsan megállítható a kliensen. Ennek megvalósításához az járul hozzá, hogy a szerver mindig csak az aktuális sort fogadja a klienstől, tehát nem a teljes kód kerül letöltésre továbbítva az Arduinónak. A direkt vezérléshez szükségünk lesz az egy-egy kapcsolathoz. A következő szkript a leállítást mutatja be.

$('#pause').on('click', function() {

if ($('#pause').html() == 'Pause') {

// pause queue on server

socket.emit('pause', 1);

$('#pause').html('Unpause');

$('#clearQ').removeClass('disabled');

} else {

socket.emit('pause', 0);

$('#pause').html('Pause');

$('#clearQ').addClass('disabled');

}

});

$('#clearQ').on('click', function() {

// if paused let user clear the command queue

socket.emit('clearQ', 1);

// must clear queue first, then unpause (click) because unpause does a sendFirstQ on server

$('#pause').click();

});

Az aktuális állás program megjelenítése a programban, a következő szkrip valósítja meg:

socket.on('serialRead', function (data) {

if ($('#console p').length > 300) {

// remove oldest if already at 300 lines

$('#console p').first().remove();

}

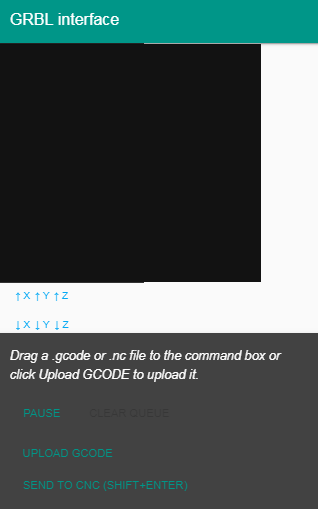
$('#console').append('<p>'+data.line+'</p>');

$('#console').scrollTop($("#console")[0].scrollHeight - $("#console").height());

});

* + 1. Kézi vezérlés

A kézi vezérlést a kurzor billenyűkkel lett megoldva, amíg telefon nézetben irányító gombok jelennek meg.



6. ábra Mobil felület

// WASD and up/down keys

$(document).keydown(function (e) {

var keyCode = e.keyCode || e.which;

if ($('#command').is(':focus')) {

// don't handle keycodes inside command window

return;

}

switch (keyCode) {

case 65:

// a key X-

e.preventDefault();

$('#xM').click();

break;

case 68:

// d key X+

e.preventDefault();

$('#xP').click();

break;

case 87:

// w key Y+

e.preventDefault();

$('#yP').click();

break;

case 83:

// s key Y-

e.preventDefault();

$('#yM').click();

break;

case 38:

// up arrow Z+

e.preventDefault();

$('#zP').click();

break;

case 40:

// down arrow Z-

e.preventDefault();

$('#zM').click();

break;

}

}).

Ehhez szükséges volt a CSS-t úgy testreszabni, hogy a dinamikusan változzon a felhasználónak az aktuális eszközére. Ez az úgynevezett responsive technológia, amely az eszköz képernyőjének paramétereithez megfelelő formájú felhasználói felületet hozz létre.

A Raspberry-n futó program az aktuális poziciót is mutatja valós időben. Ehhez az szükséges, hogy a program folyamatosan lekérdezze a GRBL vezérlőt az aktuális állásról. Ez alapján módosulnak a kliens felületen az adatok.

A kliens oldalon:

socket.on('machineStatus', function (data) {

$('#mStatus').html(data.status);

$('#mX').html('X: '+data.mpos[0]);

$('#mY').html('Y: '+data.mpos[1]);

$('#mZ').html('Z: '+data.mpos[2]);

});

A szerver oldalon:

var t = data.substr(1);

t = t.substr(0,t.length-2);

t = t.split(/,|:/);

emitToPortSockets(port, 'machineStatus', {'status':t[0], 'mpos':[t[2], t[3], t[4]]);

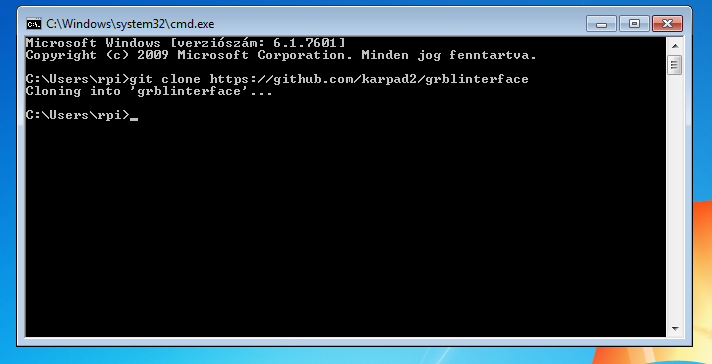
return;

}

* + 1. A szoftver telepítése

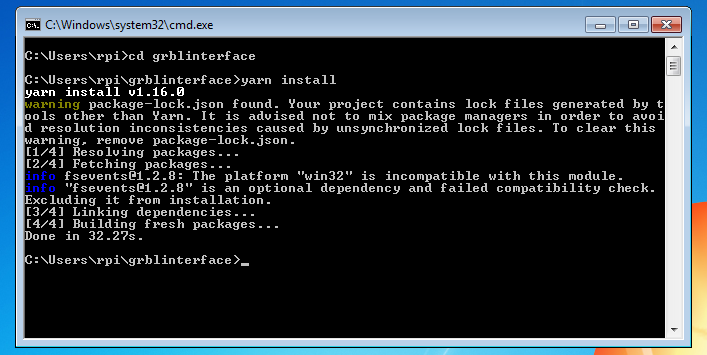
Először is szoftverhez szükséges függőségek:

* Node.JS (a Serialport könyvtár miatt 8.x verzió)
* Yarn alkalmazás
* git kliens



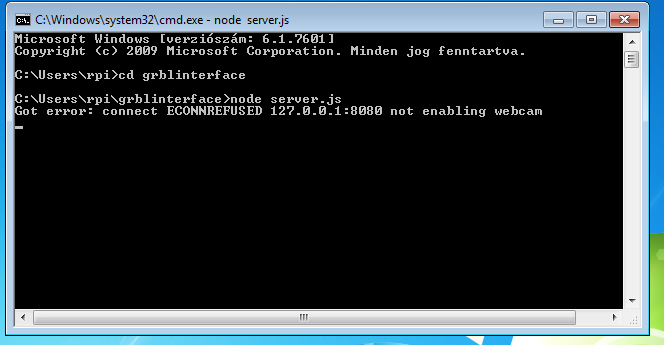
7. ábra Klónozás

Leklónozzuk a git tárolóból[5] az applikációt: Majd a Yarn a segítségével telepítjük a függőségeket:

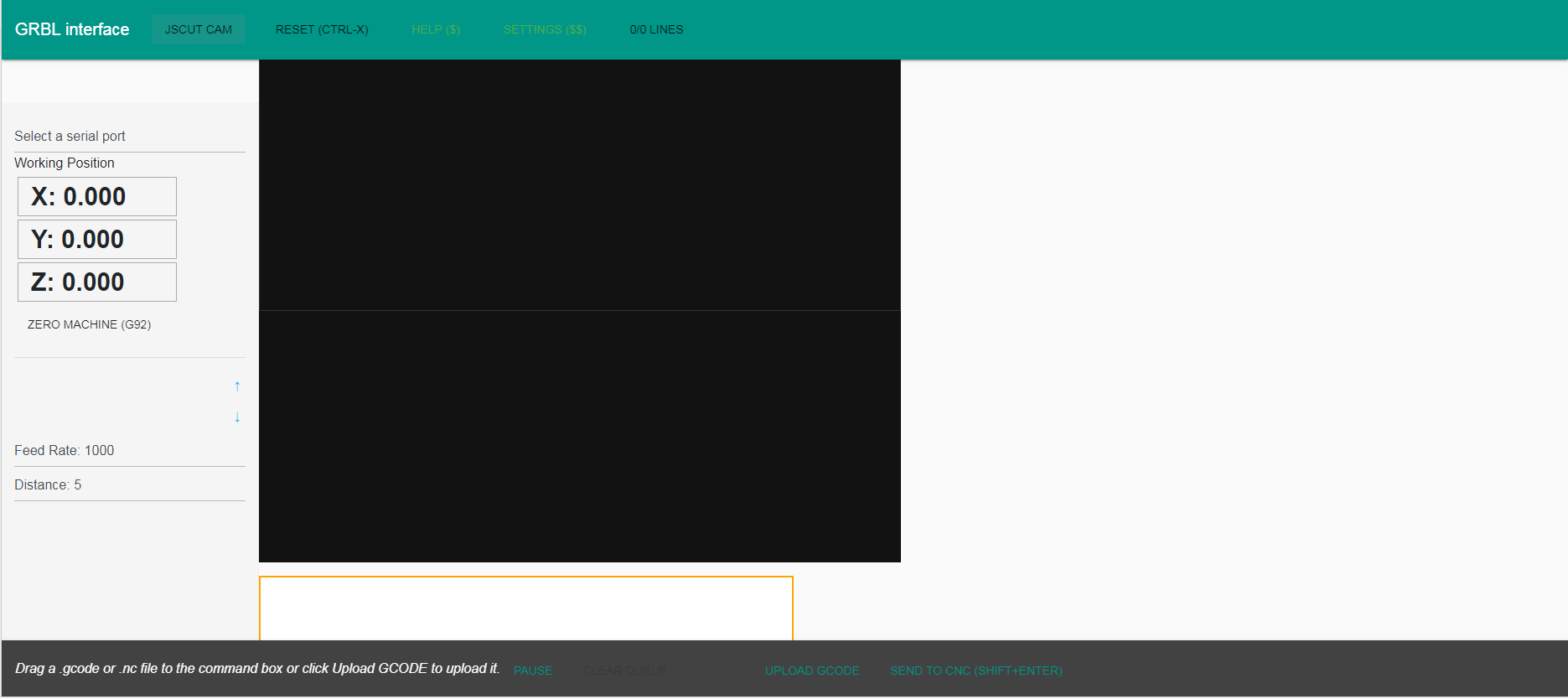


8. ábra A függőséggek telepítése

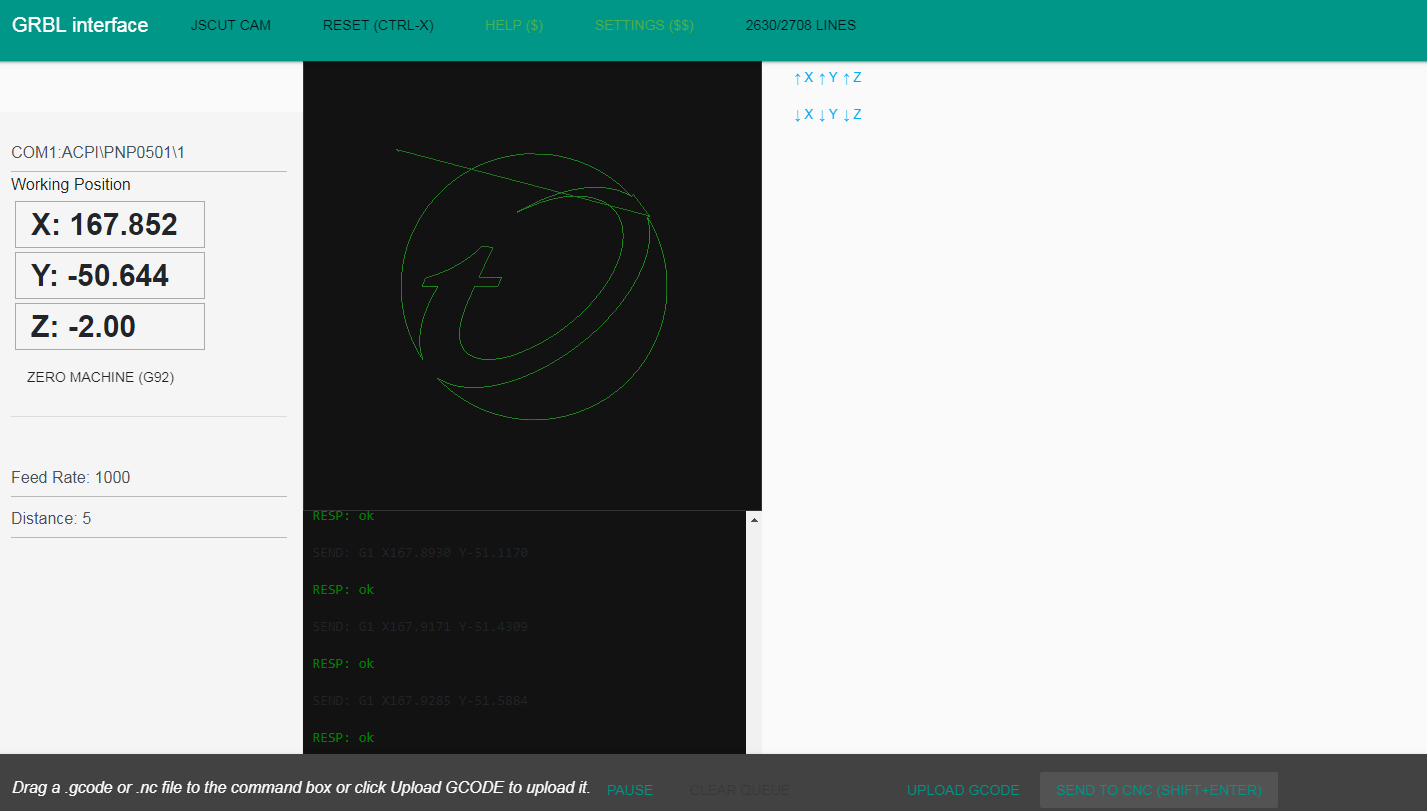
Az applikáció elindítása:



9. ábra A szerver applikáció futattás közben



10. ábra A kliens elinditás után



11. ábra A szoftver működés közben

* + 1. A felhasználói felület

A felhasználói felület építése során az egyszerűségre törekedtünk. Ezáltal minimalizálva a hiba lehetőségeket és növeltük az áttekinthetőséget.

A program több böngészőben is tesztelve lett: Vivaldi-ban, Firefox és Google Chrome, Opera-bán is fut.

A régebbi típusú böngészőket (pl. Internet Explorer 8 vagy 9) nem támogatja a szoftver, s azokat sem melyeknél nincs java script támogatottság.

1. Összegzés

Úgy vélem, hogy a szakdolgozatban elért célokat teljesítettük. Egy olyan rendszert sikerül megvalósítani, amely hardveres modulok mellett tartalmaz egy klienst és egy web alkalmazást. A kifejlesztett modulok tökéletesen működnek. Az Arduino modul a GBRL vezérlővel és G kódokkal megfelelő működést biztosít a marási folyamatban. A vezérlés kiegészítését jelentő, a Raspberry-n futó szerver és a web alkalmazás is olyan lehetőségekkel bővült, amelyeket a fejlesztés elején kitűztem:

- a marógépe vezérlő kód letölthető a kliens számítógépről;

- a marási folyamatot vizualizáltam;

- a vezérlés mobil eszközről is megvalósulhat;

A rendszer úgy lett kialakítva, hogy a jövőben bármikor tovább lehessen a fejleszteni azt. Erre a felhasználók javaslatokat is tehetnek a Githubon[5].

Irodalom

[1] G-code https://en.wikipedia.org/wiki/G-code

[2] <https://nodejs.org>, Node.JS alapok

[3] [www.inf.u-szeged.hu/miszak/utmutato-](http://www.inf.u-szeged.hu/miszak/utmutato-) Arduino – kezdő lépések

[4] <https://modi-j.webnode.hu/> - A Raspberry Pi számítógép

[5] Git tároló <https://github.com/karpad2/grblinterface>

Mellékletek

Nyomtatott melléklet

Elektronikus melléklet

CD mely tartalmazza a dokumentációt és összes forráskódot.