

Tartalomjegyzék

1. Bevezetés.....	3
2. Szakirodalom elemző feldolgozása.....	4
2.1 Szabaduló szoba.....	4
2.2 Firebase.....	4
2.3 Babel fordító könyvtár.....	6
2.4 A Blockly könyvtár.....	6
2.5 Raspberry PI 400.....	6
2.6 Manjaro ARM.....	6
2.7 NPM/Node.JS.....	7
2.8 GitHub.....	7
2.9 ESP32.....	7
2.10 RFID olvasó.....	8
2.11 OLED kijelző.....	8
2.12 Fejlesztőkörnyezet - Visual Studio Code.....	8
3. Saját munka.....	10
3.1 Firebase modell:.....	11
A szobák tárolása.....	12
Az eszközök kezelése.....	12
3.2 Hálózatkezelés:.....	17
3.3 Github Action:.....	17
3.4 Vue rész:.....	19
3.5 Szabaduló szoba programok programozása:.....	21
3.6 Kamerák kezelése:.....	34
3.7 Az Eventek kezelése:.....	34
3.8 Az eszközök kezelése:.....	35
3.9 A régebbi futamok megtekintése:.....	47
3.10 A szobák megjelenítése:.....	47
3.11 A lobby és a program futtatása:.....	51
3.12 Az ESP32 programja:.....	55
3.13 A regisztráció menete:.....	59
3.14 Az eszköz indulása:.....	60

3.15 Az eszköz bejelentkezése:.....	62
3.16 Relé üzemmód:.....	62
3.17 Bemenet üzemmód:.....	63
3.18 Megjelenítő üzemmód.....	64
3.19 RFID üzemmód.....	64
3.20 ESP32 OTA frissítés.....	65
3.21 Raspberry PI beállítása:.....	66
3.22 A szoftver futtatása Raspberry PI-n:.....	67
3.23 A szoftver tovább fejlesztési lehetőségei:.....	68
4. Összefoglalás.....	69
5. Idegen nyelvű tartalmi összefoglaló.....	70
6. Mellékletek.....	73

1. Bevezetés

A feladat egy olyan szabaduló szoba vezérlés létrehozása, mellyel akár a programot távolról is lehessen futtatni. A programnak támogatnia kell az ESP32 mikrovezérlőt, valamint tartalmazza, ábrázolja, és engedélyezze a szobák szoftverének szerkesztését.

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

A dokumentum célja a diplomamunka dokumentálása és leírása, a szakdolgozat fő célkitűzése volt, egy olyan szoftver készítése, mely a szabaduló szobáknak a létrehozásában és menedzselésében segít. A szoftvernek támogatnia kell, új szabaduló szobák létrehozását, ESP32 mikrovezérlő szoftverének vezérlését HTTP-protokollon, valamint a szobát fenntartó személyzet segítségét, a csapatok nyomon követését, és nem utolsósorban a szobák programozását.

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

2. Szakirodalom elemző feldolgozása

A szakdolgozat megvalósításához a következő technológiákat használtam fel:

- HTML5
- JavaScript
- Vue.JS
- Firebase
- Axios/Ajax
- Node.js/NPM
- CSS
- Material CSS
- Arduino IDE
- ESP32

A feladathoz az esp32 típusú mikrovezérlőt lett választva, a feladatot úgy oldottam meg, hogy a program, egy közös csatornán keresztül kommunikál a szerverként üzemelő számítógéppel, és ez úgy lett megvalósítva, hogy azonos hálózathoz, van kötve az összes kliensként dolgozó Raspberry PI 400.

A feladat megvalósításához Firebase, Vue, Blockly keretrendszereket használtam fel.

2.1 Szabaduló szoba

Szabadulósobák az utóbbi évtizedekben kezdtek elterjedni, lényege a szabaduló szobának, hogy egy olyan „Kazamatát” építsen fel, melyen a játékosok keresztül mennek, s rejtvényeket oldanak meg. Ezeknek a rejtvényeknek többféle verziója is van pl. lakat kioldása szám kombinációval, RFID-vel el látott objektumok az érzékelőre való helyezése stb. A célja a szoftvernek az időmérés illetve a játékmesterek segítése.

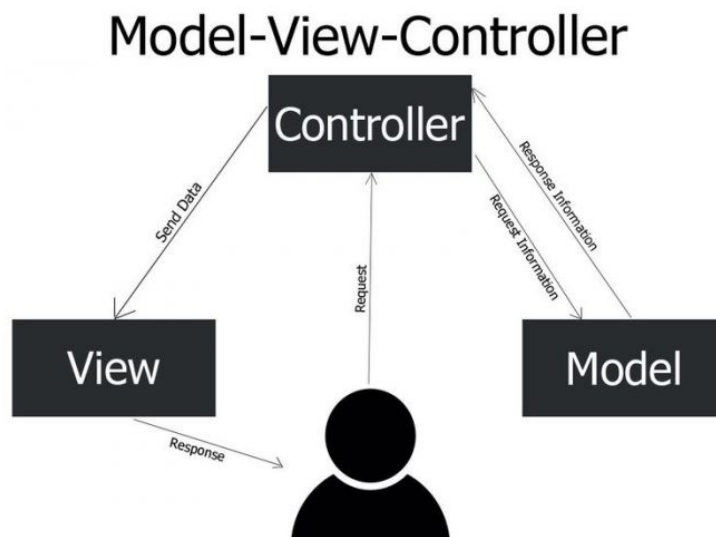
2.2 Firebase

A Firebase keretrendszer egy Google fejlesztése alatt álló adatbázis és hozzá tartozó rendszerek között használhatóak valós idejű adatbázisok, Collection-ok, valamint, Statikus adattárolás, Fájl tárolás, illetve Hosting-ra is van lehetőség. A Firebase egy úgynevezett Backend as Service alapú struktúrát követ mely segítségével az ilyen alkalmazások létrehozása, skálázott és a megfelelő használatban ingyenes is.

A Firebase adatmodellek a MongoDB-szerűen Node-okba tárolják el az adatokat. S ehhez megfelelően használtam fel én is. Én az adatbázissal való összekötésre, az úgynevezett MVC modellt használtam, mely alapján az adatot tároljuk módosítjuk, annak az adatját Kontrolleren kezeljük, illetve megjelenítjük.

Az adatbázis úgynevezett nem relációs adatbázisokon keresztül kommunikál a JavaScript felület, illetve a mikrokontroller.

A Firebase kezeli a felhasználók azonosítást is.



ábra 1 MVC modell

Ezek végeztével elkezdtem kidolgozni a feladathoz szükséges csomópontok elemeit, később feltelepítettem a Firebase könyvtárait a keretrendszerbe, e program segítségével elkezdtem elkészíteni felhasználói felületet, mely Vue.JS JavaScript támogatásával üzemel.

Vue 3.0 segítségével elkezdtem az egy oldalas alkalmazás létrehozását, mely csak Ajax használatával kommunikál a rendszerrel, a többi egy előre fordított JavaScript-en keresztül tölti be.

A modulokat létrehoztam, majd elkezdtem a weboldalhoz tartozó weboldalakat létrehozni. Ezek egy Vue.js könyvtárban találhatóak, melyeknek direkt kommunikációval rendelkeznek a weboldalhoz. Ezt egy úgy nevezett¹ Axios.JS csomagon keresztül intézi.

¹ https://medium.com/@joespinelli_6190/mvc-model-view-controller-ef878e2fd6f5

2.3 Babel fordító könyvtár

Ez a könyvtár lehetővé teszi újabb alkalmazások pl. (Vue, React, vagy Angular) típusú alkalmazások felhasználást régebbi böngészőre, valamint a kódot optimalizálja. Az így létrehozott Stílus és JavaScript fájl nem több pár megabiténál ezzel levéve a súlyos fileok mozgatását a szerverről.

Ezt használja fel a WebPack technológiának is.

2.4 A Blockly könyvtár

Google által fejlesztett grafikus programozási program, mely képes több nyelvre is lefordítani a lehelyezett elemeket, támogatott a php, JavaScript, go stb.

Az itt felhasznált része annyiban merül ki, hogy a felhasználó ebben rakja össze az ő általa írt programokat, és majd ezeket lementi és lefuttatja a kezelő felület.

Ehhez szükséges volt egy külön konzolt létrehozni, mely lehetővé teszi a felhasználóknak a Blockly által támogatott blokkok futtatását, valamint azok reakcióját a valós rendszerrel.

2.5 Raspberry PI 400

A szolgáltatás futtatására én a Raspberry PI 400-as mikroszámítógépet használtam fel. Ezt használja a felhasználó az eszköz kezelésére. 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. A 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.

2.6 Manjaro ARM

Manjaro ARM olyan operációs rendszer, mely a Raspberry PI 400-on fut. A Manjaro egy Arch típusú Linux disztribúció.

2.7 NPM/Node.JS

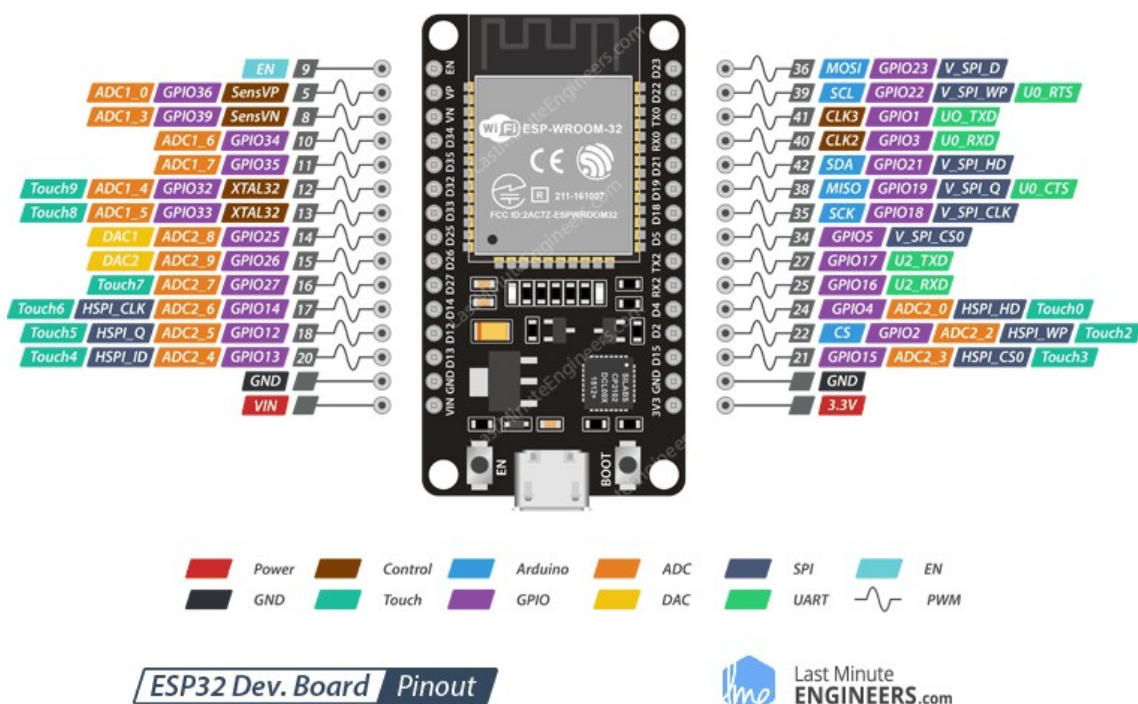
A rendszer az npm könyvtárkezelő segítségével építi fel magát.

2.8 GitHub

A szoftver szervesen támaszkodik GitHub adattárolóra, egy rész a fejlesztés itt történt, valamint a GitHub Action automata feltöltése a Firebase hosting-ra is itt történt, valamint az ESP32 is innen tölti le a legújabb verzióját az ESP32-nek.

2.9 ESP32

Az esp32 az Espress If által tervezett és gyártott mikrokontroller. Ezt használtam fel a feladat megoldásához, ez az eszköz kommunikál a Raspberry Pi-vel a Firebase API csatornáján és állítja be a szoftvert.



ábra 2 ESP32

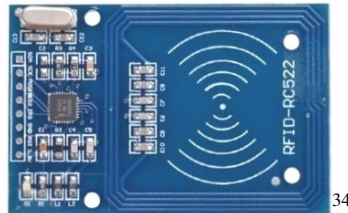
Az ESP32 támogat többféle protokollt, egyelőre 4 dolog lett kidolgozva, képes relé üzemmódban dolgozni, érzékeli azt és elküldi az RFID kártya ID-jét a szervernek, valamint képes bemeneti módban is dolgozni azáltal, hogy a megjelölt (25) portot rövidre zárjuk a

² <https://circuits4you.com/2018/12/31/esp32-devkit-esp32-wroom-gpio-pinout/>

földdel. Nem utolsósorban képes az eszköz megjelenítő eszközként is dolgozni amennyiben egy OLED kijelzőt kötünk rá.

2.10 RFID olvasó

Az RFID egy rádiós technológián alapuló azonosító eszköz. Különféle eszközök használhatnak RFID tag-eket. S az ESP32 is képes ilyenek olvasására az MFRC522 lappal.



ábra 3 MFRC522 RFID olvasó

2.11 OLED kijelző

Az OLED kijelző egy grafikus megjelenítő eszköz, melynek segítségével szöveget írhatunk ki az ESP32 eszközről.



ábra 4 OLED kijelző

2.12 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

³<https://github.com/playfultechnology/arduino-rfid-MFRC522>

⁴

⁵ https://www.rpibolt.hu/termek/096_128x64_oled_grafikus_kijelzo_-_monokrom_feher.html

- Automatikus kódformázás

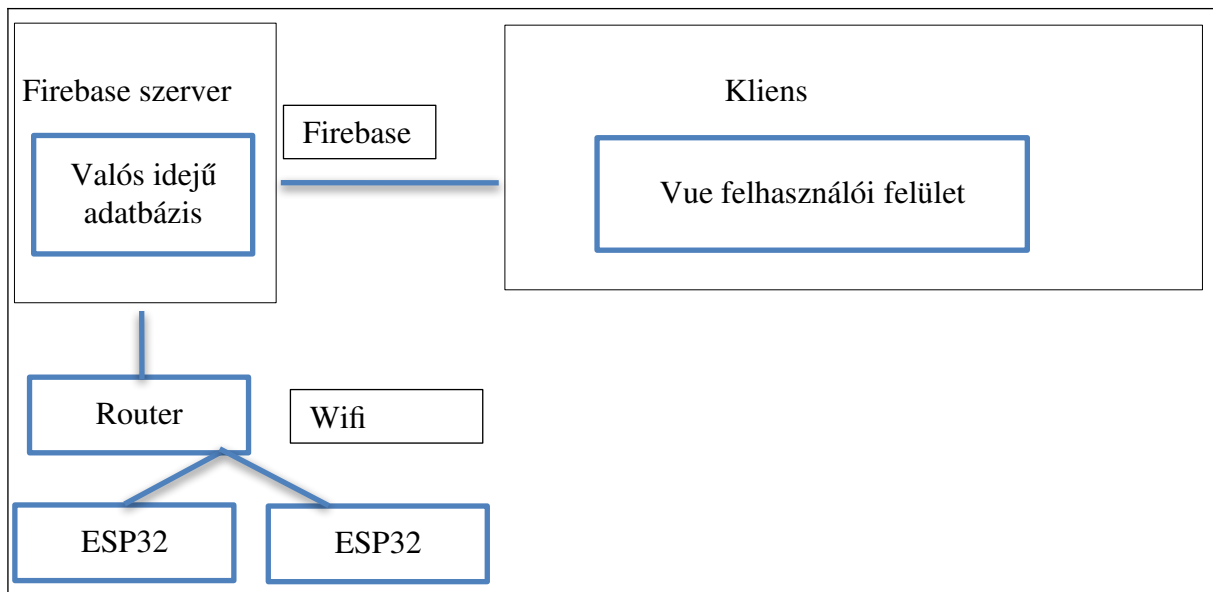
A Visual Studio Code azért is rendkívül jó IDE, mert rengeteg programozási nyelvet támogat. Csak, hogy néhányat említsek:

- JavaScript
- Node.js
- HTML
- CSS
- C#
- C++
- F#
- CoffeeScript
- TypeScript

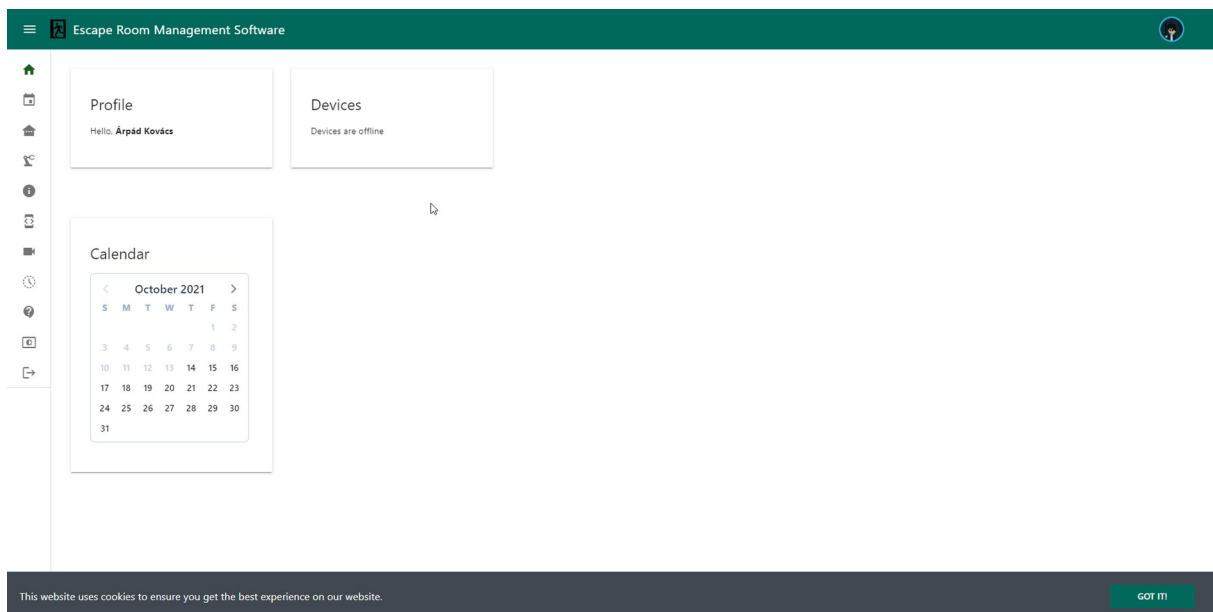
A szakdolgozat megvalósításánál ebben készítettem a JavaScript, Node.js, HTML és CSS részeket.

3. Saját munka

Ebben a fejezetben fogom ismertetni a szakdolgozat megvalósítását. A szakdolgozat két részből áll. Az első maga a szerver elkészítése, a második 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:



ábra 5 A rendszer blokkvázlata



ábra 6 Az alkalmazás cím oldala

A Firebase keretrendszert választottam majd, elkezdtem létrehozni a kezdő struktúrát, kezdve a felhasználók struktúrájával.

Ehhez a Vue 3.0 starterkit-jét telepítettem, mely támogatja a CRFS védelmet, valamint a botok elleni védelmet is, ennek a telepítő kódja:

```
npm init obudai_diplomamunka
npm install vue
npm run dev
```

3.1 Firebase modell:

Ez alapján kezdtem el kidolgozni az MVC modellt. Meg kell említeni, hogy a rendszer elsődlegesen a Firebase adatbázis kommunikációra támaszkodik adatkommunikációt használ, míg a program kinézetek előre vannak fordítva az app.js file-ba, a Babel könyvtár segítségével.

Modellekkel kezdtem.

A Firebase hozzáférés kezelése:

```
{
  "rules":{
    //".read": true,

    "users": {
      "$uid": {
        ".write": "$uid === auth.uid",
        ".read": "$uid === auth.uid",
        "rooms":
          {
```

```

    "$roomid":{
      "devices":
      {
        "$devId":{
          ".write": true,
          ".read": true,
        }
      }
    }
  }
}

```

A szobák tárolása

A szobákat a felhasználók alatt egy AuthID található /rooms Node-ban tárolja a rendszer.

Szobák tartalmazzák a legfontosabb adatokat, eszközöket, illetve az ehhez tartozó beállításokat is.

A szoba tárolja az eszközöket, szenzorokat, előző szabadulási időket, csapatokkal, bejegyzett időpontokat, kapcsolódási pontot (Wifi), illetve programokat, s ezek beállítását is.

Az eszközök kezelése

Ezt a hozzáférést egy ún. REST API-n keresztül történik. S majd ezt az API-t hozzáférésről a .json fájl kiterjesztésével lehet lekérdezni, s frissíteni.

Mely azt jelenti, hogy az alkalmazásnak HTTP kéréseken keresztül lehet elérni.

Ez azt jelenti, hogy ugye a biztonsági táblázat meghatározza az eszközöknek a hozzáférést, s hogy adatmódosításának biztonsági rést alkot, ha valaki tudja, az eszköz ID-jét. ezért éreztem szükségesnek még a szobának az ID-jébe is beágyazni, s ezáltal lehet csak az eszköz állapotát módosítani.

Azért választottam a Rest API-t nem az MQTT protokoll mert egyszerűbbnek éreztem az implementálását többféle eszközre, ezzel a megoldással akár, PLC-t is lehet kapcsolgatni, mivel egy JSON kódolva kapja meg az adatokat így egyszerűbb ennek a lekérdezési formátuma.

Az eszközök az esp32-k programozhatóak weboldalról is, ezt egy ún. Serial.JS könyvtár teszi lehetővé mely segítségével a soros porton keresztül az alkalmazás ellenőrzi, hogy a program aktív, valamint beállítja a kérdezési útvonalat pl.

```
https://escaperoom-b4ae9-default-rtdb.europe-west1.firebaseio.com/users/6ZMB5XNK1gd2PsJA8ae8vJs9jkP2/rooms/-MibGQVxewjkAo66aY67/devices/-Mieerw2m_Ess8--VHdF.json
```

Majd ennek végeztével, egy HTTP lekérdezést csinált 3 másodpercenként. Érdemes megemlíteni a Firebase korlátjait ebben az esetben. Mivel a Firebase az egy használat után emelkedik az ára én az ingyenes csomagot használtam, ezért 5 GB adatforgalom van megengedve havonta a valós idejű adatbázisra.

Itt frissíti a bázisban található időt közvetlenül lekérdezés után, sajnos ezt az eszköznek kell végrehajtania, mivel a bázis csak a prémiumnál engedélyezi a “belső szkript futtatását”.

Így az eszköznek kell frissítenie az órát mely eltérhet a Belgiumi időzónától ezért szükséges az ESP32 még, mindennek előtt egy NTP szerverrel összehangolni, mely segítségével ilyen probléma nem fordul elő.

A programok tárolása:

A programokat a szoftver a szoba Room - nodeban tárolja a programs Node alatt.

itt is a felhasználók korlátlanul tudnak hozzáadni a programokat, amelyeket a Blockly felülettel hoznak létre. Azért jó ez a “hierarchikus” elrendezés mivel így az adatok minimálisan keverednek, de viszont létrejöhetnek duplikátumok ugyanazon adatokból.

A programokat mentés után, mely automatikusan történik változtatás után, a program BASE64-be lekódolja és ez kerül bele a “program_xml” mezőbe. Ezért arra volt szükség, hogy a szoftver ne tároljon két másolatot ugyanarról a szoftver, egy JavaScript alapút, s egy XML alapú.

E kódolás változtatható, mivel a program tárolja azt, hogy mivel lett kódolva, s ez alapján vissza is lehessen oldani, de ehhez a felhasználó nem fér hozzá.

A programokat lehet biztonsági menteni is, ilyenkor a program a teljes Node tartalmát, egy JSON file-ba kiírja és letölteti a felhasználóval.

Ezt a fájlt ugyanígy vissza is lehet tölteni.

Az aktív programnak kiválasztását az Active_program cella tárolja, itt található blokkba a felhasználó kijelöli az Aktív programot, s ennek a programnak az ID-jét, a szoftver beírja a cellába.

Ezek végeztével elkezdtem kidolgozni a nézeteket is. Ezeknek a használatához igénybe vettem a Vue.JS könyvtárat. Majd párhuzamosan kidolgoztam az útvonalat (Route-t) melyeket VueRoute modullal hoztam létre és a hozzá tartozó nézetet. Mivel a cél az egy oldalas alkalmazás készítése volt, ezért a rendszer technológia Stackje megfelel már az egy oldalas feltételnek.

Ezt a modult feltelepítve létrehoztam a következő rendszert.

A vue route javascript file tartalma:

```
import Vue from 'vue';
import VueRouter from 'vue-router';
import {FirebaseAuth} from '../firebase';
Vue.use(VueRouter);

const router = new VueRouter ({
  mode: 'history',
  base: '/',
  routes: [
    {
      path: '*',
      name: 'error404',
      component: Error404,
    },
    {
      path: '/',
```

```
component: Index,
meta: {
  requiresAuth: true
},
children: [
  {
    path: 'home',
    name: 'home',
    component: Home,
  },
  {
    path: 'rooms',
    name: 'rooms',
    component: Rooms,
  },
  {
    path: 'devices',
    name: 'devices',
    component: Devices,
  },
  {
    path: 'programs',
    name: 'programs',
    component: Programs,
  },
  //
  {
    path: 'cameras',
    name: 'cameras',
    component: Cameras,
  },
  {
    path: 'pruns',
    name: 'pruns',
    component: PreviousRuns,
  },
  {
    path: 'room',
    name: 'room',
    component: Room,
  },
  {
    path: 'info',
    name: 'info',
    component: Info,
  },
],
```

```

    {
      path: 'logout',
      name: 'logout',
      component: Info,
    },
    {
      path: 'events',
      name: 'event',
      component: Events,
    },
    {
      path: 'support',
      name: 'support',
      component: Support,
    },
    {
      path: '/room/:rid',
      component: Room,
    },
    {
      path: '/room/:rid/device/:did',
      component: Device,
    },
    {
      path: '/room/:rid/program/:pid',
      component: Program,
    },
    {
      path: '/room/:rid/event/:eid',
      component: Event,
    },
    {
      path: '/room/:rid/lobby',
      component: Lobby,
    },
    {
      path: '/room/:rid/camera/:cid',
      component: Camera,
    },
    {
      path: '/room/:rid/prun/:prid',
      component: PreviousRun,
    },
  ]
},

```



```

    {
      path: '/account',
      component: AccountIndex,
      children: [
        {
          path: '',
          name: 'account-info',
          component: AccountInfo,
        },
        {
          path: 'login',
          name: 'account-login',
          component: AccountLogin,
        }
      ]
    }
  ]
});

router.beforeEach((to, from, next) => {
  let currentUser = FirebaseAuth.currentUser;
  let requiresAuth = to.matched.some(record => record.meta.requiresAuth);
  if(requiresAuth && !currentUser) next('/account/login');
  else next();
})

export default router;

```

Ha a felhasználó nem jelentkezett be automatikus visszautasítja a rendszer a bejelentkezési képernyőre.

3.2 Hálózatkezelés:

A hálózati összekötésre a REST API-t használja a rendszer. S azonnal frissíti a rendszer magát.

Mivel a böngészők automatikusan lekezelik a https kérését, de azonban az ESP32-nek szükséges az Igazolás az eszközön, hogy rajta legyen. Ezt szükséges volt egy böngészőből importálni. S ezt a fájlt az ESP32 SPIFFS fájlrendszerébe lementeni.

3.3 Github Action:

A github Action egy speciális szkript, amely lehetővé teszi, 1 bizonyos műveletek végrehajtását, időközönként, vagy valamilyen akcióra reagáljon. Ebben az esetben Push relációra automatikusan

lefordítja ezáltal ellenőrizve, hogy van-e benne hiba és ha nincs benne hiba akkor, deploy-olja a weboldalt a következő címre:

<https://escaperoom-b4ae9.web.app/>

name: Deploy to Firebase Hosting on merge

'on':

push:

branches:

- main

jobs:

build_and_deploy:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v2

- run: echo '\${{ secrets.CREDENTIALJS }}' > ./src/firebase/credentials.js

- run: npm install && npm run build

- uses: FirebaseExtended/action-hosting-deploy@v0

with:

repoToken: '\${{ secrets.GITHUB_TOKEN }}'

firebaseServiceAccount:
{ {secrets.FIREBASE_SERVICE_ACCOUNT_ESCAPEROOM_B4AE9 } }

'\$

channelId: live

```
projectId: escaperoom-b4ae9
```

Az ESP32 felöli autoupdate:

Ha Pusholás történik akkor a Github egy virtuális gép segítségével automatikusan lefordítja s feltölti az új bin fileoka.

3.4 Vue rész:

A Vue kinézeti fileok a következő képen épülnek fel:

```
<template>
Itt található a Vue elemekből összeállított modell (Html)
</template>
<script>
Itt funckionalitást rakjuk össze.(JS)
export default {
  components: {
  },
  data: ()=>
  {
    Itt találhatóak program változói,
  },
  methods:
  {
    program()
  },
  mounted: mounted()
  {
    Meghíváskor lefutó szkript
  }
}
</script>
<style>
Míg a stilusban kinézeti css-t állítjuk be. (CSS)
</style>
```

A felhasznált szkript a következő képen vannak eltárolva:

- Komponensek: több azonos modult felhasználó elemeket tárolok, itt vannak Vue eredeti filéja is pl. Checkbox.vue.
- Az oldalak: Létre van hozva egy Components nevű mappa ott találhatóak Vue oldalak, ami komponensekből áll.
- Rendszert konfiguráló beállítások: Ezek azok a fileok, amik meghatározzák az alábbi fileok működését, pl. Itt van meghatározva, hogy az elemek Tailwind css-t használnak.

A program kiinduló pontja:

```
import Vue from 'vue';
import router from './router';
import ForkeMeOnGithub from 'fork-me-on-github-vue';
import { IconsPlugin } from 'bootstrap-vue';
import BootstrapVue from "bootstrap-vue";
import VueNoty from 'vuejs-noty';
import VCalendar from 'v-calendar';
import AudioRecorder from 'vue-audio-recorder';
import VueMaterial from 'vue-material'; // TODO: import only needed component, not all
import 'vue-material/dist/vue-material.min.css';
Vue.use(VueMaterial);
//Vue.use(ForkeMeOnGithub);

Vue.use(AudioRecorder);
Vue.use(BootstrapVue);
import "bootstrap/dist/css/bootstrap.css";
import "bootstrap-vue/dist/bootstrap-vue.css";

Vue.use(IconsPlugin);
Vue.use(VCalendar, {})

Vue.use(VueNoty, {
  timeout: 4000,
  progressBar: true,
  layout: 'topRight'
});

import App from './App.vue';

let app = null;
Vue.config.productionTip = false;

import { getAuth, onAuthStateChanged } from "firebase/auth";
```

```

const auth = getAuth();
onAuthStateChanged(auth, (user) => {
  if (!app) {
    app = new Vue({
      router,
      render: h => h(App)
    });
    //app.use(BootstrapIconsPlugin);
    app.$mount('#app');
  }
});

```

3.5 Szabaduló szoba programok programozása:

A programnak szükséges támogatnia a Blockly könyvtárat, melynek fő használata program.vue fájlban belül található. A Blockly segítségével grafikusán lehet programozni.

S itt ez a szkript akkor hatódik le amennyiben a program a szoba lobby fázisba kerül a Blockly lefordítja az eltárolt Base64-be letárolt xml-t lefordítja JavaScript formátumba.

S A startra egy ún. Sandbox-ba rakja.

```

<template>
<div class="center">
  <h2>Program ~ {{ program.program_name }} </h2>

  <md-field>
    <label>Program name:</label>
    <md-input @change="namechange" v-model="program.program_name"></md-input>
  </md-field>
</div id="blocklyDiv"></div>
<Blocks :devices="devices" />

<md-button class="md-raised md-primary" @click="duplicateprogram()">Duplicate program</md-button>
<md-button class="md-raised md-secondary" @click="get_config()">Download Program</md-button>

<md-field>
<md-button class="md-raised md-secondary" @click="showDeleteDialog = true">Delete Program</md-button>

```

```

</md-field>
<md-dialog-confirm
  :md-active.sync="showDeleteDialog"
  md-title="Delete this program?"
  md-content="Your program will be deleted."
  md-confirm-text="Continue"
  md-cancel-text="Cancel"
  @md-cancel="onCancel()"
  @md-confirm="delete_pr()" />

</div>
</template>

```

A program szkriptje:

```

<script>
import * as Blockly from 'blockly/core';
import * as En from "blockly/msg/en";
import BJavascript from "blockly/javascript";

import 'blockly/blocks';
import Blocks from "@components/parts/Blocks";
import { send_data, devices, init, get_data, send_finish } from "@components/BlocklyJS";
//import { media } from "blockly/media";
import { FireDb, FirebaseAuth, userId } from "@firebase";
import { ref, set, onValue, get, child, push, runTransaction } from "firebase/database";
import { add_program } from "@mod_data/set_data";
import { get_encoding } from "@mod_data/get_data";
import { delete_program } from "@mod_data/del_data";
import { encode, decoding } from "@datas";
import { saveTextAsFile } from "@file";

export default {
  name: 'Programs',
  data: () => ({
    program: {},
    showDialog: false,
    a_program_name: "",
    a_program_xml: "",

```

```

a_program_javascript: "",
camera:{},
devices:[],
showDeleteDialog:false,
Workspace:null,
prev:"",
js_code:""
}),
comments:{
  Blockly,
},
components:
{
  Blocks
},
methods: {
myUpdateFunction(event) {},
save() {
  if (this.program.program_javascript != "") {
    const userId = FirebaseAuth.currentUser.uid;
    let _ref= ref(FireDb, `/users/${userId}/rooms/${this.$route.params.rid}/programs/${this.$route.params.pid}/program_xml`);
    set(_ref,this.program.program_xml);

  }
  console.log("WELP");
},
delete_pr()
{
  delete_program(this.$route.params.rid,this.$route.params.pid);
},
onCancel () {
  //this.value = 'Disagreed'
},
achange()
{
  if(this.prev==this.a_program_xml) return;
  const userId = FirebaseAuth.currentUser.uid;
  //console.log(this.device.mode);
  let _ref= ref(FireDb,
`/users/${userId}/rooms/${this.$route.params.rid}/programs/${this.$route.params.pid}/program_xml`);
  this.program.program_xml=encode(this.a_program_xml,get_encoding(this.$route.params.rid,this.$route.params.pid));
  set(_ref,this.program.program_xml);
  this.prev=this.a_program_xml;
  this.$noty.success("Saved!");
},

```

```

    namechange(){
        const userId = FirebaseAuth.currentUser.uid;
        let _ref= ref(FireDb,
`/users/${userId}/rooms/${this.$route.params.rid}/programs/${this.$route.params.pid}/
program_name`);
        set(_ref,this.program.program_name);
    },

    auto_compile() {
        console.log("im here");
        //this.program.program_javascript = Blockly.Javascript.workspaceToCode(this.Workspace);
        let xml = Blockly.Xml.workspaceToDom(this.Workspace);
        console.log(xml);
        if (xml !== "") {
            this.a_program_xml = Blockly.Xml.domToText(xml);
            console.log(this.a_program_xml);
            this.achange();
        }
    },
    start()
    {

        //let blocklyDefault = document.getElementById("blocklyDefault");

        //media: "/media/",
        if(this.program.program_xml==undefined||this.program.program_xml==null) return;
        console.log(get_encoding(this.$route.params.rid,this.$route.params.pid));
        this.a_program_xml=decoding(this.program.program_xml,get_encoding(this.$route.params.rid,this.
$route.params.pid));

        let workspace_default = Blockly.Xml.textToDom(this.a_program_xml);
        Blockly.Xml.appendDomToWorkspace(workspace_default,this.Workspace);

        this.auto_setup();
    },
    duplicateprogram()
    {
        add_program(this.$route.params.rid,this.program.program_name,this.program);
        this.$noty.success("Success!");
    },
    get_config()
    {
        let filename="program.json";

```



```

        let l={ data:this.program};
        saveTextAsFile(JSON.stringify(l),filename);}
    },
    mounted() {
        Blockly.setLocale(En);
        this.Workspace = Blockly.inject("blocklyDiv", {
            toolbox: document.getElementById("toolbox"),
            scrollbar: false,
        });

        this.Workspace.addChangeListener(() => {
            this.auto_compile(this.Workspace);
        });
        console.log(this.$route.params);
        //localStorage.setItem('device',JSON.stringify(null));
        const userId = FirebaseAuth.currentUser.uid;
        const devId = this.$route.params.pid;
        const room_id=this.$route.params.rid;

        onValue(ref(FireDb,
`/users/${userId}/rooms/${room_id}/programs/${devId}`),(sn)=>{
            if(sn.exists())
            {
                this.program=sn.val();
                //this.select=this.device.mode;
            }
        });

        onValue(ref(FireDb, `/users/${userId}/rooms/${room_id}/devices`),(sn)=>{
            if(sn.exists())
            {
                sn.forEach((a)=>{
                    this.devices.push(
                        {
                            devID:a.key,
                            data:a.val()
                        }
                    );
                });
            }
            this.start();
        });
    },
}

```

Stílusa a kinézetnek:

```

</script>
<style lang="scss" scoped>
#blocklyDiv
{
  width: 800px;
  height: 480px;
}
#text_programjavascript
{
}
#text_programxml
{
  display: none;
}
</style>

```

Mindegyik Blockot definiálni kell, ez alapján jelenhet meg az adott blocknak a kódja a fordítóban illetve a Sandboxban is.

Pl. A Blocks.vue fájlban találhatóak az elérhető blokkok melyek között ott vannak a gyári logikai blokkok is (IF, While, For return, String kezelő blokkok).

```

<category name="Blocks Library" colour="%{BKY_MATH_HUE}">

  <block type="start_room"></block>

  <block type="send_data"></block>

  <block type="get_data"></block>

  <block type="send_finish"></block>

</category>

```

Ezeknek a blokkoknak a leírását egy külön Javascript file foglalkozik melyben csak megvannak hívva és külön fájlban vannak pontosan meghatározva az egyszerűség kedvéért.

```
import * as Blockly from 'blockly/core';

import * as En from "blockly/msg/en";

const send_data= require("./send_data");

const devices= require("./devices");

const init= require("./init");

const get_data= require("./get_data");

const send_finish= require("./send_finish");

const start_room = require("./start_room");

import Blocks from "@components/parts/Blocks";


export {

  send_data,

  devices,

  init,

  get_data,

  send_finish,

  start_room

}
```

get_data fájl tartalma:

```
import * as Blockly from 'blockly/core';
```

```
import "blockly/javascript";
```

```
Blockly.defineBlocksWithJsonArray([
```

```
{
```

```
  "type": "get_data",
```

```
  "message0": "get device %1 %2",
```

```
  "args0": [
```

```
    {
```

```
      "type": "input_value",
```

```
      "name": "device",
```

```
      "check": ["String"]
```

```
    },
```

```
    {
```

```
      "type": "input_value",
```

```
      "name": "mode",
```

```
      "check": ["String"]
```

```
    },
```

```
  ],
```

```
  "message1": "Then, do %1",
```

```
  "args1": [
```

```
    {
```

```
      "type": "input_statement",
```

```
      "name": "DO0"
```

```

    },

    ],

    "previousStatement": true,

    "nextStatement": true,

    "inputsInline": true,

    "colour": '#0ddb69',

    "tooltip": "",

    "helpUrl": "",

  },

  ]);

Blockly.JavaScript['get_data'] = function(block) {

  var device = Blockly.JavaScript.statementToCode(block, 'device');

  var mode = Blockly.JavaScript.statementToCode(block, 'mode');

  var code = `get_data(${device},${mode});`;

  return [code, Blockly.JavaScript.ORDER_FUNCTION_CALL];

```

Amit a rendszeren keresztül meg kap a vue.js.

```

function encode(a_program_xml,format="base64")
{
  if(format=="base64")
  {
    let encoding=CryptoJS.enc.Utf8.parse(a_program_xml);

```

```

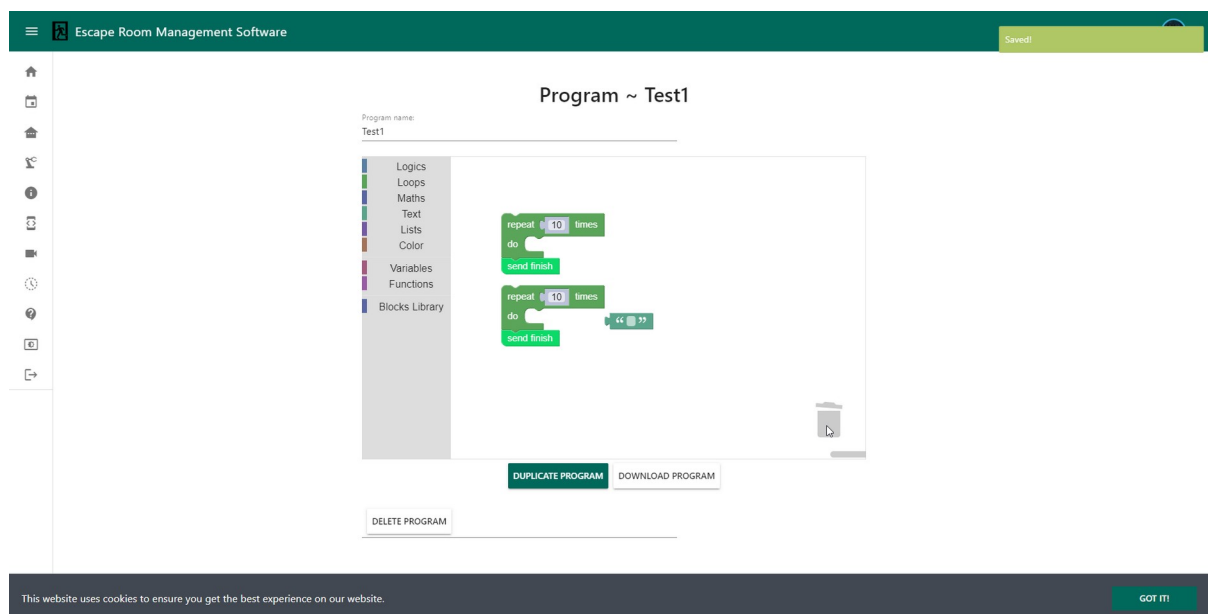
        return CryptoJS.enc.Base64.stringify(encoding);
    }
    else return ""; //
}
function decoding(program_xml,format="base64")
{
    if(format=="base64")
    {
        let encodedWord = CryptoJS.enc.Base64.parse(program_xml); // encodedWord via Base6
4.parse()
        return CryptoJS.enc.Utf8.stringify(encodedWord);
    }
    else return ""; //
}

```

A program átadja az xml, és a javascript változatát a programnak, ezt a Blockly felhasználja és a Vue beilleszti a két kijelölt textarea html tag-be. Minden egyes változtatásnál a felületen a Blockly.JS nevű könyvtár legenerálja az xml, valamint a Javascript változatot az aktuális beállításhoz képest.

Változás esetére a blockly automatikus lefordítja Base64-be lekódolja az xml verziót, s ezt feltölti a szobába a Firebase könyvtár segítségével.

A program nem támogatja a kódok kézi hozzáadását.



ábra 7 Program szerkesztő

És az alábbi szkript amit a program generált base64-be lekódolt xml-struktúra.

Programok futtatása:

A programok futtatása az úgynevezett lobby.vue fájlban található meg. Ez gyakorlatilag egy Sandbox-ot hoz létre az aktuális futó szkriptje, és ebben fut a leadott program is. Először is létrehozza az aktuális „futamot”, melynél elmenti a programot, a csapat id-t, a kezdeti dátumot, és NULL értéket ad a befejezési dátumnak (innen tudjuk, hogy aktív a futam).

A programot betöltés után a mounted függvény megnyitja és a Acorn fordító Sandbox-ába téve futtatja le. Szükséges megemlíteni, hogy a homokozóban 10 000 maximális lépés van meghatározva.

A programnak van egy belső konzolja amivel „kommunikál” az Axios keretrendszeren keresztül a perifériákkal, és ezek segítségével állítja át az ESP32-k tulajdonságait, pl.

```
beforeMount()
{
  this.status=status_run();
  localStorage.setItem("room_id",this.$route.params.rid);
  localStorage.setItem("user_id",FirebaseAuth.uid);
},
```

```

mounted()
{
  this.get_data();
  // console.log(this.room);
  console.log(this.sounds)
  Blockly.setLocale(En);
  const room_id=this.$route.params.rid;
  localStorage.setItem("roomID",room_id);
  this.cameras=get_data_fromroomdb(room_id,"cameras");
  this.devices=get_data_fromroomdb(room_id,"devices");
  this.Workspace = Blockly.inject("blocklyDiv", {
    toolbox: document.getElementById("toolbox"),
    scrollbar: false});
  Blockly.JavaScript.INFINITE_LOOP_TRAP = null;
  this.a_xml=decoding(this.program.program_xml,get_encoding(this.$route.params.rid));
  console.log(this.a_xml);
  let workspace_default = Blockly.Xml.textToDom(this.a_xml);
  Blockly.Xml.appendDomToWorkspace(workspace_default,this.Workspace);
  this.a_js = Blockly.Javascript.workspaceToCode(this.Workspace);

},

```

S ezt használja ki a Blockly-ba írt program is.

```

import * as Blockly from 'blockly/core';
import "blockly/javascript";
import axios from 'axios';

Blockly.defineBlocksWithJsonArray([
  {
    "type": "send_data",
    "message0": "set device %1, mode: %2, status: %3",
    "args0": [
      {
        "type": "field_variable",
        "name": "device",
        "check": ["String"]
      },
      {
        "type": "field_dropdown",
        "name": "mode",
        "variableTypes": [""],
        "options": [
          ["Relay", "relay"],
          ["RFID reader", "rfid"],

```



```

        ["Input", "input"],
        ["OLED Display", "oled"],
    ]
},
{
    "type": "input_value",
    "name": "value",
    "check": ["String"]
}
],
"inputsInline": true,
"previousStatement": true,
"nextStatement": true,
"colour": 160,
"tooltip": "",
"helpUrl": "",
},
]);
Blockly.JavaScript['send_data'] = function(block) {
    var device = Blockly.JavaScript.statementToCode(block, 'device');
    var mode = Blockly.JavaScript.statementToCode(block, 'mode');
    var value = Blockly.JavaScript.statementToCode(block, 'value');
    var code = `set_data(${device},${mode},${value});\n`;
    return [code, Blockly.JavaScript.ORDER_FUNCTION_CALL];
};
function set_data(device,mode,value)
{
    const res = axios.patch(build_link(), { mode: mode,status:value });

}
function build_link( device_id="")
{
    let user=localStorage.getItem("user_id");
    let room=localStorage.getItem("room_id");
    return "https://escaperoom-b4ae9-default-rtdb.europe-west1.firebaseio.com/users/"+user+"/rooms/"+room+"/devices/"+device_id+".json";
}

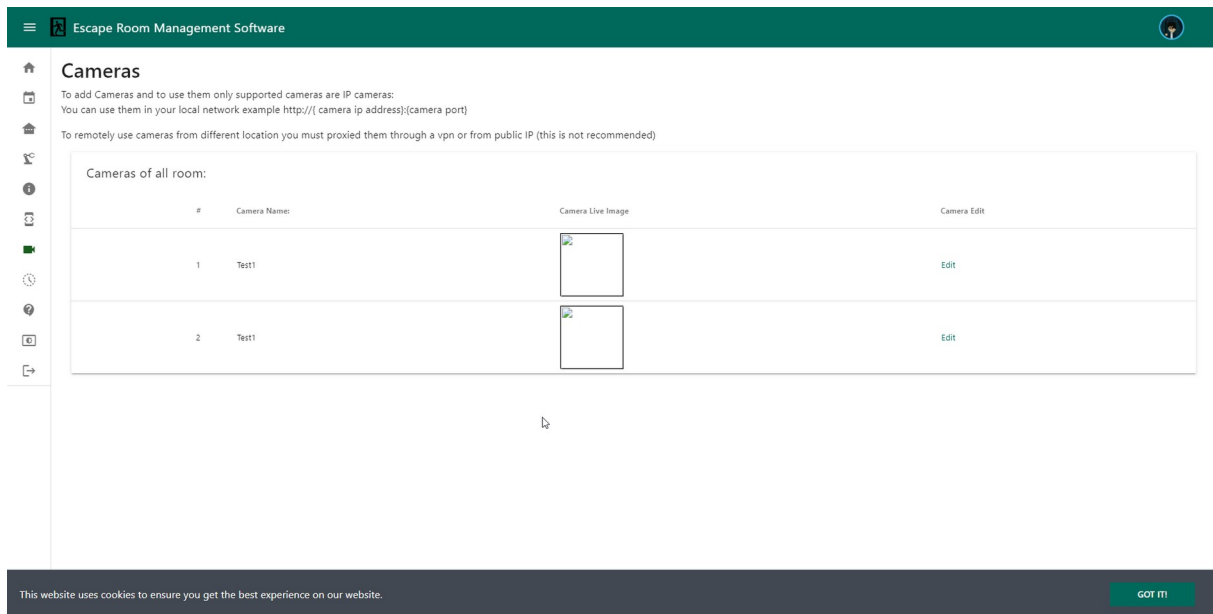
export {
    set_data
}

```

3.6 Kamerák kezelése:

Mint minden szabaduló szobában szükséges megfigyelő eszközöket elhelyezni a játékosok tippjeinek, vagy megakadásának megsegítésére. Ezért szükséges volt a programba beültetni egy Kamera kezelő modult, mely az aktuális szobára képes IP webkameráról képet adni.

Ezt úgy érjük el, hogy az ESP32 hálózatba csatoljuk a webkamerákat is, és megadjuk azok streaming címét, ezáltal a program ha rákattintunk az adott kamerára képes annak képét megjeleníteni.





Escape Room Management Software

Cameras

To add Cameras and to use them only supported cameras are IP cameras:
You can use them in your local network example `http://{camera ip address}:{camera port}`

To remotely use cameras from different location you must proxied them through a vpn or from public IP (this is not recommended)

Cameras of all room:

#	Camera Name:	Camera Live Image	Camera Edit
1	Test1		Edit
2	Test1		Edit

This website uses cookies to ensure you get the best experience on our website. [GOT IT](#)

3.7 Az Eventek kezelése:

A szoftver támogatja események hozzáadását, szobánként ezáltal nyomon követhetővé téve illetve ezek az event-ek megjelennek a címloldalon is.

3.8 Az eszközök kezelése:

Az eszközök kezelését két fő részre lehet bontani, van az automatikus kezelés (program általi), és van kézi vezérlés.

Az alábbi részben a kézi vezérlésről lesz szó.

A vue rész ami kezeli az eszközöket:

A template-je a weboldalnak:

```
<template>
<div >
<h2>Devices</h2>
<p>How to add device?</p>
<p>There is two method.</p>
<p>First you will need to add a device manually on the website.</p>
<p>Our method is using a "config.json" and flash this file to microcontroller, this file will contains the data for reaching server,
and Network settings for it.</p>
<div class="section" v-if="devices.length>0">
  <md-table md-card>
    <md-table-toolbar>
      <h1 class="md-title">Devices</h1>
    </md-table-toolbar>
    <md-table-row>
      <md-table-head md-numeric>#</md-table-head>
      <md-table-head>Device Name:</md-table-head>
      <md-table-head>Device Mode:</md-table-head>
      <md-table-head>Device is Online?</md-table-head>
      <md-table-head>Devices Edit:</md-table-head>
```

```

</md-table-row>

<md-table-row v-for="(row,index) in devices" :key="index">
  <md-table-cell md-numeric>{{ index+1 }}</md-table-cell>
  <md-table-cell>{{ row.data.device_name }}</md-table-cell>
  <md-table-cell>{{ row.data.mode }}</md-table-cell>
  <md-table-cell><activatedevice :lastonline="row.data.lastonline"/></md-table-cell>
  <md-table-cell><md-button class="md-raised md-primary" @click="edit(`/room/${
row.room_id}/device/${row.dev_id}`)">Edit Device</md-button></md-table-cell>
</md-table-row>

</md-table>
<md-button class="md-raised md-primary" @click="showDDialog = true">Add Device </md-
button>
</div>
<md-empty-state v-else
  md-icon="precision_manufacturing"
  md-label="You do not have devices"
  md-description="You do not have registered device. You can create device inside the Room. :/">
</md-empty-state>

</div>
</template>

```

A program szkript.

```

<script>
import Activedevice from "@/components/parts/Activatedevice";
import ElapsedTime from "@/components/ElapsedTime";
import router from "@/router";
import { FireDb,FirebaseAuth,userId } from "@/firebase";
import { ref, set ,onValue,get, child,push,runTransaction } from "firebase/database";
import { BIconCheck2,BIconPlus } from 'bootstrap-icons-vue'
import { get_data_from_allroomdb } from "@/mod_data/get_data";
export default {

  name: 'Devices',
  data: () => ({
    showDialog: false,
    scan:null,
    found_devices:[],
    showDDialog:false

  }),
  components:{

```

```

    Activeddevice
  },
  beforeMount()
  {
    const userId = FirebaseAuth.currentUser.uid;

    /*
      még nincs implementálva a böngészőkben OwO
      this.scan = navigator.bluetooth.requestLEScan({ "acceptAllAdvertisements":true });
      navigator.bluetooth.addEventListener('advertisementreceived',this.findADevice(event));
      //
    */
  },
  methods:

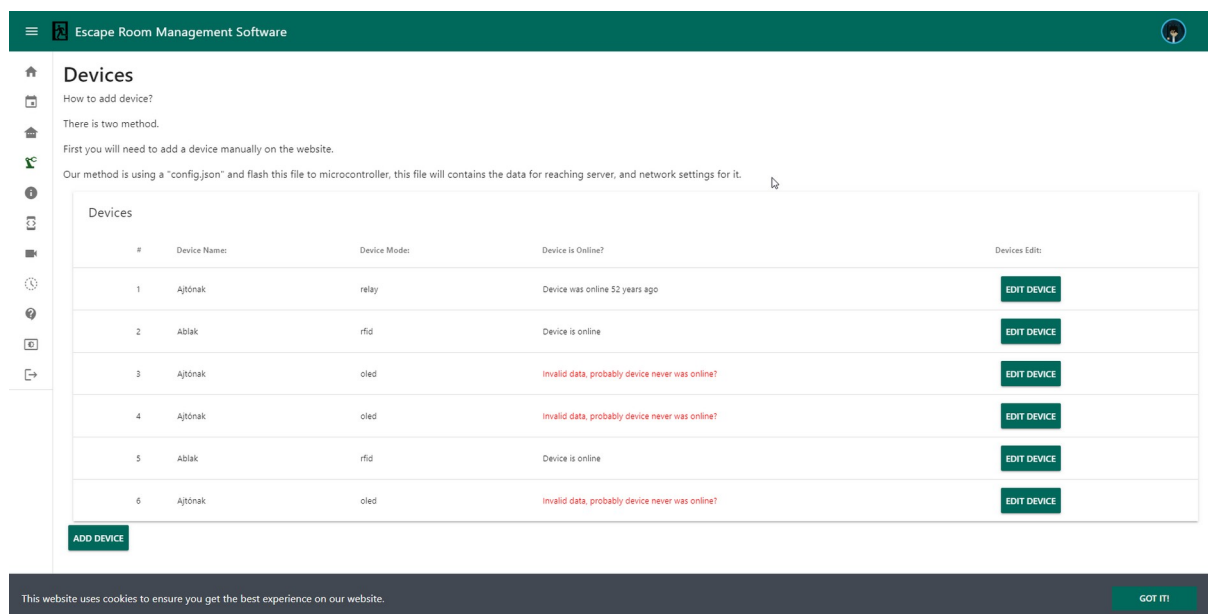
  edit(l)
  {
    router.push(l);
  },

  },
  computed:
  {
    devices()
    {
      return get_data_from_allroomdb("devices");
    }
  }

}

</script>

```



ábra 8 Saját eszközök áttekintése

Valamint van lehetőség az eszközöknek a saját konfigurációjára is.

A template skeletonja:

```
<template>
<div >
<div class="section">
<md-card md-with-hover >

  <md-card-header>
    <div class="md-title" >Device ~ {{ device.device_name }} </div>
  </md-card-header>
  <md-card-content>
    <md-field>
      <label for="name">Device name:</label>
      <md-input id="name" @change="namechange" v-model="device.device_name"></md-input>
    </md-field>
    <p><Deviceactivity :lastonline="device.last_online" /></p>

    <md-field>
      <p >Choose mode:</p>
      <b-select id="select" @change="achange" v-model="device.mode">
        <b-select-option v-for="mode in devm" :key="mode.type" :value="mode.type">{{ mode.name }} </b-
select-option>
      </b-select>
    </md-field>
    <DeviceInput :mode="device.mode" :v-model="device.status" />
  </md-card-content>
</div>
</div>
</template>
```

```

    <md-card-actions>
    <md-button class="md-raised md-primary" @click="duplicateDevice">Duplicate device with same settings</md-button>
    <md-button class="md-raised md-primary" @click="download_file">Download Config file</md-button>
    <md-button class="md-raised md-secondary" @click="showDeleteDialog = true">Delete Device</md-button>
    </md-card-actions>
  </md-card>
</div>
<div class="section">
<md-card md-with-hover v-if="serial_supported">
  <md-card-header>
    <div class="md-title">Flashing Device</div>
  </md-card-header>
  <md-card-content>
    <p>Only supported devices are <a href="https://en.wikipedia.org/wiki/ESP32">ESP32</a> family</p>
    Connected device : {{ serial.device_name }};

    <div >
      <p></p>
    </div>

  </md-card-content>

  <md-card-actions>
    <md-button class="md-raised md-secondary" :v-if="bluetooth_active" @click="bluetooth_connect">Try to connect with Bluetooth (alpha)</md-button>
    <md-button class="md-raised md-primary" @click="check_ports_usb">Connect to device through Serial port</md-button>
    <md-button class="md-raised md-primary" @click="flash_config">Flash config</md-button>
  </md-card-actions>
</md-card>
<md-dialog-confirm
  :md-active.sync="showDeleteDialog"
  md-title="Delete this device?"
  md-content="Your device settings will be deleted and cant be recovered."
  md-confirm-text="Agree"
  md-cancel-text="Disagree"
  @md-cancel="onCancel()"
  @md-confirm="adelete()" />
</div>
</div>
</template>

```

A program szkriptje:

```
<script>
import {Firebase,FirebaseAuth,userId} from "@/firebase";
import {ref, set ,onValue,get, child,push,runTransaction } from "firebase/database";
import Deviceactivity from '@/components/parts/Deviceactivity';
import DeviceInput from '@/components/parts/DeviceInput';
import {check_serial_supported,check_bluetooth_supported,list_coms} from '@/components/
flash_device';
import {devicemodes} from "@/datas";
import BluetoothTerminal from "@/bluetooth/BluetoothTerminal";

import {get_data_fromroomitemdb} from "@/mod_data/get_data";
import {add_device} from "@/mod_data/set_data";
import {delete_device} from "@/mod_data/del_data";
import {saveTextAsFile} from "@/file";
export default {
  name: 'Device',
  data: () => ({
    showDialog: false,
    showDeleteDialog:false,
    device:{device_name:"_Basic",mode:"relay",status:false},
    devm:devicemodes,
    select:"relay",
    showDDialog:false,
    device_name:"",
    serial_supported:false,
    serial_device:null,
    serial_reader:null,
    connected:false,
    defaultDeviceName:"mcu",
    serial:{device_name:"",terminalcontainer:""},
    bluetooth_uuid:"00001101-0000-1000-8000-00805F9B34FB",
    terminal : null,
    bluetooth_active:false,
    encoder:null,
    decoder:null,
    signals:null,
    reader: null,
    writer: null

  )),
  components:{
    Deviceactivity,
    DeviceInput
```



```

},
mounted()
{

    console.log(this.$route.params);
    //localStorage.setItem('device',JSON.stringify(null));
    const userId = FirebaseAuth.currentUser.uid;
    const devId = this.$route.params.did;
    const room_id=this.$route.params.rid;
    localStorage.setItem("mods",JSON.stringify(this.devm));
    this.serial_supported=check_serial_supported();
    this.bluetooth_active=check_bluetooth_supported();
    onValue(ref(FireDb, `/users/${userId}/rooms/${room_id}/devices/${devId}`),(sn)=>{
    if(sn.exists())
    {this.device=sn.val();
    this.select=this.device.mode;
    }
    });

    if(this.device.status==null)
    {
        set(ref(FireDb, `/users/${userId}/rooms/${room_id}/devices/${devId}/status`),false);
    }
    /*

    még nincs implementálva a böngészőkben OwO
    navigator.bluetooth.addEventListener('advertisementreceived',this.findADevice(event));
    */
    //setTimeout(this.serial_log, 3000);
    if(this.serial_supported)
    {
        navigator.serial.addEventListener("connect",(event)=> {this.connect_device(event)});
        navigator.serial.addEventListener("disconnect",(event)=> {this.disconnect_device(event)});
    }
    console.log(navigator.bluetooth);
    if(this.bluetooth_active)
    {
        let serveUuid = 0xFFE0, characteristicUuid = 0xFFE1;
        this.terminal=new BluetoothTerminal(serveUuid,characteristicUuid,'\n','\n');
    }
    this.encoder = new TextEncoder();
    this.decoder = new TextDecoder();

},

```

```

methods:
{
  adelete()
  {
    delete_device(this.$route.params.rid,this.$route.params.did);
  },
  onCancel () {
    //this.value = 'Disagreed'
  },
  get_type()
  {
    let b=[];
    b=JSON.parse(localStorage.getItem("mods"));
    let l="";
    console.log(b);
    if(this.device.mode===null) return "text";
    if(!b||b===null) return "text"; //to b or not 2[b]

    b.forEach((k)=>{
      if(this.device.mode===k.type)
      {
        l=k.control;
        return;
      }
    });
    //this.device.mode
    return l;
  },
  get_readonly()
  {
    let b=[];
    b=JSON.parse(localStorage.getItem("mods"));
    let l="";
    if(this.device.mode===null) return "false";
    if(!b||b===null) return "false";
    console.log(b);

    b.forEach((k)=>{
      if(this.device.mode===k.type)
      {
        l=k.readonly;
        return;
      }
    });
    //this.device.mode
    return l;
  },

```

```

async check_ports()
{
  let config_for_join={ baudRate: 9600 };
  const filters=[
    { usbVendorId: 4292, usbProductId: 60000 }
  ]

  this.serial_device = await navigator.serial.requestPort({ filters});
  //this.serial_device.open(config_for_join);
  console.log( this.serial_device);
  console.log(this.serial_device.getInfo());

  await this.serial_device.open(config_for_join);

  //this.serial_reader=this.serial_device.readable.getReader();
  this.serial_log();
},
async check_ports_usb()
{
  let config_for_join={ baudRate: 115200 };
  const filters=[
    { usbVendorId: 4292, usbProductId: 60000 }
  ]

  this.serial_device = await navigator.serial.requestPort({ filters});
  //this.serial_device.open(config_for_join);
  console.log( this.serial_device);
  console.log(this.serial_device.getInfo());

  await this.serial_device.open(config_for_join);

  this.writer = this.serial_device.writable.getWriter();
  this.reader = this.serial_device.readable.getReader();

  //this.serial_reader=this.serial_device.readable.getReader();
  this.serial_log();

},

flash_config()
{

},

achange()

```

```

    {
      const userId = FirebaseAuth.currentUser.uid;
      console.log(this.device.mode);
      let _ref= ref(Firebase, `/users/${userId}/rooms/${this.$route.params.rid}/devices/${this.$route.params.did}/mode`);
      set(_ref,this.device.mode);
      _ref= ref(Firebase, `/users/${userId}/rooms/${this.$route.params.rid}/devices/${this.$route.params.did}/status`);
      set(_ref,false);
    },
    namechange()
    {
      const userId = FirebaseAuth.currentUser.uid;
      console.log(this.device.mode);
      let _ref= ref(Firebase, `/users/${userId}/rooms/${this.$route.params.rid}/devices/${this.$route.params.did}/device_name`);
      set(_ref,this.device.device_name);
    },
    duplicatedevice()
    {
      add_device(this.$route.params.rid,this.device.device_name,this.device);
    },
    connect_device()
    {
      if(this.connected) return;
      this.$noty.success("Device connected!");
      this.check_ports();
      this.connected=true;
    },
    disconnect_device()
    {
      if(!this.connected) return;
      this.$noty.success("Device disconnected!");
      this.connected=false;
    },
    get_config()
    {
      const userId = FirebaseAuth.currentUser.uid;

      let l={
        data:this.device,wifiname:get_data_fromroomitemdb(this.$route.params.rid,"wifi_name"),
        wifipassword:get_data_fromroomitemdb(this.$route.params.rid,"wifi_password"),
        device_id:this.$route.params.did,
        room:this.$route.params.rid,
        user:userId,
        user_email:FirebaseAuth.currentUser.user_email,

```

```

device_mode:this.device.mode,
database_url:"https://escaperoom-b4ae9-default-rtdb.europe-west1.firebaseio.com",
api_key:"Rko9QOoaBDbXAndM0UGhKbZBFFZWFGsl6KiUNnGt",
mod:"setup",
accessToken:FirebaseAuth.currentUser.accessToken
};

return l;
},
bluetooth_connect()
{
  /* var serviveUuid = null;
  var characteristicUuid = null;
  navigator.bluetooth.requestDevice({acceptAllDevices: true})
    .then(device => {
      console.log(device);
    });*/
  if(this.bluetooth_active)
  {
    this.terminal.connect().
    then(() => {
      this.serial.device_name = this.terminal.getDeviceName() ?
        this.terminal.getDeviceName() : this.defaultDeviceName;
    });
  }
},

download_file()
{
  let filename="config.json";
  let l=this.get_config();
  saveTextAsFile(JSON.stringify(l),filename);
},
async serial_log()
{
  let string="";

  if( !ReadableStream.prototype.pipeTo ) {
    this.$noty.error( "Your browser doesn't support pipeTo");
    return;
  }

  // Listen to data coming from the serial device.

```

```

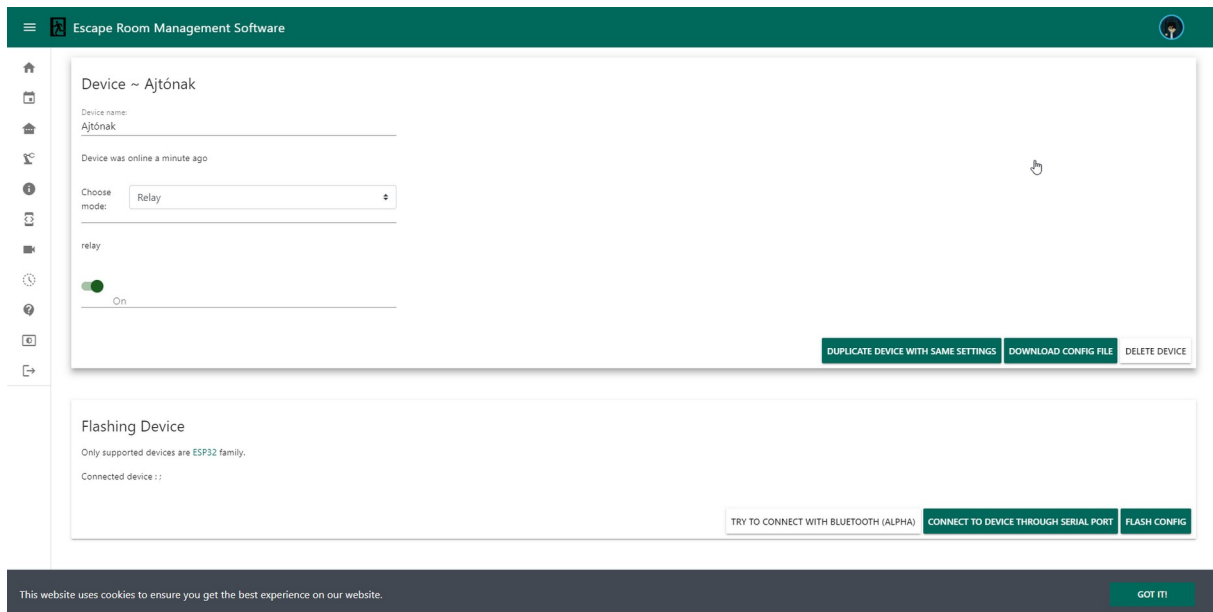
/*eslint no-constant-condition: ["error", { "checkLoops": false }]*/

},
write_to_serial(text){
  if (this.serial_device && this.serial_device.writable) {
    //const value = parseInt(text);
    const bytes = new Uint8Array([text]);
    const writer = this.serial_device.writable.getWriter();
    writer.write(bytes);
    writer.releaseLock();

  }

},
async write(data){
  const dataArrayBuffer = this.encoder.encode(data);
  return await this.writer.write(dataArrayBuffer);
},
async read() {
  try {
    const readerData = await this.reader.read();
    return this.decoder.decode(readerData.value);
  } catch (err) {
    const errorMessage = `error reading data: ${err}`;
    console.error(errorMessage);
    return errorMessage;
  }
}
}
</script>

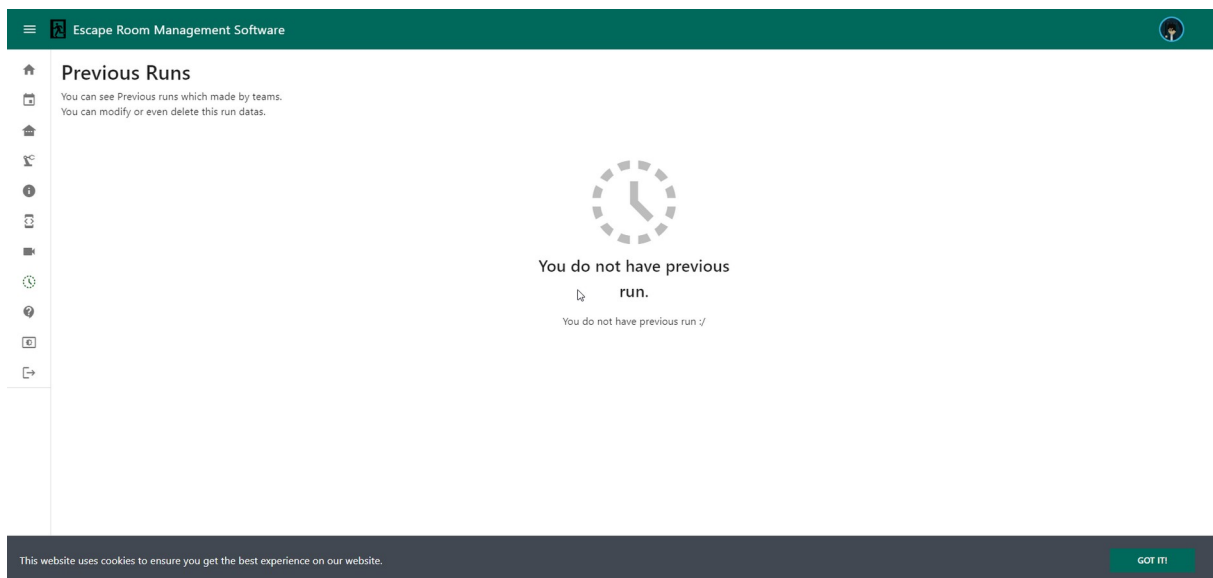
```



ábra 9 Az eszköz manuális beállítása

3.9 A régebbi futamok megtekintése:

A régebbi futamokat is meglehet a szoftverben tekinteni, mely segítségével megjelenik.



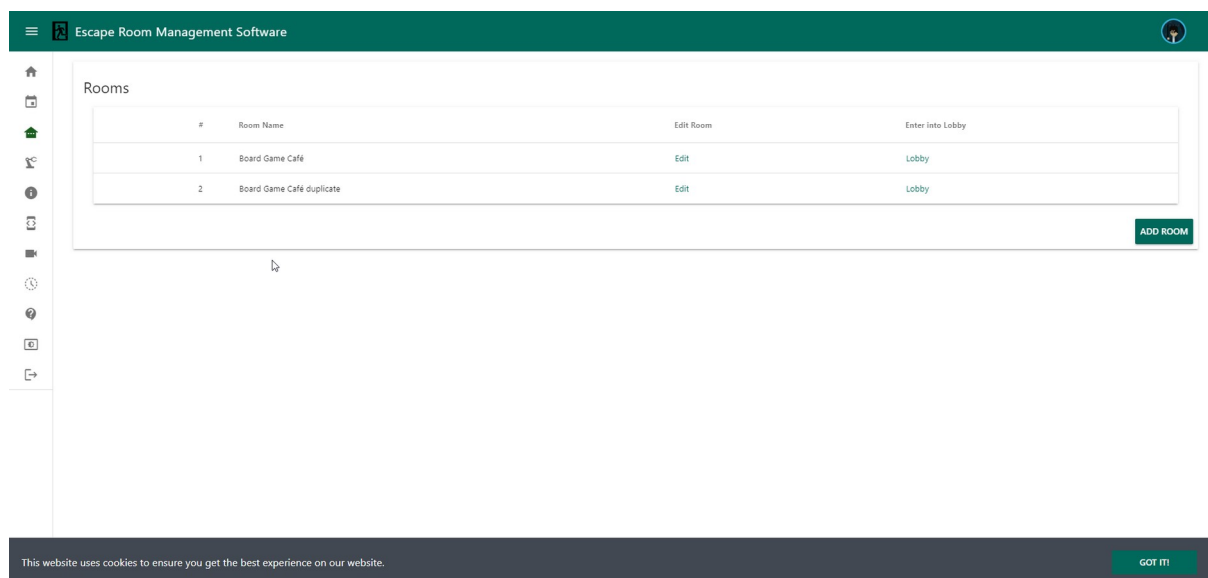
ábra 10 Előző futamok megjelenítése

Ki listázza szobánként az előző futamokat.

A szobákban található previous_runs-ből kieszedi az előző futamok, idejét és megjeleníti őket.

3.10 A szobák megjelenítése:

A szobák megjelenítése



ábra 11 a Szobák megjelenítése

A felhasználók node-jából ki listázza a szobákat.

A program váza:

```
<template>
<div >

  <div v-if="rooms.length>0">

    <md-card >
      <md-card-header><div class="md-title">Rooms</div></md-card-
header>
      <md-card-content>
        <md-table md-card>

          <md-table-row>
            <md-table-head md-numeric>#</md-table-head>
            <md-table-head>Room Name</md-table-head>
            <md-table-head>Edit Room</md-table-head>
            <md-table-head>Enter into Lobby</md-table-head>

          </md-table-row>

          <md-table-row v-for="(row,index) in rooms" :key="row.devID">
            <md-table-cell md-numeric>{{index+1}}</md-table-cell>
            <md-table-cell>{{row.data.room_name}}</md-table-cell>
            <md-table-cell><router-link :to="{ path: '/'
room/' +row.devID}">Edit</router-link></md-table-cell>
            <md-table-cell><router-link :to="{ path: '/'
room/' +row.devID+ '/lobby/'}">Lobby</router-link></md-table-cell>
```



```

</md-table-row>

</md-table>
</md-card-content>
<md-card-actions>
  <md-button class="md-raised md-primary" @click="showDialog = true">Add room</md-button>
</md-card-actions>
</md-card>
</div>

<md-empty-state v-else
  md-icon="other_houses"
  md-label="Create your first room"
  md-description="Create beautiful project with this program, and
you'll be able to create great things.">
  <md-button class="md-primary md-raised" @click="showDialog = true" >Add Room</md-button>
</md-empty-state>

<md-dialog-prompt
  :md-active.sync="showDialog"
  v-model="room_name"
  md-title="Add Room"
  md-input-maxlength="30"
  md-input-placeholder="Room name ..."
  md-confirm-text="Done"
  :md-confirm="padd_room()" />

</div>
</template>

```

A következő szkript hatodik végre:

```

<script>
import {Firestore,FirestoreAuth,userId} from "@firebase";

import {ref, set ,onValue,get, child,push,runTransaction } from "firebase/database";
import {get_data_fromroomdb,get_rooms} from "@mod_data/get_data";
import {add_room} from "@mod_data/set_data";

export default {
  name: 'Rooms',

```

```

data(){
  return {
    showDialog: false,
    rooms:[],
    room_keys:[],
    room_name:"",
    test:"aaa",
    rows:[],
    devices:[],
    users:[]
  },

  beforeMount()
  {
    this.rooms=[];
    this.rooms=get_rooms();
    //console.log(this.rooms);
  },
  methods:{
    padd_room() {
      add_room(this.room_name);
      //this.showDialog=false;
      this.room_name="";
    },

    get_active_devices(index)
    {
      this.devices=get_data_fromroomdb(index,"devices");
      let active=0,inactive=0,k;

      this.devices.forEach(element => {
        console.log(element.data.lastonline);
        k=Date.now()-Date(element.data.lastonline);
        if(k<120)
        {
          active++;
        }
        else
        {
          inactive++;
        }

        // console.log(k);
      });
    }
  }
}

```

```

        if(inactive==0)
        {
            return "All device online";
        }
        else if(active==0)
        {
            return "All device offline";
        }
        else
        {
            return `${active} are online, and ${inactive} are offline`;
        }
    },
}

}

</script>

```

A stílusa programnak:

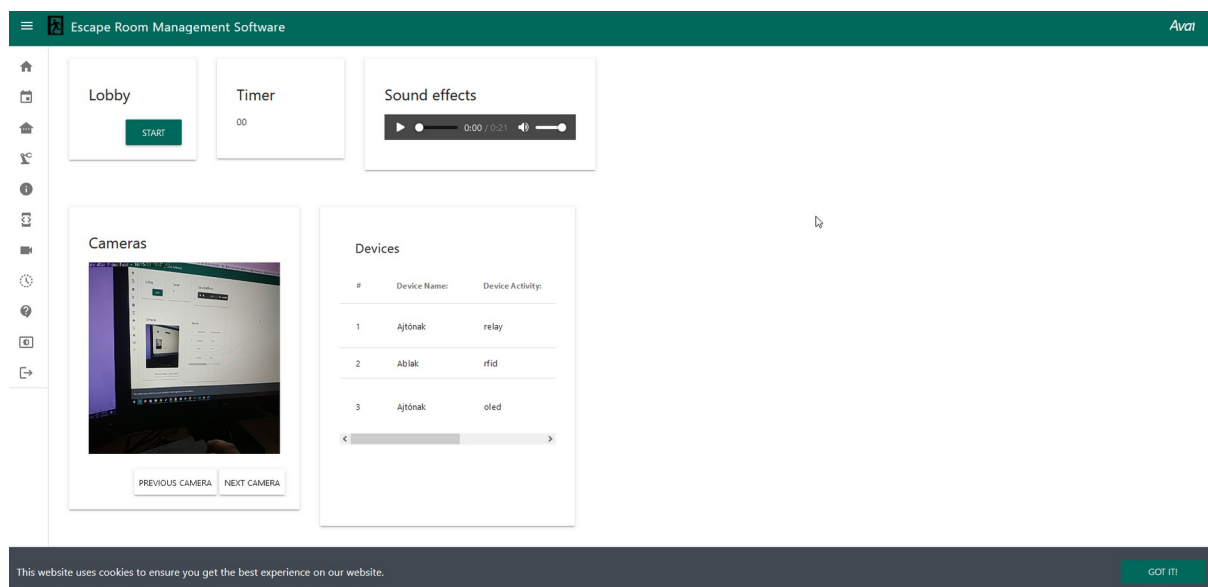
```

<style lang="scss" scoped>
    .md-list {
        width: 320px;
        max-width: 100%;
        display: inline-block;
        vertical-align: top;
        border: 1px solid rgba(#000, .12);
    }
</style>

```

3.11 A lobby és a program futtatása:

A maga a Lobby és a Program futtatása az egyik legfontosabb pillére a programnak ezért ezzel is tölteném a legfontosabb információkat.



ábra 12 Lobby még nem indult el a folyamat

Ahogy említettem előzőleg maga a blockly egy lementett base64-ben kódolt kódot ment el. Itt ez a fájlt át fordítjuk Javascriptre, S ezt a fájlt az eval() függvény segítségével a böngésző navigatorjában futtattjuk a kódot.

Először is szükséges Custom elementeket létrehozni ilyenek pl. a Beolvasó, kiírató, Státusz ellenőrző, illetve a befejező, valamint a Console-ba kiírató parancsok.

A blockly element létrehozása:

Először is deklaráljuk a modellt erre példának az eszközre kiíratást fogom használni:

XML schema:

```
Blockly.defineBlocksWithJsonArray([
  {
    "type": "send_data",
    "message0": "set device %1, mode: %2, status: %3",
    "args0": [
      {
        "type": "field_dropdown",
        "name": "device",
        "variableTypes": [""],
        "options": this.get_devices_to_array()
      },
      {
        "type": "field_dropdown",
        "name": "mode",
        "variableTypes": [""],
        "options": [
          ["Relay", "relay"],
          ["RFID reader", "rfid"],

```

```

        ["Input", "input"],
        ["OLED Display", "oled"],
    ],
    },
    {
        "type": "input_value",
        "name": "value",
        "check": ["String"]
    }
],
"inputsInline": true,
"previousStatement": true,
"nextStatement": true,
"colour": 160,
"tooltip": "",
"helpUrl": "",
},
);

```

Ezt hozzáadva látható, hogy a bemenő argumentumoknál 3 elem van meghatározva:

- Eszköz
- Üzem mód
- Státusz

Majd ennek az elemnek létrehozzuk, a javascript változatát is.

```

Blockly.JavaScript["send_data"] = function(block) {
    var device = block.getFieldValue('device');
    var mode = block.getFieldValue('mode');
    var value = Blockly.JavaScript.valueToCode(block, 'value',
Blockly.JavaScript.ORDER_MEMBER) || '\'\';
    var code = `this.set_data("${device}","${mode}",${value});\n
n`;
    return code;
};

```

S hasonlóan a többi függvényt is létrehozzuk.

Majd szükséges volt, egy független kezelő modult létrehozni melynek segítségével könnyedén lehessen módosítani az értékeket.

A Rest API-hoz használtam az Axios könyvtáron keresztül.

set_data függvény tartalma:

```

set_data(device,mode_input,value)
{

```

```

        let res=axios.patch(this.build_link(device),
        {mode:mode_input,status:value});
        //res.data.headers['Content-Type'];
        console.log("Setted device");
    },

```

Mivel egy kvázi homokozóban fut az applikáció ezért itt is a parancsoknak úgy kellet létrehozni, hogy könnyen lehessen módosítani, s programozhatóvá tenni őket. A programnak az XML verzióját, lerendereljük az XML verzióját, s azután megkapjuk a Javascript verziót is.

```

    this.a_xml=decoding(this.program.program_xml,get_encoding(this.
$route.params.rid));

    let xml= Blockly.Xml.textToDom(this.a_xml);
    console.log(xml);
    try {
        Blockly.Xml.domToWorkspace(xml,this.Workspace);
    }
    catch (error) {
        console.error(error);
    }

    try {

        this.a_js = Blockly.JavaScript.workspaceToCode(this.Work
space);
    } catch (error) {
        console.error(error);
    }

    console.log(this.a_js);
    console.log(this.run);
    if(this.run==null) this.started=false;
    else {
        this.started=this.run.active;
        if (this.started) {
            setTimeout(()=>{
                this.startCountdown();
                this.start_runprocess();
            },2000);
        }
    }
}

```

Ellenőrizzük, hogy a `current_status`-ba van e érték. Ha van akkor folytatjuk a szoftver futtatásával. Ha nincs akkor a szoftver még nem lesz elindítva.

Várjuk a felhasználót, hogy elindítja a folyamatot.

A programnak a visszaszámlálónak, az elindítása folyamán kezdjük el ellenőrizni, hogy a szoftvernek szánt időből kifutott a szoftver, s akkor lezárja a futamot:

```
setInterval(()=>{
    if(this.run.active)
    {
        if(this.run.finishing_time<=Date.now())
        {
            console.log("Run out of time flag!");
            this.send_finish();
        }
    }
}, 1000);
```

3.12 Az ESP32 programja:

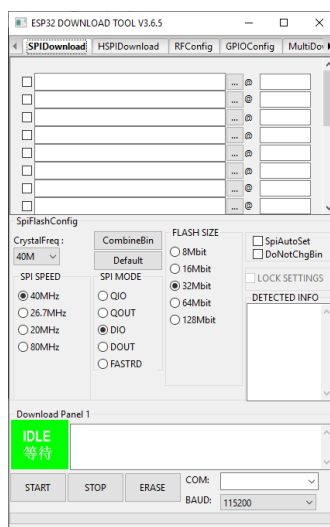
Az ESP32-nek is először szükséges a program feltöltése a programot kétféle képen lehet feltölteni az eszközre az első lépés, hogy a fejlesztő környezetet használjuk.

Az ESP32-t a Platform IO környezetben fejlesztettem, s itt ha felhasználó leklónozza a következő linket:

https://github.com/karpad2/obudai_diplomamunka/tree/esp32_szoftver

S lefordítja a Platform IO-val, s telepíti.

A második mód:



ábra 13 ESP32 gyári feltöltő szoftver

A bin könyvtárak manuális feltöltése a gyári feltöltő szoftverrel. Mellyel feltöltjük a SPIFFS, valamint az ESP32 „factory.bin”-jét.

S ezek segítségével képes a külvilággal is kommunikálni.

Én a programozáshoz a Visual Studio Code-t használtam a Platform IO kiegészítővel.

Az eszköz írásához szükséges a platform IO partíciós tábla használata:

```
# Name, Type, SubType, Offset, Size, Flags
```

```
nvs, data, nvs, 0x9000, 0x5000,
```

```
otadata, data, ota, 0xe000, 0x2000,
```

```
app0, app, ota_0, 0x10000, 0x1E0000,
```

```
app1, app, ota_1, 0x1F0000, 0x1E0000,
```

```
spiffs, data, spiffs, 0x3D0000, 0x30000,
```


NTP idő lekérdezés

Mivel szükséges, hogy az eszköz látható legyen, hogy aktív ezért szükséges volt NTP szerver használata.

Amikor a rendszer lekérdezi az időpontot ezután szükséges, még hogy a rendszer frissítse a rendszer időpontját.

A rendszer órája az UTC időpontra van beállítva, s innen tudjuk a világszinten, hogy az alkalmazás valós időben fut-e.

```
configTime(0, 0, ntpServer); //UTC időpont

Serial.print(F("Waiting for NTP time sync: "));
time_t nowSecs = time(nullptr);
while (nowSecs < 8 * 3600 * 2) {
    delay(500);
    Serial.print(F("."));
    yield();
    nowSecs = time(nullptr);
}
```

Fontos volt még beállítani a DNS szerveret az eszközön, sajnos erre az Espress if nem adott más lehetőséget, mint először DHCP kliensként, bejelentkezni, s azután Statikus IP címként meghatározni a DNS szerveret.

Erre azért volt szükség mivel sajnos másképp, problémás volt a github oldal elérése a frissítéseknél.

```
Serial.print("Connected to ");
Serial.println(wifi_name);
Serial.print("IP address: ");
Serial.println(WiFi.localIP());
if (MDNS.begin("ESP32-Controller")) {
    Serial.println("MDNS responder started");
    blinking(3);
}
WiFi.config(WiFi.localIP(), WiFi.gatewayIP(), WiFi.subnetMask(),
IPAddress(8,8,8,8));
```

Majd a csatlakozás után, szükséges volt a HTTPS kapcsolatra, muszáj volt a hitelesítő dokumentumok eltárolása is. Erre a Spiiffs tár helyét használtam fel.

Spiiffs tárhely az ESP32 egy 0x3000 byte tárhelyű tár ahol, különböző nem Hard kódolt elemek vannak letárolva. Itt található a config.json, firebase_cert.txt, és a github_cert.txt file is.

Az eszköz aktivitását ellenőrző rész:

```
client.setCACert((char*)firebase_cert.c_str());
String link=build_link(user,room,device);
Serial.print("Firebase link:");Serial.println(link);
HTTPClient http;

if(http.begin(client,(char*)link.c_str())) {
  dataMillis = millis();

  httpCode=http.GET();
  json_response="";
  if (httpCode == HTTP_CODE_OK) {
    Serial.printf("[HTTPS] GET... code: %d\n", httpCode);
    json_response=http.getString();
    Serial.println(json_response);
    http.end();
    //time_for_response

    link=build_link(user,room,device);
    if(http.begin(client,(char*)link.c_str())) {
      String httpRequestData = "";
      http.addHeader("Content-Type", "application/json");

      DynamicJsonDocument timejson(2048);
      timejson["lastonline"]=printLocalTime();
      serializeJson(timejson,httpRequestData);
      httpCode=http.PATCH(httpRequestData);
      http.end();
    }
  }
}
```

3.13 A regisztráció menete:

Először is az ESP32 használja a SPIFFS tárolót, és itt be van állítva egy config.json file.

Ebből a fájlból lehet beállítani az eszköznek a saját felhasználó ID-jét, valamint ahhoz tartozó Szoba ID-jét, de mivel ez egy központosított rendszer, ezért több eszközről is betud jelentkezni.

Küldi a rendszer a Webserial-API -n keresztül küldi az adatokat az esp32nek soros porton keresztül:

S a következő szkript fogadja:

```
void parse_serial(String text)
{
    deserializeJson(doc, (char*)text.c_str());

    if(doc["mode"]=="setup")
    {
        DynamicJsonDocument parsejson(2048);
        String _text=a_config_read();
        deserializeJson(parsejson,_text);
        String tmp="";
        tmp=doc["wifi_name"].as<String>();
        parsejson["wifi_name"]=(tmp=="null"?parsejson["wifi_name"]
.as<String>() : tmp);
        tmp=doc["wifi_password"].as<String>();
        parsejson["wifi_password"]=(tmp=="null"?parsejson["wifi_password"]
.as<String>() : tmp);
        tmp=doc["user"].as<String>();
        parsejson["user"]=(tmp=="null"?parsejson["user"].as<String>() : tmp);
        tmp=doc["room"].as<String>();
        parsejson["room"]=(tmp=="null"?parsejson["room"].as<String>() : tmp);
        tmp=doc["device"].as<String>();
        parsejson["device"]=(tmp=="null"?parsejson["device"].as<String>() : tmp);

        serializeJson(parsejson,_text);

        Serial.println(_text);

        a_config_write(_text);
    }
}
```

Majd a config.json fájlba lementjük az adatokat.

```
void a_config_write(String text)
{
  File file = SPIFFS.open("/config.json", FILE_WRITE);
  if(!file){
    Serial.println("There was an error opening the file for writing");
    return ;
  }
  if(file.print(text)) {
    Serial.println("File was written");
  }else {
    Serial.println("File write failed");
  }
  file.close();
  ESP.restart();
}
```

Majd az eszközt újraindítjuk.

Ezután a rendszer válaszbán megadja a lementi a wifi adatokat, illetve a felhasználónak a beállításait, lementi a config file-t.

Majd az eszközt újra indítjuk.

3.14 Az eszköz indulása:

Az eszköz indulásánál először is a SpiFFs fájlt beolvassuk majd.

```
void a_setup_filesystem()
{
  delay(200);
  Serial.println(F("Inizializing FS..."));
  if(SPIFFS.begin()) Serial.println(F("done. "));
  else {
    Serial.println(F("fail. "));
    SPIFFS.format();
  }
}
```

Ha a spiFFs nem létezik létrehozzuk.

Majd folytatjuk a config file beolvasásával:

```
String a_file_read(String filename)
{
    Serial.println("Try to read "+filename);

    String configtxt="";
    File file = SPIFFS.open("/"+filename);
    if(!file){
        Serial.println("There was an error opening the file,try to recreate it.");
        file.close();
        //a_config_write(config_json);

        return a_file_read(filename);
        //return "";
    }
    else{
        Serial.println("Try to open file:");
        if(file.available()) configtxt=file.readString();

    }

    file.close();
    //Serial.println(configtxt);
    return configtxt;
}
```

S beolvassuk az ESP32 adatait.

```
String a_config_read()
{
    return a_file_read("config.json");
}
```

Ha az adatok, megfelelőek a rendszer bejelentkezik a Wifi-re majd, beállítjuk az IO paramétereit.

```
void setup_io()
{
    SPI.begin();
```

```
pinMode(relaypin, OUTPUT);
pinMode(boot_pin, INPUT);
pinMode(led, OUTPUT);
attachInterrupt(interruptpin, isr, FALLING);
}
```

Elkezd 3 másodpercenként ellenőrizni a változásokat.

3.15 Az eszköz bejelentkezése:

Mint az előbb említettem a rendszernek a bejelentkezéshez szükséges a jelszó használata.

Az eszköznek 4 féle módja van előre letárolva, de lehet, hogy ez a lista még később bővül.

- Relé üzemmód.
- RFID olvasó üzemmód.
- Bemeneti üzemmód.
- OLED kijelző üzemmód.

Ebben a megvalósításban az eszköz egyszerre csak egy módszert támogat, még pedig azt amit a hálózatról kiolvas.

```
DynamicJsonDocument device_json(2048);

deserializeJson(device_json,(char*)json_response.c_str());
String aaa=device_json["mode"].as<String>();
```

3.16 Relé üzemmód:

Ammenyiben a bemenetre a relé üzemmód van kapcsolva, akkor az eszköz elkezd a változóknak az elemeit értelmezi az értékét és egyet villant a led-en.

```
if(aaa=="relay")

{ String a=device_json["status"];
  Serial.println(a);
  bool status=a=="true"?true:false;
  relay(status);
}
```

A relay függvény tartalma:

```
void relay(bool _status)
{
    digitalWrite(relaypin,_status);
    if(_status) Serial.println("Relay: on");
    else Serial.println("Relay: off");
}
```

3.17 Bemenet üzemmód:

A bemeneti üzemmódban a kijelölt láb (25) és a GND (föld) közötti rövidre zárásként reagál. Ebben az üzemmódban egy kapcsoló reagálásra, pl. Reed relé összekapcsolására reagál.

```
else if(aaa=="input")
{
    {
        timejson["status"]=inputflag;
        if(http.begin(client,(char*)link.c_str())) {
            httpRequestData = "";
            http.addHeader("Content-Type", "application/json");
            serializeJson(timejson,httpRequestData);
            httpCode=http.PATCH(httpRequestData);
            http.end();
        }
    }
    inputflag=false;
}
```

S a program reagál erre a lábra és bejelentkezik a szerverre.

```
void IRAM_ATTR isr() {
    inputflag=true;
    blinking(3);
}
```

Ezt Majd ezt a változást lementi az inputflag-be s ha átállítódik a mód input állapotra akkor a program elküldi státusz ként a Firebase-be.

3.18 Megjelenítő üzemmód

A megjelenítő üzemmód lehető teszi, hogy egy OLED kijelzőre lehetséges legyen szöveget kiírhatni.

Megjelenítése az ESP32:

```
else if(aaa=="oled")
{
    display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
    display.clearDisplay();

    // Display Text
    display.setTextSize(1);
    display.setTextColor(WHITE);
    display.setCursor(0,28);
    String text_oled= device_json["status"].as<String>();
    display.println((char*)text_oled.c_str());
    display.display();
    delay(2000);
    display.clearDisplay();
}
else
{
    Serial.println("Fail");
}
```

Fogadó szkript mely kezeli a megjelenítést:

3.19 RFID üzemmód

A szoftver képes RFID kártya olvasására és jelezni a szoftver számára.

Az esp32 szkript:

```
else if(aaa=="rfid")
{
    Serial.print(F("PICC type: "));
    MFRC522::PICC_Type piccType = rfid.PICC_GetType(rfid.uid.sak);
    Serial.println(rfid.PICC_GetTypeName(piccType));
    if (piccType != MFRC522::PICC_TYPE_MIFARE_MINI &&
        piccType != MFRC522::PICC_TYPE_MIFARE_1K &&
        piccType != MFRC522::PICC_TYPE_MIFARE_4K) {
        Serial.println(F("Your tag is not of type MIFARE Classic."));
        return;
    }
}
```



```

    for (byte i = 0; i < 4; i++) nuidPICC[i] = rfid.uid.uidByte[i];
    Serial.println(F("The NUID tag is:"));
    Serial.print(F("In hex: "));
    readed_rfid_code="";
    for(byte i=0;i<rfid.uid.size;i++)
    {
        readed_rfid_code+=String(rfid.uid.uidByte[i],HEX);
    }

    if(readed_rfid_code=="")
    {
        Serial.println("Not readed card");
    }
    else
    {
        timejson["status"]=readed_rfid_code;

        if(http.begin(client,(char*)link.c_str())) {
            httpRequestData = "";
            http.addHeader("Content-Type", "application/json");

            serializeJson(timejson,httpRequestData);
            httpCode=http.PATCH(httpRequestData);
            http.end();
        }

        readed_rfid_code="";
    }

}

```

Mint látható az eszköz elküldi a kártya azonosító számát és beírja a Firebase status mezőjébe az értéket.

3.20 ESP32 OTA frissítés

Az ESP32 képes magát a hálózatról frissíteni, le ellenőrzi a szoftver, hogy a GitHubon található-e újabb verzió, s majd ha a verzió nagyobb akkor a rendszer frissíti magát.

```

void system_update()
{
    HTTPClient http;

```

```

bool update=true;
client.setCACert((char*)github_cert.c_str());
http.begin(fw_link);

httpCode=http.GET();
if(httpCode>0)
{
    String text=http.getString();
    deserializeJson(version_tester,(char*)text.c_str());
    double ghversion=version_tester["version"].as<double>();
    update=ghversion>version;
    Serial.println("Github version: "+String(ghversion)+" , Local version: "+String(version)+" , Thats w
hy device is: "+(update?"updating":"not updating"));
    String txt_json="";
    String _text=a_config_read();
    deserializeJson(doc,_text);
    Serial.println(_text);

    if(update)
    {
        doc["version"]=ghversion;
        serializeJson(doc,_text);
        a_config_write(_text);

        http.begin(fw_link_bin);
        httpUpdate.update(http,fw_link_bin);
    }
    else
    {
        Serial.printf("[HTTPS] GET... failed, error: %s\n", http.errorToString(httpCode).c_str());
    }
    http.end();
}

```

3.21 Raspberry PI beállítása:.

Zerotier alkalmazást feltelepítve beállítottam, hogy induláskor először lecsatlakozon a hálózatról, majd vissza csatlakozón azért, mert a képfájl ami a github tárolóban fent van, ne jelentsen ütközést a legelső indulásnál.

```

zerotier-cli leave 1d719394045cf130
zerotier-cli join 1d719394045cf130

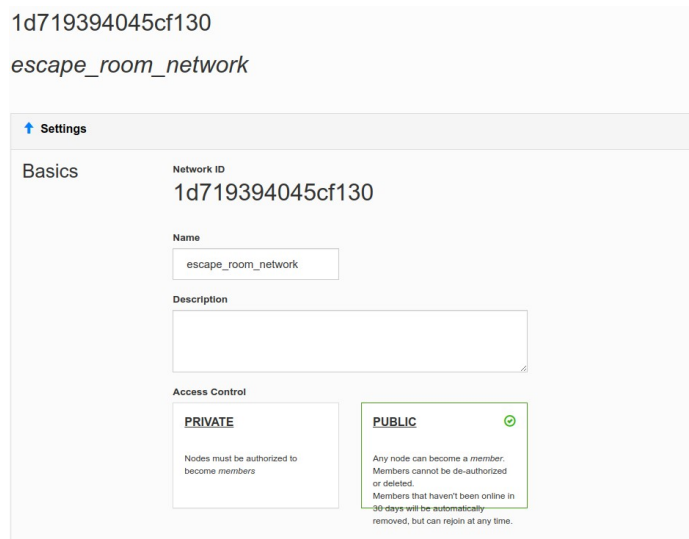
```

Rasp Ap nyílt hozzáférésű szoftvert felraktam Manjaro operációs rendszerre. Majd beállítottam, hogy a Zerotier hálózatra adjon hozzáférést a beállított SSID, és wifi jelszó segítségével.

SSID: TP-Link

Jelszó: asdfghjkl123#

Majd felállítottam egy Webmin szolgáltatást is a rendszeren, ahol átirányítottam a kéréseket a VPN szolgáltatásra. Szükséges megemlíteni, hogy ha azt akarjuk, hogy a felhasználók használhassák a kamera szolgáltatást távolról ezért szükséges használni a VPN szolgáltatást.



ábra 14 Zerotier kezelő felülete

3.22 A szoftver futtatása Raspberry PI-n.

Először is szükséges leklónozni a Github Repository tárolót.

```
git clone https://github.com/karpad2/obudai_diplomamunka
```

Ennek végeztével szükséges felrakni, Manjaro operációs rendszerre a yay nevezetű szoftvert.

```
sudo pacman -Sy yay
```

Majd ha ez a program feltelepült a következő szkriptje futtatjuk le:

```
yay -S node
```

```
yay -S composer
```

Belépünk a repositoryba mappába.

A szükséges Javascript függőségeket feltelepítjük.

```
npm install  
npm run serve
```

Majd elkészítjük az optimalizált verziót a futtatásra.

Vagy elnavigálunk a Firebase által hostingolt linkre és bejelentkezünk :

<https://escaperoom-b4ae9.web.app/>

S a kameránál engedélyezzük a http (nem biztonságos) forrás engedélyezését.

3.23 A szoftver tovább fejlesztési lehetőségei:

A szoftvert akár a végtelenségig lehetne fejleszteni több modullal megoldási segédletekkel, de egy ami a szoftverből talán mégis hiányzik PLC-re való optimalizálás (REST API-n való kezelés), illetve az ARM processzorokra való Docker lehetősége. A többi hiányosságot az ESP32 frissítéssel, illetve a Weboldal frissítésével megoldhatjuk.

4. Összefoglalás

A szakdolgozatban elért célokat teljesítettük. Egy olyan rendszert sikerül megvalósítani, amely hardveres modulok mellett tartalmaz egy klienst és egy web alkalmazást. A kifejlesztett modulok működnek.

Az ESP32 és a webalkalmazás közötti kommunikáció megfelelő működést biztosít. A vezérlés kiegészítését jelentő, a Raspberry-n futó szerver és a web alkalmazás is olyan lehetőségekkel bővült, amelyeket a fejlesztés elején kitűztem:

- a vezérlés mobil eszközről is megvalósulhat;
- Programokat lehet készíteni, valamint futtatni;
- Az eszközök állapota nyomon követhető a menüben is;

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

A alkalmazást lehetne tovább fejleszteni Dockerbe MongoDB-vel, de mivel a Docker ARM támogatottsága még elége limitált, ezért ezt a célt még nem sikerült megvalósítani, tehát egy „kattintással elindítani”.

5. Idegen nyelvű tartalmi összefoglaló

The objective of this thesis is to design an escape room management software. A complete system of different types of hardware and software is applied. The thesis contains a web application, and client.

The ESP32 and the webapplication communication are working. The control between the Raspberry and the devices is excluded by different features:

- Control from a mobile device
- Can be programmed from a website
- Device status can be monitored from the menu.

The system contains a support tab, where users can leave issues, and ideas for upgrade, and they can leave suggestions on Github.

The application can be integrated with the Docker using only MongoDB, but the Docker ARM is quite limited, as it cannot be started with a simple application.

Irodalomjegyzék

- [1] Szabaduló szoba https://hu.wikipedia.org/wiki/Szabaduló_játék
- [2] <https://nodejs.org>, Node.JS alapok
- [3] 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/obudai_diplomamunka
- [6] Git tároló: https://github.com/karpad2/obudai_diplomamunka/tree/esp32_szoftver
- [7] A program logója <https://www.lastingimpressionsonline.co.uk/image/cache/catalog/Product%20Images/Section%203/BS%20Fire%20Exit/03205-500x500.png>
- [8] Blockly <https://developers.google.com/blockly>
- [9] Vue <https://vuejs.org/resources/themes.html>

Felhasznált modulok:

- VueMaterial: <https://www.creative-tim.com/vuematerial/>
- VueNoty: <https://madewithvuejs.com/vue-noty>
- Axios: <https://axios-http.com/docs/intro>
- CryptoJS: <https://cryptojs.gitbook.io/docs/?cacheBust=1634631452035>
- lzma: <https://github.com/LZMA-JS/LZMA-JS>
- moment: <https://momentjs.com/>
- vue-cookie-law: <https://www.npmjs.com/package/vue-cookie-law>
- vue-countdown: <https://github.com/fengyuanchen/vue-countdown>
- blockly: <https://github.com/google/blockly>
- webpack: <https://webpack.js.org/>

6. Mellékletek

Elektronikus melléklet

CD, mely tartalmazza a dokumentációt és összes forráskódot.