|  |  |  |
| --- | --- | --- |
|  | | |
| **Závěrečná studijní práce**  **dokumentace** | | |
| **Teploměr do udírny** | | |
| Adam Rychtář | | |
|  | | |
|  | |  |
| **Obor:** | 18-20-M/01 INFORMAČNÍ TECHNOLOGIE  se zaměřením na počítačové sítě a programování | |
| **Třída:**  **Školní rok:** | IT4  2021/2022 | |

###### 

###### **Poděkování**

* *Děkuji panu učiteli Ing. Petru Grussmannovi za pomoc s vybráním správných technologií a panu Mgr. Markovi Lučnému za pomoc s vyřešením kódových chyb.*

Prohlašuji, že jsem závěrečn ou 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 31. 12. 2021

*podpis autora práce*

**Anotace**

Projekt se zabývá tvorbou zařízení, pomocí kterého lze měřit teplotu v udírně a zobrazit si uložené teploty kdekoli s přístupem na internet. Skládá se z hardwarové a softwarové části. Hlavní část hardwarové části tvoří vývojová deska Raspberry Pi 3 model B. Zařízení pomocí teplotní sondy snímá teplotu v udírně a vypisuje ji na displej a zároveň hodnotu posílá do databáze, ze které se pak vykresluje graf ve službě Grafana přístupná na vlastním serveru. Softwarovou část tvoří aplikace napsaná v jazyce C a CMake. Součástí front-endu je služba Grafana a v rámci back-endu jsou využité služby Influxdb a Node Red. Uživatel si může v Grafaně zobrazit teploty v grafu na časové ose.

**Klíčová slova**

Raspberry Pi 3 model B; Grafana; C; CMake; Influxdb; Node Red

OBSAH

[Úvod 5](#_Toc92654859)

[1 Výroba teploměru do udírny 6](#_Toc92654860)

[2 Využité technologie, kód a postup 7](#_Toc92654861)

[2.1 Hardware 7](#_Toc92654862)

[2.1.1 Seznam součástek 7](#_Toc92654863)

[2.1.2 Raspberry Pi 3 Model B 7](#_Toc92654864)

[2.1.3 Teplotní sonda DS18B20 8](#_Toc92654865)

[2.1.4 7 segmentový displej TM1637 9](#_Toc92654866)

[2.1.5 Bzučák 9](#_Toc92654867)

[2.2 Software 9](#_Toc92654868)

[2.2.1 Jazyk C 9](#_Toc92654869)

[2.2.2 CMake 9](#_Toc92654870)

[2.2.3 Grafana 9](#_Toc92654871)

[2.2.4 InfluxDB 10](#_Toc92654872)

[2.2.5 MQTT 10](#_Toc92654873)

[2.2.6 Docker 11](#_Toc92654874)

[2.2.7 Time Series Admin 11](#_Toc92654875)

[3 Výsledky řešení 12](#_Toc92654876)

[3.1 Podoba hardwarového zařízení 12](#_Toc92654877)

[3.2 Uživatelský front-end Grafana 13](#_Toc92654878)

[3.3 Bezdrátová komunikace 13](#_Toc92654879)

[Závěr 14](#_Toc92654880)

[Seznam použitýCH INFORMAČNÍCH ZDROJů 15](#_Toc92654881)

Úvod

Rozhodl jsem se pro vytvoření teploměru do udírny s webovým rozhraním Grafana pro vykreslování grafů. Velice mě zaujalo složení a naprogramování hardwarových součástek společně s webovými technologiemi.

Hlavním cílem projektu je přečíst teplotu z teplotní sondy a tu pak vypsat na displej a zároveň odesílat data skrze technologii MQTT do databáze Influxdb a ty pak vykreslovat do grafu, o který se stará webová technologie Grafana. Záměrem vykreslování grafu je zjištění teploty v udírně bez neustálého stání přímo u teploměru. Jelikož se jedná o jeden z prvních mých větších projektů využil jsem tedy známých technologií s mnoha návody, jak s nimi zacházet. Celá webová aplikace je spouštěna přes docker-compose, se kterým jsem se již setkal v hodinách počítačových sítí. Hardwarovou část tvoří hlavně vývojová deska Raspberry Pi 3 model B a následně další komponenty, o kterých se zmíním v dalších částech dokumentace. Celý program běžící na Raspberry je naprogramovaný v programovacím jazyce C, který je zkompilovaný pomocí CMake. Komunikaci mezi vývojovou deskou a databází řeší technologie MQTT.

V této dokumentaci podrobně popisuji výrobu teploměru a princip jeho fungování. Na začátku zmiňuji určité problémy, se kterými jsem se při vývoji setkal, poté následuje popis technologií nezbytných k jeho výrobě i k jeho funkčnosti. Zmiňuji fungování samotného teploměru a webová aplikace potřebné pro zjištění teploty na dálku. V další části vysvětluji vzájemnou komunikaci a popisuji jednotlivé úkony obou částí systému. V závěru hodnotím odvedenou práci.

# Výroba teploměru do udírny

Můj projekt již existoval, ale na platformě Arduino UNO, která nemá tolik možností, jako třeba nativní komunikaci se vzdáleným serverem. Základní součástky, např. displej TM1637 a teplotní sondu DS18B20 jsem tedy již měl nakoupené a využil je na tento projekt. Z minulosti se mi doma také povalovalo nevyužité Raspberry Pi 3 Model B, proto jsem se jej rozhodl použít, i přesto že pro tento projekt je zbytečně velké. Zapojení jsem převzal z Arduino verze mého projektu a přízpůsobil tak aby fungovalo s Raspberry Pi.

Následovalo zvolení technologií a programovacích jazyků. Největší zkušennost mám s jazykem Python, proto jsem na začátku zvolil tento jazyk, stále jsem ale neměl úplně jasno v tom, jak bude celý projekt fungovat a jaké technologie bude využívat. Po konzultaci s panem učitelem Ing. Petrem Grussmannem, který mi pomohl ve výběru správných technologií, které mám použít. Následně mi kamarád propůjčil jeho VPS (virtual private server) a jeho doménu vaqe.net, na který jsem nainstaloval Grafanu jako front-end a jako back-end jsem využil Node-RED ve spojení s InfluxDB. Pro přenos dat mezi Raspberry Pi a vzdáleným serverem jsem zvolil MQTT.

Prvně jsem si vytvořil hrubou strukturu adresáře a kódu a vyzkoušel funkčnost displeje TM1637 a teplotní sondy DS18B20. Po úspěšném naprogramování a úpravě některých funkcí jsem přešel na zprovoznění VPS serveru, kde jsem pomocí Docker compose vytvořil kontejery pro Grafanu, Node-RED a InfluxDB. Chvíli jsem si pohrával s myšlenkou provozování vlastního MQTT brokera, ale to je pro můj projekt zbytečné a dostačuje mi veřejný broker projektu EclipseProjects.io. Poté jsem přešel ke zkoušce funkčnosti MQTT brokera a transportu informací o teplotě z Raspberry Pi umístěného v naší pergole, kde máme udírnu a vlastně jakéhokoliv místa kde je schopné se Raspberry Pi připojit k internetu na server. K tomuto účelu jsem si našel knihovnu a naprogramoval jednoduchý kód v Pythonu pro server i Raspberry Pi.

Nechtěl jsem kombinovat více programovacích jazyků a většina knihoven, které jsem potřeboval použít ve svém projektu, byla již napsaná v programovacím jazyce C, upustil jsem tedy od Pythonu úplně.

Na straně serveru za mě toto řeší technologie Node-RED.

# Využité technologie, kód a postup

## Hardware

### Seznam součástek

* Minipočítač Raspberry Pi 3 Model B
* Teplotní sonda DS18B20
* 7 segmentový displej TM1637
* Bzučák

### Raspberry Pi 3 Model B

Minipočítač Raspberry Pi se proslavil svou malou velikostí předurčující jej pro IoT použití. Původním záměrem však byla dostupná alternativa k desktopovým počítačům pro nenáročné aplikace a výuku ve vývojových zemích. Já ve svém projektu používám Raspberry Pi 3 Model B.

*Obrázek č. 1 / Raspberry Pi 3*



### Teplotní sonda DS18B20

DS18B20 je 9/12bitová digitální teplotní sonda která měří v rozsahu od -55 do +125 °C což jsou pro můj projekt plně dostačující vlastnosti. Mezi -10 °C do +85 °C je schopná měřit v přesnosti až na +-0,5°C. S Raspberry Pi komunikuje pomocí tří vodičů (+5V, Data, GND) a pro usnadnění manipulace se připojuje skrze malou desku plošných spojů se svorkovnicí pro připojení teplotní sondy a konektory Dupont pro připojení k GPIO pinům Raspberry Pi. Měření probíhá přibližně co 2-3 sekundy. Jako alternativa DS18B20, by se dala použít LM35 s teplotním rozsahem od -55 do +150 °C. Ale pro můj projekt není moc praktická, protože DS18B20 má i voděodolnou verzi a je tedy vhodná pro venkovní projekty, což LM35 nemá.

// Čte teplotu z čidla

float ReadTemperature(const Sensor \*sensor) {

long deviceFileSize;

char \*buffer;

FILE \*deviceFile = sensor->SensorFile;

fseek(deviceFile, 0, SEEK\_END);

deviceFileSize = ftell(deviceFile);

fseek(deviceFile, 0, SEEK\_SET);

buffer = calloc(deviceFileSize, sizeof(char));

fread(buffer, sizeof(char), deviceFileSize, deviceFile);

char \*temperatureComponent = strstr(buffer, "t=");

if(!temperatureComponent) {

free(buffer);

return -1;

}

temperatureComponent +=2; //move pointer 2 spaces to compensate for t=

float temperatureFloat = atof(temperatureComponent);

temperatureFloat = temperatureFloat / 1000;

if(temperatureFloat < -55)

temperatureFloat = -55;

else if(temperatureFloat > 125)

temperatureFloat = 125;

free(buffer);

return temperatureFloat;

}

### 7 segmentový displej TM1637

Jako výstupní zařízení jsem použil LCD displej 1602, který dokáže zobrazit 32 znaků na dvou řádcích (16 x 2). Kvůli omezenému počtu pinů na NodeMcu jsem pro připojení využil I2C sériové rozhraní, které mi umožnilo připojit displej pouze dvěma vodiči. Funkčnost displeje obstarává knihovna LiquidCrystal\_I2C.

### Bzučák

Slouží pro indikaci příliš vysoké teploty.

## Software

### Jazyk C

Programovací jazyk je velice spolehlivý, který je široce využívaný pro téměř všechny druhy software. Pohání například Linuxové jádro a různé mikrokontroléry. Já jsem jej použil, protože většina knihoven, které v projektu používám je napsáná právě tímto programovacím jazykem. Obsluhuji s ním displej TM1637 a teplotní sondu DS18B20 a také MQTT klienta pro publikování hodnot teploty do daného MQTT topicu.

### CMake

CMake je software pro automatizování kompilace programů napsaných v C/C++ pro použití v růůzných operačních systémech, používá se pro multiplatformní projekty, což není můj případ, nicméně jeho user-friendly „ovládání“ se mi líbí, a proto jej používám pro kompilaci mého programu obsluhující 7 segmentový displej, teplotní sondu DS18B20, bzučák a MQTT klienta.

### Grafana

Grafana je multiplatformní open-source webová aplikace pro analýzu a interaktivní vizualizaci, která poskytuje různé druhy s šírokými možnostmi importu dat do ní, např. time-series NoSQL databázi InfluxDB kterou ve svém projektu používám pro ukládání historie teploty.

### InfluxDB

Jako úložiště dat jsem použil InfluxDB což je time-series NoSQL databáze. Data se v ní ukládají v závislosti na čase.

### MQTT

Pro komunikaci mezi Raspberry Pi a serverem používám protokol MQTT, který byl vyvinutý právě pro potřeby IoT zařízení. Komunikaci mezi koncovými zařízeními zajišťuje tzv. Broker, v mém případě jsem chtěl nejprve využít veřejný broker od EclipseProjects.io, ale jsou zde omezení na počet poslaných dat. Rozhodl jsem se tedy vytvořit na mém VPS serveru vlastního Brokera.

// Připojuje se na MQTT brokera a posílá data o teplotě do topicu

void publish(char stemp[4]) {

int rc;

struct mosquitto \* mosq;

mosquitto\_lib\_init();

mosq = mosquitto\_new("publisher", true, NULL);

rc = mosquitto\_connect(mosq, "mqtt.eclipseprojects.io", 1883, 60);

if(rc != 0){

printf("Nelze se pripojit k brokerovi! Error Code: %d\n", rc);

mosquitto\_destroy(mosq);

}

printf("Pripojeno k brokerovi!\n");

mosquitto\_publish(mosq, NULL, "Rychtar/uzak", 2, stemp, 0, false);

mosquitto\_disconnect(mosq);

mosquitto\_destroy(mosq);

mosquitto\_lib\_cleanup();

}

### Docker

Na VPS serveru používám Docker což je „virtualizační“ software na úrovni operačního systému dostupný pro všechny běžné platformy (Windows, Linux, macOS) který software rozčleňuje do balíčků zvaných kontejnery. Každá služba tedy běží ve svém vlastním kontejneru nezávisle, nebo závisle na druhé což zvyšuje bezpečnost a také replikaci na jakémkoliv jiném počítači na kterém Docker běží.

### Time Series Admin

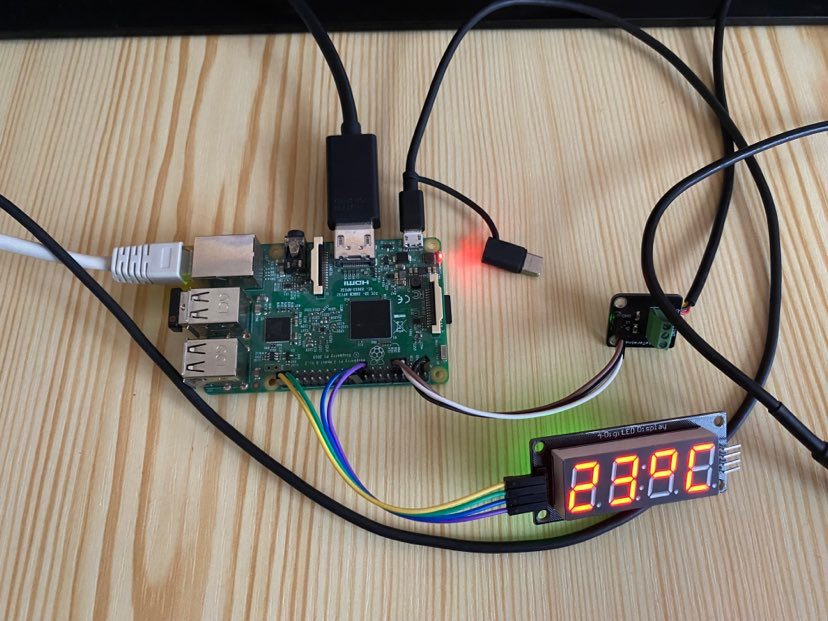
Pro debugging komunikace mezi Node-RED a InfluxDB jsem použil open-source administrátor pro databáze InfluxDB se jménem Time Series Admin, který je volně dostupný na Githubu.

# Výsledky řešení

## Podoba hardwarového zařízení

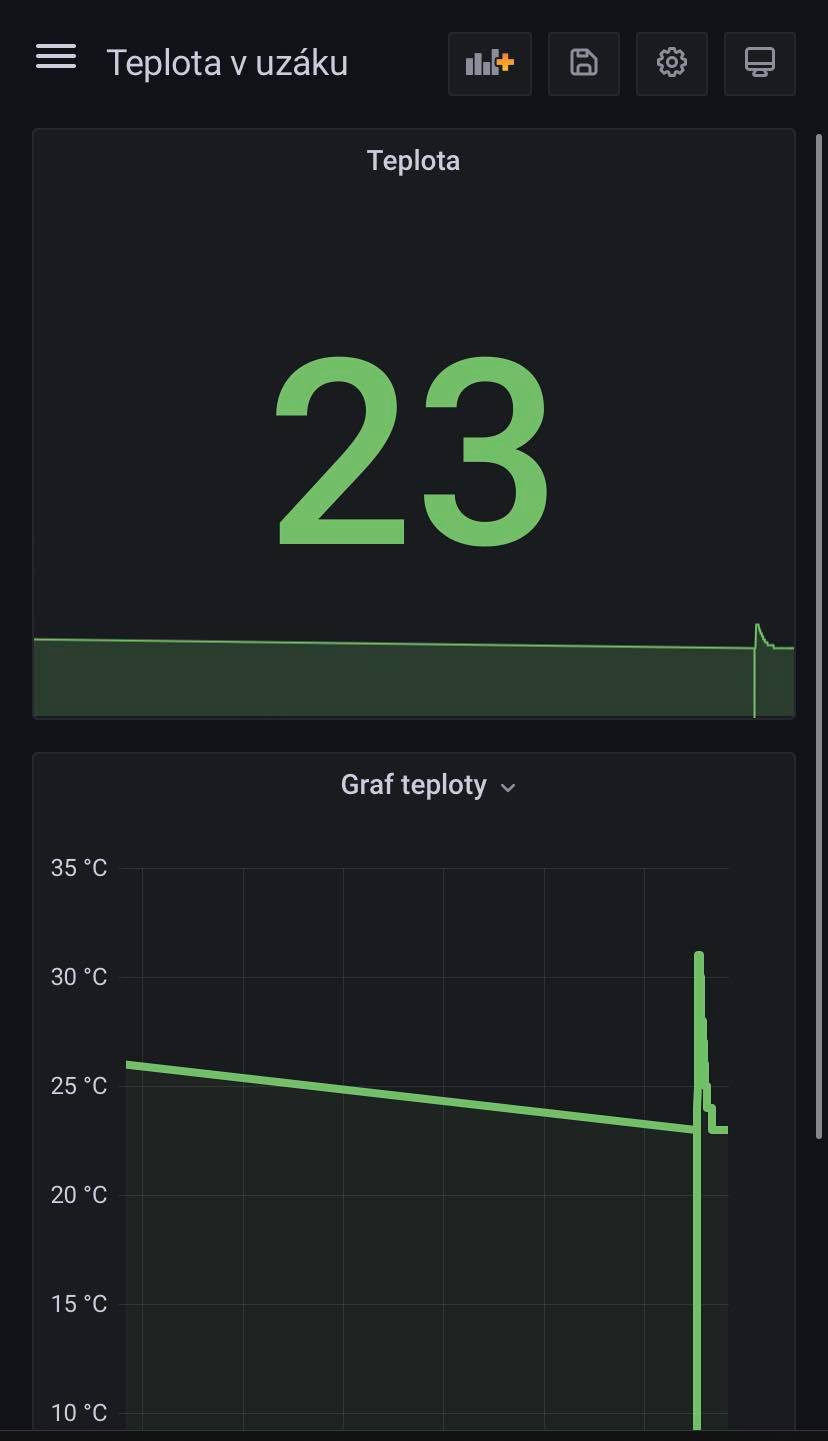
Můj teploměr do udírny jsem umístil do své vlastní krabičky navrhnuté speciálně pro tento projekt. Krabičku jsem navrhnul v CAD programu Autodesk Fusion 360 a vytisknul jí na své 3D tiskárně Ender3 V2. Všechno má tak své místo a celé zařízení je tak chráněno proti vnějším vlivům, jako je např. pád. Pro kabel od teplotní sondy jsem zvolil upevnění kabelovou průchodkou, nehrozí tak její vytržení a celé zařízení je tak částečně a krátkodobě chráněno i proti povětrnostním vlivům, např. mrholení.

*Obrázek č. 2 Teploměr do udírny v téměř hotovém stavu*



## Uživatelský front-end Grafana

Grafana je webová aplikace která zobrazuje data a aktuální naměřené teploty v případě že uživatel není přítoen u zařízení, nebo poblíž něj. V mém případě, můj otec občas při uzení odejde např. do domu a nemá tedy přehled o teplotě v udírně, toto efektivně řeší jeho problém a kdykoliv si může zobrazit aktuální teplotu ve svém telefonu a reagovat.



*Obrázek č. 3 Front-end Grafany*

*Obrázek č. 18 Tlačítko pro aktivaci alarmu*

## Bezdrátová komunikace

Komunikace mezi Grafanou a Raspberry zajišťuje protokol MQTT. Na straně serveru toto řeší sám Node-RED, který data z MQTT příjmá a ukládá ihned do databáze. Grafana následně data stahuje z databáze InfluxDB a zprostředkovává uživateli.

**Závěr**

Mým záměrem bylo vytvořit a přepracovat můj dřívější projekt teploměru do domácí udírny, postavený na základě Arduino UNO, připojit jej k internetu a zprostředkovat jej pro vzdálený přístup k naměřeným hodnotám a grafu s historií naměřených teplot. Pro tento účel jsem zvolil minipočítač Raspberry Pi. Celý projekt jsem sestavil ze součástek, které se mi již válely doma a celý projekt mě tak stál jenom pár korun za zhruba 150 g plastu použitého pro vytisknutí krabičky. Kdybych měl tento projekt dělat příště, zvolil bych nejspíše platformu ESP8266, která se zdá být pro tento projekt vhodnější než Raspberry Pi 3 Model B.

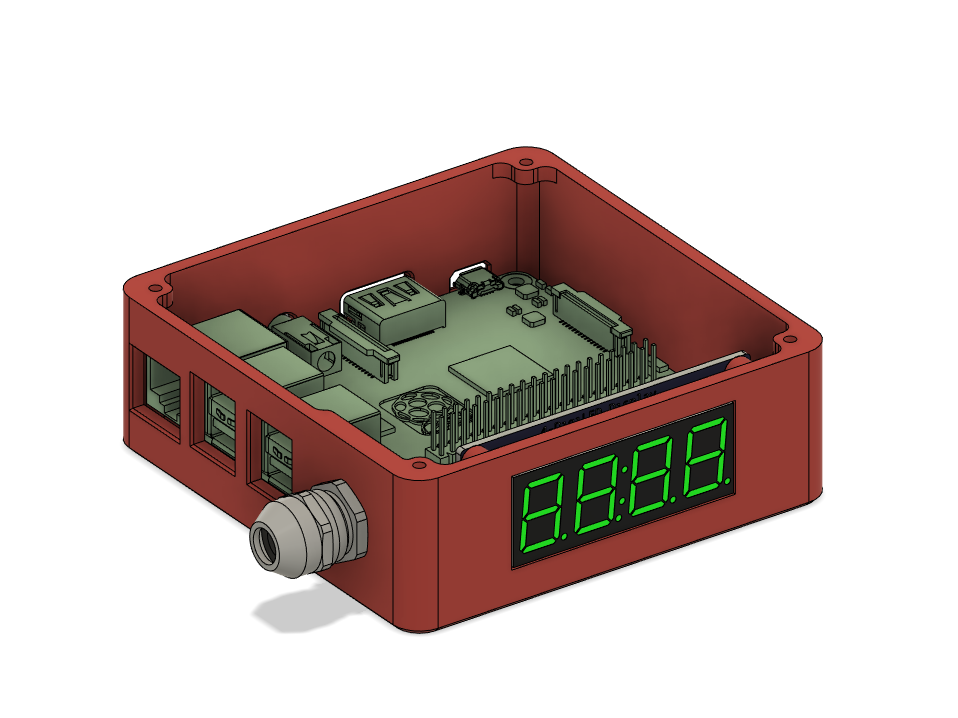
Seznam použitýCH INFORMAČNÍCH ZDROJů

[1] Dokumentace Docker. Docker [online]. [cit. 2021-12-31]. Dostupné z: https://docs.docker.com/

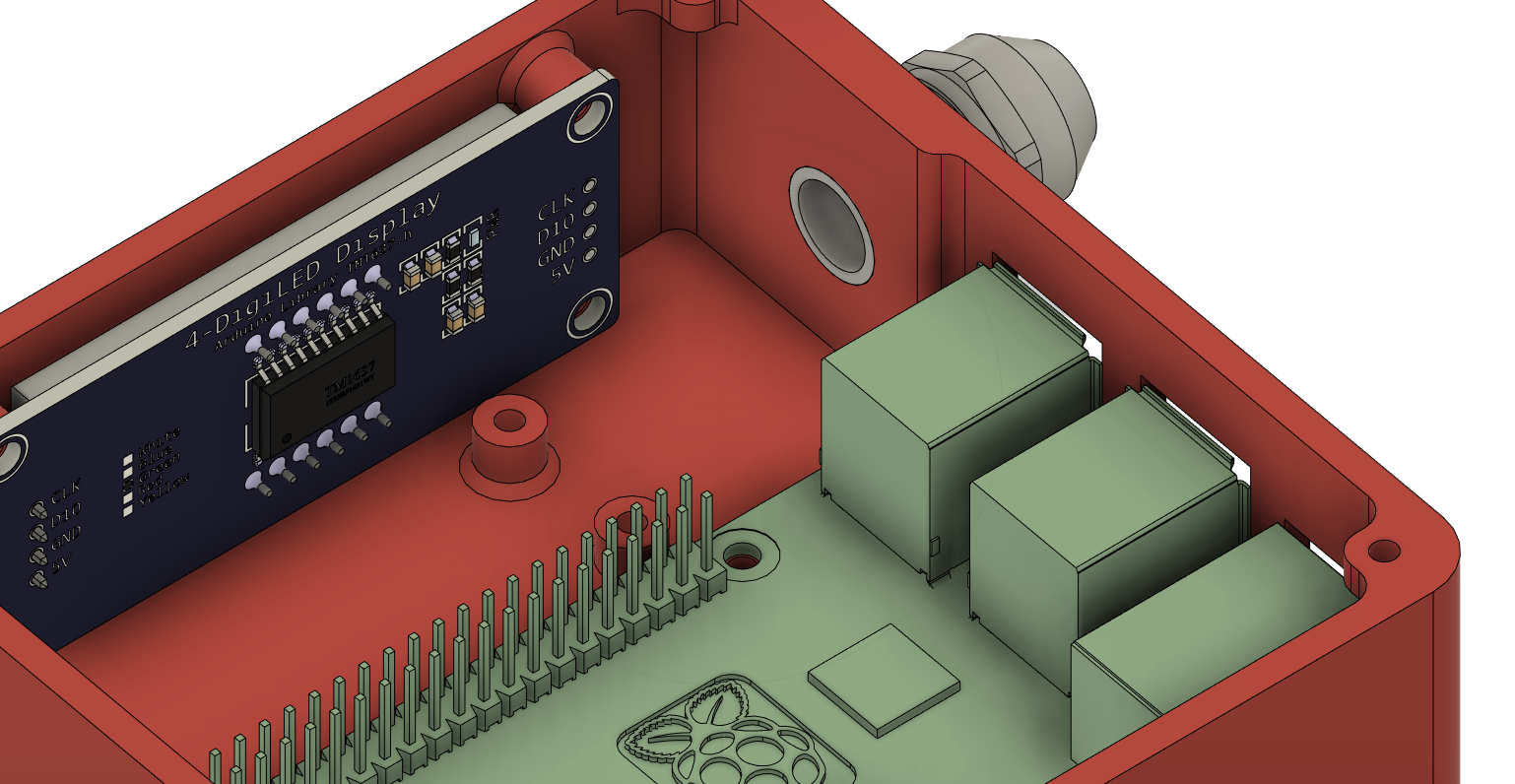
[2] Dokumentace Node-RED. Docker [online]. [cit. 2021-12-31]. Dostupné z: https://nodered.org/docs/

[3] #255 Node-Red, InfluxDB, and Grafana Tutorial on a Raspberry Pi. Node-RED [online]. [cit. 2021-12-31]. Dostupné z: https://www.youtube.com/watch?v=JdV4x925au0

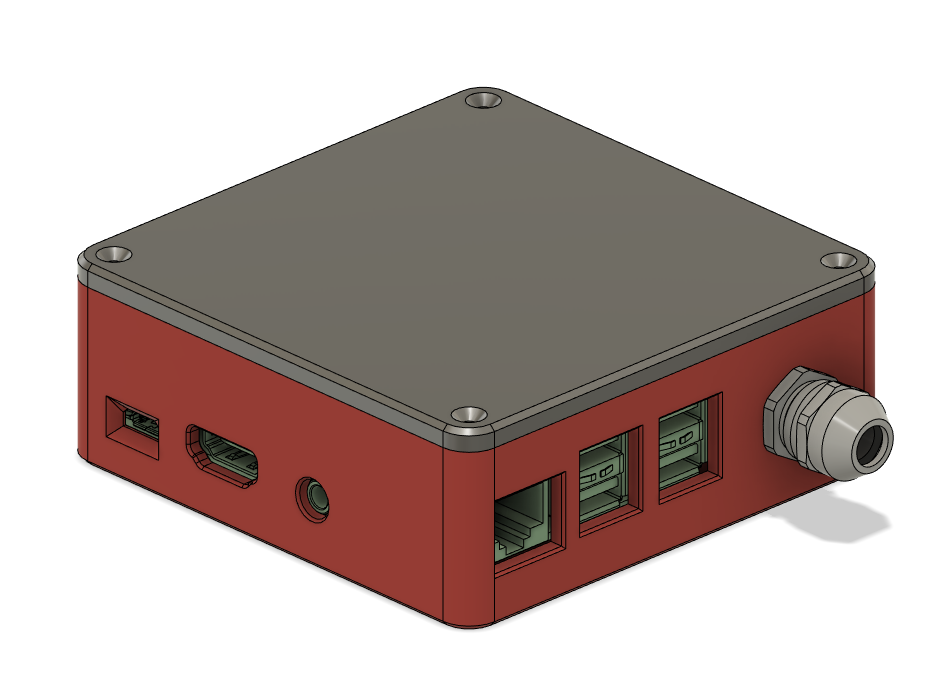
**Rendery krabičky**



3 D Render – Krabička bez víka (čelní pohled)



3 D Render – Krabička pohled dovnitř



3 D Render – Krabička



*3D Render – finální podoba*

Dokončená krabička