|  |  |  |
| --- | --- | --- |
|  | | |
| **Závěrečná studijní práce**  **dokumentace** | | |
| **Systém pro monitorovaní zdravotního stavu pacienta** | | |
| Filip Hluchník | | |
|  | | |
|  | |  |
| **Obor:** | 18-20-M/01 INFORMAČNÍ TECHNOLOGIE  se zaměřením na počítačové sítě a programování | |
| **Třída:**  **Školní rok:** | IT4  2020/2021 | |

###### Poděkování

*Děkuji panu učiteli Ing. Petru Grussmannovi za cenné rady a panu učiteli Mgr. Marcelu Godovskému za pomoc s pájením součástek.*

Prohlašuji, že jsem závěrečnou práci vypracoval samostatně a uvedl veškeré použité   
informační zdroje.

Souhlasím, aby tato studijní práce byla použita k výukovým účelům na Střední průmyslové   
a umělecké škole v Opavě, Praskova 399/8.

V Opavě 3.1.2021

*podpis autora práce*

**Anotace**

Cílem projektu bylo vytvořit systém pro monitorování zdravotního stavu pacienta na dálku přes access point s možností indikace pro přiložení prstu na senzor. Projekt je tvořen z hardwarové a softwarové části. Hardwarovou část tvoří hlavně vývojová deska ESP32 DEVKIT V1-DOIT od firmy Espressif Systems. Zařízení pomocí tří senzorů snímá hodnoty pro měření stavu pacienta. Nejdůležitější senzor MAX30100 snímá tělesnou teplotu a množství kyslíku v krvi pacienta. Další, digitální čidlo na měření tělesné teploty, jednoduché pro měření vlhkosti vzduchu a teploty v místnosti a dvě LED diody pro znamení pro přiložení prstu na oximetr. Programová část je řešena v jazyce C++ a webová část v jazyce HTML, JSON a pomocí komunikačního protokolu WebSockets. Programová část je vytvořena s využitím PlatformIO 5.0.4.

**Klíčová slova**

Zdravotní stav; Espressif Systems; ESP32; WebSockets; WiFi; MAX30100; PlatformIO 5.0.4; C++; JSON;

OBSAH

[Úvod 5](#_Toc503478439)

[1 Výroba systému 6](#_Toc503478440)

[2 Princip fungování 7](#_Toc503478441)

[3 Využité technologie 8](#_Toc503478442)

[3.1 Použitý Hardware 8](#_Toc503478443)

[3.1.1 Seznam součástek 8](#_Toc503478444)

[3.1.2 ESP32 DEVKIT V1-DOIT 8](#_Toc503478445)

[3.1.3 Detektor srdečního tepu s oximetrem MAX30100 9](#_Toc503478446)

[3.1.4 Digitální teplotní čidlo DS18B20 9](#_Toc503478447)

[3.1.5 Teploměr a vlhkoměr DHT11 9](#_Toc503478448)

[3.2 Použitý software 9](#_Toc503478451)

[3.2.1 KiCad 9](#_Toc503478452)

[3.2.2 PlatformIO 5.0.4 10](#_Toc503478453)

[3.2.3 Visual Studio Code 1.52.1 10](#_Toc503478454)

[3.2.4 Webová aplikace 11](#_Toc503478455)

[3.2.5 Jazyk C++ 11](#_Toc503478455)

[3.2.6 Arduino IDE 1.8.13 11](#_Toc503478455)

[4 Způsoby řešení a použité postupy 12](#_Toc503478458)

[4.1 Hardwarové zařízení 12](#_Toc503478459)

[4.1.1 Struktura programu v jazyce Arduino 12](#_Toc503478461)

[4.1.2 Funkce pro inicializaci senzoru MAX30100 12](#_Toc503478462)

[4.1.3 WiFi 12](#_Toc503478463)

[4.1.4 Odesílání hodnot 12](#_Toc503478464)

[4.2 Webová aplikace 13](#_Toc503478468)

[4.2.1 Zabezpečení 14](#_Toc503478469)

[4.2.2 WebSockets 14](#_Toc503478470)

[4.2.3 Ovládání LED diod 15](#_Toc503478471)

[4.2.4 Souborový systém SPIFFS 15](#_Toc503478471)

[5 Výsledky řešení 16](#_Toc503478472)

[5.1 Podoba hardwarového zařízení 16](#_Toc503478473)

[5.2 Podoba webové aplikace 16](#_Toc503478474)

[Závěr 18](#_Toc503478476)

[Seznam použitýCH INFORMAČNÍCH ZDROJů 19](#_Toc503478477)

Úvod

Rozhodl jsem se vytvořit systém pro měření zdravotního stavu pacienta a podmínek v jeho okolí, toto zahrnuje měření teploty pacienta a hodnoty kyslíku v jeho krvi, teplotu a vlhkost v místnosti, protože mi problematika IoT, tedy internetu věcí, připadala velmi zajímavá a sám jsem si chtěl něco takového vyzkoušet. Zařízení lze použít třeba při vzdálené kontrole pacienta, který je například izolovaný doma nebo v nemocniční místnosti, přístroj by tak mohl být eventuálně umístěn například vedle lůžka pacienta. Hardwarovou část tvoří hlavně vývojová deska ESP32 a k němu připojené senzory které jsou popsány v další části dokumentace. Zařízení je naprogramováno v jazyce C++ a jeho webová část je řešena pomocí komunikačního protokolu WebSockets. Komunikace mezi aplikací a zařízením probíhá přes WiFi.

V první části mé dokumentace se věnuji hlavně použitému hardwaru a technologiím, které jsem při řešení svého projektu použil, dále popisuje způsob komunikace mezi serverem a zařízením a uživatelské rozhraní.

# Výroba systému

První částí mého projektu představovalo sestavení a spojení všech součástí na nepájivé pole. Při výběru jsem se díval na přesnost a funkčnost daného senzoru, ale také cenu a dostupnost. Jako základní desku jsem zvolil ESP32, kvůli jejímu výkonu a možností připojení k Wifi.

Jelikož jsem ještě nikdy s těmito součástkami nepracoval, rozhodl jsem se zapojovat jednu součástku po druhé do nepájivého pole a zkoušel jsem její funkčnost.

Následovalo programování zařízení a implementace komunikačního protokolu. Rozhodoval jsem se mezi protokolem SocketIO a WebSockets, ale po rozsáhlém hledání na internetu jsem zjistil že protokol WebSockets se pro projekty mého typu využívá častěji, proto jsem se rozhodl použít právě WebSockets. Následovalo programování zařízení a samotná implementace WebSockets do mého projektu. Když byla první část projektu plně funkční, následovalo psaní kódu webového rozhraní projektu a implementace signalizačních LED diod.

Podobu webové stránky jsem řešil v jazyce HTML a po funkční stránce senzorů a LED diod pomocí jazyků JavaScript a JSON. Pro přístup k rozhraní se stačí připojit k WiFi síti „kontrola“ desky ESP32 a poté přejít na URL adresu „esp.local/“.

# Princip fungování

Systém funguje následovně: když se rozsvítí zelená LED dioda, je to signál pro pacienta pro přiložení prstu na senzor MAX30100, ten změří jeho tep a hladinu kyslíku v krvi. Systém také neustále měří vlhkost a teplotu vzduchu pomocí levného senzoru DHT11, ten sice není příliš přesný, ale to pro měření venkovních není potřeba. Systém také dokáže změřit teplotu těla pacienta při strčení senzoru DS18B20 do podpaží.

Pokud není senzor DHT11 či senzor MAX30100 správně připojen, program vypíše chybovou hlášku a proces se zastaví. Pokud je vše v pořádku WebSockets naváže handshake a vydá požadavek pro update. Všechna data jsou zasílána na server a přehledně zobrazována na webové stránce, na které také lze zapínat a vypínat LED diody.

# Využité technologie

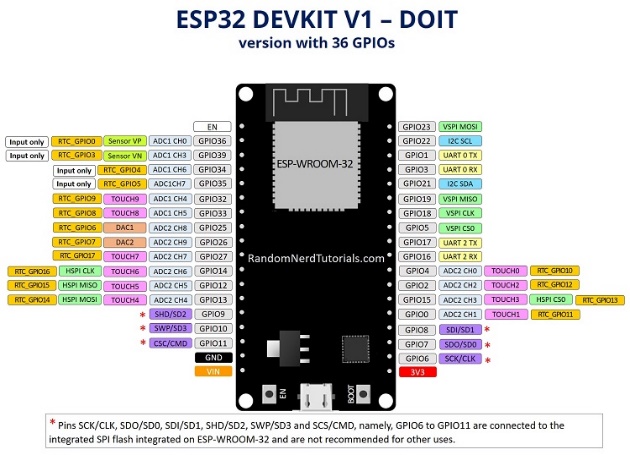
## Použitý hardware

### Seznam součástek

* Vývojová deska ESP32 DEVKIT V1-DOIT,
* Senzor MAX30100,
* Senzor DS18B20,
* Senzor DHT11,
* červená a zelená LED,
* rezistor 4.7 K Ω,
* 2x rezistor 330 Ω,
* 2 nepájivé pole se 400 kontakty.

### ESP32 DEVKIT V1-DOIT

ESP32 DEVKIT V1-DOIT - (IOT modul s ESP-32S) je vývojová deska s integrovanou anténou a RF balunem, výkonovým zesilovačem, nízkošumovými zesilovači, filtry a modulem řízení spotřeby. Tato deska má integrované dual-core processor s 2.4 GHz Wi-Fi a Bluetooth čipy s technologií TSMC 40nm s nízkou spotřebou



*Obrázek č. 1 Piny ESP32 DEVKIT V1-DOIT*

*Obrázek č. 2 ESP32 DEVKIT V1-DOIT*

### Detektor srdečního tepu s oximetrem MAX30100

Digitální sensor MAX30100 obsahuje dvě LED (infračervené a červené světlo) pro detekci pulzu a saturaci kyslíku. Senzor obsahuje dva páry přijímačů a emitorů světla které fungují na mírně odlišných vlnových délkách. Používá odlišnosti schopnosti okysličené a neokysličené krve, aby zjistil, jak moc kyslíku se v krvi nachází (hemoglobin který přenáší kyslík v červených krvinkách odráží červené světlo). Výstup tvoří tok hodnot, které označují přechod a odraz určitých vlnových délek světla. Výpočet výsledného tepu pak probíhá pomocí počítání špiček hodnot. Používá svou knihovnu MAX30100\_PulseOximeter.h.

### Digitální teplotní čidlo DS18B20

Čidlo DS18B20 je digitální teploměr s kalibrovanou **přesností ±0,5 °C a rozlišením až 12 bitů**. Informace o teplotě se z čidla do Arduina přenáší digitálně, **nedochází tak ke zkreslení naměřené hodnoty** ani změně v důsledku rušení. DS18B20 využívá jednoduché [komunikační rozhraní OneWire](https://cs.wikipedia.org/wiki/1-Wire).h, které **pro přenos dat potřebuje pouze jeden vodič**. Teplotní rozsah je -55 ÷ +125 °C a rozlišovací schopnost je až ± 0,0625 °C.

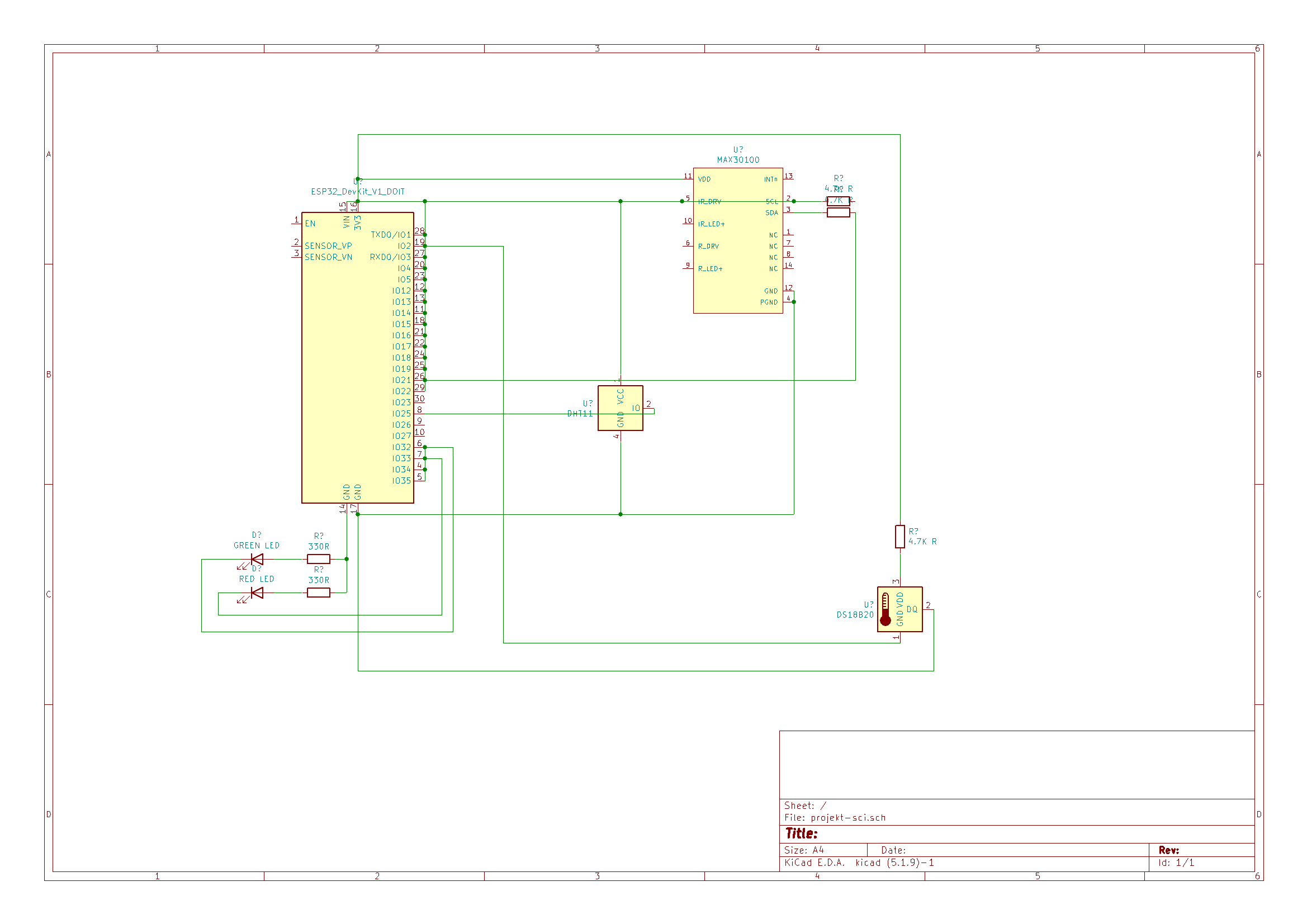
### Teploměr a vlhkoměr DHT11

Teploměr a vlhkoměr DHT11 je snadno použitelný modul, který obsahuje vše potřebné již ve svém pouzdře. Stačí ho tedy pouze připojit třemi vodiči k vývojové desce a dále už jen vyčítat nové informace o okolí, které se obnovují přibližně každé dvě vteřiny. Toto čidlo pro svou funkci využívá knihovnu DHT.h.

## Použitý software

### KiCAD

KiCad je multiplatformní Open-Source profesionální software pro návrh a projektování plošných spojů. Krom samotného návrhu schémat zapojení elektrických obvodů a jejich převodu do plošných spojů zahrnuje například také nástroje pro sestavení kusovníku a export do formátu Gerber.



*Schéma zapojení obvodu*

### PlatformIO 5.0.4

Jako vývojové prostředí jsem zvolil PlatformIO, které kombinuje IDE (tedy nástroj pro zápis zdrojového kódu), správce knihoven a nástroje pro překlad a nahrání výsledného projektu do vývojové desky. Podporovaným programovacím jazykem je C/C++.

### Visual Studio Code 1.52.1

Jako editor zdrojového kódu jsem použil Visual Studio Code. Protože mi je ze všech programů nejbližší a také kvůli jeho spojení s PlatformIO IDE. Visual Studio Code se také snadno ovládá a má přehledné uživatelské rozhraní.

### Webová aplikace

Webovou aplikaci jsem vytvořil v jazyce HTML a CSS pomocí frameworku Bootstrap 4. Celá aplikace běží na WebSockets HTTP Serveru. K provozu aplikace jsem použil funkce JavaScript a JSON, pomocí kterých vypisuji data a ovládám LED diody.

### Jazyk C++

C++ je programovací jazyk který vyvinul v roce 1985 Bjarne Stroustrup jako rozšíření jazyka C. Podporuje několik [programovacích stylů (paradigmat)](https://cs.wikipedia.org/wiki/Programovac%C3%AD_paradigma) jako je [procedurální programování](https://cs.wikipedia.org/wiki/Imperativn%C3%AD_programov%C3%A1n%C3%AD), [objektově orientované programování](https://cs.wikipedia.org/wiki/Objektov%C4%9B_orientovan%C3%A9_programov%C3%A1n%C3%AD) a [generické programování](https://cs.wikipedia.org/wiki/Generick%C3%A9_programov%C3%A1n%C3%AD). V současné době patří C++ mezi nejrozšířenější programovací jazyky.

### Arduino IDE 1.8.13

Pro testování kódu jednotlivých součástek jsem využil Arduino IDE, protože je velmi jednoduché a nenáročné, pro tvorbu celého projektu ale až tak nehodí. Vývojové prostředí Arduina sestává z textového editoru pro psaní kódu, prostoru pro zprávy, textové konzole.

# Způsoby řešení a použité postupy

## Hardwarové zařízení.

### Struktura programu v jazyce Arduino

Funkce setup() je přípravná a provádí se jen jednou na začátku programu, funkce loop() je výkonná a provádí se neustále dokola. Pro správnou činnost programu je vždy nutné použít obě tyto funkce.

### Funkce pro inicializaci senzoru MAX30100

Funkce inicializuje pomocí knihovny MAX30100\_PulseOximeter.h senzor MAX30100 a spustí pulzní oximetr pokud se jeho spuštění nepovede, vypíše se chybová hláška. Funkce také obsahuje parametr pro určování proudu, pomocí kterého se nastavuje citlivost laseru.

1. **if** (!pox.begin())
2. {
3. Serial.println("FAILED");
4. **for** (;;)
5. ;
6. }
7. **else**
8. {
9. Serial.println("SUCCESS");
10. }
11. //pox.setIRLedCurrent(MAX30100\_LED\_CURR\_46\_8MA);

*Ukázka kódu z inicializace MAX30100*

### WiFi

Vývojová deska ESP32 při spuštění spustí vlastní access point, který je zašifrovaný heslem pro větší míru zabezpečení. Stránka se nachází na URL „192.168.4.1/“.

1. WiFi.softAP(ssid, password);
2. Serial.println();
3. Serial.println("Access point bezi");
4. Serial.print("IP adresa: ");
5. Serial.println(WiFi.softAPIP());

*Ukázka kódu z funkce inicializace WiFi*

### Odesílání hodnot

Odesílání dat na server zajišťuje tato funkce, nejprve zakóduje data ze senzorů do JSON výstupu a následně pošle pomocí příkazu broadcastTXT přes protokol WebSockets.

1. String JSON\_Data = "{\"temp\":";
2. JSON\_Data += t;
3. JSON\_Data += ",\"hum\":";
4. JSON\_Data += h;
5. JSON\_Data += ",\"tep\":";
6. JSON\_Data += BPM;
7. JSON\_Data += ",\"ox\":";
8. JSON\_Data += SpO2;
9. JSON\_Data += ",\"bodytemp\":";
10. JSON\_Data += bodytemperature;
11. JSON\_Data += "}";
12. Serial.println(JSON\_Data);
13. websockets.broadcastTXT(JSON\_Data);

## Webová aplikace

*Ukázka kódu z funkce pro odesílaní JSON dat přes WebSockets*

Podoba webové aplikace je poměrně jednoduchá a přehledná, vypisuje hodnoty ze senzorů a obsahuje tlačítka pro ovládáni LED diod. Je chráněna heslem pomocí jednoduché funkce.

1. <header **class**="jumbotron jumbotron-fluid change">
2. <div **class**="center">
3. <h1 **class**="text-black display-2">System pro monitorovani­ zdravotni­ho stavu pacienta</h1>
4. </div>
5. </header>
6. <main **class**="container center">
7. <div **class**="row" style="margin: 10px 10px;">
8. <div **class**="col-12">
9. <h3> LED RED </h3>
10. <button onclick="button\_1\_on()">On</button><button onclick="button\_1\_off()">Off</button>
11. <h3> LED GREEN </h3>
12. <button onclick="button\_2\_on()">On</button><button onclick="button\_2\_off()">Off</button>
13. <h2>Teplota</h2><meter value="2" high="25.5" optimum="22" low="19" min="0" max="100" id="temp\_meter"> </meter>
14. <h3 id="temp\_value" style="display: inline-block;"> 2 </h3>
15. <h2>Vlhkost</h2><meter value="2" optimum="50" min="0" max="100" id="hum\_meter"> </meter>
16. <h3 id="hum\_value" style="display: inline-block;"> 2 </h3>
17. <h2>Tep</h2><meter value="2" high="120" optimum="75" min="0" max="250" id="bpm\_meter"> </meter>
18. <h3 id="bpm\_value" style="display: inline-block;"> 2 </h3>
19. <h2>Kyslik</h2><meter value="2" low="90" optimum="95" min="0" max="100" id="SpO2\_meter"> </meter>
20. <h3 id="SpO2\_value" style="display: inline-block;"> 2 </h3>
21. <h2>Teplota tela</h2><meter value="2" high="37" optimum="36.6" low="36" min="0" max="100"id="bodytemperature\_meter"> </meter>
22. <h3 id="bodytemperature\_value" style="display: inline-block;"> 2 </h3>
23. <br />
24. </div>
25. </div>
26. </main>

*Ukázka kódu z HTML podoby stránky*

### Zabezpečení

Zabezpečení je vyřešeno pomocí jednoduché funkce. Uživateli se při vstupu na stránku zobrazí okénko pro vyplnění hesla.

1. var password;
2. var pass1 = "admin1";
4. password = prompt('Zadejte heslo', ' ');
6. **if** (password == pass1)
7. alert('Spravne heslo!');
8. **else** {
9. window.location = "";
10. }

*Ukázka kódu z funkce pro požadavek hesla*

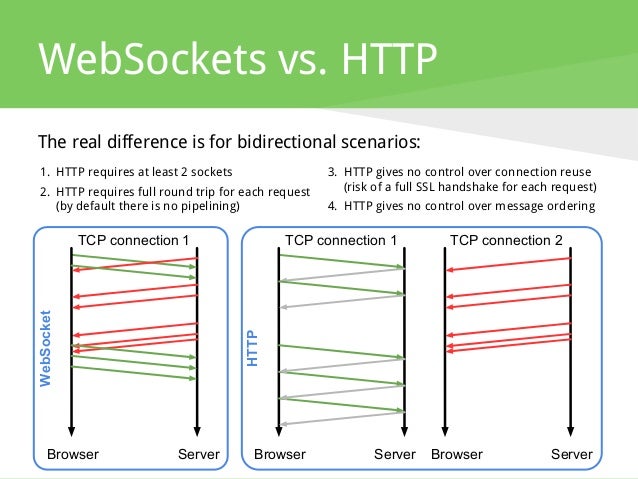
### WebSockets

Je počítačový komunikační protokol, který poskytuje plně duplexní (obousměrný) komunikační kanál přes jediné TCP připojení. Jeho jediný vztah k HTTP je, že jeho handshake (navázání spojení) je interpretováno HTTP servery jako požadavek na upgrade. Umožňuje interakci mezi prohlížečem a webovým serverem s nižší režií, usnadňuje real-time přenos dat ze serveru a na server. To je možné proto, že poskytuje standardizovaný způsob pro odeslání obsahu ze serveru do prohlížeče, aniž by to bylo na požadavek klienta (jako v [AJAXu](https://cs.wikipedia.org/wiki/AJAX" \o "AJAX)), a umožňuje předání zpráv tam i zpět, zatímco je udržováno otevřené spojení. Tímto způsobem může probíhat obousměrná (bi‑directional) komunikace mezi prohlížečem a serverem.

1. var connection = **new** WebSocket('ws://' + location.hostname + ':81/'

);

*Ukázka kódu pro vytvoření WebSocket URL*



*Obrázek č. 3 Websockets vs. HTTP*

### Ovládání LED diod

Ovládání je řešeno pomocí tlačítek ON a OFF. Jejich programová část je řešena pomocí jednoduchých JavaScript funkci.

1. function button\_1\_on() {
2. button\_1\_status = 1;
3. send\_data();
4. }
5. function button\_1\_off() {
6. button\_1\_status = 0;
7. send\_data();
8. }
9. function button\_2\_on() {
10. button\_2\_status = 1;
11. send\_data();
12. }
13. function button\_2\_off() {
14. button\_2\_status = 0;
15. send\_data();
16. }
17. function send\_data() {
18. var full\_data = '{"LED1" :' + button\_1\_status + ',"LED2":' + button\_2\_status + '}';
19. connection.send(full\_data);
20. }

Funkce prvního a druhého tlačítka v HTML kódu:

1. <h3> LED RED </h3>
2. <button onclick="button\_1\_on()">On</button>
3. <button onclick="button\_1\_off()">Off</button>
4. <h3> LED GREEN </h3>
5. <button onclick="button\_2\_on()">On</button>
6. <button onclick="button\_2\_off()">Off</button>

*Ukázka kódu funkcí tlačítek*

### Souborový systém SPIFFS

Soubory html, js a css jsou připojeny pomocí souborového systému SPIFFS, který zajištuje, aby se html stránka po připojení na ip adresu zobrazila.

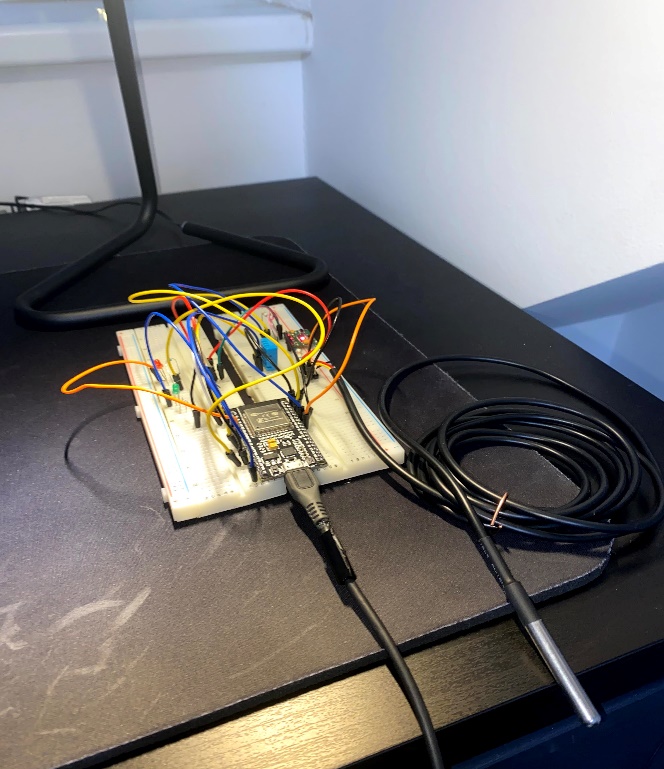
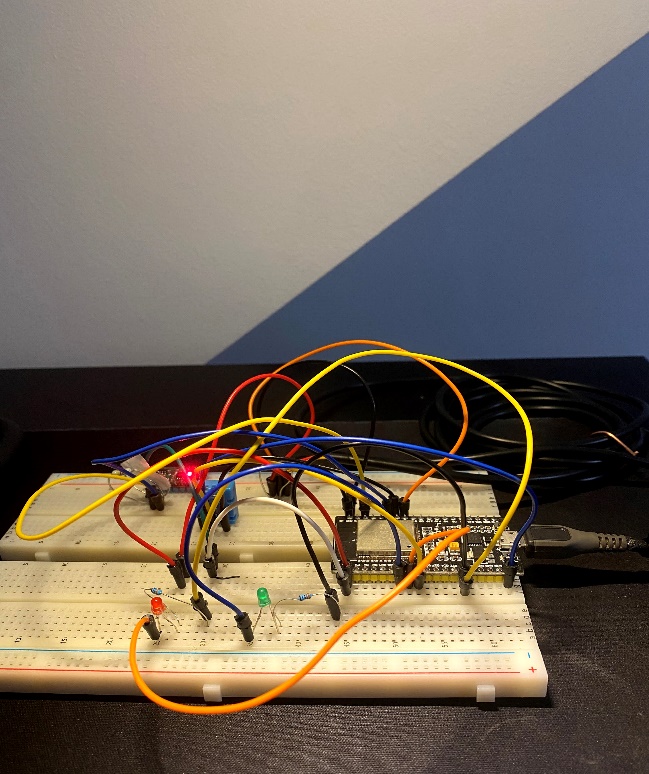
1. server.on("/", HTTP\_GET, [](AsyncWebServerRequest \*request) {
2. request->send(SPIFFS, "/index.html", String());
3. });
5. server.on("/style.css", HTTP\_GET, [](AsyncWebServerRequest \*request) {
6. request->send(SPIFFS, "/css/style.css", String());
7. });
9. server.on("/index.js", HTTP\_GET, [](AsyncWebServerRequest \*request) {
10. request->send(SPIFFS, "/js/index.js", String());
11. });

*Ukázka kódu funkcí SPIFFS*

# Výsledky řešení

## Podoba hardwarového zařízení

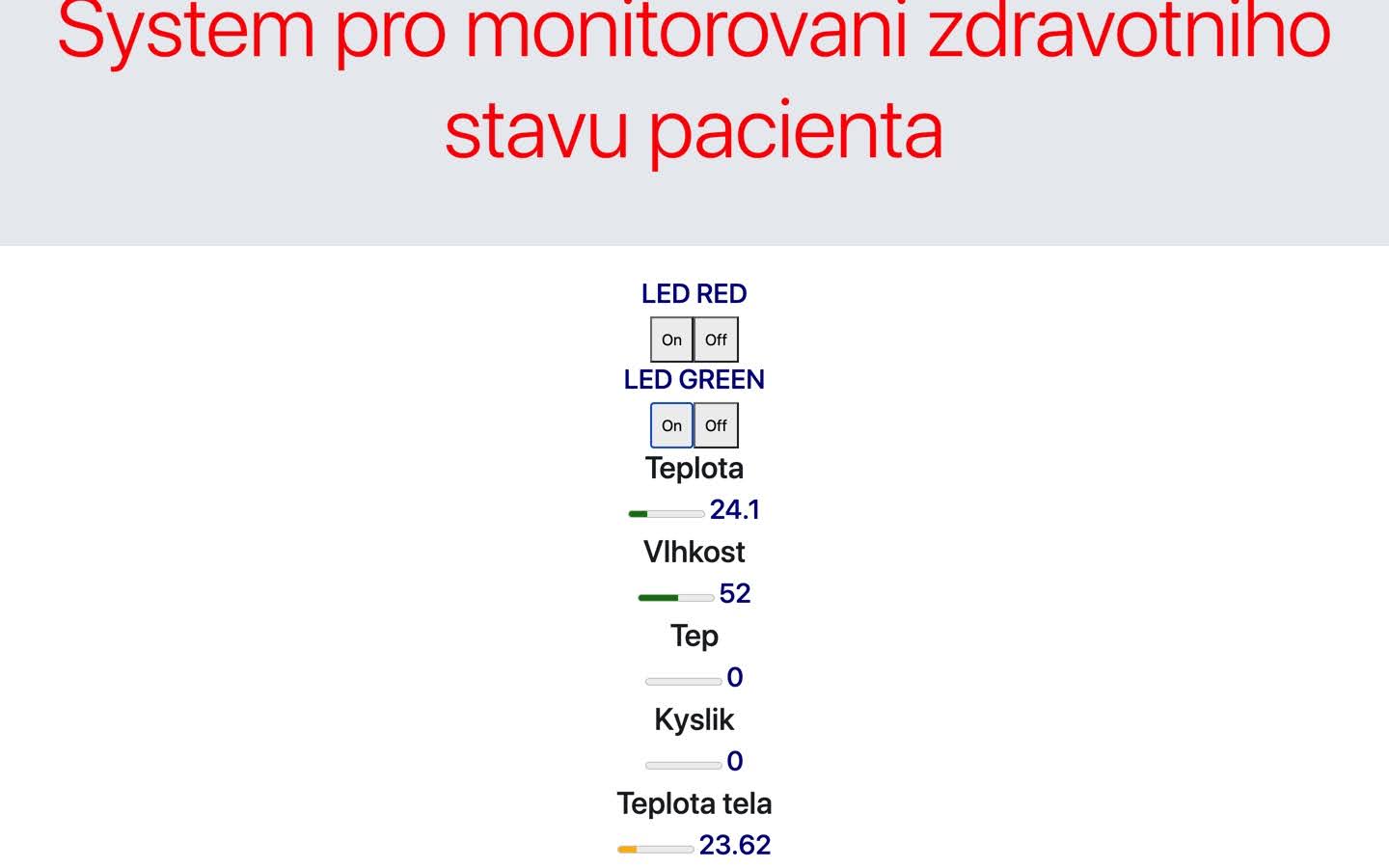
Zařízení je téměř plně funkční až na senzor MAX30100, který ukazuje nepřesné hodnoty tepu.



*Obrázek č. 4 Podoba zařízení*

## Podoba webové aplikace

Webová aplikace splňuje svůj účel, a funguje tak jak má. Uživateli se po zadání hesla zobrazí všechna potřebná data. Stupnice hodnot mění barvu podle stavu hodnoty, pokud jsou hodnoty v optimálním rozmezí, stupnice je zelená, ale pokud jsou hodnoty příliš nízké, či vysoké, stupnice zčervená.



*Obrázek č. 5 Podoba webové stránky*

**Závěr**

Cílem projektu bylo vytvořit zařízení, které by monitorovalo zdravotní stav pacienta a data byla následně posílána a zobrazována na webové stránce. Cílem projektu nebylo vyobrazení hodnot pro pacienta, ale jejich změření a následné zaslání na server pro kontrolu ošetřovatelem. V době psaní této dokumentace nefunguje úplně správně senzor MAX30100. Při vývoji mě napadlo pár nápadů, jak toto zařízení zlepšit. Možná by byla implementace LED obrazovky, která by sloužila k zobrazení hodnot tepu a teplota. Tu jsem se ovšem rozhodl nepoužit z důvodu, že je lepší, když má bezprostřední přístup datům jen ošetřovatel, nikoliv pacient. Další z možností by mohlo být přidání kamery, mikrofonu a reproduktory, díky nimž by mohl ošetřovatel na dálku komunikovat s pacientem, či jej monitorovat. Samotné součástky bych rád umístil na desku plošných spojů místo dvou nepájivých polí, čímž bych vyřešil problém s křehkostí celého zařízení.

Odkaz na GitHub:

https://github.com/filiashluch/stavpacienta/tree/master/Documents/PlatformIO/Projects/final

Seznam použitýCH INFORMAČNÍCH ZDROJů

[1] *Heart rate(HR) sensors* [online]. [cit. 2020-12-27]. Dostupné z: <https://www.chipperbits.com/article/heartrate-sensor-modules/>

[2] *Teplotní čídlo DS18B20* [online]. [cit. 2020-12-27]. Dostupné z: <https://www.hwkitchen.cz/teplotni-cidlo-ds18b20-vodotesne/>

[3] *Teploměr a vlhkoměr DHT11 a DHT22* [online]. [cit. 2020-12-27]. Dostupné z: <https://navody.dratek.cz/navody-k-produktum/teplotni-senzor-dht11.html>

[4] *#93 Cheap heart Sensors: Are they good enough? // Review* [online]. [cit. 2021-1-4]. Dostupné z: https://www.youtube.com/watch?v=57PgD6ZSV4I&ab\_channel=MickMake

[5] *How to use the MAX30100 as Arduino heart rate sensor* [online]. [cit. 2021-1-4]. Dostupné z: https://www.teachmemicro.com/max30100-arduino-heart-rate-sensor/

[6] *ESP32 Tutorial Arduino: 28. websocket server - receiving binary frames* [online].[cit. 2021-1-4]. Dostupné z: <https://www.dfrobot.com/blog->1193.html

[7] *Arduino – příručka programátora* [online]. [cit. 2021-1-4]. Dostupné z: http://www.hobbyrobot.cz/wp-content/uploads/ArduinoPriruckaProgramatora.pdf

[8] ESP32 Pinout Reference: Which GPIO pins should you use? [online]. [cit. 2021-1-4]. Dostupné z: https://randomnerdtutorials.com/esp32-pinout-reference-gpios/

[9] *Arduino-MAX30100* [online]. [cit. 2021-1-4]. Dostupné z: https://github.com/oxullo/Arduino-MAX30100

[10] *Ticker* [online]. [cit. 2021-1-4]. Dostupné z: https://github.com/sstaub/Ticker

[11] *Create Password protected HTML page in Website* [online]. [cit. 2021-1-4]. Dostupné z: https://www.youtube.com/watch?v=7W17Z0X7swo&ab\_channel=TechGeekShan

[12] *arduinoWebSockets* [online]. [cit. 2021-1-4]. Dostupné z: <https://github.com/Links2004/arduinoWebSockets>

[13] *Heart rate (HR) sensors* [online]. In: 18. 1. 2016 [cit. 2021-1-4]. Dostupné z: https://www.chipperbits.com/article/heartrate-sensor-modules/#digital-sensors-max30100rcwl-0530

[14] *JAK NA VODOTĚSNÉ TEPLOTNÍ ČIDLO DS18B20?* [online]. [cit. 2021-1-4]. Dostupné z: https://www.hwkitchen.cz/navody-hwkitchen/jak-na-vodotesne-teplotni-cidlo-ds18b20-arduino-navody/

[15] *JAK PRACOVAT SE SENZORY DHT22 A DHT11?* [online]. [cit. 2021-1-4]. Dostupné z: https://www.hwkitchen.cz/navody-hwkitchen/jak-pracovat-se-senzory-dht22-a-dht11-arduino-navody/

[16] *ESP32 I2C Communication: Set Pins, Multiple Bus Interfaces and Peripherals (Arduino IDE)* [online]. [cit. 2021-1-4]. Dostupné z: https://randomnerdtutorials.com/esp32-i2c-communication-arduino-ide/